|  |
| --- |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
| ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC  NGÀNH CÔNG NGHỆ THÔNG TIN |
| **KIỂM THỬ TỰ ĐỘNG BẰNG CÔNG CỤ SELENIUM WEBDRIVER CHO HỆ THỐNG HỌC TRỰC TUYẾN LEARNICO** |
|  |
|  |
| **GVHD:** ThS. Hoàng Quang Huy |
| **Sinh viên:** Nguyễn Thị Kim Chi  **Mã số sinh viên:** 2021603849 |
|  |
| **Hà Nội – Năm 2025** |

**LỜI CẢM ƠN**

Để được kết quả như ngày hôm nay, trước tiên em xin gửi lời cảm ơn sâu sắc đến thầy Hoàng Quang Huy, giảng viên khoa Công nghệ thông tin, Đại học Công nghiệp Hà Nội đã tận tình hướng dẫn, giúp đỡ em trong suốt quá trình nghiên cứu và thực hiện đồ án tốt nghiệp.

Em cũng xin bày tỏ lòng biết ơn chân thành đến với các thầy cô trong đại học Công nghiệp Hà Nội đặc biệt là các thầy cô trong khoa Công nghệ thông tin đã tận tụy dậy dỗ em trong quá trình học tập suốt thời gian qua, để em có được những kiến thức không chỉ để làm đồ án mà còn là học tập và rèn luyện cho công việc sau này.

Cuối cùng em xin cảm ơn đến những người bạn đã cùng em học tập, rèn luyện và hỗ trợ , giúp đỡ em trong quá trình học tập, nghiên cứu và hoàn thành đồ án tốt nghiệp.

Trong quá trình hoàn thành đồ án, mặc dù đã cố hêt sức để hoàn thiện đề tài tốt nhất có thể, nhưng chắc chắn không thể thoát khỏi những thiếu sót vì vậy em rất mong nhận được sự góp ý của thầy cô và các bạn để đồ án em hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2025

Sinh viên thực hiện

Nguyễn Thị Kim Chi

**MỤC LỤC**

[DANH MỤC CÁC CHỮ VIẾT TẮT vi](#_Toc199114687)

[DANH MỤC BẢNG BIỂU vii](#_Toc199114688)

[DANH MỤC HÌNH ẢNH viii](#_Toc199114689)

[MỞ ĐẦU 1](#_Toc199114690)

[CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ 4](#_Toc199114691)

[1.1. Kiểm thử phần mềm 4](#_Toc199114692)

[1.1.1. Khái niệm kiểm thử phần mềm 4](#_Toc199114693)

[1.1.2. Mục đích và vai trò của kiểm thử phần mềm 4](#_Toc199114694)

[1.1.3. Quy trình kiểm thử phần mềm 6](#_Toc199114695)

[1.1.4. Các loại kiểm thử 7](#_Toc199114696)

[1.1.5. Các cấp độ kiểm thử phần mềm 11](#_Toc199114697)

[1.1.6. Kỹ thuật kiểm thử hộp đen 14](#_Toc199114698)

[1.1.7. Thiết kế ca kiểm thử và tạo Bug report 17](#_Toc199114699)

[1.2. Khái quát về kiểm thử tự động 20](#_Toc199114700)

[1.2.1. Khái niệm kiểm thử tự động 20](#_Toc199114701)

[1.2.2. Quy trình kiểm thử tự động 21](#_Toc199114702)

[1.2.3. So sánh kiểm thử thủ công và kiểm thử tự động 23](#_Toc199114703)

[1.2.4. Một số công cụ kiểm thử tự động 25](#_Toc199114704)

[CHƯƠNG 2. CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM WEBDRIVER 26](#_Toc199114705)

[2.1. Công cụ kiểm thử tự động selenium 26](#_Toc199114706)

[2.1.1. Khái quát về selenium 26](#_Toc199114707)

[2.1.2. Đặc điểm của selenium 26](#_Toc199114708)

[2.1.3. Các thành phần của selenium 27](#_Toc199114709)

[2.2. Selenium Webdriver 30](#_Toc199114710)

[2.2.1. Tổng quan về Selenium Webdriver 30](#_Toc199114711)

[2.2.2. Tiền thân của Selenium Webdriver 30](#_Toc199114712)

[2.2.3. Đặc điểm nổi bật của Selenium Webdriver 31](#_Toc199114713)

[2.2.4. Cách lấy Xpath của element trên website 31](#_Toc199114714)

[2.2.5. Các câu lệnh sử dụng trong Selenium Webdriver 33](#_Toc199114715)

[2.3. Công cụ và thư viện khác được sử dụng 36](#_Toc199114716)

[2.3.1. Ngôn ngữ Python 36](#_Toc199114717)

[2.3.2. Lambdatest 37](#_Toc199114718)

[2.3.3. Thư viện OpenCV (cv2) và PyAutoGUI 38](#_Toc199114719)

[2.3.4. Thư viện Requests 38](#_Toc199114720)

[2.3.5. Thư viện CuteCharts 39](#_Toc199114721)

[2.4. Cài đặt selenium Webdriver 39](#_Toc199114722)

[2.4.1. Cài đặt Python 39](#_Toc199114723)

[2.4.2. Cài đặt PyCharm 39](#_Toc199114724)

[2.4.3. Cài đặt Selenium Webdriver 40](#_Toc199114725)

[2.4.4. Xây dựng dự án Selenium WebDriver trong PyCharm 41](#_Toc199114726)

[2.4.5. Các bước thực hiện kiểm thử tự động với Selenium WebDriver trên Lambdatest 42](#_Toc199114727)

[CHƯƠNG 3. KIỂM THỬ CHO HỆ THỐNG HỌC TRỰC TUYẾN LEARNICO 43](#_Toc199114728)

[3.1. Mô tả về hệ thống học trực tuyến Learnico 43](#_Toc199114729)

[3.1.1. Giới thiệu 43](#_Toc199114730)

[3.1.2. Biểu đồ usecase 44](#_Toc199114731)

[3.1.3. Mô tả usecase 45](#_Toc199114732)

[3.2. Kế hoạch kiểm thử 52](#_Toc199114733)

[3.2.1. Phạm vi kiểm thử 52](#_Toc199114734)

[3.2.2. Mục tiêu kiểm thử 52](#_Toc199114735)

[3.2.3. Chiến lược kiểm thử 52](#_Toc199114736)

[3.2.4. Tiêu chí kiểm thử 53](#_Toc199114737)

[3.2.5. Nguồn nhân lực 54](#_Toc199114738)

[3.2.6. Môi trường kiểm thử 54](#_Toc199114739)

[3.2.7. Lịch trình và dự toán 55](#_Toc199114740)

[3.3. Thiết kế ca kiểm thử 55](#_Toc199114741)

[3.3.1. Kiểm thử chức năng 55](#_Toc199114742)

[3.3.2. Kiểm thử tương thích hệ thống 79](#_Toc199114743)

[3.4. Xây dựng test script 81](#_Toc199114744)

[3.4.1. Xây dựng test data 81](#_Toc199114745)

[3.4.2. Xây dựng script bộ testcase 82](#_Toc199114746)

[3.4.3. Xây dựng test script toàn bộ module 85](#_Toc199114747)

[3.4.4. Xây dựng báo cáo HTML 87](#_Toc199114748)

[3.4.5. Xây dựng script cho kiểm thử tương thích sử dụng lambdatest 90](#_Toc199114749)

[3.5. Kết quả kiểm thử 91](#_Toc199114750)

[KẾT LUẬN 93](#_Toc199114751)

[TÀI LIỆU THAM KHẢO 95](#_Toc199114752)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| Từ viết tắt | Giải thích |
| SRS | Software Requirements Specification |
| API | Application Programming Interface |
| UI | User Interface |
| UAT | User Acceptance Testing |
| CI/CD | Continuous Integration/ Continuous Deployment/Delivery |
| XML | eXtensible Markup Language |
| JSON | JavaScript Object Notation |
| HTML | Hypertext Markup Language |
| ID | Identification |
| OS | Operating System |
| SQL | Structured Query Language |
| URL | Uniform Resource Locator |
| PHP | Hypertext Preprocessor |
| UX | User Experience |
| IDE | Integrated Development Environment |

# DANH MỤC BẢNG BIỂU

[Bảng 1.1: Bảng quyết định màn hình Login 16](#_Toc199114753)

[Bảng 1.2: So sánh kiểm thử thủ công và kiểm thử tự động 23](#_Toc199114754)

[Bảng 3.1: Nguồn nhân lực 54](#_Toc199114755)

[Bảng 3.2: Lịch trình và dụ toán 55](#_Toc199114756)

[Bảng 3.3: Test case Đăng Nhập 56](#_Toc199114757)

[Bảng 3.4: Test case chức năng Thêm bài giảng 58](#_Toc199114758)

[Bảng 3.5: Test case chức năng Cập nhật bài giảng 62](#_Toc199114759)

[Bảng 3.6: Test case chức năng Xóa bài giảng 66](#_Toc199114760)

[Bảng 3.7: Test case chức năng Thêm danh mục bài giảng 68](#_Toc199114761)

[Bảng 3.8: Test case chức năng Cập nhật danh mục bài giảng 70](#_Toc199114762)

[Bảng 3.9: Test case chức năng Xóa danh mục bài giảng 72](#_Toc199114763)

[Bảng 3.10: Test case chức năng Thêm loại bài giảng 74](#_Toc199114764)

[Bảng 3.11: Test case chức năng Cập nhật loại bài giảng 76](#_Toc199114765)

[Bảng 3.12: Test case chức năng Xóa loại bài giảng 78](#_Toc199114766)

[Bảng 3.13: Test case chức năng Xóa loại bài giảng 79](#_Toc199114767)

[Bảng 3.14: Bảng tổng hợp kết quả kiểm thử chức năng 91](#_Toc199114768)

[Bảng 3.15: Bảng tổng hợp kết quả kiểm thử tương thích 91](#_Toc199114769)

[Bảng 3.16: Bảng tổng hợp lỗi 91](#_Toc199114770)

# DANH MỤC HÌNH ẢNH

[Hình 1.1: Quy trình kiểm thử phần mềm 6](#_Toc199114771)

[Hình 1.2: Minh họa testcase 18](#_Toc199114772)

[Hình 1.3: Một bug report được log trên Redmine 19](#_Toc199114773)

[Hình 2.1: Cấu trúc của Xpath 32](#_Toc199114774)

[Hình 2.2: Cách lấy Xpath 33](#_Toc199114775)

[Hình 2.3: Ví dụ mở Google bằng Selenium WebDriver 42](#_Toc199114776)

[Hình 2.4: Tạo script kiểm thử tích hợp với Selenium WebDriver 43](#_Toc199114777)

[Hình 3.1: Biểu đồ phân rã usecase Quản lý bài giảng 44](#_Toc199114778)

[Hình 3.2: Biểu đồ phân rã usecase Danh mục bài giảng 44](#_Toc199114779)

[Hình 3.3: Biểu đồ phân rã usecase Loại bài giảng 45](#_Toc199114780)

[Hình 3.4: Màn hình Đăng nhập 56](#_Toc199114781)

[Hình 3.5: Màn hình Danh sách bài giảng 58](#_Toc199114782)

[Hình 3.6: Màn hình Thêm mới bài giảng 58](#_Toc199114783)

[Hình 3.7: Màn hình Cập nhật bài giảng 62](#_Toc199114784)

[Hình 3.8: Màn hình Xóa bài giảng 66](#_Toc199114785)

[Hình 3.9: Màn hình Danh mục bài giảng 67](#_Toc199114786)

[Hình 3.10: Màn hình Thêm mới danh mục bài giảng 68](#_Toc199114787)

[Hình 3.11: Màn hình Cập nhật danh mục bài giảng 70](#_Toc199114788)

[Hình 3.12: Màn hình Xóa danh mục bài giảng 72](#_Toc199114789)

[Hình 3.13: Màn hình Loại bài giảng 73](#_Toc199114790)

[Hình 3.14: Màn hình Thêm mới loại bài giảng 74](#_Toc199114791)

[Hình 3.15: Màn hình Cập nhật loại bài giảng 76](#_Toc199114792)

[Hình 3.16: Màn hình Xóa loại bài giảng 78](#_Toc199114793)

[Hình 3.17: Hàm xây dựng báo cáo html 88](#_Toc199114794)

[Hình 3.18: Kết quả test được hiển thị lên file html 89](#_Toc199114795)

[Hình 3.19: Xây dựng script cho kiểm thử tương thích sử dụng lambdatest 90](#_Toc199114796)

[Hình 3.20: Kết quả sau khi chạy testcase kiểm thử tương thích trên lambdatest 90](#_Toc199114797)

# MỞ ĐẦU

1. Lý do chọn đề tài

Trong những năm gần đây, sự phát triển mạnh mẽ của công nghệ thông tin đã góp phần thúc đẩy ngành công nghiệp phần mềm trở thành một lĩnh vực then chốt trong tiến trình công nghiệp hóa, hiện đại hóa đất nước. Khi nhu cầu về các hệ thống phần mềm ngày càng tăng, đòi hỏi các tổ chức, doanh nghiệp công nghệ không chỉ tập trung vào việc xây dựng chức năng mà còn phải đảm bảo chất lượng sản phẩm, tối ưu hóa hiệu suất cũng như giảm thiểu rủi ro về lỗi trong quá trình vận hành.

Trong bối cảnh đó, kiểm thử phần mềm là một bước không thể thiếu trong quy trình phát triển, đặc biệt là trong giai đoạn hoàn thiện và triển khai sản phẩm. Kiểm thử giúp xác định phần mềm có hoạt động đúng với yêu cầu đặt ra hay không, có phát sinh lỗi trong quá trình xử dụng không, và sản phẩm có đáp ứng được nhu cầu thực tế của người dùng hay không. Qua đó, kiểm thử phần mềm đóng vai trò then chốt trong việc nâng cao chất lượng của hệ thống.

Tuy nhiên, với các hệ thống ngày càng phức tạp, việc kiểm thử thủ công có thể mất nhiều thời gian, công sức và chi phí. Do đó, kiểm thử tự động đang trở thành xu hướng, giúp tăng tốc độ kiểm tra, nâng cao hiệu quả và giảm chi phí cho các doanh nghiệp phát triển phần mềm.

Xuất phát từ nhu cầu thực tiễn và mong muốn tìm hiểu sâu hơn về quy trình kiểm thử tự động, đồng thời rèn luyện kỹ năng chuyên môn phục vụ công việc trong tương lai, em đã quyết định lựa chọn đề tài "Kiểm thử tự động bằng công cụ Selenium webdriver cho hệ thống học trực tuyến Learnico" dưới sự hướng dẫn tận tình của Thạc sĩ Hoàng Quang Huy – giảng viên Khoa Công nghệ thông tin, Trường Đại học Công Nghiệp Hà Nội. Thông qua đề tài này, em mong muốn được tiếp cận công cụ kiểm thử tự động, hiểu rõ hơn về quy trình phát hiện và xử lý lỗi, từ đó góp phần nâng cao chất lượng sản phẩm phần mềm trong quá trình phát triển.

1. Mục tiêu

Đề tài: ***Kiểm thử tự động bằng công cụ Selenium webdriver cho hệ thống học trực tuyến Learnico*** đáp ứng được những mục tiêu:

* Nghiên cứu các kiến thức cơ bản về quy trình kiểm thử, vai trò, mục tiêu của kiểm thử phần mềm.
* Thiết kế và triển khai tập hợp các ca kiểm thử tự động để kiểm tra các chức năng của hệ thống Learnico.
* Nghiên cứu và áp dụng công cụ Selenium webdriver vào quy trình kiểm thử tự động cho hệ thống học trực tuyến Learnico.
* Xuất báo cáo thống kê số ca kiểm thử pass, fail trong quá trình kiểm thử đính kèm video, hình ảnh

1. Đối tượng và phạm vi nghiên cứu

Quyển đồ án tốt nghiệp trình bày kiến thức về kiểm thử phầm mềm.

Kiểm thử chức năng cho hệ thống website học trực tuyến cho các module như sau:

* Đăng Nhập
* Quản lý bài giảng
* Danh mục bài giảng
* Loại bài giảng

Kiểm thử phi chức năng : Kiểm thử tương thích (Compatibility testing )

1. Ý nghĩa khoa học và thực tiễn của đề tài

Trong thời đại công nghệ thông tin phát triển nhanh chóng như hiện nay, việc xây dựng và phát triển các hệ thống học trực tuyến trở nên rất quan trọng để đáp ứng nhu cầu học tập đa dạng và linh hoạt của mọi người. Đề tài “Kiểm thử tự động bằng công cụ Selenium WebDriver cho hệ thống học trực tuyến Learnico” mà em thực hiện nhằm mục đích nâng cao chất lượng phần mềm thông qua việc áp dụng kiểm thử tự động.

Về mặt khoa học, đề tài tập trung tìm hiểu từ kiến thức cơ bản đến nâng cao về quy trình kiểm thử phần mềm, đặc biệt là kiểm thử tự động bằng công cụ Selenium WebDriver. Qua đó, em muốn làm rõ vai trò, mục tiêu cũng như các kỹ thuật và phương pháp kiểm thử chức năng phần mềm, từ khâu lên kế hoạch, thiết kế test case đến thực hiện kiểm thử và báo cáo kết quả một cách tự động, hiệu quả.

Về mặt thực tiễn, đề tài hướng tới việc nâng cao hiệu quả kiểm thử cho hệ thống học trực tuyến Learnico bằng cách giảm bớt thời gian và công sức cho việc kiểm thử thủ công, đồng thời tăng độ chính xác và sự nhất quán trong quá trình kiểm thử. Việc triển khai kiểm thử tự động giúp phát hiện lỗi sớm hơn, đảm bảo hệ thống vận hành ổn định và mang lại trải nghiệm tốt hơn cho người dùng.

Tuy nhiên, trong quá trình nghiên cứu và thực hiện, em cũng nhận thấy một vài khó khăn và hạn chế. Em rất mong nhận được góp ý để có thể hoàn thiện và phát triển tốt hơn trong tương lai.

# CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ

## Kiểm thử phần mềm

### Khái niệm kiểm thử phần mềm

Kiểm thử phần mềm là hoạt động nhằm tìm kiếm, phát hiện các lỗi của phần mềm, đảm bảo sản phẩm phần mềm đáp ứng chính xác, đầy đủ và đúng theo yêu cầu của khách hàng đã đặt ra, cung cấp mục tiêu, cái nhìn độc lập về phần mềm, cho phép việc đánh giá và hiểu rõ hơn các rủi ro khi thực thi phần mềm.

Mục tiêu chính của kiểm thử phần mềm có thể khái quát:

* Phát hiện ra càng nhiều lỗi càng tốt trong thời gian kiểm thử xác định trước.
* Đảm bảo rằng kết quả cuối cùng đáp ứng các yêu cầu và phù hợp với đặc tả của nó.
* Tạo ra các testcase chất lượng nhằm tìm ra lỗi (nếu có) với chi phí thấp.

Tùy vào từng mô hình phát triển phần mềm mà thời gian thực hiện test là khác nhau. Giai đoạn kiểm thử nên được thực hiện càng sớm càng tốt. Theo quy trình phát triển phần mềm thì kiểm thử được thực hiện sau giai đoạn coding. Code xong thì các dev sẽ build và bàn giao cho tester thực hiện test. Thực tế thì tester tham gia sớm hơn, chỉ cần có tài liệu đặc tả/nghiệp vụ là tester thực hiện tìm hiểu nghiệp vụ dự án và thực hiện viết testcase.

### Mục đích và vai trò của kiểm thử phần mềm

#### Mục đích của kiểm thử phần mềm

* Phát hiện và xác định lỗi của sản phẩm.
* Ngăn ngừa lỗi của sản phẩm.
* Đảm bảo chất lượng sản phẩm.
* Cung cấp thông tin cho nhà sản xuất về chất lượng sản phẩm. Tầm quan trọn của kiểm thử.

#### Tầm quan trọng của kiểm thử

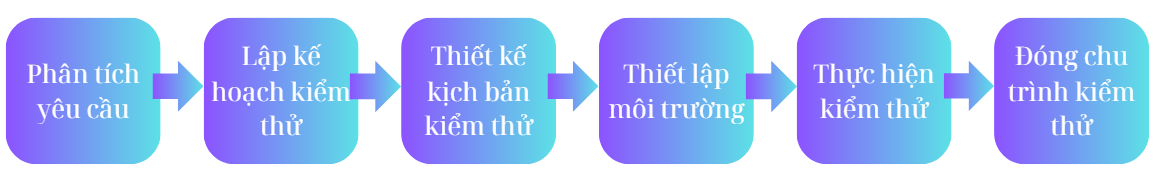
Kiểm thử phần mềm thể hiện được các “trách nhiệm” cao cả dưới đây:

* Thứ nhất, trách nhiệm hiệu quả về chi phí. Kiểm thử phần mềm giúp nhanh chóng phát hiện các lỗi của phần mềm, giúp giảm chi phí sửa chữa.
* Thứ hai, trách nhiệm bảo mật. Sản phẩm được phát hiện và sửa lỗi giúp loại bỏ các rủi ro và các vấn đề sớm, làm tăng độ tin cậy cho sản phẩm. Đối với ngành công nghệ phần mềm, vấn đề bảo mật là yếu tố cực kỳ nhạy cảm, nó liên quan trực tiếp đến việc sở hữu, sử dụng của người dùng. Vì vậy, việc kiểm thử phần mềm giúp hoàn thiện nhất sản phẩm phần mềm, tránh những lỗ hổng bảo mật đáng tiếc, tăng độ tin tưởng cho người sử dụng.
* Thứ ba, trách nhiệm về chất lượng sản phẩm. Ngoài vấn đề bảo mật như trên, sản phẩm phần mềm được kiểm tra sẽ đảm bảo được độ tin cậy, hiệu suất hoạt động cao, đảm bảo được các yêu cầu, tính năng cần thiết của nó. Sản phẩm đưa đến tay khách hàng phải là một sản phẩm đạt đủ các yêu cầu của khách hàng về hình thức, giao diện, cấu trúc, tính năng và đảm bảo không còn bất cứ lỗi nào trên sản phẩm.
* Thứ tư, trách nhiệm với niềm tin của khách hàng. Một sản phẩm càng chỉn chu, càng hoàn thiện, chất lượng càng cao sẽ tạo ra những trải nghiệm người dùng tốt nhất, từ đó càng tạo được niềm tin và uy tín với khách hàng và đối tác.

Như vậy, kiểm thử phần mềm là hoạt động không thể tách rời trong quá trình phát triển phần mềm.

### Quy trình kiểm thử phần mềm

Software testing life cycle đề cập đến một quy trình test (Testing  
process) trong đó các bước cụ thể được thực hiện theo một trình tự nhất định  
để đảm bảo mục tiêu chất lượng được đáp ứng. Trong quy trình kiểm thử  
phần mềm mỗi hoạt động được thực hiện một cách có kế hoạch và hệ thống.  
Mỗi một giai đoạn có các mục tiêu khác nhau.



Hình 1.: Quy trình kiểm thử phần mềm

* Phân tích yêu cầu kiểm thử: Đọc hiểu, nghiên cứu phân tích các yêu cầu về sản phẩm có trong các tài liệu SRS, tài liệu thiết kế…
* Lập kế hoạch kiểm thử: Xác định phạm vi kiểm thử, xác định phương pháp tiếp cận, xác định nguồn lực: con người và thiết bị, lên kế hoạch thiết kế công việc test.
* Thiết kế kịch bản kiểm thử: Review tài liệu xác định công việc cần làm, viết test case/checklist, chuẩn bị dữ liệu kiểm thử: test data, test script.
* Thiết lập môi trường: Chuẩn bị môi trường kiểm thử để có thể sẵn sàng thực hiện các test case
* Thực hiện kiểm thử: Thực hiện test theo kịch bản kiểm thử, so sánh kết quả thực tế với mong đợi và log bug lên tool quản lý lỗi, theo dõi quá trình xử lý lỗi.
* Đóng chu trình kiểm thử: Tổng kết báo cáo kết quả về việc thực thi test

### Các loại kiểm thử

#### Kiểm thử chức năng

Kiểm thử chức năng là một loại kiểm thử phần mềm nhằm xác minh rằng từng chức năng của ứng dụng phần mềm hoạt động theo các yêu cầu cụ thể và đáp ứng kỳ vọng trong các điều kiện khác nhau.

Mục tiêu của thử nghiệm chức năng là xác thực các tính năng, khả năng và tương tác của hệ thống với các thành phần khác nhau. Nó bao gồm việc thử nghiệm đầu vào và đầu ra của phần mềm, thao tác dữ liệu, tương tác của người dùng và phản ứng của hệ thống đối với các tình huống và điều kiện khác nhau. Thử nghiệm chức năng chỉ liên quan đến việc xác thực xem hệ thống có hoạt động như mong đợi hay không.

Các loại kiểm thử chức năng bao gồm:

* Kiểm thử đơn vị (Unit testing): Quá trình kiểm tra từng đơn vị nhỏ nhất của chương trình, thường là các hàm hoặc phương thức, nhằm đảm bảo chúng hoạt động đúng như mong đợi.
* Kiểm thử giao diện (Interface testing): Tập trung kiểm tra sự tương tác giữa các mô-đun hoặc giữa hệ thống và các thành phần bên ngoài, đảm bảo dữ liệu được truyền đúng định dạng và quy trình.
* Kiểm thử tích hợp (Integration testing): Được thực hiện sau kiểm thử đơn vị, nhằm đánh giá sự phối hợp giữa các mô-đun khi chúng được kết hợp lại với nhau.
* Kiểm thử hệ thống (System testing): Kiểm tra toàn bộ hệ thống như một thể thống nhất, đảm bảo các chức năng đáp ứng đầy đủ yêu cầu đã xác định trong tài liệu đặc tả.
* Kiểm thử hồi quy (Regression testing): Được thực hiện sau khi có thay đổi trong mã nguồn để đảm bảo rằng các chức năng cũ vẫn hoạt động đúng và không bị ảnh hưởng.
* Kiểm thử chấp nhận (Acceptance testing): Là giai đoạn cuối cùng trước khi triển khai sản phẩm, nhằm xác minh rằng hệ thống đáp ứng các tiêu chí chấp nhận của khách hàng hoặc người dùng cuối.

#### Kiểm thử phi chức năng

Kiểm thử phi chức năng (non-functional) được định nghĩa là một loại kiểm thử phần mềm để kiểm tra các khía cạnh phi chức năng (hiệu suất, khả năng sử dụng, độ tin cậy, v.v.) của ứng dụng phần mềm. Đảm bảo rằng hệ thống có thể xử lý các nhu cầu thực tế và cung cấp trải nghiệm người dùng liền mạch, chất lượng cao.

Mục tiêu của kiểm thử phi chức năng là đánh giá các đặc tính chất lượng của hệ thống phần mềm như hiệu suất, khả năng sử dụng, bảo mật, tính tương thích và khả năng bảo trì. Quá trình này giúp đảm bảo phần mềm hoạt động ổn định trong các điều kiện thực tế, giảm thiểu rủi ro và chi phí liên quan đến các yếu tố phi chức năng. Đồng thời, kiểm thử phi chức năng còn hỗ trợ tối ưu hóa quá trình cài đặt, vận hành và quản lý hệ thống, cung cấp số liệu phục vụ nghiên cứu, cải tiến sản phẩm và nâng cao hiểu biết về hành vi cũng như công nghệ của phần mềm.

Các loại kiểm thử phi chức năng :

* Kiểm thử độ ổn định (Stability testing): Nhằm đánh giá khả năng hệ thống duy trì hoạt động liên tục trong một khoảng thời gian dài mà không gặp sự cố của hệ thống
* Kiểm thử khả năng chịu tải (Load testing): Được thực hiện để đo lường hiệu suất của hệ thống dưới điều kiện tải bình thường hoặc gần ngưỡng tối đa.
* Kiểm thử áp lực (Stress testing): Tập trung kiểm tra hệ thống trong điều kiện vượt quá giới hạn tải nhằm đánh giá khả năng chịu đựng và phục hồi khi gặp sự cố.
* Kiểm thử tính khả dụng (Usability testing): Đánh giá mức độ thân thiện và dễ sử dụng của hệ thống từ góc nhìn người dùng cuối.
* Kiểm thử bảo trì (Maintainability testing): Xác định mức độ dễ dàng trong việc sửa lỗi, nâng cấp và cải tiến hệ thống sau khi triển khai.
* Kiểm thử độ tin cậy (Reliability testing): Nhằm đảm bảo hệ thống hoạt động chính xác và ổn định trong thời gian dài mà không phát sinh lỗi.
* Kiểm thử tính tương thích (Portability testing): Kiểm tra khả năng hoạt động của phần mềm trên các nền tảng, thiết bị hoặc môi trường khác nhau.

#### Kiểm thử cấu trúc

Kiểm thử cấu trúc (Structural testing), còn gọi là kiểm thử hộp trắng (White-box testing), là phương pháp kiểm thử phần mềm dựa trên cấu trúc bên trong của mã nguồn. Thay vì chỉ tập trung vào đầu vào và đầu ra như kiểm thử hộp đen, kiểm thử cấu trúc yêu cầu người kiểm thử hiểu rõ logic xử lý bên trong chương trình để thiết kế các ca kiểm thử phù hợp.

Mục tiêu của kiểm thử cấu trúc là đảm bảo rằng các thành phần bên trong của phần mềm, bao gồm các câu lệnh, nhánh điều kiện, vòng lặp và các cấu trúc điều khiển, được kiểm tra đầy đủ và toàn diện. Thông qua việc phân tích trực tiếp mã nguồn, kiểm thử cấu trúc giúp phát hiện sớm các lỗi logic, sai sót trong thuật toán hoặc các đoạn mã chưa được thực thi. Ngoài ra, phương pháp này còn góp phần nâng cao độ bao phủ mã (code coverage), từ đó tăng độ tin cậy và chất lượng tổng thể của phần mềm. Kiểm thử cấu trúc không chỉ hỗ trợ việc xác minh tính đúng đắn của các thuật toán được cài đặt mà còn giúp cải thiện hiệu quả phát triển và bảo trì hệ thống trong dài hạn.

Các loại kiểm thử cấu trúc phổ biến:

* Kiểm thử mức câu lệnh (Statement testing): Kiểm tra xem tất cả các câu lệnh trong chương trình có được thực thi ít nhất một lần hay không.
* Kiểm thử mức nhánh (Branch testing): Đảm bảo rằng tất cả các nhánh trong cấu trúc điều kiện (như if, switch) đều được kiểm tra.
* Kiểm thử mức điều kiện (Condition testing): Tập trung kiểm tra từng điều kiện logic trong các biểu thức điều kiện, đảm bảo mỗi điều kiện có thể được đánh giá cả giá trị đúng và sai.
* Kiểm thử đường dẫn (Path testing): Kiểm tra tất cả các đường dẫn logic khả thi qua chương trình để đảm bảo toàn bộ luồng xử lý được đánh giá.

#### Kiểm thử xác nhận

Kiểm thử xác nhận (Confirmation testing) là quá trình kiểm tra lại phần mềm sau khi một lỗi đã được phát hiện và sửa chữa. Mục tiêu chính của kiểm thử xác nhận là xác minh rằng lỗi đã được sửa thành công và phần mềm hoạt động đúng như mong đợi sau khi sửa lỗi.

Khi thực hiện kiểm thử xác nhận, điều quan trọng nhất là phải đảm bảo rằng các trường hợp kiểm thử được thực hiện chính xác như lần kiểm thử đầu tiên, sử dụng cùng một đầu vào, dữ liệu và môi trường kiểm thử. Điều này giúp đảm bảo rằng các lỗi đã được sửa chữa và hệ thống hoạt động đúng như mong đợi. Tuy nhiên, tester cần nhận thức rằng trong quá trình kiểm thử sau khi vá lỗi, khả năng phát sinh lỗi mới là điều hoàn toàn có thể xảy ra. Vì vậy, mặc dù kiểm thử xác nhận giúp xác minh sự khắc phục của lỗi hiện tại, việc chỉ thực hiện kiểm thử xác nhận trên phiên bản phần mềm hiện tại vẫn chưa đủ. Để phát hiện những vấn đề ngoài ý muốn do thay đổi hoặc sửa lỗi, việc thực hiện kiểm thử hồi quy là cần thiết để đảm bảo rằng các thay đổi không ảnh hưởng tiêu cực đến các chức năng đã hoạt động ổn định trước đó.

#### Kiểm thử hồi quy

Kiểm thử hồi quy (Regression testing) là quá trình kiểm tra phần mềm sau khi có sự thay đổi (như sửa lỗi hoặc thêm tính năng mới) nhằm đảm bảo rằng những thay đổi này không gây ảnh hưởng xấu đến các chức năng đã hoạt động đúng trước đó. Kiểm thử hồi quy cũng được thực hiện khi có sự thay đổi trong môi trường phần mềm.

Mục tiêu của kiểm thử hồi quy là đảm bảo rằng các thay đổi, dù là sửa lỗi hay cập nhật tính năng, không làm phát sinh lỗi mới hoặc ảnh hưởng đến các phần khác của hệ thống. Kiểm thử hồi quy còn giúp xác nhận rằng hệ thống vẫn đáp ứng các yêu cầu và duy trì tính ổn định sau khi có sự thay đổi.

### Các cấp độ kiểm thử phần mềm

#### Kiểm thử đơn vị

Kiểm thử đơn vị (Unit testing) là một trong những bước quan trọng nhất trong quy trình kiểm thử phần mềm, thuộc nhóm kiểm thử mức thấp.

Mục tiêu chính của kiểm thử đơn vị là xác minh tính đúng đắn của từng đơn vị nhỏ nhất trong chương trình – thường là các hàm, phương thức, hoặc module – một cách độc lập trước khi chúng được tích hợp vào hệ thống lớn hơn. Đây là giai đoạn kiểm thử đầu tiên được thực hiện ngay sau khi lập trình viên hoàn thành mã nguồn.

Việc thực hiện kiểm thử đơn vị giúp phát hiện sớm các lỗi logic, lỗi cú pháp, hoặc các hành vi không mong muốn ngay trong nội bộ chức năng của chương trình. Bằng cách cô lập từng thành phần và kiểm tra riêng biệt, kiểm thử đơn vị giúp đảm bảo rằng mỗi phần tử hoạt động đúng như thiết kế ban đầu. Điều này góp phần giảm đáng kể chi phí và thời gian sửa lỗi trong các giai đoạn phát triển sau này, vì lỗi được phát hiện càng sớm thì chi phí khắc phục càng thấp.

Kiểm thử đơn vị thường được thực hiện bởi chính lập trình viên và có thể được tự động hóa bằng các framework kiểm thử như JUnit (cho Java), NUnit (cho .NET), hoặc unittest/pytest (cho Python). Các công cụ này không chỉ hỗ trợ việc viết và thực thi các test case mà còn giúp theo dõi tỷ lệ bao phủ mã (code coverage) – một chỉ số quan trọng trong đánh giá chất lượng kiểm thử.

#### Kiểm thử tích hợp

Kiểm thử tích hợp (Integration testing) là giai đoạn kiểm thử trung gian, được thực hiện sau kiểm thử đơn vị và trước kiểm thử hệ thống.

Mục tiêu của kiểm thử tích hợp là kiểm tra sự tương tác giữa các module hoặc thành phần riêng lẻ trong hệ thống phần mềm, nhằm đảm bảo rằng chúng hoạt động chính xác khi được kết hợp với nhau. Dù từng đơn vị riêng lẻ có thể đã hoạt động đúng trong kiểm thử đơn vị, nhưng khi tích hợp lại, lỗi vẫn có thể phát sinh do các vấn đề về giao tiếp, chia sẻ dữ liệu, hoặc logic tương tác giữa các module.

Quá trình kiểm thử tích hợp giúp phát hiện sớm các lỗi liên quan đến giao diện giữa các thành phần, chẳng hạn như lỗi truyền dữ liệu, định dạng dữ liệu không khớp, hoặc các lỗi logic xảy ra khi các module thực hiện tuần tự các chức năng phụ thuộc lẫn nhau. Kiểm thử tích hợp đặc biệt quan trọng trong các hệ thống có kiến trúc phân lớp, hệ thống phân tán, hoặc các hệ thống có thành phần đến từ các nhà phát triển khác nhau.

Các phương pháp kiểm thử tích hợp :

* Kiểm thử Big Bang: Tất cả các module được tích hợp cùng một lúc và sau đó tiến hành kiểm thử toàn bộ hệ thống. Phương pháp này đơn giản nhưng khó xác định nguyên nhân lỗi khi phát sinh.
* Kiểm thử Top-down: Tiến hành kiểm thử từ các module cấp cao xuống cấp thấp, sử dụng các module giả thay thế cho các thành phần chưa phát triển. Giúp phát hiện lỗi sớm ở các tầng điều khiển chính.
* Kiểm thử Bottom-up: Bắt đầu kiểm thử từ các module cấp thấp (gần với dữ liệu) rồi dần dần tích hợp lên các module cấp cao, dùng module giả (driver) để kiểm tra phần chưa sẵn sàng. Tốt cho kiểm tra các chức năng cơ sở.
* Kiểm thử Sandwich: Kết hợp giữa top-down và bottom-up, tiến hành kiểm thử đồng thời cả từ trên xuống và dưới lên. Phù hợp với các hệ thống lớn có cấu trúc phân lớp rõ ràng.

#### Kiểm thử hệ thống

Kiểm thử hệ thống (System testing) là thực hiện kiểm thử một hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đúng yêu cầu của phần mềm. Đây là cấp độ kiểm thử cao hơn kiểm thử tích hợp, được thực hiện sau khi tất cả các thành phần đã được tích hợp thành một hệ thống hoàn chỉnh.

Mục tiêu của kiểm thử hệ thống là đảm bảo rằng phần mềm hoạt động chính xác trong một môi trường giống như thực tế, với đầy đủ các thành phần, dữ liệu và điều kiện hoạt động. Quá trình này không chỉ kiểm tra các chức năng chính của hệ thống mà còn đánh giá các yếu tố phi chức năng như hiệu suất, bảo mật, tính khả dụng, khả năng tương thích và được thực hiện bởi tester.

#### Kiểm thử chấp nhận

Kiểm thử chấp nhận (Acceptance testing) là giai đoạn cuối cùng trong quy trình kiểm thử phần mềm, nhằm xác nhận rằng sản phẩm đã sẵn sàng để bàn giao cho khách hàng hoặc triển khai trong môi trường thực tế. Đây là bước kiểm thử mang tính quyết định, thường được thực hiện bởi khách hàng, người dùng cuối hoặc nhóm kiểm thử đại diện cho họ, với mục tiêu đánh giá xem hệ thống có đáp ứng đầy đủ các yêu cầu và kỳ vọng đã đặt ra ban đầu hay không.

Mục tiêu của kiểm thử chấp nhận không chỉ là phát hiện lỗi mà còn là xác minh rằng phần mềm đã đạt chất lượng mong muốn, hoạt động đúng với mục đích sử dụng và có thể vận hành ổn định trong môi trường thực tế. Các bài kiểm thử ở giai đoạn này được xây dựng dựa trên các tiêu chí chấp nhận đã thống nhất từ trước trong tài liệu yêu cầu hoặc hợp đồng.

Kiểm thử chấp nhận được chia thành 2 mức khác nhau:

* Kiểm thử alpha: được thực hiện tại nơi phát triển phần mềm bởi những người trong tổ chức nhưng không tham gia phát triển phần mềm.
* Kiểm thử beta: được thực hiện tại bởi khách hàng/ người dùng cuối tại địa điểm của người dùng cuối.

### Kỹ thuật kiểm thử hộp đen

Kiểm thử hộp đen (Black-box testing) là một kỹ thuật kiểm thử phần mềm tập trung vào việc đánh giá chức năng của hệ thống mà không cần quan tâm đến cấu trúc bên trong hoặc mã nguồn. Người kiểm thử chỉ cần biết các yêu cầu đầu vào và kỳ vọng đầu ra để xác định phần mềm có hoạt động đúng theo yêu cầu hay không.

Mục tiêu chính của kiểm thử hộp đen là đảm bảo rằng phần mềm thực hiện chính xác các chức năng được xác định trong tài liệu đặc tả, đồng thời phát hiện các lỗi liên quan đến xử lý sai logic, thiếu chức năng, hoặc các vấn đề về giao diện, hiệu năng và khả năng sử dụng.

#### Phân Vùng tương đương

Phân vùng tương đương là kỹ thuật chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy có thể test một giá trị đại diện trong vùng tương đương.

VD: User ID – Six number digits

Nhập 6 ký tự số

Nhập < 6 ký tự số

Nhập > 6 ký tự số

#### Phân tích giá trị biên

Đây là một trong những kỹ thuật kiểm thử phần mềm, trong đó các test case được thiết kế bao gồm các giá trị tại các biên. Test tất cả các giá trị ở vùng biên của dữ liệu vào và dữ liệu ra. Các tester sẽ tập trung vào các giá trị biên chứ không test toàn bộ dữ liệu. Do đó, thay vì phải kiểm thử toàn bộ dữ liệu vào và ra, ta có thể test từ 4 - 6 case mà vẫn đảm bảo hệ thống hoạt động tốt.

Giá trị biên là các vị trí ở giữa, trên và dưới các biên của lớp tương đương. Khi áp dụng kỹ thuật phân tích giá trị biên, người kiểm thử sẽ chọn các giá trị:

* Giá trị nhỏ nhất
* Giá trị ngay dưới giá trị nhỏ nhất
* Giá trị bình thường
* Giá trị lớn nhất
* Giá trị ngay trên giá trị lớn nhất

VD. Số điểm ứng viên cần đạt là từ [24-40] thì mới qua bài test. Áp dụng kỹ thuật phân tích giá trị biên sẽ có các giá trị:

* Giá trị nhỏ nhất: 24
* Giá trị ngay dưới giá trị nhỏ nhất: 23
* Giá trị bình thường: 32
* Giá trị ngay trên giá trị lớn nhất: 41
* Giá trị lớn nhất: 40

#### Bảng quyết định

Đối với các trường dữ liệu đơn như textbox, các tester thường sử dụng các phương pháp phân vùng tương đương hay phân tích giá trị biên. Đối với kiểm thử hành vi của hệ thống với nhiều trường dữ liệu, Bảng quyết định (Decision table) sẽ giúp chúng ta phân loại và định hình được kịch bản kiểm thử một cách chính xác và rõ ràng hơn.

Bảng quyết định là một kỹ thuật tốt để áp dụng cho những trường hợp cần nhiều sự kết hợp. Kỹ thuật này hỗ trợ việc lựa chọn Test Case tối thiểu một cách có hệ thống kỹ thuật với độ bao phủ tối đa.

VD: Tại màn hình Login có 2 field: “User Name” và “Password”. Chỉ login thành công nếu nhập đúng cả 2 field “User Name” và “Password”. Các trường hợp còn lại được hệ thống hiển thị thông báo “Username hoặc password không chính xác. Vui lòng nhập lại”

Bảng 1.: Bảng quyết định màn hình Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | Rule 1 | Rule 2 | Rule 3 | Rule 4 |
| Username | T | T | F | F |
| Password | T | F | T | F |
| Output | Sign in successfully | Sign in failed | Sign in failed | Sign in failed |

#### Đoán lỗi

Đoán lỗi là kỹ thuật mô tả hành động phỏng đoán lỗi thường gặp của hệ thống dựa trên trực giác và kinh nghiệm của các tester. Người kiểm thử sẽ liệt kê các loại lỗi có thể xảy ra và cho vào Test Case để kiểm tra xác minh vấn đề.

Phương pháp này đặc biệt dựa vào kinh nghiệm và kiến thức của tester. Kỹ thuật đoán lỗi không tuân theo bất kỳ quy tắc cụ thể nào, Test Case có thể được thiết kế tùy thuộc vào nhiều yếu tố như: Đặc trưng hoạt động của phần mềm, lỗi đã xuất hiện ở các dự án tương tự khác…

Các yếu tố mà người kiểm thử hay sử dụng để đoán lỗi:

* Bài học từ các lần release sản phẩm
* Trực giác của tester
* Từ những lỗi đã từng gặp
* Từ các checklist, reviews, common testcase
* Các kết quả kiểm thử
* Các quy định chung về kiểm thử
* Kinh nghiệm các lần UAT

### Thiết kế ca kiểm thử và tạo Bug report

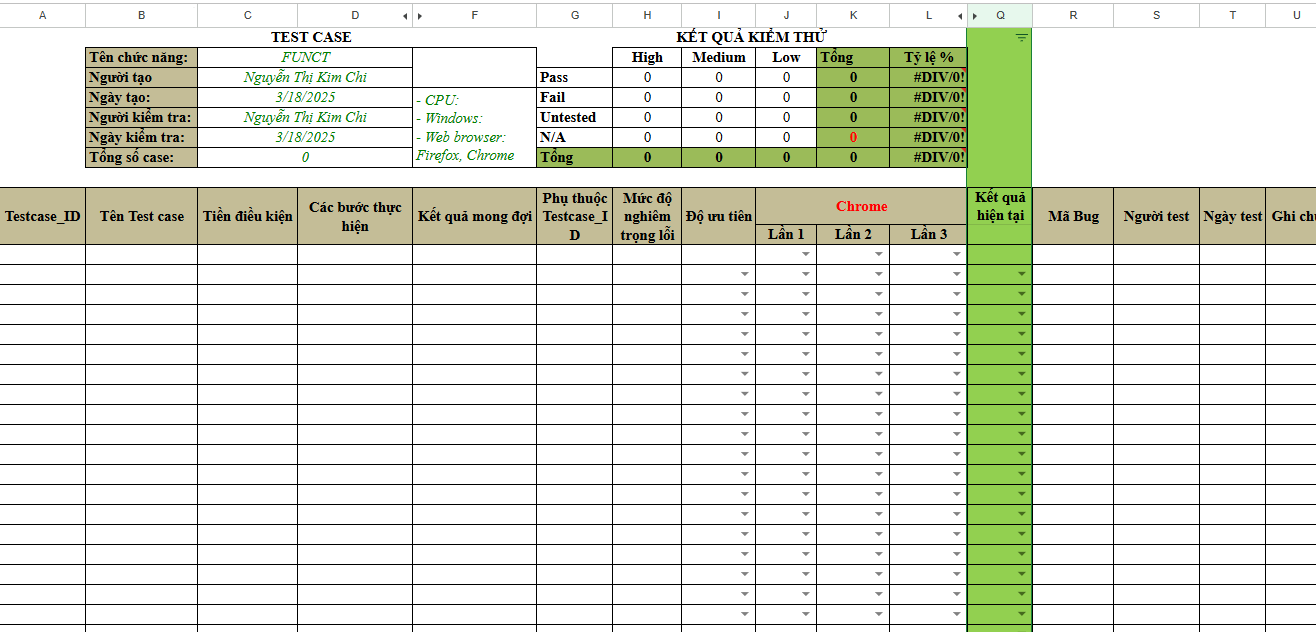
#### Thiết kế ca kiểm thử

Quá trình phát triển ca kiểm thử có thể giúp tìm ra lỗi trong các yêu cầu hoặc thiết kế của ứng dụng, vì nó đòi hỏi phải tư duy hoàn toàn thông qua các hoạt động của ứng dụng. Vì lý do này, việc chuẩn bị ca kiểm thử sớm nhất có thể trong quy trình phát triển phần mềm là rất hữu ích. Các trường hợp kiểm thử phải bao phủ được toàn bộ luồng xử lý chức năng mô tả trong tài liệu phân tích và thiết kế, các yêu cầu về bảo mật an toàn thông tin, yêu cầu hiệu năng của hệ thống.

Các ca kiểm thử thường có cấu trúc:

* Test Case ID: Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.
* Tên test case
* Tiền điều kiện: Nếu có
* Các bước thực hiện: Các bước thực hiện 1 ca kiểm thử.
* Kết quả mong đợi: Kết quả mong đợi từ các bước thực hiện trên.
* Mức độ nghiêm trọng : Cao, trung bình, thấp
* Kết quả hiện tại: Kết quả thực tế khi chạy chương trình, pass, fail
* Mã bug: Mã bug khi được lưu trên công cụ quản lý bug
* Người test
* Ngày test
* Ghi chú: Cột này dùng để ghi chú những thông tin liên quan khi thực hiện ca kiểm thử.

Ngoài ra tùy theo từng doanh nghiệp mẫu testcase sẽ có sự thay đổi khác nhau.



Hình 1.2: Minh họa testcase

Các bước xác định Ca kiểm thử:

Bước 1: Xác định mục đích kiểm thử: cần hiểu rõ đặc tả yêu cầu của khách hàng.

Bước 2: Xác định chức năng cần kiểm tra: cần phải biết làm thế nào phần mềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau. Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button, v.v.

Bước 3: Xác định các yêu cầu phi chức năng: yêu cầu phần cứng, hệ điều hành, các khía cạnh an ninh.

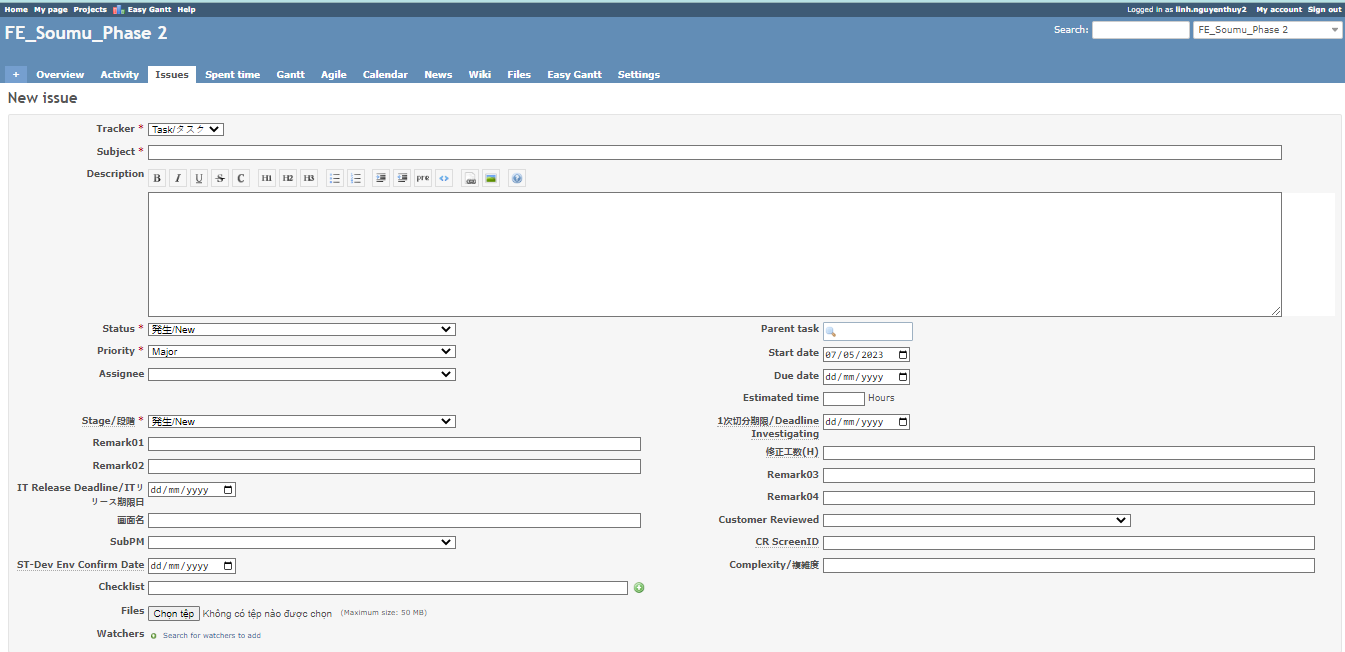
Bước 4: Xác định biểu mẫu cho Ca kiểm thử: bao gồm giao diện UI, chức năng, khả năng tương thích và hiệu suất.

Bước 5: Xác định tính ảnh hưởng giữa các nguyên tắc mô-đun: mỗi một ca kiểm thử nên được thiết kế để có thể che phủ được sự ảnh hưởng của các mô-đun với nhau ở mức độ cao nhất.

#### Tạo bug report

Cấu trúc của một Bug report:

* Project: tên của dự án phần mềm.
* Reported by: kiểm thử viên tạo ra Bug report.
* Bug Name, Bug ID và Date: tên của bug, ID và ngày tạo report.
* Assigned to: cá nhân hoặc tổ chức phát triển phần mềm đó.
* Status: Trạng thái thực hiện của report.
* Summary/Description: mô tả ngắn gọn về bug.
* Environments (OS/Browser): môi trường chạy thử phần mềm.
* Step to reproduce: mô tả lại các bước thực hiện gây ra bug.
* Actual results: kết quả thực tế.
* Expected results: kết quả mong đợi.
* Severity: mức độ nghiêm trọng của bug.
* Priority: mức độ ưu tiên của bug.
* Attachment: đính kèm với bug (tệp, đường dẫn URL, ảnh, v.v.)



Hình 1.3: Một bug report được log trên Redmine

Một số yêu cầu khi tạo Bug report:

* Tiêu đề phải rõ ràng: Khi lập trình viên đọc bug, thứ đầu tiên đập vào mắt là Bug name. Nó cũng là phần được đọc nhiều nhất, không phải là description. Một bug name tốt phải ngắn gọn và diễn tả được bug một cách tối giản.
* Phải mô phỏng lại được quá trình gây ra bug: nếu không mô phỏng lại được, bug sẽ không thể được khắc phục.
* Không viết luận trong description: viết ngắn gọn và vào trọng tâm. Cố gắng viết ít chữ nhất có thể nhưng vẫn đầy đủ ý.

## Khái quát về kiểm thử tự động

Kiểm thử phần mềm tốn nhiều chi phí, nhân lực và thời gian. Trong một  
số dự án, chi phí kiểm thử phần mềm chiếm 40% tổng giá trị của dự án. Do  
đó một trong các mục tiêu của kiểm thử là tự động hóa nhiều, nhờ đó mà  
giảm thiểu chi phí, giảm lỗi, đặc biệt giúp việc kiểm thử dễ dàng và nhanh  
chóng hơn.

### Khái niệm kiểm thử tự động

Kiểm thử tự động (Automated testing) là một phương pháp kiểm thử phần mềm trong đó các trường hợp kiểm thử được thực thi tự động thông qua các công cụ hoặc kịch bản kiểm thử, thay vì thực hiện thủ công bởi con người.

Mục đích chính của kiểm thử tự động là cải thiện hiệu suất và độ chính xác của quá trình kiểm thử. Thông qua việc sử dụng các công cụ kiểm thử như Selenium, Appium, JUnit hay TestNG, các nhà phát triển và kiểm thử có thể thực thi hàng loạt ca kiểm thử một cách nhanh chóng, lặp đi lặp lại nhiều lần mà vẫn đảm bảo tính nhất quán. Điều này đặc biệt hữu ích trong kiểm thử hồi quy, nơi cần đảm bảo rằng các chức năng hiện tại không bị ảnh hưởng sau khi phần mềm được cập nhật hoặc thay đổi.

### Quy trình kiểm thử tự động

Một quy trình kiểm thử tự động thường trải qua những bước cơ bản như sau:

* Lựa chọn công cụ kiểm thử tự động
* Xác định phạm vi
* Lập kế hoạch
* Thiết kế kịch bản kiểm thử tự động
* Phát triển kịch bản kiểm thử tự động
* Thực thi kiểm thử (Execution)
* Cập nhật và Bảo trì

#### Lựa chọn công cụ kiếm thư tự động

Mục tiêu của việc lựa chọn công cụ kiểm thử là chọn ra công cụ kiểm thử tự động phù hợp nhất với loại ứng dụng (web, mobile, API...) và môi trường triển khai (trình duyệt, thiết bị, hệ điều hành...).

Việc lựa chọn công cụ phù hợp sẽ giúp tăng hiệu quả kiểm thử, tận dụng được tối đa tính năng của công cụ, đồng thời giảm thiểu rủi ro kỹ thuật trong quá trình phát triển và thực thi kịch bản kiểm thử. Công cụ có thể là mã nguồn mở (như Selenium, Appium) hoặc thương mại (như Katalon, TestComplete), tùy theo ngân sách và trình độ kỹ thuật của nhóm phát triển.

#### Xác định phạm vi

Không phải tất cả các ca kiểm thử đều phù hợp để tự động hóa. Mục đích của việc xác định phạm vi là phân tích, đánh giá và xác định rõ các phần chức năng nào trong hệ thống có thể và nên được tự động hóa. Việc này nhằm tối ưu tài nguyên, tập trung kiểm thử vào những luồng nghiệp vụ quan trọng, lặp đi lặp lại hoặc có khả năng tái sử dụng cao. Đồng thời, bước này cũng giúp tránh lãng phí thời gian khi cố gắng tự động hóa những phần không ổn định hoặc có tần suất thay đổi cao.

#### Lập kế hoạch

Lập kế hoạch nhằm xây dựng kế hoạch tổng thể cho quá trình kiểm thử tự động bao gồm xác định phạm vi, nguồn lực, thời gian, ngân sách và các rủi ro có thể gặp phải.

Lập kế hoạch giúp nhóm kiểm thử định hướng rõ ràng về lộ trình triển khai, phân bổ nhân lực và chuẩn bị tốt cho các giai đoạn tiếp theo. Một kế hoạch chi tiết và thực tế sẽ đảm bảo kiểm thử được thực hiện hiệu quả, có thể đo lường và kiểm soát tiến độ.

#### Thiết kế kịch bản kiểm thử tự động

Mục tiêu của việc thiết kế kịch bản kiểm thử là xác định và xây dựng cấu trúc logic của các kịch bản kiểm thử dựa trên yêu cầu nghiệp vụ.

Các kịch bản cần được thiết kế rõ ràng, đầy đủ và có thể kiểm soát được nhằm đảm bảo khi phát triển thành mã nguồn, chúng sẽ thực hiện đúng yêu cầu. Việc thiết kế tốt ngay từ đầu giúp giảm thiểu lỗi trong quá trình phát triển và bảo trì sau này.

#### Phát triển kịch bản kiểm thử tự động

Các kịch bản được mã hóa thành các đoạn mã kiểm thử (test scripts) có thể thực thi bằng công cụ đã chọn. Mục đích là biến các kịch bản lý thuyết thành kiểm thử có thể chạy tự động trên hệ thống thực tế.

Các kịch bản phải đảm bảo ổn định, chính xác, dễ bảo trì, đồng thời có khả năng ghi nhận kết quả và log khi kiểm thử. Đây là giai đoạn đòi hỏi kỹ thuật cao và thường chiếm nhiều thời gian trong quy trình.

#### Thực thi kiểm thử

Mục tiêu của việc thực thi kiểm thử là chạy các kịch bản kiểm thử đã phát triển trên hệ thống thực tế để xác định xem phần mềm có hoạt động đúng với các yêu cầu đã đặt ra hay không.

Kết quả thực thi sẽ được ghi lại để phân tích, đánh giá và làm căn cứ cho báo cáo chất lượng phần mềm. Đây là bước then chốt giúp phát hiện lỗi và xác định tính ổn định của hệ thống trước khi phát hành.

#### Cập nhật và bảo trì

Sau mỗi lần cập nhật hoặc nâng cấp phần mềm, các kịch bản kiểm thử có thể cần được sửa đổi để tương thích với thay đổi mới. Mục tiêu của bước này là đảm bảo bộ kiểm thử tự động luôn phản ánh đúng trạng thái hiện tại của ứng dụng, giúp phát hiện lỗi phát sinh khi có thay đổi. Bảo trì tốt sẽ giúp hệ thống kiểm thử tự động duy trì hiệu quả lâu dài và đáng tin cậy qua nhiều chu kỳ phát triển phần mềm.

### So sánh kiểm thử thủ công và kiểm thử tự động

Trong lĩnh vực kiểm thử phần mềm, kiểm thử tự động (Automation testing) và kiểm thử thủ công (Manual testing) là hai phương pháp phổ biến, mỗi phương pháp đều có những ưu điểm và hạn chế riêng, tùy thuộc vào mục tiêu, phạm vi và yêu cầu cụ thể của dự án phần mềm.

Bảng 1.2: So sánh kiểm thử thủ công và kiểm thử tự động

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Kiểm thử thủ công** | **Kiểm thử tự động** |
| Định nghĩa | Test case được thực hiện thủ công bởi tester | Tester phải viết test script và lựa chọn công cụ để tự động hóa việc test |
| Thời gian xử lý | Cần nhiều thời gian và nhân lực | Thời gian kiểm thử nhanh hơn so với kiểm thử thủ công |
| Exploratory Testing/ Kiểm thử khám phá | Exploratory Testing/ Kiểm thử khám phá được thực hiện | Không cho phép kiểm thử khám phá |
| Thay đổi UI | Sự thay đổi nhỏ như ID, Class hoặc 1 button nhưng không ảnh hưởng đến thực thi test | Chỉ 1 vài thay đổi nhỏ trong UI, người dùng phải update script để đảm bảo có kết quả như mong đợi |
| Độ tin cậy | Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi do con người | Do được thực thi bằng tool và scripts nên kết quả đáng tin cậy hơn |
| Đầu tư | Cần nhiều nguồn lực | Bắt buộc phải đầu tư tool để test và những kỹ sư auto |
| Báo cáo | Thường lưu lại kết quả ở Excel, Word... | Tất cả stakeholders có thể đăng nhập vào hệ thống auto và kiểm tra lại kết quả test |
| Sự quan sát của con người | Cần có sự quan sát của con người để giúp cho hệ thống thân thiện với người dùng | Không có sự quan sát của con người |
| Kiểm thử hiệu năng/Performance Testing | Không thực hiện được Kiểm thử hiệu năng | Kiểm thử hiệu năng phải được thực hiện bởi 1 tool phù hợp |
| Kiến thức lập trình | Không cần có khả năng code | Phải có kiến thức về lập trình để tạo ra các test script |
| Cách tiếp cận tốt | Hữu ích khi chúng ta chạy lại bộ testcase 1 hoặc 2 lần | Hữu ích khi chạy lại script nhiều lần |
| Khi nào sử dụng | Phù hợp cho kiểm thử khám phá, khả năng sử dụng và kiểm thử dựa vào thực tế | Thích hợp cho kiểm thử hồi quy, hiệu năng hoặc các trường hợp có khả năng lặp lại nhiều lần |

### Một số công cụ kiểm thử tự động

Công cụ kiểm thử giao diện người dùng:

* Selenium
* Katalon studio
* Appium,…

Công cụ kiểm thử API/Web services:

* Postman
* SoapUI,…

Công cụ kiểm thử hiệu năng :

* Jmeter
* LoadRunner,…

Công cụ kiểm thử tương thích:

* Lambdatest
* BrowserStack,…

# CÔNG CỤ KIỂM THỬ TỰ ĐỘNG SELENIUM WEBDRIVER

## Công cụ kiểm thử tự động selenium

### Khái quát về selenium

Selenium là một bộ công cụ kiểm thử tự động dành cho các ứng dụng web, hỗ trợ hoạt động trên nhiều trình duyệt và nền tảng khác nhau như Windows, Mac, Linus… Với Selenium có thể viết các testscript bằng các ngôn ngữ lập trình khác nhau như Java, PHP, C#, Ruby hay Python …

Selenium là công cụ được sử dụng để tự động hóa các thao tác tương tác với trình duyệt web, mô phỏng hành vi của người dùng thực khi sử dụng một ứng dụng web. Cụ thể, Selenium cho phép lập trình các thao tác như mở trình duyệt, truy cập liên kết, nhập liệu vào biểu mẫu, thu thập thông tin từ trang web, thực hiện thao tác tải lên hoặc tải xuống dữ liệu, v.v.

Với Selenium, người dùng có thể xây dựng các kịch bản kiểm thử tự động một cách linh hoạt và hiệu quả, đồng thời tận dụng khả năng tùy biến cao để mở rộng chức năng theo nhu cầu. Ngoài mục tiêu chính là phục vụ kiểm thử phần mềm, Selenium còn có thể được ứng dụng vào việc tự động hóa các tác vụ thủ công, lặp đi lặp lại trong công việc hàng ngày, góp phần tăng năng suất và giảm thiểu sai sót do con người.

### Đặc điểm của selenium

Mã nguồn mở: Đây là điểm mạnh nhất của Selenium khi so sánh với các test tool khác. Vì là mã nguồn mở nên chúng ta có thể sử dụng mà không phải lo lắng về phí bản quyền hay thời hạn sử dụng.

Cộng đồng hỗ trợ: Vì là mã nguồn mở nên Selenium có một cộng đồng hỗ trợ khá mạnh mẽ. Bên cạnh đó, Google là nơi phát triển Selenium nên chúng ta hoàn toàn có thể yên tâm về sự hỗ trợ miễn phí khi có vấn đề về Selenium. Tuy nhiên, đây cũng là một điểm yếu của Selenium. Vì công cụ này hoàn toàn miễn

phí, cộng đồng lại đông nên một vấn đề có thể nhiều giải pháp, và có thể một số giải pháp là không hữu ích. Mặt khác, chúng ta không thể hối thúc hay ra deadline cho sự hỗ trợ.

Selenium hỗ trợ nhiều ngôn ngữ lập trình.

Selenium hỗ trợ chạy trên nhiều hệ điều hành khác nhau với mức độ chỉnh sửa script hầu như không có. Thực sự điều này phụ thuộc phần lớn vào khả năng viết script của người dùng.

Chạy test case ở background. Khi chúng ta thực thi một test script, chúng ta hoàn toàn có thể làm việc khác trên cùng một máy tính. Điều này hỗ trợ chúng ta không cần tốn quá nhiều tài nguyên máy móc khi chạy test script.

Không hỗ trợ Win app. Selenium thực sự chỉ hỗ trợ chúng ta tương tác với Browser mà không hỗ trợ chúng ta làm việc với các Win app, kể cả Win dialog như Download/Upload. Vậy nên, để xử lý các trường hợp cần tương tác với hệ thống hay một app thứ ba, chúng ta cần một hay nhiều thư viện khác nhau như AutoIt hay Coded UI.

Là một công cụ hỗ trợ kiểm tra tính năng nên Selenium không có khả năng giả lập nhiều người dùng ảo cùng một lúc. Công việc của nó là chạy kiểm thử tự động dựa trên một kịch bản đã được thiết kế từ trước. Qua đó chúng ta có thể chắc chắn rằng đối tượng kiểm thử có hoạt động đúng như mong đợi hay không.

### Các thành phần của selenium

Selenium được chia thành ba thành phần chính: Selenium IDE, Selenium RC, Selenium Webdriver, và Selenium Grid. Mỗi thành phần có vai trò và phạm vi ứng dụng riêng, hỗ trợ linh hoạt cho quá trình kiểm thử tự động ứng dụng web ở nhiều cấp độ khác nhau.

#### Selenium IDE (Integrated Development Environment)

Selenium IDE là một công cụ phát triển tích hợp dưới dạng tiện ích mở

rộng cho trình duyệt web như Firefox và Chrome. Đây là công cụ thân thiện với người dùng, cho phép ghi lại và phát lại các hành động của người dùng trên trình duyệt. Khi người dùng thao tác trên trang web, Selenium IDE sẽ ghi nhận các bước và tự động tạo ra mã kiểm thử tương ứng, giúp người dùng dễ dàng tạo test case mà không cần biết lập trình.

**Ưu điểm:**

* Giao diện trực quan, dễ sử dụng.
* Phù hợp với người mới bắt đầu hoặc dùng để tạo kịch bản kiểm thử nhanh.
* Cho phép xuất kịch bản sang nhiều ngôn ngữ như Java, Python, C#, v.v.

**Nhược điểm:**

* Khả năng tùy biến hạn chế.
* Không phù hợp với kiểm thử phức tạp hoặc kiểm thử hồi quy lớn.
* Không hỗ trợ tốt cho kiểm thử trên nhiều trình duyệt hoặc môi trường khác nhau.

#### Selenium Webdriver

Selenium Webdriver là thành phần cốt lõi và mạnh mẽ nhất trong bộ công cụ Selenium. Cho phép lập trình viên viết mã kiểm thử để điều khiển trực tiếp các trình duyệt web một cách tự nhiên và chính xác. Webdriver hoạt động bằng cách tương tác trực tiếp với trình duyệt thông qua các native API của trình duyệt, thay vì thông qua lớp trung gian như Selenium IDE.

WebDriver hỗ trợ nhiều ngôn ngữ lập trình phổ biến như Java, Python, C#, Ruby, JavaScript, và có thể tương thích với hầu hết các trình duyệt hiện đại như Chrome, Firefox, Safari, Edge, Opera, v.v. Điều này cho phép lập trình viên xây dựng các bộ kiểm thử linh hoạt, mở rộng và dễ dàng tích hợp vào các quy trình kiểm thử tự động chuyên nghiệp.

**Ưu điểm:**

* Communicate trực tiếp với trình duyệt
* Tương tác với trình duyệt giống như thao tác của một người dùng thật
* Tốc độ nhanh hơn so với Selenium IDE
* Thao tác dễ dàng hơn với các phép tính toán logic hay các điều kiện phức tạp

**Nhược điểm:**

* Cài đặt phức tạp hơn so với Selenium IDE
* Đòi hỏi người dùng phải có kĩ năng lập trình

#### Selenium Grid

Selenium Grid là công cụ hỗ trợ mở rộng khả năng thực thi kiểm thử bằng cách chạy song song các kịch bản kiểm thử trên nhiều trình duyệt, hệ điều hành và thiết bị khác nhau. Grid bao gồm hai thành phần chính: Hub và Node. Hub đóng vai trò là trung tâm điều phối các kịch bản kiểm thử, trong khi các Node là nơi thực thi các kịch bản đó trên từng môi trường cụ thể.

Điều này đặc biệt hữu ích khi cần:

* Kiểm thử chéo trình duyệt
* Kiểm thử chéo nền tảng
* Giảm thời gian kiểm thử bằng cách chạy nhiều ca song song.

#### Selenium RC (Remote Control)

Selenium RC là một trong những thành phần đầu tiên của Selenium, cho phép viết mã kiểm thử bằng nhiều ngôn ngữ lập trình khác nhau và điều

khiển trình duyệt từ xa thông qua một máy chủ trung gian (Selenium Server).

**Ưu điểm:**

* Hỗ trợ nhiều ngôn ngữ lập trình: Java, C#, PHP, Python, Ruby...
* Có thể chạy kiểm thử trên trình duyệt từ xa thông qua HTTP proxy.

**Nhược điểm:**

* Cấu hình phức tạp do cần khởi động Selenium Server.
* Tốc độ chậm và không tương tác trực tiếp với trình duyệt.
* Đã lỗi thời và được thay thế bởi Selenium WebDriver.

## Selenium Webdriver

### Tổng quan về Selenium Webdriver

Selenium WebDriver là thành phần cốt lõi và hiện đại nhất trong bộ công cụ Selenium, được sử dụng phổ biến trong lĩnh vực kiểm thử tự động các ứng dụng web. Khác với các công cụ kiểm thử thủ công đòi hỏi người kiểm thử phải thao tác trực tiếp với giao diện người dùng, Selenium Webdriver giúp tự động hóa toàn bộ quá trình này bằng cách điều khiển trình duyệt thông qua mã lập trình.

Sự ra đời của WebDriver đã đánh dấu bước chuyển mình lớn trong lĩnh vực kiểm thử phần mềm, giúp nâng cao hiệu quả, giảm thiểu chi phí và tối ưu thời gian kiểm thử.

### Tiền thân của Selenium Webdriver

Trước khi có Webdriver, Selenium sử dụng một thành phần có tên là Selenium RC (Remote Control). Selenium RC cho phép kiểm thử tự động bằng cách sử dụng một máy chủ trung gian (Selenium Server) để "tiêm" JavaScript vào trình duyệt nhằm giả lập các hành động người dùng. Mặc dù RC hỗ trợ đa ngôn ngữ và trình duyệt, nhưng việc phụ thuộc vào máy chủ khiến nó trở nên chậm chạp và khó mở rộng.

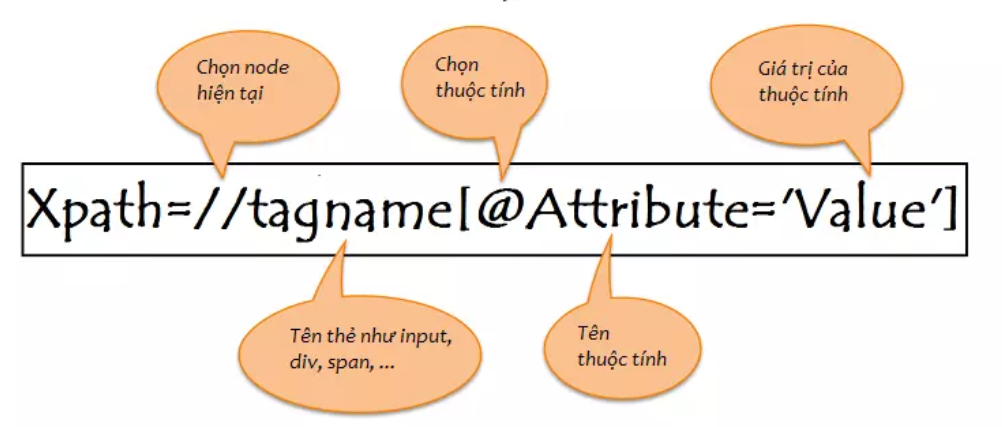
Nhằm khắc phục những điểm yếu đó, Selenium Webdriver ra đời như một giải pháp mới, không cần máy chủ trung gian và tương tác trực tiếp với trình duyệt thông qua API gốc (native API). Đây là lý do khiến Webdriver ngày nay trở thành lựa chọn mặc định cho các dự án kiểm thử tự động.

### Đặc điểm nổi bật của Selenium Webdriver

* Không cần máy chủ trung gian: Webdriver tương tác trực tiếp với trình duyệt thay vì sử dụng proxy như Selenium RC, giúp tăng tốc độ kiểm thử và giảm độ trễ.
* Hỗ trợ đa trình duyệt: Có thể làm việc với hầu hết các trình duyệt phổ biến như Google Chrome, Mozilla Firefox, Microsoft Edge, Safari...
* Hỗ trợ đa ngôn ngữ: Cho phép lập trình bằng nhiều ngôn ngữ như Java, Python, C#, Ruby, JavaScript (Node.js)...
* Khả năng mở rộng cao: Có thể tích hợp dễ dàng với các framework kiểm thử như JUnit, TestNG, PyTest cũng như các hệ thống CI/CD như Jenkins.
* Hỗ trợ kiểm thử nâng cao: Webdriver có thể xử lý nhiều tình huống phức tạp như thao tác với JavaScript, cảnh báo (alerts), chuyển tab, xử lý cookies, kéo thả, tải lên/tải xuống tệp, v.v.

### Cách lấy Xpath của element trên website

XPath được định nghĩa là đường dẫn XML. Nó là một cú pháp hoặc ngôn ngữ để tìm kiếm bất kỳ phần tử nào trên trang web bằng cách sử dụng biểu thức XML path. XPath được sử dụng để tìm vị trí của bất kỳ phần tử nào trên trang web bằng cách sử dụng cấu trúc DOM HTML.



Hình 2.1: Cấu trúc của Xpath

Có 2 loại Xpath: Xpath tuyệt đối và Xpath tương đối.

Xpath tuyệt đối:

* Xpath tuyệt đối bắt đầu bằng dấu gạch chéo đơn "", cho phép xác định một đường dẫn tuyệt đối đến đối tượng UI.
* Xpath tuyệt đối được xem là là cách tìm kiếm phần tử dễ dàng nhất, tuy nhiên nhược điểm của nó là nếu có bất kỳ thay đổi nào trên đường dẫn của element thì Xpath lấy sẽ sai.

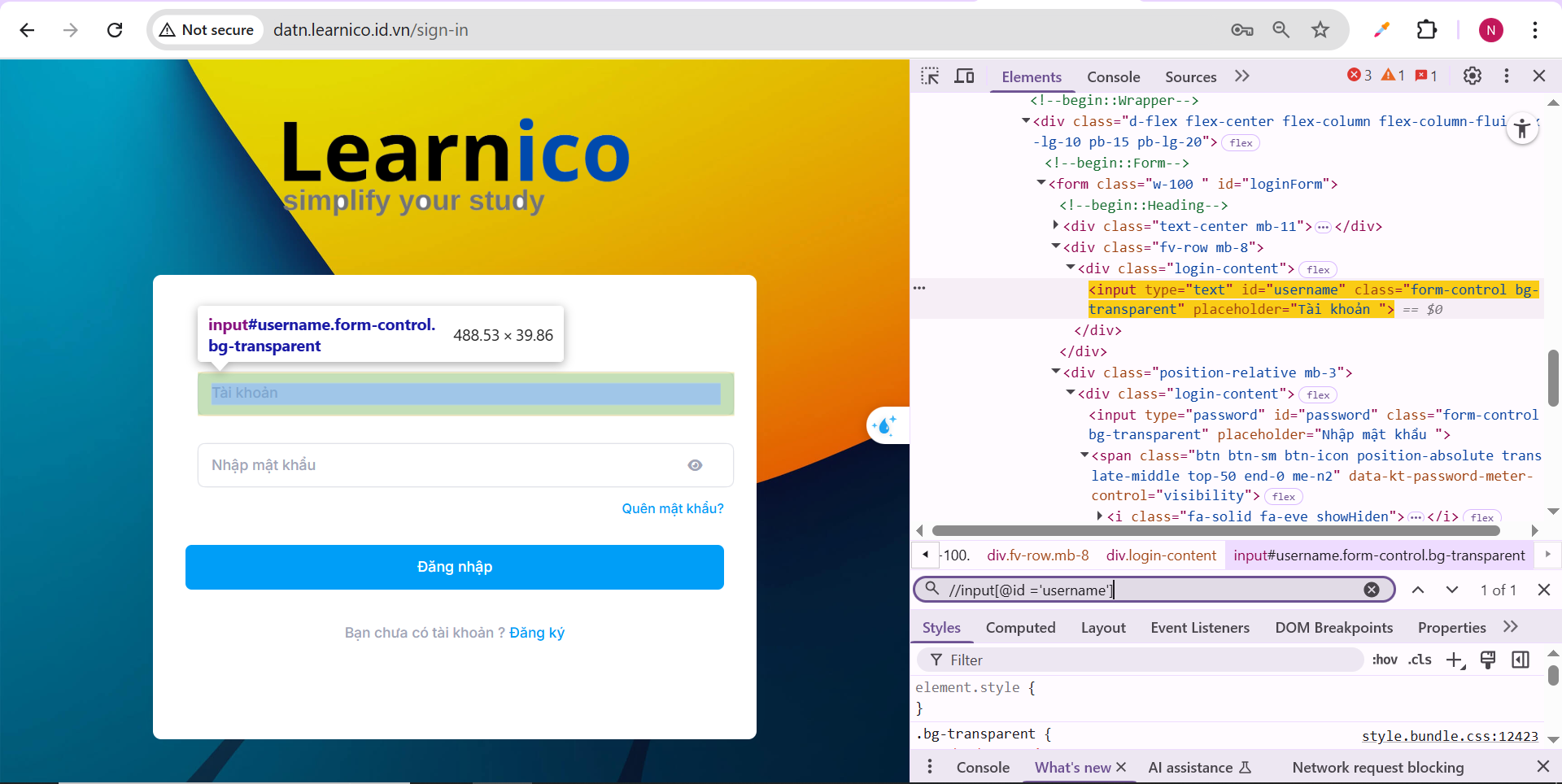
Xpath tương đối: Xpath tương đối bắt đầu bằng 2 dấu gạch chéo "//", cho phép xác định một đối tượng UI ở bất kỳ đâu trên trang web, không cần bắt đầu bởi thẻ html trong đường dẫn.

Cách lấy Xpath từ Inspect của Chrome:

Bước 1: Chạy và khởi động trình duyệt.

Bước 2: Trỏ chuột vào vị trí muốn lấy. Sau đó, bấm chuột phải rồi chọn Inspect.

Bước 3: Kiểm tra lại xem vị trí đã được xác định đúng chưa. Sau đó, bấm chuột phải rồi chọn Copy và chọn Copy Xpath.



Hình 2.2: Cách lấy Xpath

### Các câu lệnh sử dụng trong Selenium Webdriver

#### Lệnh khởi tạo trình duyệt và truy cập trang web

* Mục đích: Mở trình duyệt và truy cập một trang web cụ thể
* Cú pháp:

**from** selenium **import** webdriver

driver = webdriver.Chrome() *# Khởi tạo trình duyệt Chrome*

driver.get("https://example.com") *# Truy cập URL*

#### Lệnh tìm kiếm các phần tử trên trang (Element Locators)

* Mục đích: Xác định phần từ web để tương tác
* Cú pháp:

**from** selenium.webdriver.common.by **import** By

driver.find\_element(By.ID, "username") *# Tìm theo ID*

driver.find\_element(By.NAME, "email") *# Tìm theo name*

driver.find\_element(By.CLASS\_NAME, "form-input") *# Tìm theo class*

driver.find\_element(By.TAG\_NAME, "input") *# Tìm theo thẻ HTML*

driver.find\_element(By.LINK\_TEXT, "Đăng nhập") *# Tìm theo link text*

driver.find\_element(By.XPATH, "//input[@id='username')*# Theo XPath*

driver.find\_element(By.CSS\_SELECTOR, "input#username")*#Theo CSS*

#### Lệnh tương tác các phần tử

* + - * 1. **sen\_keys()**
* Mục đích: Gõ nội dung vào ô iput
* Cú pháp và ví dụ:

driver.find\_element(By.ID, "username").send\_keys("admin")

* + - * 1. **click()**
* Mục đích nhấn vào nút hoặc liên kết
* Cú pháp và ví dụ: driver.find\_element(By.ID, "submitBtn").click()
  + - * 1. **clear()**
* Mục đích: Xóa nội cung đã nhập
* Cú pháp và ví dụ: driver.find\_element(By.ID, "username").clear()
  + - * 1. **submit()**
* Mục đích: Gửi biểu mẫu
* Cú pháp và ví dụ: driver.find\_element(By.ID, "loginForm").submit()

#### Lệnh truy xuất thông tin phần tử

* + - * 1. **text**
* Mục đích: Lấy nội dung văn bản
* Cú pháp và ví dụ: driver.find\_element(By.ID, "msg").text
  + - * 1. **get\_attribute()**
* Mục đích: Lấy giá trị thuộc tính
* Cú pháp và ví dụ:

driver.find\_element(By.ID, "email").get\_attribute("value")

* + - * 1. **is\_displayed()**
* Mục đích: Kiểm tra có hiển thị không
* Cú pháp và ví dụ: driver.find\_element(By.ID, "btn").is\_displayed()

#### Lệnh chờ đợi

* + - * 1. **sleep()**
* Mục đích: Chờ đợi trong một thời gian nhất định
* Cú pháp và ví dụ:

import time

time.sleep(3)

* + - * 1. **Explicit Wait**
* Mục đích: Chờ phần tử xuất hiện hoặc có điều kiện trước khi thao tác
* Cú pháp và ví dụ:

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.ID, "username"))

)

#### Điều hướng trang web

* + - * 1. **driver.back()**
* Mục đích: Quay về trang trước
* Cú pháp: driver.back()
* Ví dụ cụ thể:

**from** selenium **import** webdriver

import time

driver = webdriver.Chrome()

driver.get("https://www.google.com")

time.sleep(2)

driver.get("https://www.wikipedia.org")

time.sleep(2)

driver.back() *# Quay lại trang Google*

time.sleep(2)

driver.quit()

* + - * 1. **driver.refresh()**
* Mục đích: Tải lại trang
* Cú pháp: driver.refresh()
* Ví dụ cụ thể:

**from** selenium **import** webdriver

import time

driver = webdriver.Chrome()

driver.get("https://www.example.com")

time.sleep(2)

driver.refresh() *# Làm mới lại trang*

time.sleep(2)

driver.quit()

## Công cụ và thư viện khác được sử dụng

Trong quá trình xây dựng hệ thống kiểm thử tự động cho ứng dụng web sử dụng Selenium WebDriver, bên cạnh Selenium, một số công cụ và thư viện khác cũng được tích hợp nhằm hỗ trợ tối ưu quy trình kiểm thử, ghi nhận kết quả, và tạo báo cáo trực quan.

### Ngôn ngữ Python

Python là một trong những ngôn ngữ lập trình phổ biến và được ưa chuộng nhất hiện nay nhờ vào cú pháp đơn giản, dễ đọc, dễ viết và cộng đồng phát triển mạnh mẽ. Được phát triển bởi Guido van Rossum và phát hành lần đầu vào năm 1991, Python giúp người lập trình dễ dàng tiếp cận, đặc biệt là những người mới bắt đầu học lập trình.

Với khả năng tương thích đa nền tảng và hàng nghìn thư viện mã nguồn mở, Python được ứng dụng rộng rãi trong nhiều lĩnh vực như: phát triển ứng dụng web (Flask, Django), khoa học dữ liệu (NumPy, Pandas, Matplotlib), trí tuệ nhân tạo và học máy (TensorFlow, Scikit-learn), tự động hóa (Selenium, PyAutoGUI), v.v.

Trong lĩnh vực kiểm thử phần mềm, Python là một lựa chọn lý tưởng để viết các kịch bản kiểm thử tự động nhờ khả năng tích hợp linh hoạt với các công cụ như Selenium WebDriver, unittest, pytest,... Việc sử dụng Python giúp kiểm thử viên dễ dàng mô tả hành vi người dùng trên giao diện web, thực hiện các hành động như nhập liệu, nhấn nút, kiểm tra dữ liệu trả về, và tự động ghi nhận kết quả. Bên cạnh đó, Python cũng hỗ trợ tốt việc sinh báo cáo, ghi log, quay video và chụp ảnh màn hình trong quá trình test, góp phần nâng cao hiệu quả và độ chính xác trong kiểm thử phần mềm.

### Lambdatest

LambdaTest là một nền tảng kiểm thử đám mây (Cloud-based Testing Platform) cho phép thực hiện kiểm thử giao diện và chức năng của ứng dụng web trên nhiều tổ hợp hệ điều hành và trình duyệt khác nhau. Thông qua việc tích hợp với các công cụ kiểm thử tự động như Selenium, Cypress, Playwright, LambdaTest hỗ trợ người dùng thực hiện các bài kiểm thử tự động một cách hiệu quả mà không cần đầu tư hạ tầng vật lý phức tạp.

Với LambdaTest, người kiểm thử có thể kiểm tra khả năng tương thích của hệ thống trên hàng nghìn cấu hình trình duyệt và hệ điều hành ngay trên nền tảng web, giúp tiết kiệm thời gian, chi phí và công sức trong việc bảo trì môi trường kiểm thử truyền thống

**Ưu điểm:**

* Hỗ trợ đa nền tảng: LambdaTest cung cấp hơn 3000 tổ hợp cấu hình trình duyệt và hệ điều hành khác nhau, bao gồm Windows, macOS, Linux và các phiên bản khác nhau của Chrome, Firefox, Edge, Safari...
* Không cần cài đặt máy ảo: Người dùng không cần tự tạo máy ảo hay cấu hình môi trường thủ công, mọi cấu hình kiểm thử đều có sẵn trên nền tảng LambdaTest.
* Hỗ trợ kiểm thử tự động: Dễ dàng tích hợp với Selenium, Appium, Cypress,... để chạy kiểm thử tự động.
* Báo cáo chi tiết và trực quan: Hệ thống cung cấp video ghi lại quá trình kiểm thử, log chi tiết.

**Nhược điểm:**

* Hạn chế với gói miễn phí: Gói dùng thử hoặc miễn phí giới hạn số lượng phút kiểm thử, số lượng phiên kiểm thử đồng thời và các tính

năng nâng cao.

* Phụ thuộc vào kết nối Internet: Vì là nền tảng đám mây nên cần có kết nối mạng ổn định để thực hiện kiểm thử.
* Thời gian khởi tạo phiên làm việc có thể chậm: Khi hệ thống đông người dùng hoặc đường truyền không ổn định, quá trình tạo phiên làm việc kiểm thử có thể bị chậm hoặc bị gián đoạn.

### Thư viện OpenCV (cv2) và PyAutoGUI

Để ghi lại quá trình kiểm thử dưới dạng video, thư viện OpenCV (cv2) được sử dụng kết hợp với PyAutoGUI. Trong đó, PyAutoGUI đảm nhận việc chụp liên tục các ảnh màn hình, còn OpenCV xử lý và ghép chúng thành video. Điều này giúp minh họa trực quan quá trình kiểm thử và phục vụ cho việc kiểm tra lại khi phát sinh lỗi.

### Thư viện Requests

Thư viện requests được sử dụng để gửi các yêu cầu HTTP đến các API upload ảnh hoặc video. Điều này hỗ trợ việc tự động hóa quá trình đính kèm dữ liệu minh họa (ảnh, video) vào báo cáo kiểm thử.

### Thư viện CuteCharts

CuteCharts là một thư viện giúp tạo biểu đồ HTML đơn giản nhưng sinh động. Trong dự án, thư viện này được sử dụng để trực quan hóa thống kê số lượng test case đạt, lỗi và thất bại theo dạng biểu đồ tròn hoặc cột.

## Cài đặt selenium Webdriver

### Cài đặt Python

Bước 1: Tải Python

* Truy cập trang chủ: <https://www.python.org/>
* Vào menu Downloads → chọn Download Python x.x.x (bản mới nhất).

Bước 2: Cài đặt

* Mở file .exe vừa tải xuống.
* Tick vào Add Python to PATH (thêm vào biến môi trường).
* Chọn Install Now để tiến hành cài đặt mặc định.

Bước 3: Kiểm tra cài đặt

* Mở Command Prompt (cmd) và gõ lệnh: python –version
* Nếu hiển thị phiên bản Python (Ví dụ: Python 3.13.3), quá trình cài đặt thành công

### Cài đặt PyCharm

#### Giới thiệu về PyCharm

PyCharmlà một môi trường phát triển tích hợp (IDE) chuyên dụng cho ngôn ngữ Python, được phát triển bởi JetBrains. Công cụ này cung cấp giao diện thân thiện, hỗ trợ gợi ý mã, kiểm tra lỗi, quản lý thư viện và tích hợp sẵn công cụ chạy lệnh, giúp quá trình phát triển phần mềm trở nên dễ dàng và hiệu quả hơn.

Các phiên bản của PyCharm:

* Community Edition *(Miễn phí)* – đủ dùng cho các dự án Python thông thường.
* Professional Edition *(Trả phí)* – hỗ trợ thêm các công nghệ web (Django, Flask, v.v.).

Trong phạm vi đồ án, bản Community Edition là đủ dùng.

#### Các bước cài đặt PyCharm trên Windows

Bước 1: Tải PyCharm

* Truy cập trang chủ: <https://www.jetbrains.com/pycharm/>
* Nhấn nút Download → chọn Community Edition.

Bước 2: Cài đặt

* Mở file cài đặt .exe vừa tải xuống.
* Chọn các tuỳ chọn cần thiết (nên chọn “Add launcher to PATH”).
* Nhấn Install để bắt đầu quá trình cài đặt.
* Sau khi hoàn tất, chọn Run PyCharm.

Bước 3: Cấu hình lần đầu

* Lựa chọn giao diện (Light hoặc Dark).
* Chọn thiết lập mặc định hoặc tuỳ chỉnh plugin.

Bắt đầu sử dụng.

### Cài đặt Selenium Webdriver

#### Cài đặt thư viện Selenium Webdriver

Để cài đặt thư viện selenium, người dùng sử dụng trình quản lý gói pip của Python với câu lệnh sau trong cửa sổ Terminal hoặc Command Prompt: pip install selenium

Sau khi cài đặt thành công, có thể kiểm tra lại bằng cách mở Python và nhập:

import selenium

print(selenium.\_\_version\_\_)

#### Cài đặt WebDriver cho trình duyệt

Selenium WebDriver cần trình điều khiển (driver) tương ứng với trình duyệt web mà người dùng muốn tự động hóa. Trong đồ án này, sử dụng trình duyệt Google Chrome, do đó cần cài đặt ChromeDriver.

Các bước cài đặt ChromeDriver:

* Truy cập trang: <https://sites.google.com/chromium.org/driver/>
* Chọn phiên bản phù hợp với phiên bản Chrome đang cài trên máy (kiểm tra tại chrome://settings/help).
* Tải về tệp .zip, giải nén và đặt tệp chromedriver.exe vào thư mục hệ thống (hoặc cấu hình đường dẫn trong mã Python).

### Xây dựng dự án Selenium WebDriver trong PyCharm

* Mở PyCharm → Chọn File > New Project
* Đặt tên cho dự án, ví dụ: selenium\_automation\_project
* Chọn New Environment using: Virtualenv hoặc sử dụng interpreter đã có sẵn.
* Nhấn Create để khởi tạo dự án.
* Chuột phải vào thư mục project → chọn New > Python File
* Đặt tên, ví dụ: test\_google.py



Hình 2.3: Ví dụ mở Google bằng Selenium WebDriver

### Các bước thực hiện kiểm thử tự động với Selenium WebDriver trên Lambdatest

Bước1: Đăng ký và đăng nhập tài khoản LambdaTest

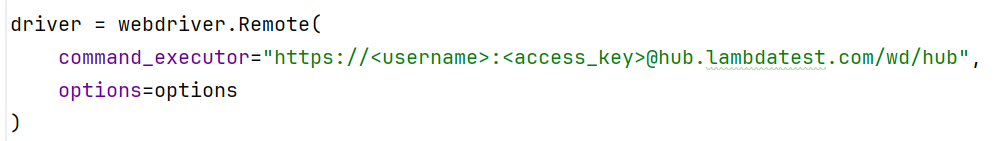
* Truy cập trang chủ <https://www.lambdatest.com> và tạo tài khoản.
* Sau khi đăng nhập, người dùng sẽ nhận được thông tin Username và Access\_key để tích hợp vào mã kiểm thử.

Bước 2: Cấu hình thông số cho trình duyệt và hệ điều hành

* Sử dụng các cấu hình như: browser name, browser version, OS, OS version,... để lựa chọn môi trường kiểm thử phù hợp.

Bước 3: Tạo script kiểm thử với Selenium WebDriver

* Sử dụng ngôn ngữ lập trình Python (hoặc Java, JavaScript...) để viết script kiểm thử.
* Thay vì sử dụng webdriver.Chrome(), dùng webdriver.Remote() để kết nối tới LambdaTest Grid qua URL



Hình 2.4: Tạo script kiểm thử tích hợp với Selenium WebDriver

Bước 4: Thực hiện kiểm thử và ghi nhận kết quả

* Script tiến hành kiểm thử và có thể sử dụng driver.execute\_script ("lambda-status=passed/failed") để thông báo trạng thái.
* LambdaTest sẽ tự động lưu lại log, video cho mỗi phiên làm việc kiểm thử.

Bước 5: Xem kết quả kiểm thử trên Dashboard

* Truy cập trang quản lý tại: <https://automation.lambdatest.com>
* Tại đây có thể xem video, log, trạng thái kiểm thử và tải về phục vụ cho việc tổng hợp báo cáo.

# KIỂM THỬ CHO HỆ THỐNG HỌC TRỰC TUYẾN LEARNICO

## Mô tả về hệ thống học trực tuyến Learnico

### Giới thiệu

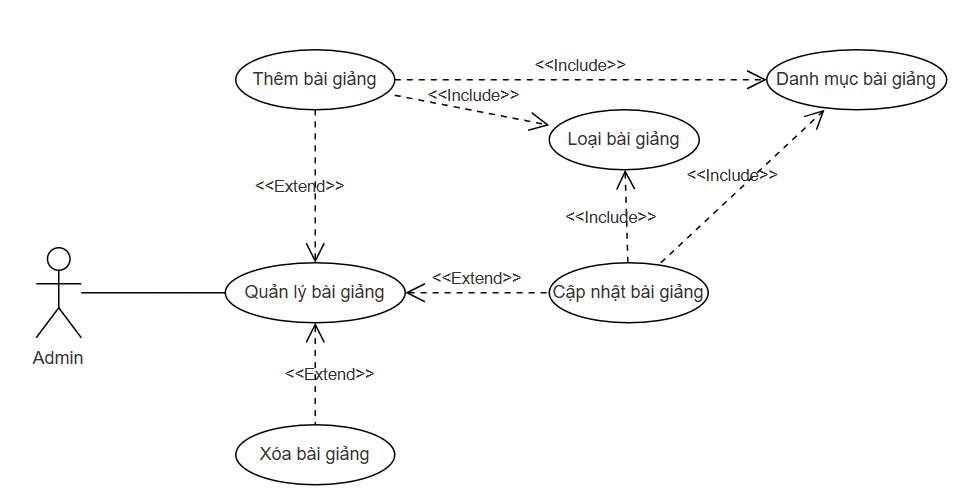
Hệ thống em thực hiện kiểm thử là một hệ thống học trực tuyến được xây dựng nhằm cung cấp các khóa học trực tuyến, giúp cho khách hàng dù ở đâu cũng có thể học trực tiếp trên hệ thống mà không cần đến lớp giảng dạy như truyền thống

Hệ thống bao gồm các hoạt động chính như: Quản lý bài giảng, quản lý khóa học, quản lý học viên, giáo viên. Học viên có thể mua khóa học, học khóa học, làm bài kiểm tra trên khóa học.

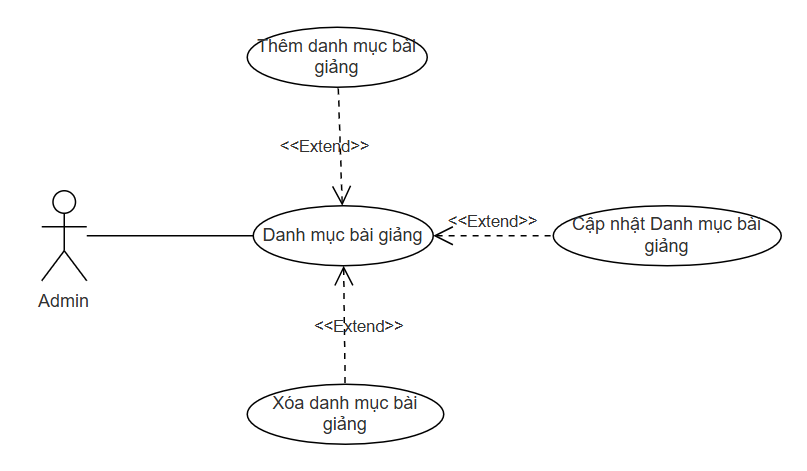
Bên cạnh những chức năng nghiệp vụ, hệ thống phải đảm bảo những đặc tính sau:

* Hệ thống phải đảm bảo tỉnh bảo mật cao.
* Thời gian đáp ứng của hệ thống phải đủ nhanh, có thể đáp ứng được số lượng truy cập lớn trên một đơn vị thời gian.
* Hệ thống có giao diện thân thiện với người sử dụng.
* Hệ thống có khả năng mở rộng, dễ dàng tích hợp với các hệ thống khác.

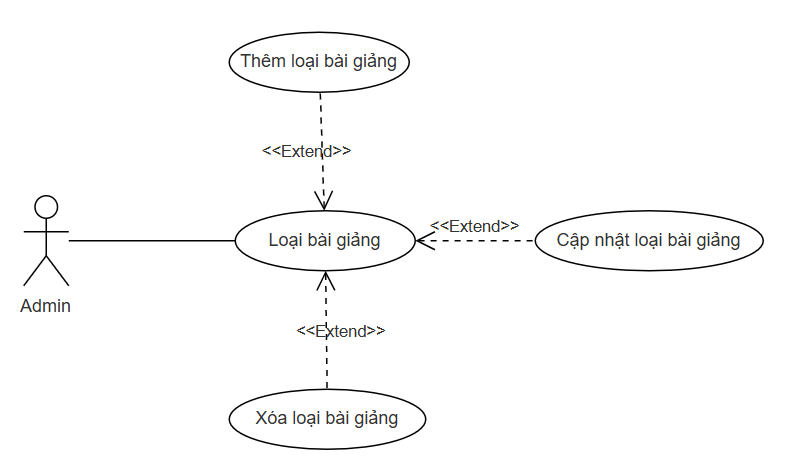
### Biểu đồ usecase



Hình 3.1: Biểu đồ phân rã usecase Quản lý bài giảng



Hình 3.2: Biểu đồ phân rã usecase Danh mục bài giảng



Hình 3.3: Biểu đồ phân rã usecase Loại bài giảng

### Mô tả usecase

Trong khuôn khổ đồ án, em sẽ thực hiện kiểm thử một vài chức năng của hệ thống như: Đăng nhập, quản lý bài giảng, loại bài giảng, danh mục bài giảng

#### Mô tả usecase đăng nhập

**Mô tả vắn tắt:**

Use case này cho phép thành viên (học sinh, giảng viên, quản trị viên/admin) có thể đăng nhập vào hệ thống.

**Luồng sự kiện:**

***Luồng cơ bản:***

1. Use case này bắt đầu khi người dùng kích vào nút “Đăng nhập” ở góc trên bên phải trang chủ. Hệ thống sẽ hiển thị màn hình yêu cầu nhập tên đăng nhập và mật khẩu.

2. Người dùng nhập tên đăng nhập và mật khẩu sau đó kích vào nút “Đăng nhập”. Hệ thống kiểm tra tên đăng nhập, mật khẩu và quyền truy nhập trong bảng Account, nếu thông tin người dùng hợp lệ, hệ thống gửi thông báo thành công và chuyển về màn hình trang chủ.

***Luồng rẽ nhánh:***

Tại bước 2 của luồng cơ bản, nếu người dùng nhập sai tên đăng nhập hoặc mật khẩu hệ thống sẽ gửi thông báo và yêu cầu người dùng đăng nhập lại. Use case kết thúc.

Tại bước 2 của luồng cơ bản, nếu người dùng chưa kích hoạt tài khoản qua email đã dùng để đăng ký tài khoản, hệ thống sẽ gửi thông báo “Tài khoản chưa được kích hoạt. Vui lòng kiểm tra email để kích hoạt tài khoản”. Use case kết thúc.

**Tiền điều kiện:** Người dùng chưa đăng nhập vào hệ thống.

**Hậu điều kiện:** Người dùng đã đăng nhập thành công và có thể sử dụng chức năng của hệ thống tương ứng với mỗi quyền của tài khoản.

**Các yêu cầu đặc biệt:** Không có.

**Điểm mở rộng:** Không có.

#### Mô tả usecase quản lý bài giảng

**Mô tả vắn tắt:**

Use case này cho phép người quản trị (quản trị viên/giảng viên) quản lý thông tin bài giảng.

**Luồng sự kiện:**

***Luồng cơ bản:***

1.Use case này bắt đầu khi người quản trị click vào mục “Quản lý bài giảng” trong thanh menu. Hệ thống sẽ truy vấn bảng Lectures và hiển thị danh sách bài giảng lên màn hình.

2. Tạo mới bài giảng:

a) Người quản trị chọn vào phần “Thêm mới”. Hệ thống hiển thị form nhập thông tin chi tiết cho bài giảng mới gồm tên, loại bài giảng (video, audio, văn bản), chọn video bài giảng (nếu là bài giảng loại video), chọn phụ đề (nếu có), chọn audio (nếu là bài giảng loại audio)…

b) Người quản trị nhập thông tin và click vào nút “Lưu”. Hệ thống sẽ một bản ghi mới trong bảng Lectures và hiển thị thông báo cập nhật thành công.

3. Tìm kiếm bài giảng: Người quản trị có thể tìm kiếm khoá học bất kì bằng một trong các cách nhập tên, mã, loại… vào ô tìm kiếm, hệ thống sẽ truy vấn bảng Lectures và hiển thị thông tin bài giảng cần tìm lên màn hình.

4. Sửa bài giảng:

a) Khi người quản trị click vào biểu tượng “Sửa” trên một dòng thông tin về bài giảng. Hệ thống sẽ hiển thị các trường thông tin cũ của bài giảng cho phép người quản trị sửa thông tin.

b) Người quản trị thay đổi thông tin bài giảng và click vào nút “Cập nhật”. Hệ thống kiểm tra thông tin thay đổi, cập nhật bảng Lectures và hiển thị thông báo thành công.

5. Xóa bài giảng: Khi quản trị viên click vào biểu tượng xóa của một bài giảng, hệ thống sẽ hiển thị thông báo xác nhận. Người quản trị click vào “Xoá”, hệ thống sẽ xóa bài giảng trong bảng Courses. Usecase kết thúc.

***Luồng rẽ nhánh:***

1. Tại bước 2 trong luồng cơ bản nếu người quản trị nhập tên bài giảng và danh mục bài giàng trùng khớp với bài giảng đã tồn tại, hệ thống hiển thị thông báo lỗi

2. Tại bước 5 trong luồng cơ bản nếu người quản trị click vào nút “Hủy” hoặc đóng cửa sổ hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách bài giảng trong bảng Lectures.

3. Tại bất kì thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị thông báo lỗi và use case kết thúc.

**Tiền điều kiện:** Người quản trị phải đăng nhập với quyền quản trị trước khi thực hiện use case.

**Hậu điều kiện:** Nếu use case kết thúc thành công thì thông tin về bài giảng sẽ được cập nhật trong cơ sở dữ liệu.

**Các yêu cầu đặc biệt:**

1. Use case này chỉ cho phép người quản trị thực hiện.

2. Người quản trị có vai trò trong hệ thống là giảng viên thì chỉ quản lý được các bài giảng là của mình.

**Điểm mở rộng:** Không có.

#### Mô tả usecase danh mục bài giảng

**Mô tả vắn tắt:**

Use case này cho phép người quản trị (quản trị viên) thêm danh mục bài giảng

**Luồng sự kiện:**

***Luồng cơ bản:***

1. Use case này bắt đầu khi người quản trị click vào mục “Danh mục bài giảng” trong thanh menu. Hệ thống sẽ truy vấn bảng Lecture Category và hiển thị danh sách danh mục bài giảng lên màn hình.

2. Tạo mới bài giảng:

a) Người quản trị chọn vào phần “Thêm mới”. Hệ thống hiển thị form nhập thông tin chi tiết gồm tên, cấp cha, thứ tự hiển thị, mô tả

b) Người quản trị nhập thông tin và click vào nút “Lưu”. Hệ thống sẽ một bản ghi mới trong bảng Lecture Category và hiển thị thông báo thêm mới thành công.

3. Sửa danh mục bài giảng:

a) Khi người quản trị click vào tên danh mục bài giảng. Hệ thống sẽ hiển thị các trường thông tin cũ của danh mục bài giảng cho phép người quản trị sửa thông tin.

b) Người quản trị thay đổi thông tin bài giảng và click vào nút “Lưu”. Hệ thống kiểm tra thông tin thay đổi, cập nhật bảng Lecture Category và hiển thị thông báo thành công.

4. Xóa bài giảng: Khi người quản trị click vào biểu tượng xóa của một bài giảng, hệ thống sẽ hiển thị thông báo xác nhận. Người quản trị click vào “Xoá”, hệ thống sẽ xóa danh mục bài giảng trong bảng Lecture Category kết thúc.

***Luồng rẽ nhánh:***

Tại bước 2 trong luồng cơ bản nếu người quản trị nhập tên đã tồn tại hệ thống sẽ hiển thị thông báo lỗi

Tại bước 2 trong luồng cơ bản nếu người dùng click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách danh mục bài giảng trong bảng Lecture Category

Tại bước 3 trong luồng cơ bản nếu người dùng click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác sửa và hiển thị danh sách danh mục bài giảng trong bảng Lecture Category

4. Tại bước 4 trong luồng cơ bản nếu người quản trị click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách bài giảng trong bảng Lecture Category

5. Tại bất kì thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị thông báo lỗi và use case kết thúc.

**Tiền điều kiện**: Người quản trị phải đăng nhập với quyền quản trị trước khi thực hiện use case.

**Hậu điều kiện**: Nếu use case kết thúc thành công thì thông tin về danh mục bài giảng sẽ được cập nhật trong cơ sở dữ liệu.

**Các yêu cầu đặc biệt**:

1. Use case này chỉ cho phép người quản trị thực hiện.

**Điểm mở rộng**: Không có.

#### Mô tả usecase loại bài giảng

**Mô tả vắn tắt:**

Use case này cho phép người quản trị (quản trị viên) thêm loại bài giảng

**Luồng sự kiện:**

***Luồng cơ bản:***

1. Use case này bắt đầu khi người quản trị click vào mục “Loại bài giảng” trong thanh menu. Hệ thống sẽ truy vấn bảng Lecture Type và hiển thị danh sách loại bài giảng lên màn hình.

2. Tạo mới bài giảng:

a) Người quản trị chọn vào phần “Thêm mới”. Hệ thống hiển thị form nhập thông tin chi tiết gồm Loại bài giảng, mô tả, ngày tạo (disable)

b) Người quản trị nhập thông tin và click vào nút “Lưu”. Hệ thống sẽ một bản ghi mới trong bảng Lecture Type và hiển thị thông báo thêm mới thành công thành công.

3. Sửa loại bài giảng:

a) Khi người quản trị click biểu tượng sửa. Hệ thống sẽ hiển thị các trường thông tin cũ của loại bài giảng cho phép người quản trị sửa thông tin.

b) Người quản trị thay đổi thông tin loại bài giảng và click vào nút “Lưu”. Hệ thống kiểm tra thông tin thay đổi, cập nhật bảng Lecture Type và hiển thị thông báo thành công.

4. Xóa bài giảng: Khi người quản trị click vào biểu tượng xóa của một loại bài giảng, hệ thống sẽ hiển thị thông báo xác nhận. Người quản trị click vào “Xoá”, hệ thống sẽ xóa loại bài giảng trong bảng Lecture Type kết thúc.

5. Tìm kiếm loại bài giảng: Người quản trị có thể tìm kiếm loại bài giảng bất kì bằng một trong các cách nhập tên vào ô tìm kiếm, hệ thống sẽ truy vấn bảng Lecture Type và hiển thị thông tin loại bài giảng cần tìm lên màn hình

***Luồng rẽ nhánh:***

3. Tại bước 2 trong luồng cơ bản nếu người quản trị nhập loại bài giảng đã tồn tại hệ thống sẽ hiển thị thông báo lỗi

2. Tại bước 2 trong luồng cơ bản nếu người dùng click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác thêm mới và hiển thị danh sách loại bài giảng trong bảng Lecture Type

3. Tại bước 3 trong luồng cơ bản nếu người dùng click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác sửa và hiển thị danh sách loại bài giảng trong bảng Lecture Type

4. Tại bước 5 trong luồng cơ bản nếu người quản trị click vào nút “Hủy” hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách bài giảng trong bảng Lecture Type.

5. Tại bất kì thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị thông báo lỗi và use case kết thúc.

**Tiền điều kiện:** Người quản trị phải đăng nhập với quyền quản trị trước khi thực hiện use case.

**Hậu điều kiện**: Nếu use case kết thúc thành công thì thông tin về danh mục bài giảng sẽ được cập nhật trong cơ sở dữ liệu.

**Các yêu cầu đặc biệt:**

1. Use case này chỉ cho phép người quản trị thực hiện.

**Điểm mở rộng:** Không có.

## Kế hoạch kiểm thử

### Phạm vi kiểm thử

Thực hiện tạo ca kiểm thử, viết test script và thực thi các chức năng của website học trực tuyến :

* Đăng nhập
* Quản lý bài giảng
* Danh mục bài giang
* Loại bài giảng

### Mục tiêu kiểm thử

Kiểm tra các chức năng của hệ thống hoạt động đúng với mô tả

Kiểm tra tương thích của hệ thống với các hệ điều hành, trình duyệt khác nhau

### Chiến lược kiểm thử

#### Đối với tạo các ca kiểm thử

Các ca kiểm thử được xây dựng dựa trên mô tả của hệ thống Learnico.

Trong quá trình tạo test case, các kỹ thuật kiểm thử được áp dụng bao gồm:

* Phân vùng tương đương (Equivalence Partitioning)
* Phân tích giá trị biên (Boundary Value Analysis)
* Bảng quyết định (Decision Table)
* Kỹ thuật đoán lỗi (Error Guessing)

Các kỹ thuật này giúp đảm bảo bao phủ đủ các trường hợp hợp lệ, không hợp lệ và bất thường có thể xảy ra khi người dùng tương tác với hệ thống.

#### Đối với viết test script

Các ca kiểm thử được tự động hóa bằng cách viết các test script sử dụng Selenium WebDriver và Python. Mỗi script tương ứng với một hoặc nhiều ca kiểm thử. Các script sẽ được viết theo thứ tự logic của hệ thống

Việc viết test script được tối ưu theo hướng:

* Tái sử dụng mã.
* Tách riêng dữ liệu test (sử dụng file JSON).
* Quản lý báo cáo tự động (ghi lại video, ảnh chụp và kết quả test).

### Tiêu chí kiểm thử

#### Tiêu chí bắt đầu kiểm thử

Quá trình kiểm thử tự động được bắt đầu khi:

* Hệ thống Learnico đã được triển khai đầy đủ các chức năng thuộc phạm vi kiểm thử.
* Các test case đã được xây dựng và rà soát kỹ, sẵn sàng để thực thi.
* Môi trường kiểm thử (trình duyệt, dữ liệu test, tài khoản người dùng...) đã được cấu hình hoàn chỉnh.

#### Tiêu chí đình chỉ kiểm thử

Các hoạt động kiểm thử sẽ bị đình chỉ khi:

* Các thành viên trong dự án chưa thực sự hiểu luồng hoạt động của hệ thống.
* Khách hàng yêu cầu thay đổi luồng hoạt động của chức năng đangkiểm thử.
* Đang thực thi kiểm thử thì gặp lỗi nghiêm trọng dẫn đến không thể tiếp tục thực thi kiểm thử.

#### Tiêu chí kết thúc kiểm thử

Quá trình kiểm thử được xem là hoàn tất khi đáp ứng các điều kiện sau:

* Tỷ lệ test case được thực thi thành công đạt yêu cầu (ví dụ: ≥ 95% hoặc theo mục tiêu ban đầu đã đề ra).
* Tất cả các chức năng thuộc phạm vi kiểm thử của hệ thống Learnico (như: quản lý bài giảng, loại bài giảng, danh mục bài giảng…)đều đã được kiểm thử đầy đủ.
* Toàn bộ các lỗi phát hiện trong quá trình kiểm thử đã được xử lý và xác nhận.
* Báo cáo kiểm thử được sinh ra đầy đủ, bao gồm: biểu đồ kết quả, chi tiết từng test case, ảnh chụp màn hình và video quá trình chạy test.

### Nguồn nhân lực

Bảng 3.1: Nguồn nhân lực

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Phân quyền** | **Trách nhiệm** | **Người thực hiện** |
| 1 | Tester | Tạo các ca kiểm thử  Viết test script  Thực hiện kiểm thử  Báo cáo bug | Nguyễn Thị Kim Chi |

### Môi trường kiểm thử

Kiểm thử chức năng của hệ thống:

* Link hệ thống: <http://datn.learnico.id.vn/admin>
* Trình duyệt kiểm thử: Chrome
* Hệ điều hành: Windows 10

Kiểm thử tương thích của hệ thống:

* Link hệ thống: <http://datn.learnico.id.vn/admin>
* Trình duyệt kiểm thử: Google Chrome, Mozilla Firefox, Microsoft Edge
* Hệ điều hành: Windows, macOS, Linux

### Lịch trình và dự toán

Bảng 3.2: Lịch trình và dụ toán

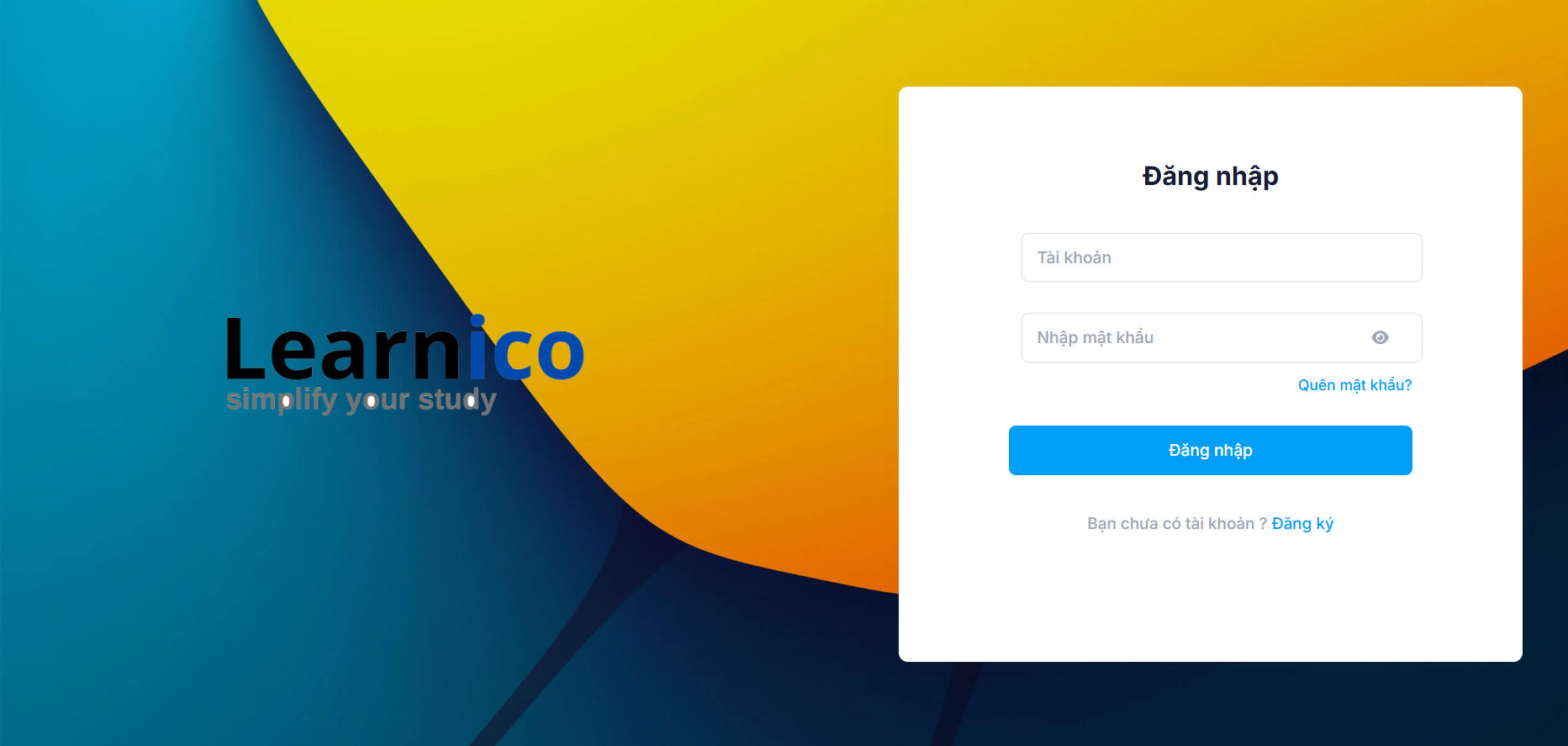
|  |  |  |  |
| --- | --- | --- | --- |
| **Công việc** | **Ngày bắt đầu** | **Ngày kết thúc** | **Ngày hoàn thành thực tế** |
| Lập kế hoạch kiểm thử | 17/3/2025 | 23/3/2025 | Chưa xác định |
| Thiết kế các ca kiểm thử | 24/3/2025 | 6/4/2025 | Chưa xác định |
| Viết test script | 7/4/2025 | 4/5/2025 | Chưa xác định |
| Thực thi kiểm thử | 5/5/2025 | 17/5/2025 | Chưa xác định |

## Thiết kế ca kiểm thử

### Kiểm thử chức năng

#### Đăng Nhập

*Tiền điều kiện*: Truy cập vào link hệ thống : <http://datn.learnico.id.vn/admin>

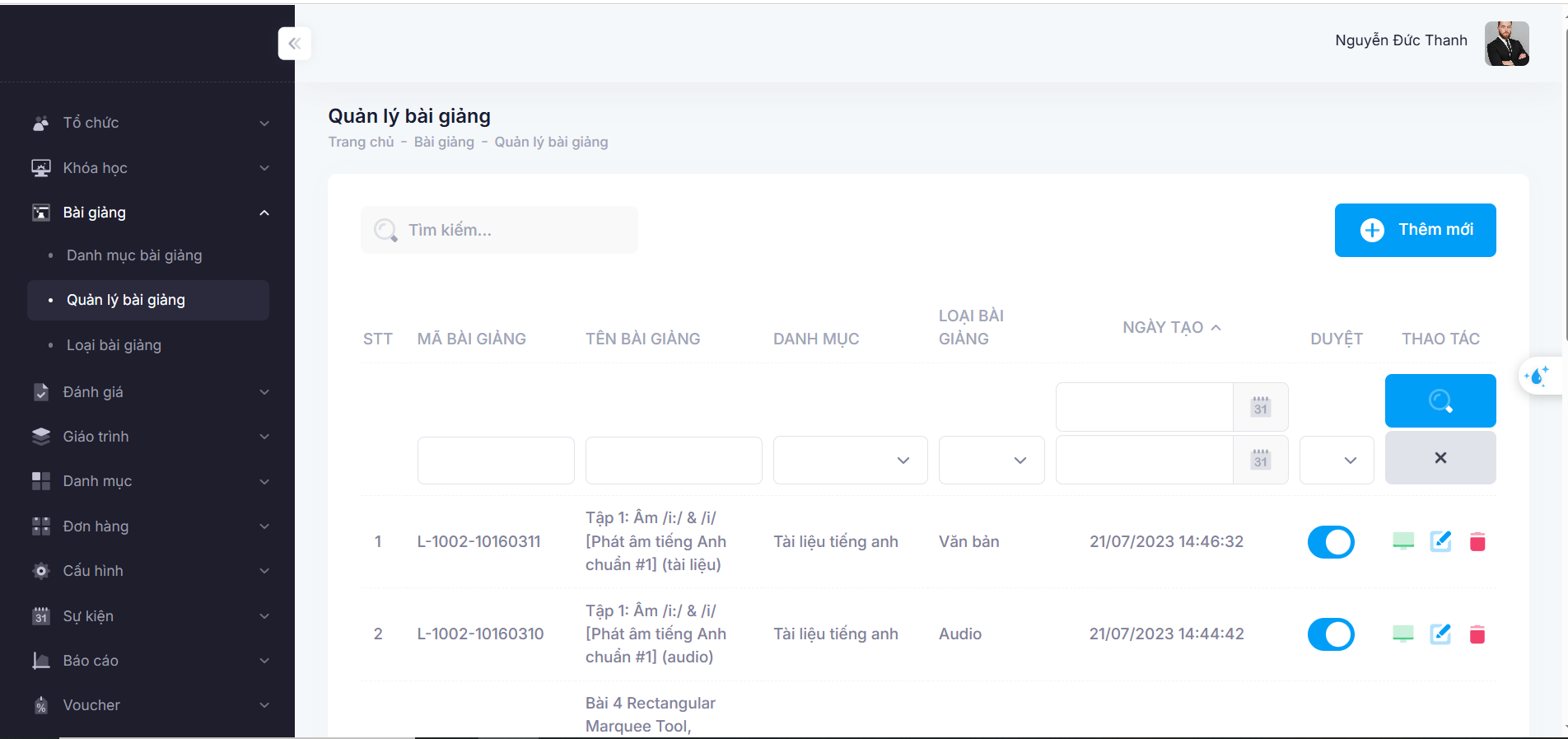


Hình 3.4: Màn hình Đăng nhập

Bảng 3.3: Test case Đăng Nhập

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| DN\_1 | Đăng nhập thành công | 1.Nhập tài khoản hợp lệ  2. Nhập mật khẩu hợp lệ  3. Click vào button Đăng nhập | Hiển thị thông báo “Đăng nhập thành công” | pass |
| DN\_2 | Đăng nhập sai tên tài khoản | 1.Nhập tên sai khoản sai  2. Nhập mật khẩu hợp lệ  3. Click vào button Đăng nhập | Hiển thị thông báo “Đăng nhập thất bại” | pass |
| DN\_3 | Đăng nhập sai mật khẩu | 1.Nhập tên tài khoản hợp lệ  2. Nhập sai mật khẩu  3. Click vào button Đăng nhập | Hiển thị thông báo “Đăng nhập thất bại” | Pass |
| DN\_4 | Đăng nhập để trống tên tài khoản | 1.Bỏ trống tên tài khoản  2. Nhập mật khẩu đúng  3. Click vào button Đăng nhập | Hiển thị thông báo “Đăng nhập thất bại” | Pass |
| DN\_5 | Đăng nhập để trống mật khẩu | 1.Nhập tên tài khoản đúng  2. Bỏ trống mật khẩu  3. Click vào button Đăng nhập | Hiển thị thông báo “Đăng nhập thất bại” | Pass |
| DN\_6 | Đăng nhập với tài khoản chưa được kích hoạt | 1.Nhập tên tài khoản và mật khẩu của tài khoản chưa được kích hoạt  2. Click button Đăng nhập | Hiển thị thông báo “Tài khoản chưa được kích hoạt. Vui lòng kiểm tra email để kích hoạt tài khoản” | fail |

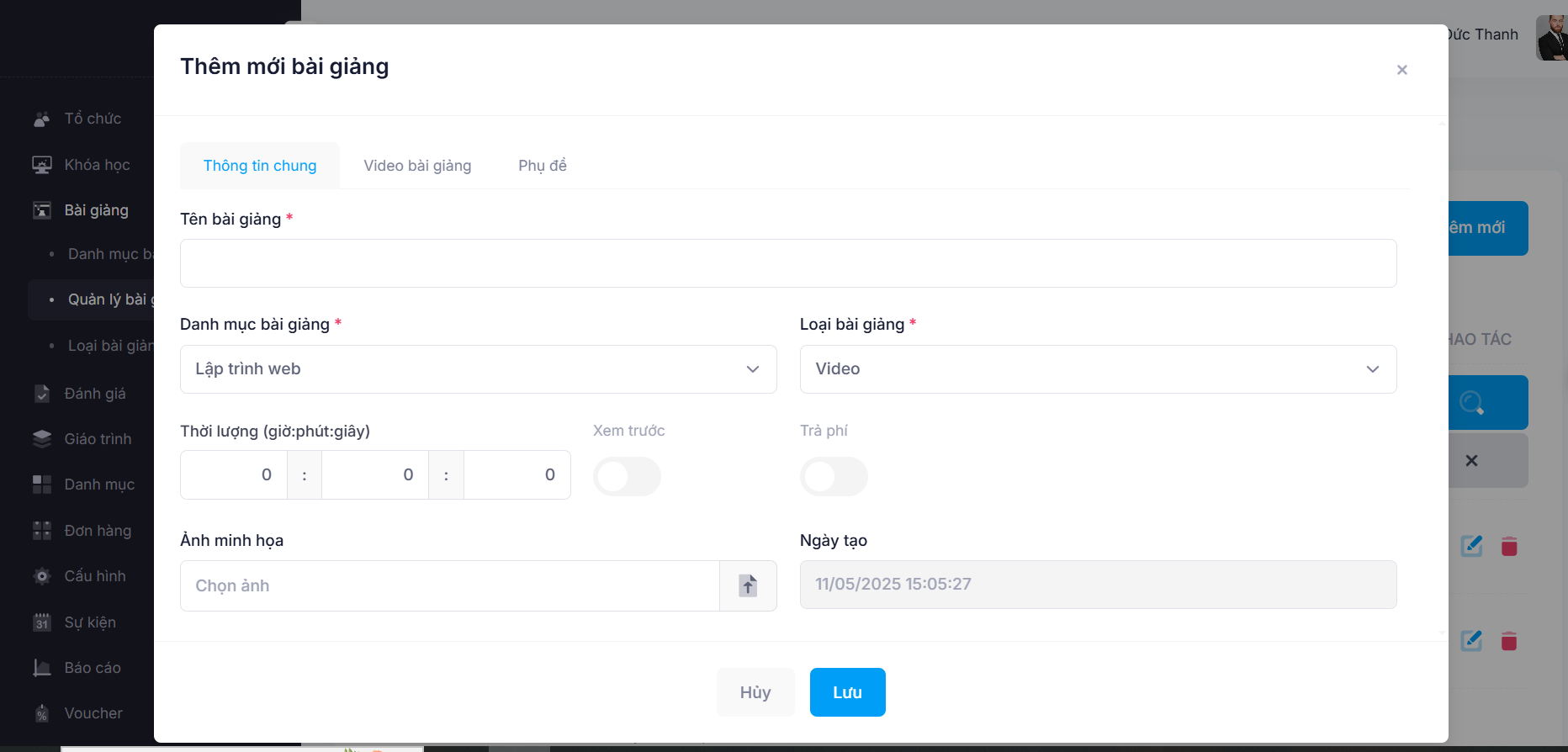
#### Quản lý bài giảng



Hình 3.5: Màn hình Danh sách bài giảng

* + - * 1. Thêm mới bài giảng

*Tiền điều kiện:* Click vào button “Thêm mới”



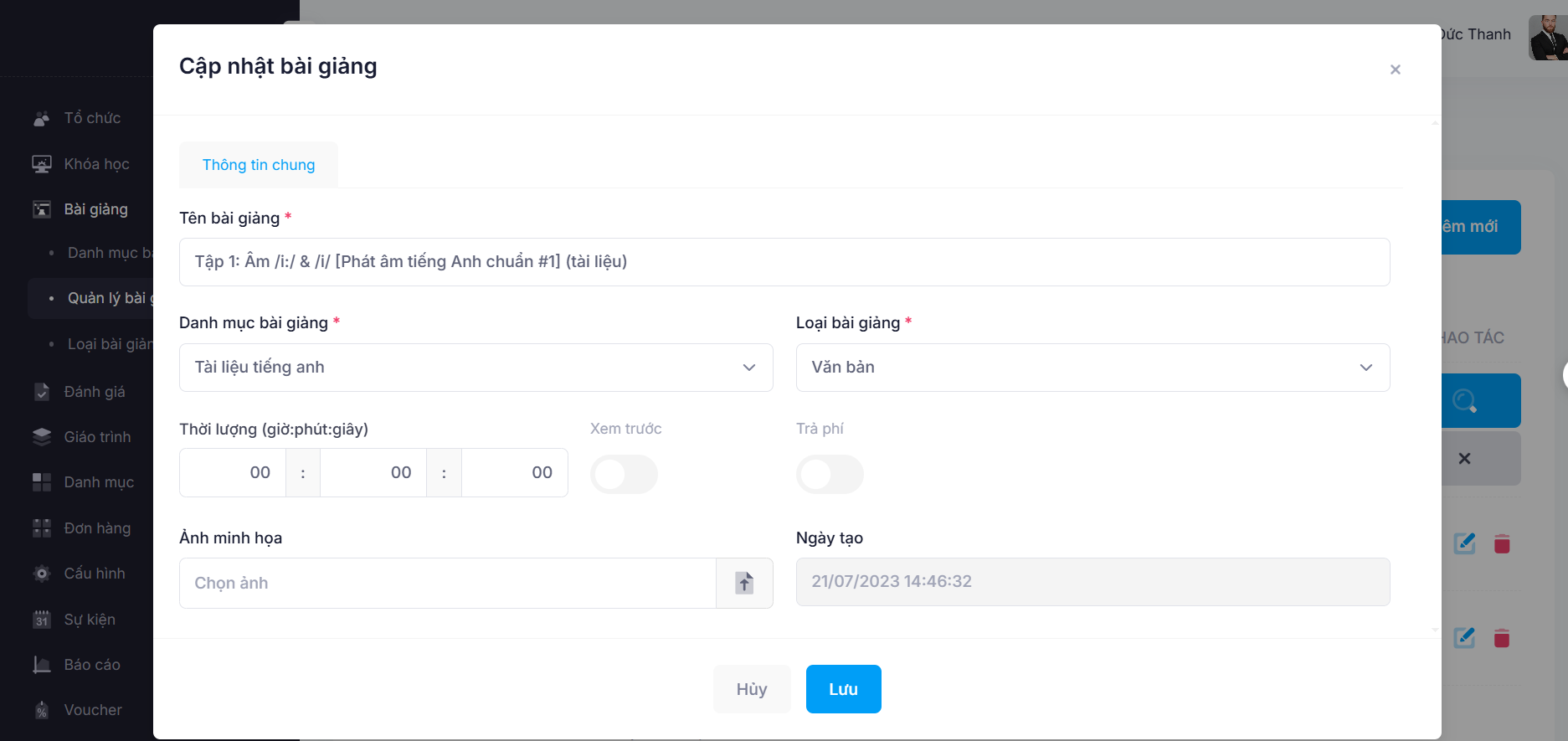
Hình 3.6: Màn hình Thêm mới bài giảng

Bảng 3.4: Test case chức năng Thêm bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| QLBG\_T\_1 | Thêm bài giảng khi nhập tất cả các trường | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập bài giảng vừa thêm mới > Enter | - Hiển thị thêm mới thành công và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_T\_2 | Thêm bài giảng khi chỉ nhập trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập bài giảng vừa thêm mới > Enter | - Hiển thị thông báo thêm mới thành công  và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_T\_3 | Thêm mới bài giảng khi chọn loại bài giảng là video | 1. Chọn loại bài giảng là video  2. Click vào video bài giảng > Chọn video  3.Nhập dữ liệu vào trường bắt buộc  4. Click vào lưu > lưu  5. Click tìm kiếm  6. Nhập bài giảng vừa thêm mới > Enter  7. Nhập loại bài giảng là video > Click icon tìm kiếm | Hiển thị thông báo thêm mới thành công  và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_T\_4 | Thêm mới bài giảng khi chọn loại bài giảng là audio | 1. Chọn loại bài giảng là audio  2.Nhập dữ liệu vào tất cả các trường còn lại  3. Click vào lưu > lưu  4. Click tìm kiếm  5. Nhập bài giảng vừa thêm mới > Enter  6. Nhập loại bài giảng là audio > Click icon tìm kiếm | Hiển thị thông báo thêm mới thành công  và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_T\_5 | Thêm bài giảng khi nhập trùng tên bài giảng và danh mục bài giảng đã tồn tại | 1.Nhập tên bài giảng và danh mục bài giảng đã tồn tại trong hệ thống  2. Nhập dữ liệu vào trường bắt buộc còn lại  3. Click Lưu > Lưu | Hiển thị thông báo “Bài giảng đã tồn tại” | Pass |
| QLBG\_T\_6 | Thêm bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc tên bài giảng  2. Click Lưu > Lưu | Hiển thị thông báo “Tên bài giảng không được để trống” | Pass |
| QLBG\_T\_7 | Kiểm tra click vào nút Hủy của popup thêm mới bài giảng | 1.Click vào nút Hủy | Hiển thị tên danh sách “Quản lý bài giảng” | Pass |
| QLBG\_T\_8 | Thêm bài giảng với file ảnh > 25MB | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Nhập file ảnh > 25MB | Hiển thị thông báo lỗi “Tệp mà bạn đã chọn quá lớn. Kích thước tối đa là 25 MB” | Pass |

* + - * 1. Cập nhật bài giảng

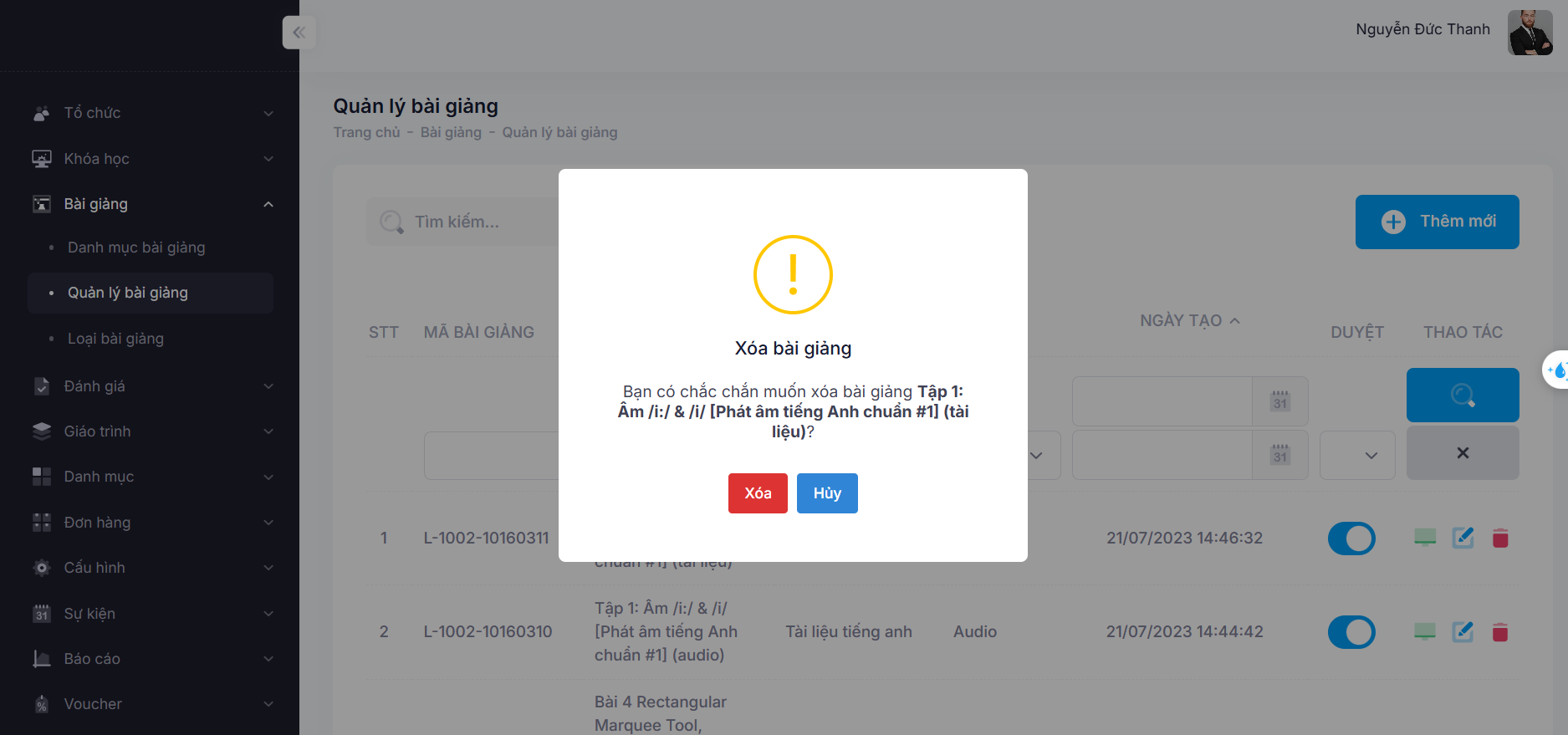
*Tiền điều kiện:* Click vào button “Cập nhật”



Hình 3.7: Màn hình Cập nhật bài giảng

Bảng 3.5: Test case chức năng Cập nhật bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| QLBG\_S\_1 | Kiểm tra cập nhật bài giảng hiển thị đúng dữ liệu | 1.Click vào button thêm mới  2. Nhập dữ liệu vào các trường bắt buôc  3. Lưu > Lưu  4. Click tìm kiếm  5. Nhập tên bài giảng vừa thêm mới > Enter  6. Click vào icon cập nhật trên dòng dữ liệu vừa thêm mới  7. So sánh dữ liệu vừa thêm mới và dữ liệu vừa ở màn cập nhật | Dữ liệu trong màn cập nhật hiển thị trùng khớp với dữ liệu vừa được thêm mới | fail |
| QLBG\_S\_2 | Sửa bài giảng khi nhập sửa tất cả các trường | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập bài giảng vừa sửa> Enter | - Hiển thị sửa thành công và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_S\_3 | Sửa bài giảng khi chỉ sửa trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập bài giảng vừa sửa > Enter | - Hiển thị sửa thành công và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_S\_4 | Sửa bài giảng khi chọn loại bài giảng là video | 1. Chọn loại bài giảng là video  2. Click vào video bài giảng > Chọn video  3.Nhập dữ liệu vào trường bắt buộc  4. Click vào lưu > lưu  5. Click tìm kiếm  6. Nhập bài giảng vừa sửa > Enter  7. Nhập loại bài giảng là video > Click icon tìm kiếm | Hiển thị sửa thành công và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_S\_5 | Sửa bài giảng khi chọn loại bài giảng là audio | 1. Chọn loại bài giảng là audio  2.Nhập dữ liệu vào tất cả các trường còn lại  3. Click vào lưu > lưu  4. Click tìm kiếm  5. Nhập tên bài giảng vừa sửa > Enter  6. Nhập loại bài giảng là audio > Click icon tìm kiếm | Hiển thị sửa thành công và hiển thị bài giảng trên danh sách | Pass |
| QLBG\_S\_6 | Sửa bài giảng khi nhập trùng tên bài giảng và danh mục bài giảng đã tồn tại | 1.Nhập tên bài giảng và danh mục bài giảng đã tồn tại trong hệ thống  2. Nhập dữ liệu vào trường bắt buộc còn lại  3. Click Lưu > Lưu | Hiển thị thông báo “Bài giảng đã tồn tại” | Pass |
| QLBG\_S\_7 | Sửa bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc tên bài giảng  2. Click Lưu > Lưu | Hiển thị thông báo “Tên bài giảng không được để trống” | Pass |
| QLBG\_S\_8 | Kiểm tra click vào nút Hủy của popup cập nhật bài giảng | 1.Click vào nút Hủy | Hiển thị tên danh sách “Quản lý bài giảng” | Pass |
| QLBG\_S\_9 | Sửa bài giảng với file ảnh > 25MB | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Nhập file ảnh > 25MB | Hiển thị thông báo lỗi “Tệp mà bạn đã chọn quá lớn. Kích thước tối đa là 25 MB” | Pass |

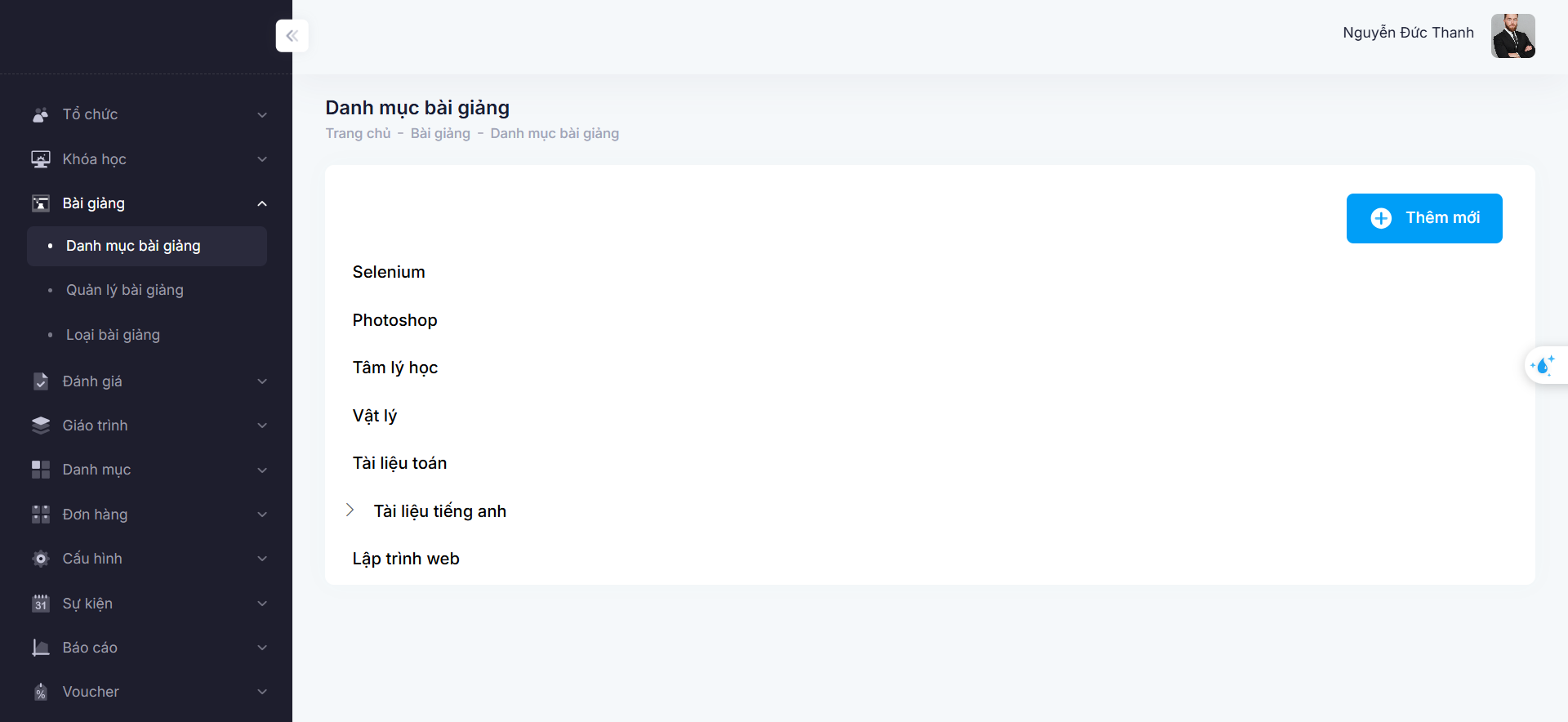
* + - * 1. Xóa bài giảng 

Hình 3.8: Màn hình Xóa bài giảng

Bảng 3.6: Test case chức năng Xóa bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| QLBG\_X\_1 | Xóa bài giảng thành công | 1. Click vào icon xóa  2. Click vào button xóa  3. Tìm kiếm tên bài giảng vừa xóa  4. So sánh tên bài giảng vừa xóa và tên bài giảng trên danh sách | Không tìm thấy tên bài giảng vừa xóa trên danh sách Hoặc hiển thị “Không tìm thấy kết quả” | Pass |
| QLBG\_X\_2 | Kiểm tra xóa bài giảng khi click vào nút “Hủy” | 1.Click vào icon Xóa  2. Click vào nút Hủy  3. Click vào nút tìm kiếm  4. Nhập tên bài giảng vừa chọn để xóa vào textbox > Enter | Hiển thị tên danh sách “Quản lý bài giảng” và bản ghi không bị xóa khỏi danh sách | Pass |

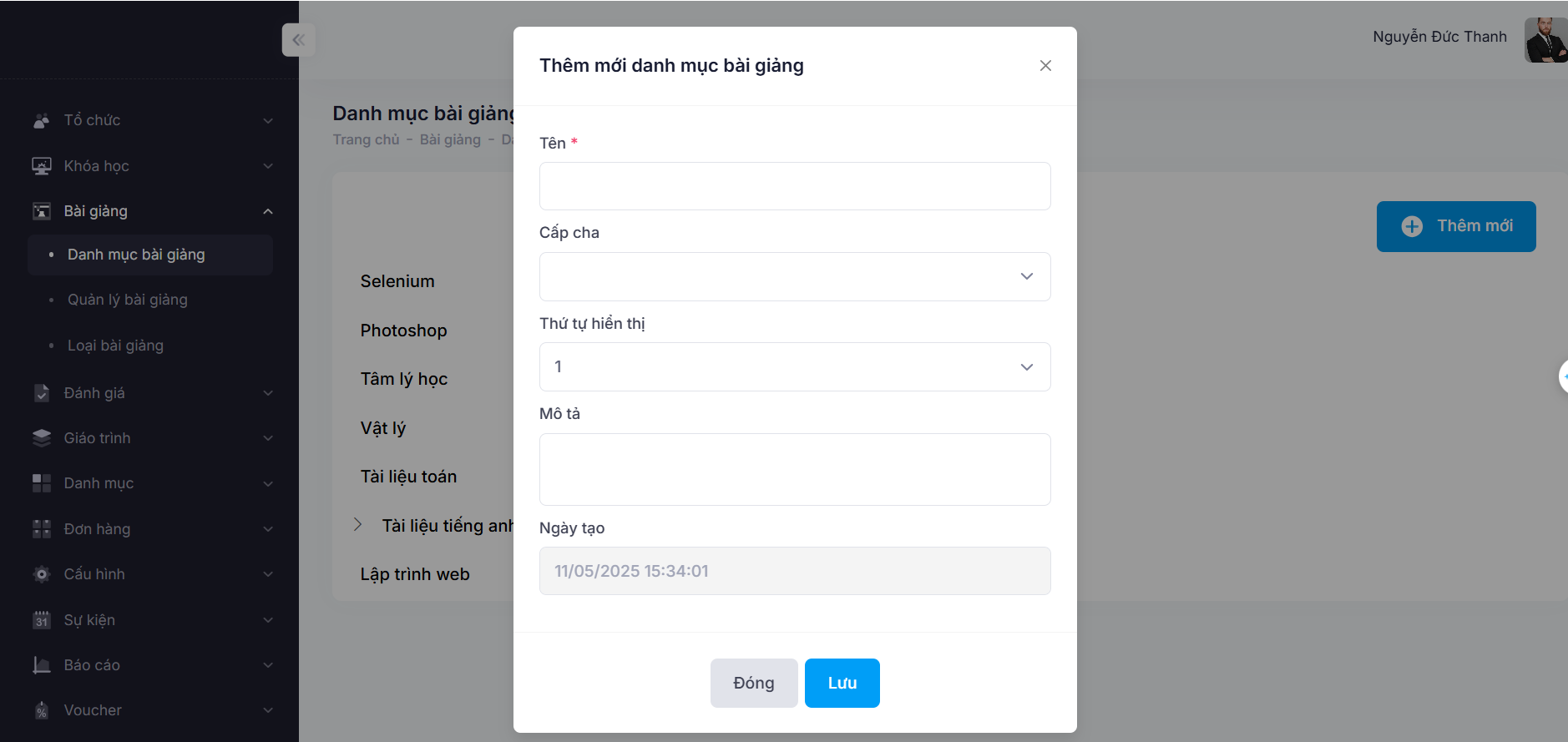
#### Danh mục bài giảng



Hình 3.9: Màn hình Danh mục bài giảng

* + - * 1. Thêm mới danh mục

*Tiền điều kiện:* Click vào button Thêm mới

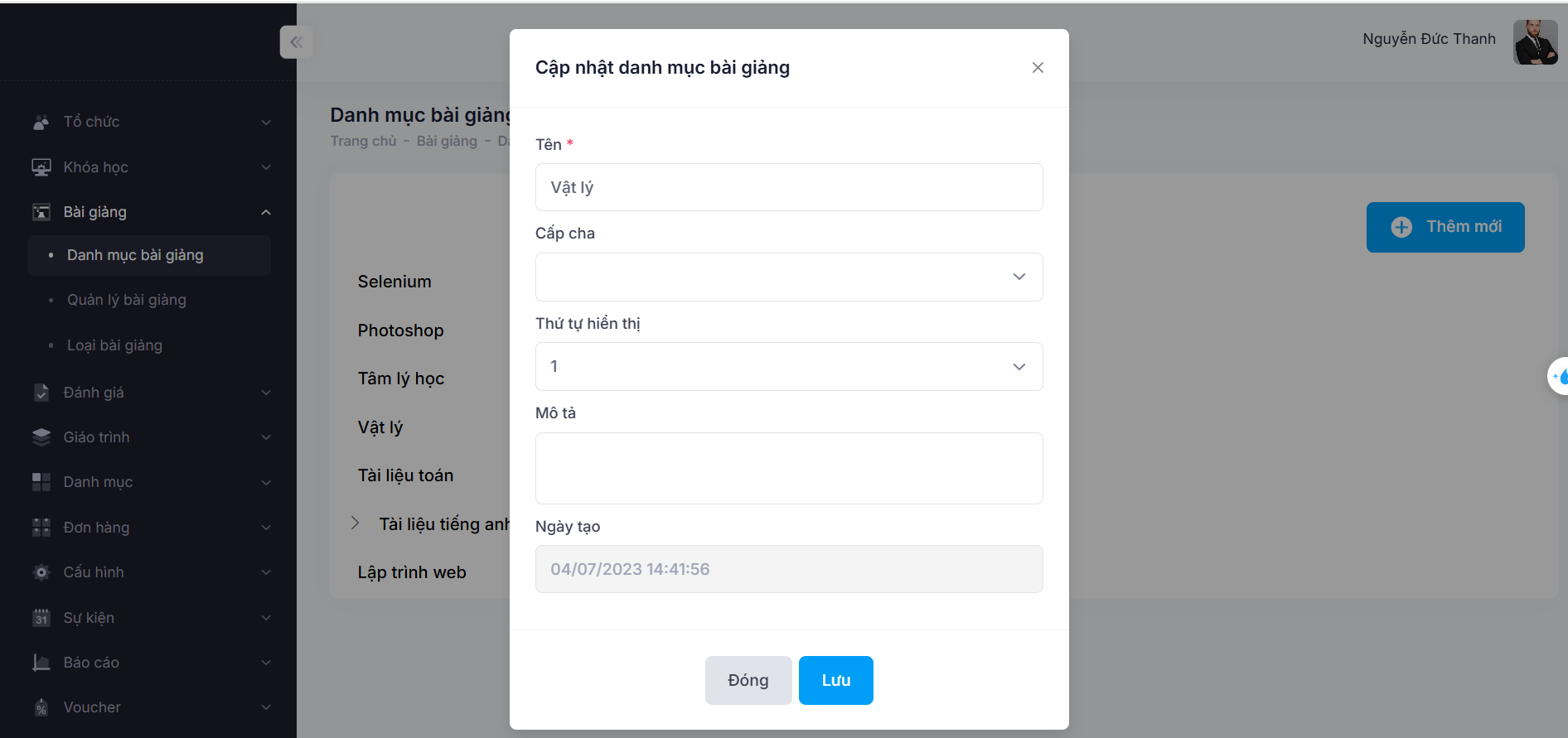


Hình 3.10: Màn hình Thêm mới danh mục bài giảng

Bảng 3.7: Test case chức năng Thêm danh mục bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| DMBG\_T\_1 | Thêm danh mục bài giảng khi nhập tất cả các trường | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. So sánh tên danh mục và thêm và tên danh mục trên danh sách | Hiển thị thêm mới thành công và tên danh mục trên danh sách trùng khớp với danh mục vừa thêm | Pass |
| DMBG\_T\_2 | Thêm danh mục bài giảng khi chỉ nhập trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. So sánh tên danh mục và thêm và tên danh mục trên danh sách | Hiển thị thêm mới thành công và tên danh mục trên danh sách trùng khớp với danh mục vừa thêm | Pass |
| DMBG\_T\_3 | Thêm danh mục bài giảng khi nhập trùng tên danh mục đã tồn tại | 1.Nhập trường tên đã tồn tại trong hệ thống  3. Click Lưu > Lưu | Hiển thị thông báo “Tên danh mục bài giảng đã tồn tại.  ” | Pass |
| DMBG\_T\_4 | Thêm danh mục bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc Tên  2. Click Lưu > Lưu | Hiển thị thông báo “Tên không được để trống” | Pass |
| DMBG\_T\_5 | Kiểm tra click vào nút Đóng của popup thêm mới danh mục bài giảng | 1.Click vào nút Đóng | Hiển thị tên danh sách: “Danh mục bài giảng” | Pass |

* + - * 1. Cập nhật danh mục bài giảng

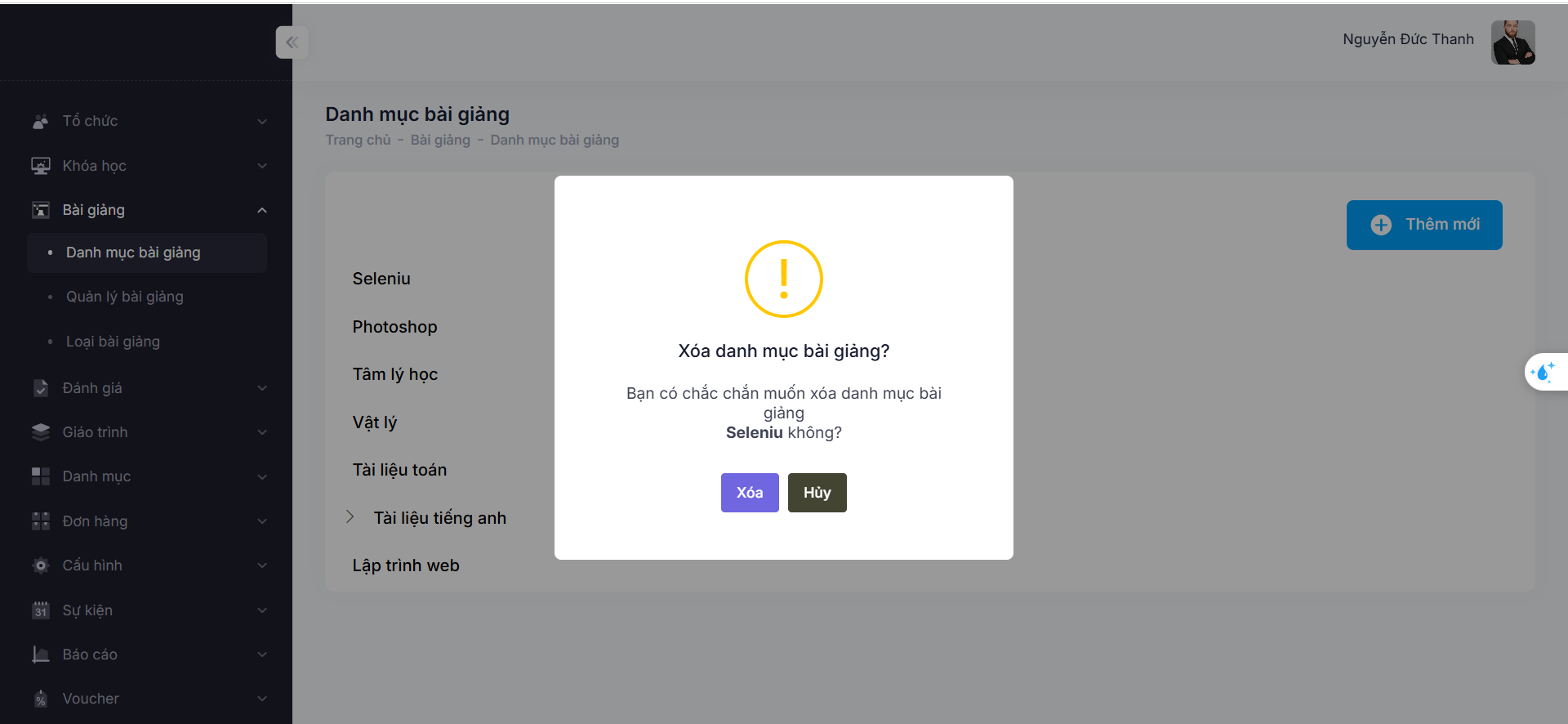


Hình 3.11: Màn hình Cập nhật danh mục bài giảng

Bảng 3.8: Test case chức năng Cập nhật danh mục bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| DMBG\_S\_1 | Sửa danh mục bài giảng khi nhập tất cả các trường | 1.Click vào tên danh mục muốn sửa  2. Nhập dữ liệu hợp lệ vào tất cả các trường  3. Click Lưu > Lưu  4. So sánh tên danh mục và thêm và tên danh mục trên danh sách | Hiển thị cập nhật thành công và hiển thị danh mục bài giảng vừa sửa trên danh sách | Pass |
| DMBG\_S\_2 | Sửa danh mục bài giảng khi chỉ nhập trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Click Lưu > Lưu  3. So sánh tên danh mục và thêm và tên danh mục trên danh sách | Hiển thị cập nhật thành công và hiển thị danh mục bài giảng vừa sửa trên danh sách | Pass |
| DMBG\_S\_3 | Sửa danh mục bài giảng khi nhập trùng tên danh mục đã tồn tại | 1.Nhập trường tên đã tồn tại trong hệ thống  3. Click Lưu > Lưu | Hiển thị thông báo “Tên danh mục bài giảng đã tồn tại.  ” | Pass |
| DMBG\_S\_4 | Sửa danh mục bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc Tên  2. Click Lưu > Lưu | Hiển thị thông báo “Tên không được để trống” | Pass |
| DMBG\_S\_5 | Kiểm tra click vào nút Đóng của popup Cập nhật danh mục bài giảng | 1.Click vào nút Đóng | Hiển thị tên danh sách: “Danh mục bài giảng” | Pass |

* + - * 1. Xóa danh mục bài giảng

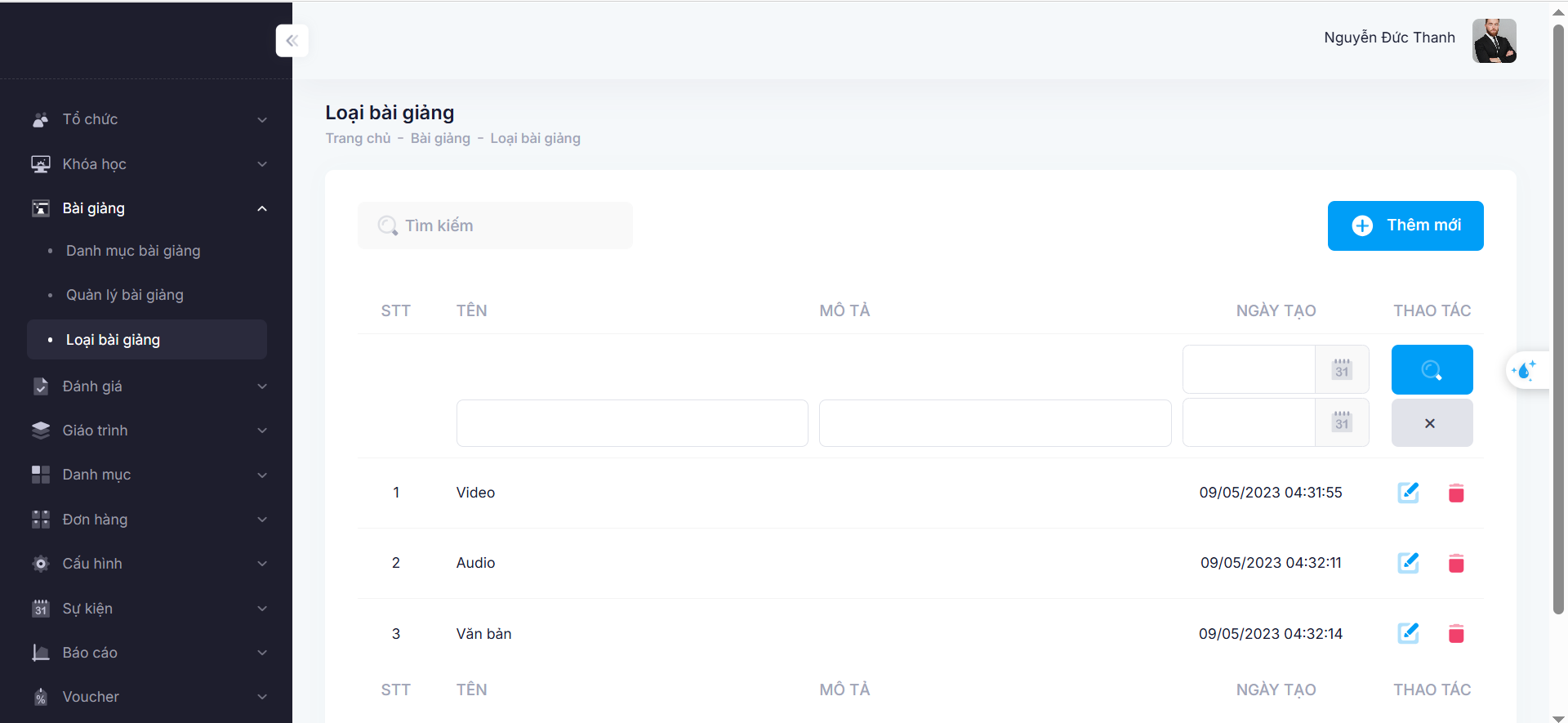


Hình 3.12: Màn hình Xóa danh mục bài giảng

Bảng 3.9: Test case chức năng Xóa danh mục bài giảng

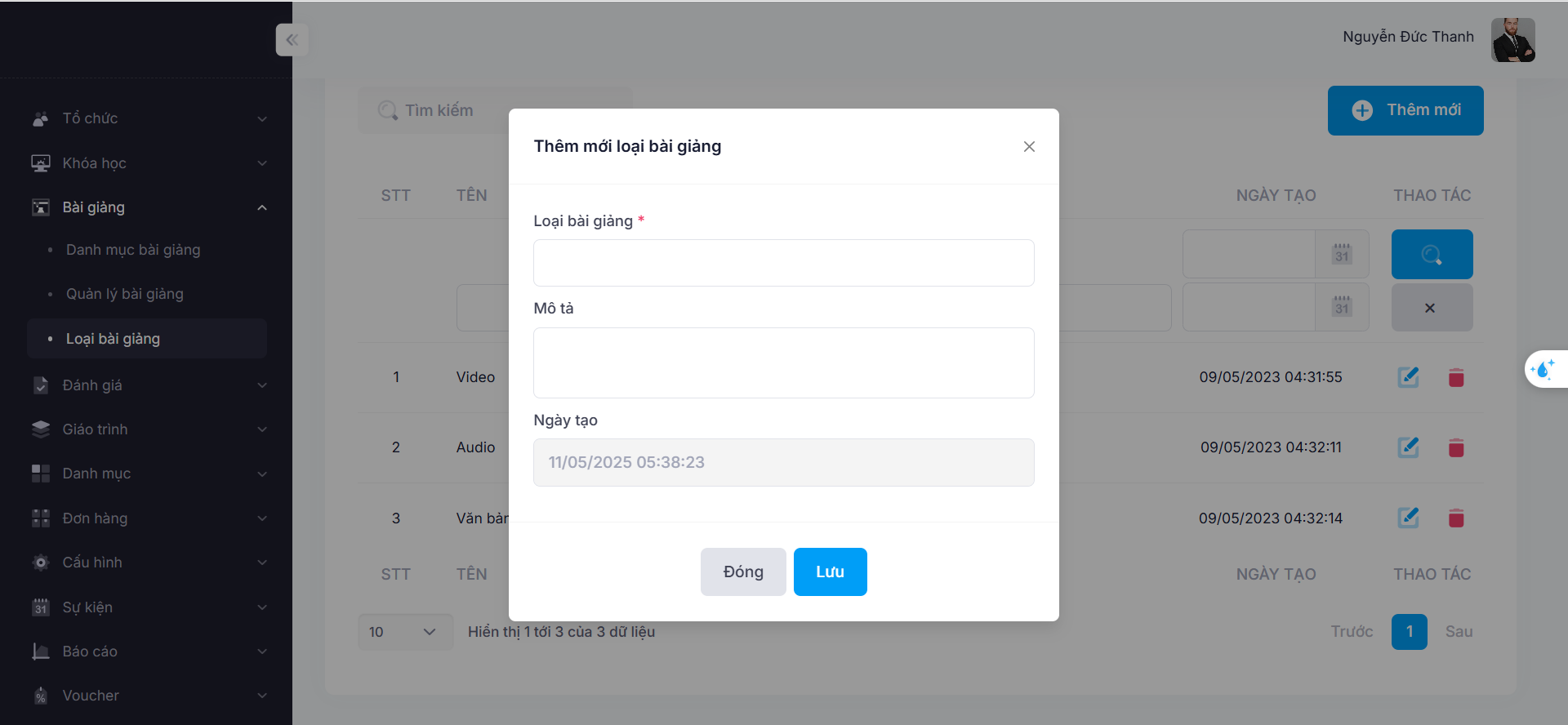
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| DMBG\_X\_1 | Xóa danh mục bài giảng cấp con thành công | 1. Click vào cấp cha  2. Click vào icon xóa của cấp con trong cấp cha  3. Click vào button xóa  4. Tìm kiếm tên bài giảng vừa xóa  5. So sánh tên bài giảng vừa xóa và tên bài giảng trên danh sách | Không tìm thấy tên bài giảng vừa xóa trên danh sách | Pass |
| DMBG\_X\_2 | Xóa danh mục bài giảng câp cha thành công | 1. Click vào icon xóa  2. Click vào button xóa  3. Tìm kiếm tên bài giảng vừa xóa  4. So sánh tên bài giảng vừa xóa và tên bài giảng trên danh sách | Không tìm thấy tên bài giảng vừa xóa trên danh sách | Pass |
| DMBG\_X\_3 | Kiểm tra xóa danh mục bài giảng khi click vào nút “Hủy” | 1.Click vào nút Hủy | Hiển thị tên danh sách “Danh mục bài giảng” | Pass |

#### Loại bài giảng



Hình 3.13: Màn hình Loại bài giảng

* + - * 1. Thêm loại bài giảng

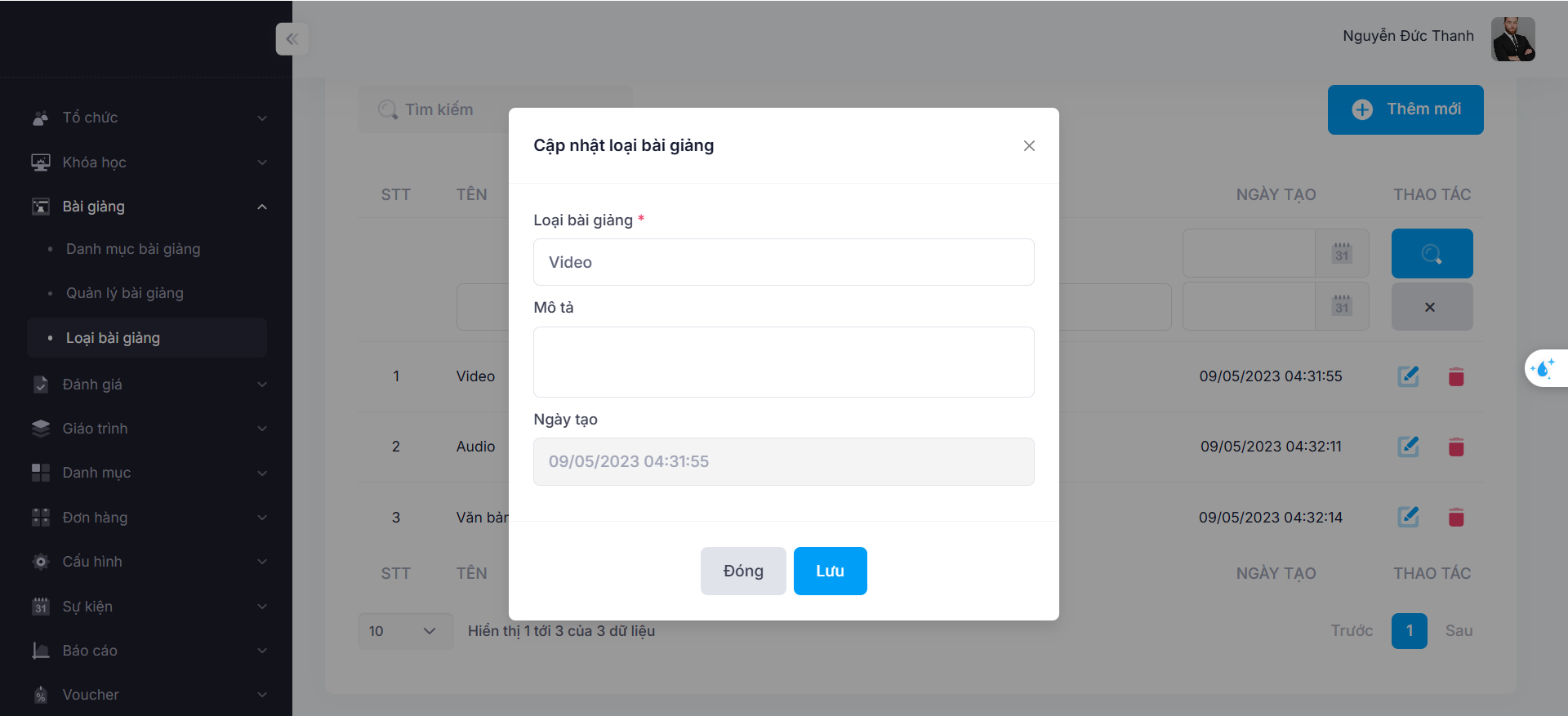


Hình 3.14: Màn hình Thêm mới loại bài giảng

Bảng 3.10: Test case chức năng Thêm loại bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| LBG\_T\_1 | Thêm loại bài giảng khi nhập tất cả các trường | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập loại bài giảng vừa thêm mới > Enter | Hiển thị thêm mới thành công và hiển thị loại bài giảng trên danh sách | Pass |
| LBG\_T\_2 | Thêm loại bài giảng khi chỉ nhập trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập loại bài giảng vừa thêm mới > Enter | Hiển thị thông báo thêm mới thành công  và hiển thị loại bài giảng trên danh sách | Pass |
| LBG\_T\_3 | Thêm loại bài giảng khi nhập trùng loại bài giảng đã tồn tại | 1.Nhập loại bài giảng đã tồn tại trong hệ thống  2. Nhập dữ liệu vào trường bắt buộc còn lại  3. Click Lưu > Lưu | Hiển thị thông báo “Loại bài giảng đã tồn tại” | Pass |
| LBG\_T\_4 | Thêm loại bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc Loại bài giảng  2. Click Lưu > Lưu | Hiển thị thông báo “Loại bài giảng không được để trống” | Pass |
| LBG\_T\_5 | Kiểm tra click vào nút Hủy của popup thêm mới bài giảng | 1.Click vào nút Hủy | Hiển thị tên danh sách “Loại bài giảng” |  |

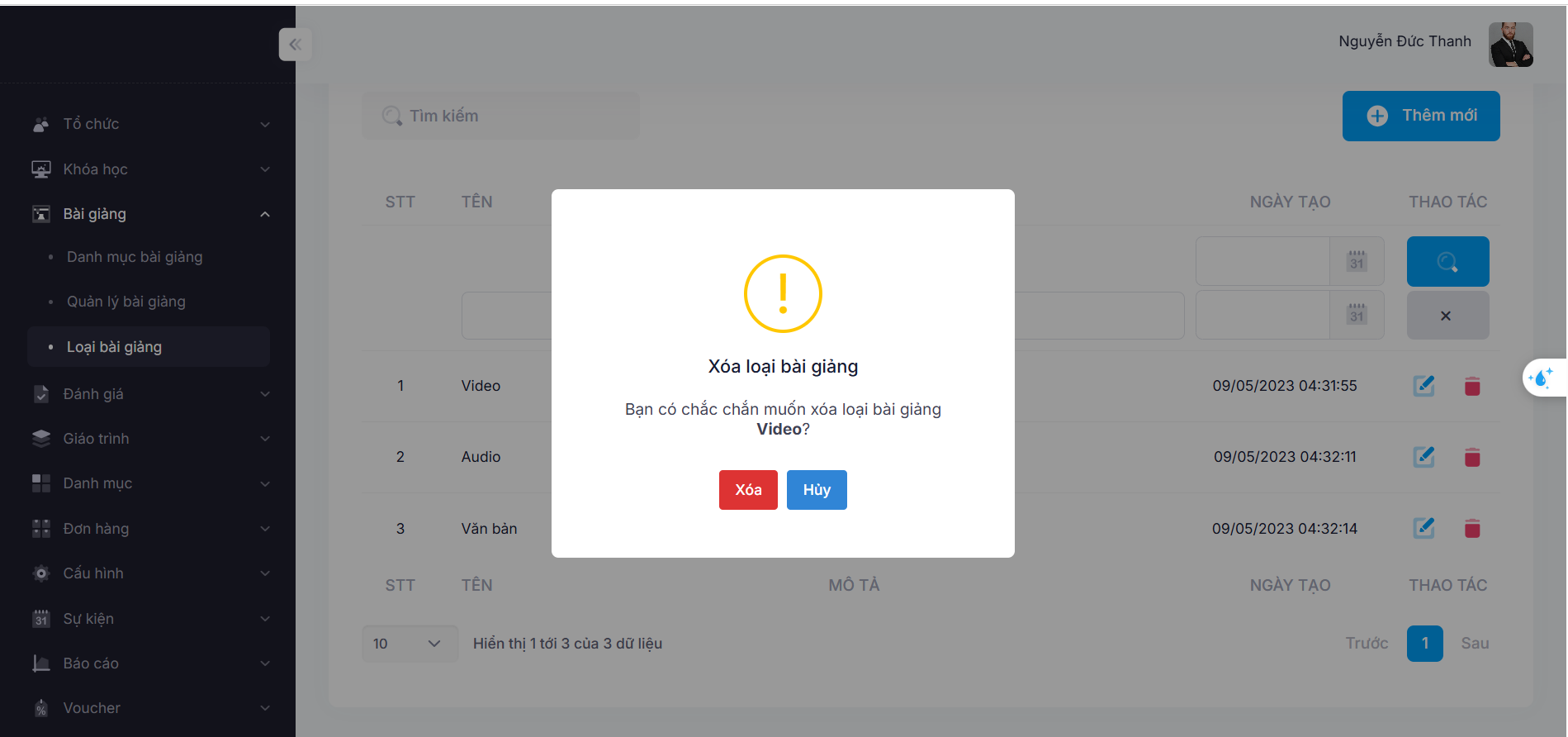
* + - * 1. Sửa loại bài giảng



Hình 3.15: Màn hình Cập nhật loại bài giảng

Bảng 3.11: Test case chức năng Cập nhật loại bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| LBG\_S\_1 | Kiểm tra cập nhật loại bài giảng hiển thị đúng dữ liệu | 1.Click vào button thêm mới  2. Nhập dữ liệu vào các trường bắt buôc  3. Lưu > Lưu  4. Click tìm kiếm  5. Nhập tên loại bài giảng vừa thêm mới > Enter  6. Click vào icon cập nhật trên dòng dữ liệu vừa thêm mới  7. So sánh dữ liệu vừa thêm mới và dữ liệu vừa ở màn cập nhật | Dữ liệu trong màn cập nhật hiển thị trùng khớp với dữ liệu vừa được thêm mới | Pass |
| LBG\_S\_2 | Sửa loại bài giảng khi nhập sửa tất cả các trường | 1.Nhập dữ liệu hợp lệ vào tất cả các trường  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập loại bài giảng vừa sửa> Enter | Hiển thị sửa thành công và hiển thị loại bài giảng trên danh sách | Pass |
| QLBG\_S\_3 | Sửa loại bài giảng khi chỉ sửa trường bắt buộc | 1.Nhập dữ liệu hợp lệ vào trường bắt buộc  2. Click Lưu > Lưu  3. Click tìm kiếm  4. Nhập loại bài giảng vừa sửa > Enter | - Hiển thị sửa thành công và hiển thị loại bài giảng trên danh sách | Pass |
| QLBG\_S\_4 | Sửa loại bài giảng khi nhập trùng loại bài giảng đã tồn tại | 1.Nhập loại bài giảng đã tồn tại trong hệ thống  2. Nhập dữ liệu vào trường bắt buộc còn lại  3. Click Lưu > Lưu | Hiển thị thông báo “Loại bài giảng đã tồn tại” | Pass |
| QLBG\_S\_5 | Sửa loại bài giảng khi không nhập trường bắt buộc | 1.Không nhập dữ liệu vào trường bắt buộc tên bài giảng  2. Click Lưu > Lưu | Hiển thị thông báo “Loại bài giảng không được để trống” | Pass |
| QLBG\_S\_6 | Kiểm tra click vào nút Đóng của popup cập nhật loại bài giảng | 1.Click vào nút Đóng | Hiển thị tên danh sách “Loại bài giảng” | Pass |

* + - * 1. Xóa loại bài giảng 

Hình 3.16: Màn hình Xóa loại bài giảng

Bảng 3.12: Test case chức năng Xóa loại bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| LBG\_X\_1 | Xóa bài giảng thành công | 1. Click vào icon xóa  2. Click vào button xóa  3. Tìm kiếm loại bài giảng vừa xóa  4. So sánh loại bài giảng vừa xóa và loại bài giảng trên danh sách | Không tìm thấy loại bài giảng vừa xóa trên danh sách Hoặc hiển thị “Không tìm thấy kết quả” | Pass |
| LBG\_X\_2 | Kiểm tra xóa loại bài giảng khi click vào nút “Hủy” | 1.Click vào icon Xóa  2. Click vào nút Hủy  3. Click vào nút tìm kiếm  4. Nhập loại bài giảng vừa chọn để xóa vào textbox > Enter | Hiển thị tên danh sách “Loại bài giảng” và loại bài giảng không bị xóa khỏi danh sách | Pass |

### Kiểm thử tương thích hệ thống

Bảng 3.13: Test case chức năng Xóa loại bài giảng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên test case** | **Các bước thực hiện** | **Kết quả mong muốn** | **Kết quả hiện tại** |
| KTTT\_1 | Kiểm thử toàn hệ thống trên Chrome + Windows 10 | 1. Truy cập http://learnico.id.vn  2. Đăng nhập với tài khoản đúng/sai  3. Thêm/sửa/xóa danh mục bài giảng  4. Thêm/sửa/xóa loại bài giảng  5. Thêm/sửa/xóa bài giảng  6. Quan sát giao diện, bố cục | Hệ thống hoạt động ổn định  Không lỗi chức năng  UI hiển thị đúng | Pass |
| KTTT\_2 | Kiểm thử toàn hệ thống trên Edge + Linux | 1. Truy cập http://learnico.id.vn  2. Đăng nhập với tài khoản đúng/sai  3. Thêm/sửa/xóa danh mục bài giảng  4. Thêm/sửa/xóa loại bài giảng  5. Thêm/sửa/xóa bài giảng  6. Quan sát giao diện, bố cục | Hệ thống hoạt động ổn định  Không lỗi chức năng  UI hiển thị đúng | Pass |
| KTTT\_3 | Kiểm thử toàn hệ thống trên Firefox + macOS Monterey | 1. Truy cập http://learnico.id.vn  2. Đăng nhập với tài khoản đúng/sai  3. Thêm/sửa/xóa danh mục bài giảng  4. Thêm/sửa/xóa loại bài giảng  5. Thêm/sửa/xóa bài giảng  6. Quan sát giao diện, bố cục | Hệ thống hoạt động ổn định  Không lỗi chức năng  UI hiển thị đúng | fail |

## Xây dựng test script

### Xây dựng test data

Trong quá trình kiểm thử tự động hệ thống học trực tuyến Learnico, việc xây dựng dữ liệu kiểm thử (test data) đóng vai trò quan trọng nhằm đảm bảo độ chính xác và khả năng tái sử dụng trong nhiều kịch bản kiểm thử khác nhau.

Để dễ dàng quản lý và mở rộng dữ liệu, việc sử dụng định dạng JSON (JavaScript Object Notation) để lưu trữ các bộ dữ liệu đầu vào cần thiết cho từng trường hợp kiểm thử. File test\_data.json:

{

**"danhmucbaigiang"**: {

**"themfull"**: {

**"ten"**: "Danh mục test 2",

**"capcha"**: "Vật lý",

**"mota"**: "Mô tả danh mục test 2"

},

**"suafull"**: {

**"ten"**: "Danh mục test sửa 2",

**"mota"**: "Mô tả danh mục test sửa"

},

**"thembatbuoc"**: {

**"ten"**: "Danh mục test"

},

**"suabatbuoc"**: {

**"ten"**: "Danh mục test sửa"

},

**"fail"**: {

**"ten"**: "Vật lý"

}

},

**"loaibaigiang"**: {

**"add"**: {

**"ten"**: "Loại bài giảng test",

**"mota"**: "Mô tả loại bài giảng test"

},

**"edit"**: {

**"ten"**: "Loại bài giảng test sửa",

**"mota"**: "Mô tả loại bài giảng test sửa"

}

},

**"quanlybaigiang"**: {

**"add"**: {

**"tenbg"**: "Selenium",

**"mota"**: "Kiểm thử tự động"

},

**"edit"**: {

**"tenbg"**: "Kiểm thử phần mềm",

**"lbg"**: "video",

**"dmbg"**: "Vật lý",

**"mota"**: "Kiểm thử"

}

},

**"login"**: {

**"pass"**: {

**"username"**: "admin",

**"password"**: "123456"

},

**"fail"**: {

**"username"**: "user",

**"password"**: "1234567"

},

**"chuakichhoat"**: {

**"username"**: "Chi",

**"password"**: "123456"

}

}

}

### Xây dựng script bộ testcase

Tạo folder Testcase để lưu trữ toàn bộ các testcase của từng màn hình

#### Test case màn Đăng nhập

Ví dụ với case: Login thành công

**def** DN\_1(driver,p\_username, p\_password):

driver.find\_element(By.XPATH, "//input[@id='username']").clear()

driver.find\_element(By.XPATH, "//input[@id='password']").clear()

driver.find\_element(By.XPATH,

"//input[@id='username']").send\_keys(p\_username)

driver.find\_element(By.XPATH,

"//input[@id='password']").send\_keys(p\_password)

driver.find\_element(By.XPATH, "//button[@id='btnLogin']").click()

*# time.sleep(3)*

time.sleep(2)

#### Test case màn Quản lý bài giảng

Ví dụ với case: Thêm quản lý bài giảng thành công

**def** Addquanlybaigiang(driver, Ttenbaigiang,Stenbb,Sloaibaigiang, tmotaqlbg):

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//button[@id='btnAdd']"))).click()

time.sleep(2)

driver.find\_element(By.XPATH,

"//input[@id='lectureTitle']").send\_keys(Ttenbaigiang)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//label[contains(text(),'Danh mục bài giảng')]//../span"))).click()

driver.find\_element(By.XPATH, "//input[@class='select2-

search\_\_field']").send\_keys(Stenbb)

driver.find\_element(By.XPATH, "//input[@class='select2-

search\_\_field']").send\_keys(Keys.ENTER)

time.sleep(1)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//label[contains(text(),'Loại bài giảng')]/../span"))).click()

driver.find\_element(By.XPATH, "//input[@class='select2-

search\_\_field']").send\_keys(Sloaibaigiang)

driver.find\_element(By.XPATH, "//input[@class='select2-

search\_\_field']").send\_keys(Keys.ENTER)

driver.find\_element(By.XPATH,

"//input[@id='lecturePreview']").click()

driver.find\_element(By.XPATH,

"//input[@id='lectureIsPayment']").click()

price = 300000

WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.XPATH,

"//input[@id='lecturePrice']"))).clear()

WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.XPATH,

"//input[@id='lecturePrice']"))).send\_keys(price)

time.sleep(2)

driver.find\_element(By.XPATH, "//label[contains(text(),'Ảnh minh

họa')]/..//span").click()

time.sleep(2)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//button[@id='btnUploadFiles']"))).click()

time.sleep(2)

folder = r"D:\DATN\Anh"

filename = "Toan.jpg"

full\_path = os.path.join(folder, filename)

pyautogui.write(full\_path)

time.sleep(2)

pyautogui.press('enter')

time.sleep(2)

driver.find\_element(By.XPATH, "(//div[@id = 'file-explorer-

content']/div/div/div)[2]").click()

driver.find\_element(By.XPATH,

"//button[@id='btnChooseImage']").click()

time.sleep(2)

driver.find\_element(By.XPATH,

"//textarea[@id='lectureDescription']").send\_keys(tmotaqlbg)

driver.find\_element(By.XPATH, "//button[@id='saveData']").click()

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//button[contains(text(),'Lưu')]"))).click()

time.sleep(2)

#### Test case màn Danh mục bài giảng

Ví dụ case: Thêm danh mục bài giảng thành công

**def** DMBG\_T\_1 (driver,Tten,Tcapcha, Tmota ):

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable(

(By.XPATH, "(//h1[contains(text(),'Danh mục bài

giảng')]/../../../..//button)[1]"))).click()

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//input[@id='name']"))).send\_keys(Tten)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"(//h4[contains(text(),'Thêm mới danh mục bài giảng ')]/../..//div[2]//div[2]//span)[1]"))).click()

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//input[@type = 'search']"))).send\_keys(Tcapcha)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//input[@type = 'search']"))).send\_keys(Keys.ENTER)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//textarea[@id='description']"))).send\_keys(

Tmota)

driver.find\_element(By.XPATH,

"//button[@id='btnUpdateItem']").click()

driver.find\_element(By.XPATH,

"(//button[contains(text(),'Lưu')])[2]").click()

time.sleep(2)

#### Testcase màn Loại bài giảng

Ví dụ case: Thêm loại bài giảng thành công

**def** Addloaibaigiang(driver, Tloaibaigiang, Tmotalbg):

driver.find\_element(By.XPATH, "(//h1[contains(text(),'Loại bài

giảng')]/../../../..//button)[1]").click()

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.ID,

"name"))).send\_keys(Tloaibaigiang)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.ID,

"description"))).send\_keys(Tmotalbg)

driver.find\_element(By.ID, "submitButton").click()

driver.find\_element(By.XPATH,

"(//button[contains(text(),'Lưu')])[2]").click()

### Xây dựng test script toàn bộ module

Xây dựng script chạy toàn bộ test case liên tục cho từng màn và cả folder

#### Xây dựng test script cho folder Bài giảng

Xây dựng script chạy toàn bộ folder:

**def** baigiang(driver, tentrung, Tten,Tcapcha, Tmota,Stendmbg, Smotadmbg,

Ttenbb,Stenbb, Tloaibaigiang, Tmotalbg, Sloaibaigiang, Smotalbg, Ttenbaigiang, tmotaqlbg, stenbaigiang, sdmbg, Slbg, smotaqlbg,f\_username, p\_password, p\_username, f\_password,c\_username, c\_password,video\_path):

name\_module = "baigiang"

DangNhap(driver, f\_username, p\_password, p\_username, f\_password,

c\_username, c\_password, video\_path, name\_module)

Danhmucbaigiang(driver, tentrung, Tten,Tcapcha, Tmota,Stendmbg,

Smotadmbg, Ttenbb,Stenbb, video\_path)

time.sleep(3)

WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.XPATH,

"(//span[contains(text(),'Loại bài giảng')])[1]"))).click()

Loaibaigiang(driver, Tloaibaigiang, Tmotalbg, Sloaibaigiang,

Smotalbg, video\_path)

WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.XPATH,

"(//span[contains(text(),'Bài giảng')])[1]"))).click()

Quanlybaigiang(driver, Ttenbaigiang, Stenbb, Sloaibaigiang,

tmotaqlbg, stenbaigiang, sdmbg, Slbg, smotaqlbg, video\_path)

#### Xây dựng test script cho module Đăng nhập

Ví dụ case: Đăng nhập thành công

**def** DangNhap(driver, p\_password, p\_username, name\_module):

*# Test case Đăng nhập thành công*

DN\_1(driver,p\_username, p\_password)

b = WebDriverWait(driver, 5).until(

EC.presence\_of\_element\_located((By.XPATH,

"//h1[contains(text(),'Thống kê')]"))

)

**if** b.is\_displayed():

**print**("Pass: Đăng nhập thành công")

*#Mảng chứa tên test case, trạng thái, link video*

save\_test\_result("Dang\_nhap\_thanh\_cong", "pass", video\_path)

**else**:

**print**("Fail: Xem lại chức năng đăng nhập")

*#Chụp ảnh khi lỗi*

screenshot = take\_screenshot(driver, "DN\_1.png", name\_module)

*# Mảng chứa tên test case, trạng thái, link video, ảnh fail*

save\_test\_result("Dang\_nhap\_thanh\_cong", "fail", video\_path,

screenshot)

#### Xây dựng test script cho module Quản lý bài giảng

Ví dụ case: Thêm quản lý bài giảng thành công

**def** Quanlybaigiang(driver, Ttenbaigiang, Stenbb,Sloaibaigiang,tmotaqlbg):

Addquanlybaigiang(driver, Ttenbaigiang,Stenbb,Sloaibaigiang,

tmotaqlbg)

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH,

"//input[@id='table\_search\_all']"))).send\_keys(Ttenbaigiang)

driver.find\_element(By.XPATH,

"//input[@id='table\_search\_all']").send\_keys(Keys.ENTER)

k = driver.find\_elements(By.XPATH,

'//table[@id="tableData"]//tbody//tr')

j = len(k)

v\_kq= 0

**if** (j > 0):

time.sleep(2)

v\_test = driver.find\_element(By.XPATH, "((//table[@id='tableData']//tbody//tr//td)[3]//span)[1]").text

**print**(v\_test)

**if** v\_test == Ttenbaigiang:

**print**("Pass: Thêm thành công")

save\_test\_result("Thêm bài giảng", "pass", video\_path)

**else**:

screenshot =take\_screenshot(driver, "add\_quanlybaigiang.png",

name\_module)

**print**("Failse: Xem lại chức năng thêm")

save\_test\_result("Thêm bài giảng", "fail", video\_path,

screensh)

#### Xây dựng test script cho module Danh mục bài giảng

Ví dụ case: Thêm danh mục bài giảng thành công

**def** Danhmucbaigiang(driver, Tten,Tcapcha, Tmota):

DMBG\_T\_1 (driver,Tten,Tcapcha, Tmota )

WebDriverWait(driver, 5).until(

EC.presence\_of\_element\_located((By.XPATH,

"//span[contains(text(),'Vật lý')]/../..//i"))

).click()

v\_test = driver.find\_element(By.XPATH, "//span[contains(text(),'Vật

lý')]/../..//li").text

time.sleep(1)

**if** v\_test == Tten:

**print**("Pass: Thêm thành công")

save\_test\_result("Them\_thanh\_cong\_danh\_muc\_bai\_giang", "pass",

video\_path)

**else**:

**print**("Failse: Xem lại chức năng thêm")

screenshot = take\_screenshot(driver, "DMBG\_T\_1.png", name\_module)

save\_test\_result("Them\_thanh\_cong\_danh\_muc\_bai\_giang", "fail",

video\_path, screenshot)

#### Xây dựng test script cho module Loại bài giảng

Ví dụ case: Thêm loại bài giảng thành công

**def** Loaibaigiang(driver, Tloaibaigiang, Tmotalbg ):

Addloaibaigiang(driver, Tloaibaigiang, Tmotalbg)

search\_box = WebDriverWait(driver,

10).until(EC.element\_to\_be\_clickable((By.ID,

"table\_search\_all")))

search\_box.clear()

search\_box.send\_keys(Tloaibaigiang)

search\_box.send\_keys(Keys.ENTER)

rows = driver.find\_elements(By.XPATH,

'//table[@id="tableData"]//tbody//tr//td')

**if** len(rows) > 1:

**print**("Pass: Thêm mới thành công")

save\_test\_result("Thêm loại bài giảng", "pass", video\_path)

**else**:

screenshot = take\_screenshot(driver, "e\_add\_loaibaigiang.png",

name\_module)

**print**("Fail: Xem lại chức năng thêm mới")

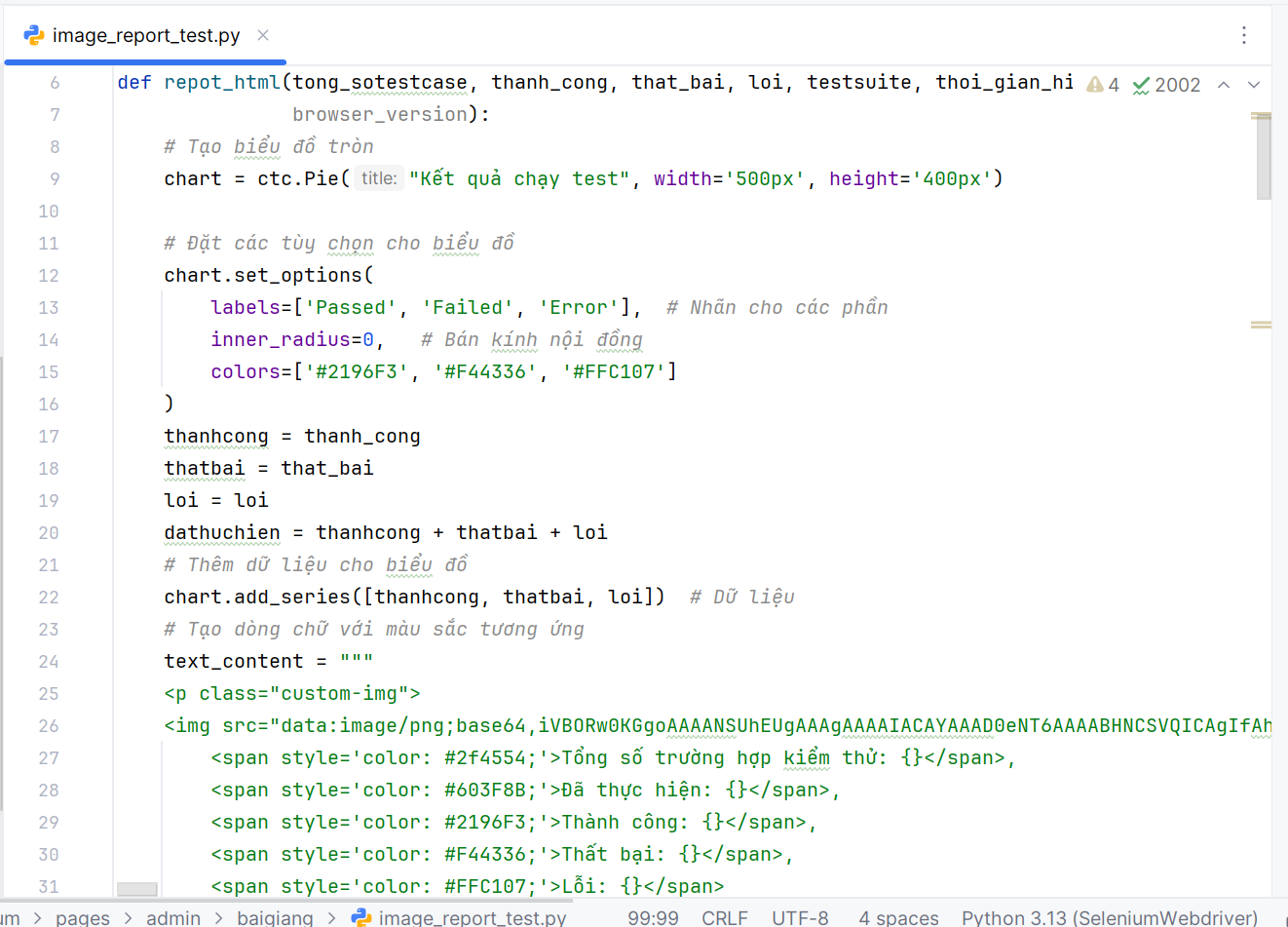
save\_test\_result("Thêm loại bài giảng", "fail", video\_path,

screenshot)

### Xây dựng báo cáo HTML

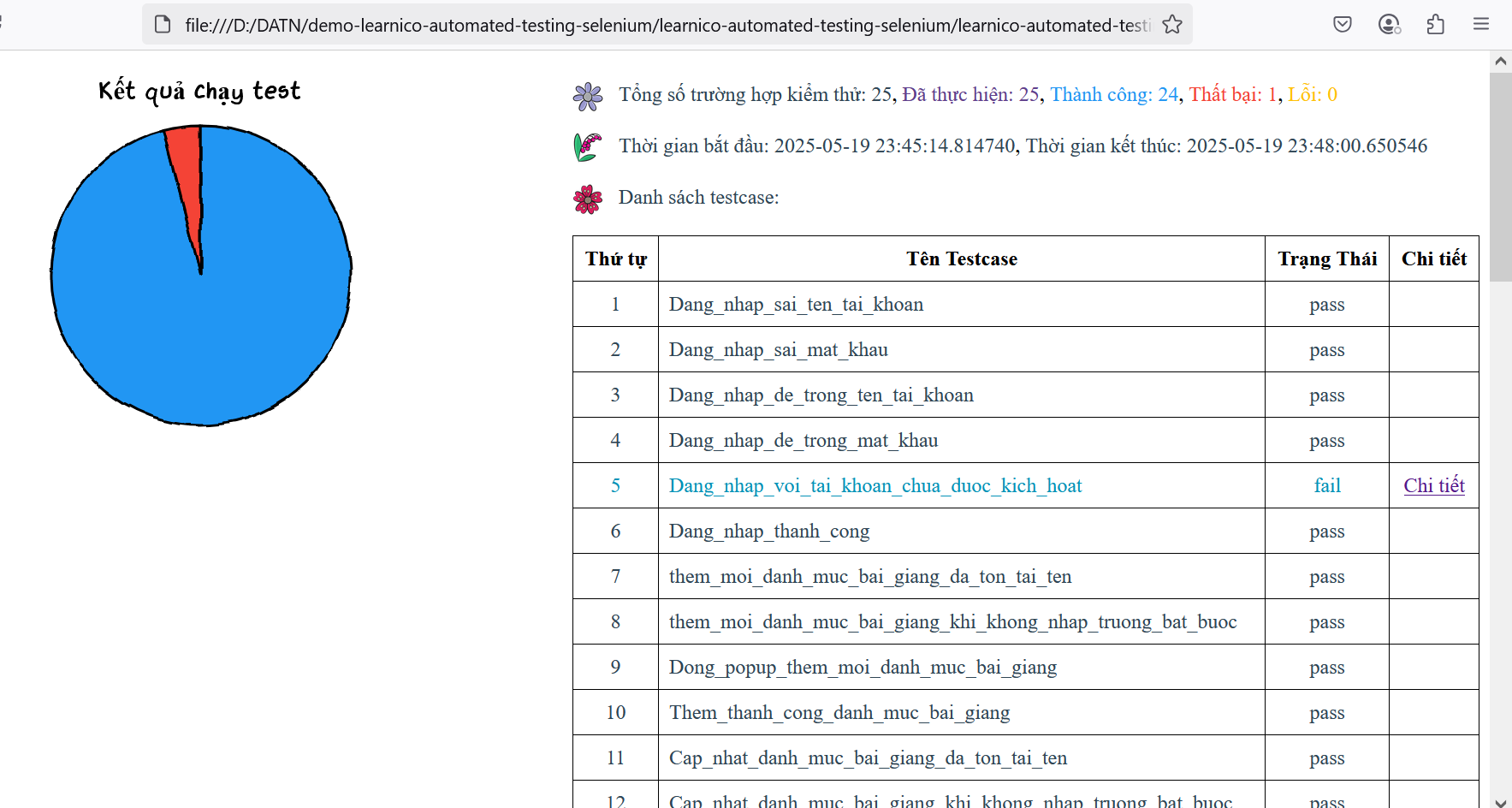
Báo cáo kiểm thử được xây dựng dưới dạng giao diện HTML trực quan, nhằm hỗ trợ việc theo dõi, phân tích và đánh giá kết quả kiểm thử một cách dễ dàng. Nội dung báo cáo bao gồm các thành phần chính như sau:

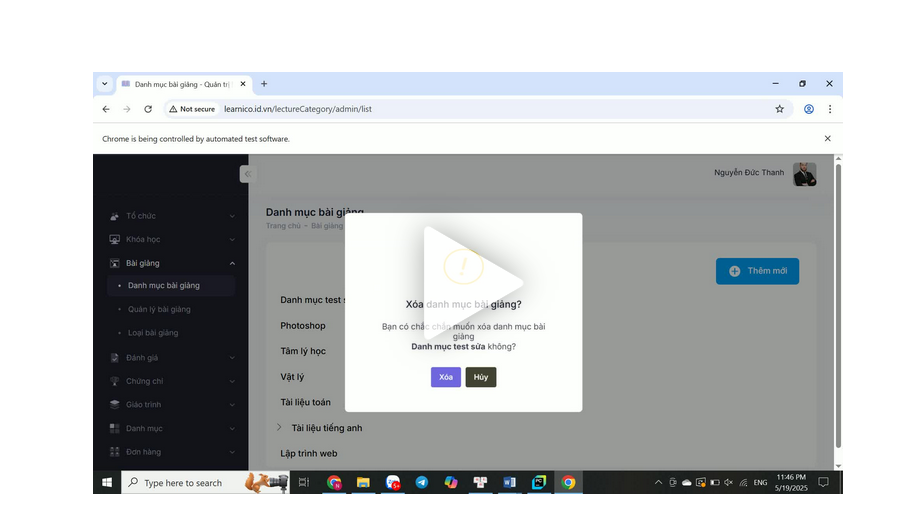
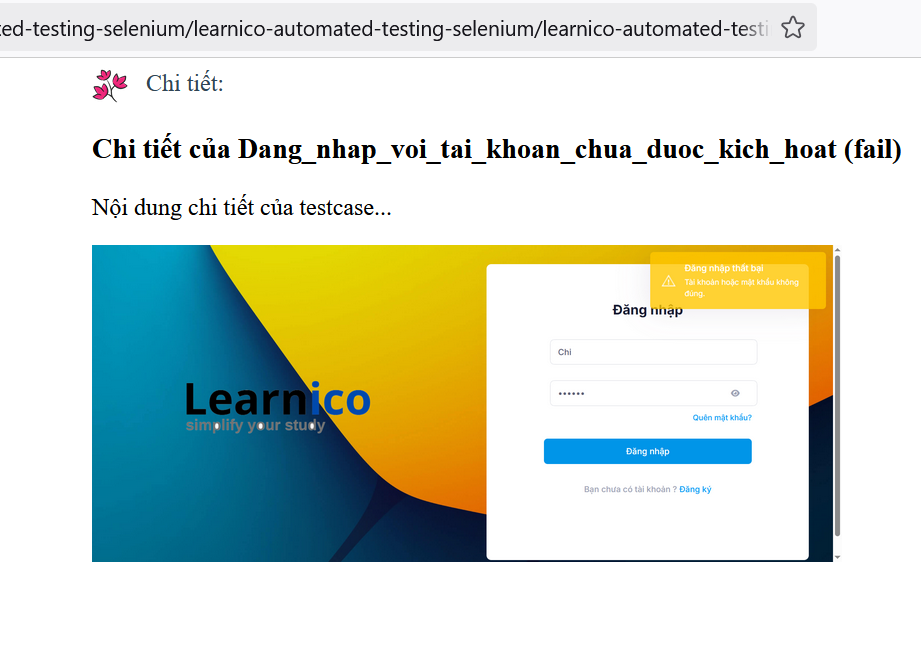
* Biểu đồ thống kê: Tổng hợp số lượng test case đạt (Pass), không đạt (Fail) và gặp lỗi (Error) dưới dạng biểu đồ tròn, giúp đánh giá nhanh hiệu quả kiểm thử.
* Thời gian kiểm thử: Ghi nhận thời điểm bắt đầu và kết thúc quá trình kiểm thử
* Danh sách test case: Hiển thị bảng thông tin gồm tên test case, trạng thái kết quả (Pass/Fail/Error), click chi tiết hiển thị gồm ảnh và video nếu là testcase fail
* Ảnh chụp lỗi: Với các test case thất bại hoặc lỗi, báo cáo đính kèm ảnh chụp màn hình tại thời điểm xảy ra sự cố để hỗ trợ việc phân tích nguyên nhân.
* Video quay màn hình: Ngoài ảnh chụp, các test case không thành công còn được ghi lại video toàn bộ quá trình thực thi để phục vụ việc debug hiệu quả hơn.



Hình 3.17: Hàm xây dựng báo cáo html

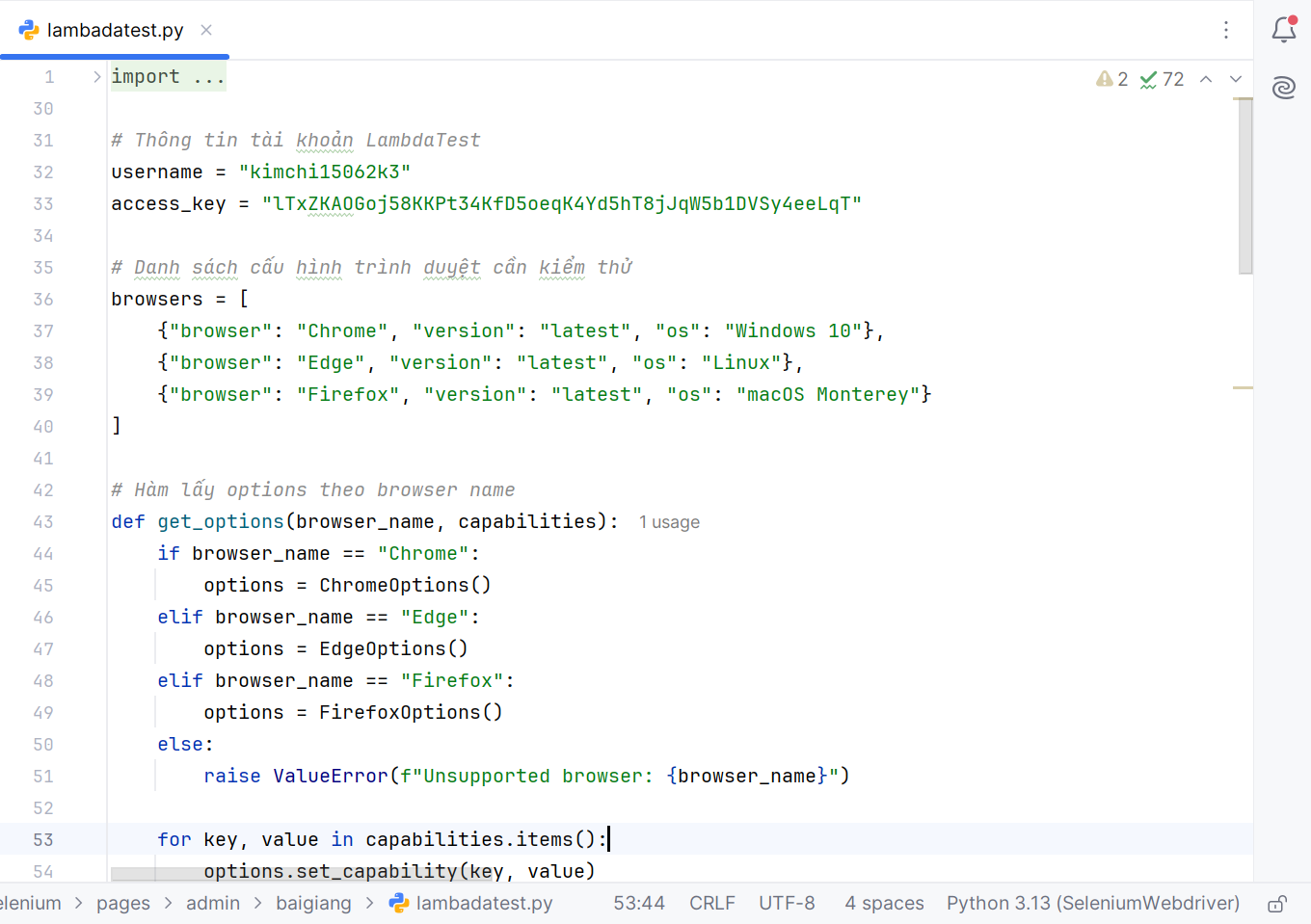
Sau khi chạy file bài giảng báo cáo sẽ hiển thị trong file image\_report\_test. Mở file trên trình duyệt để xem báo cáo thống kê.



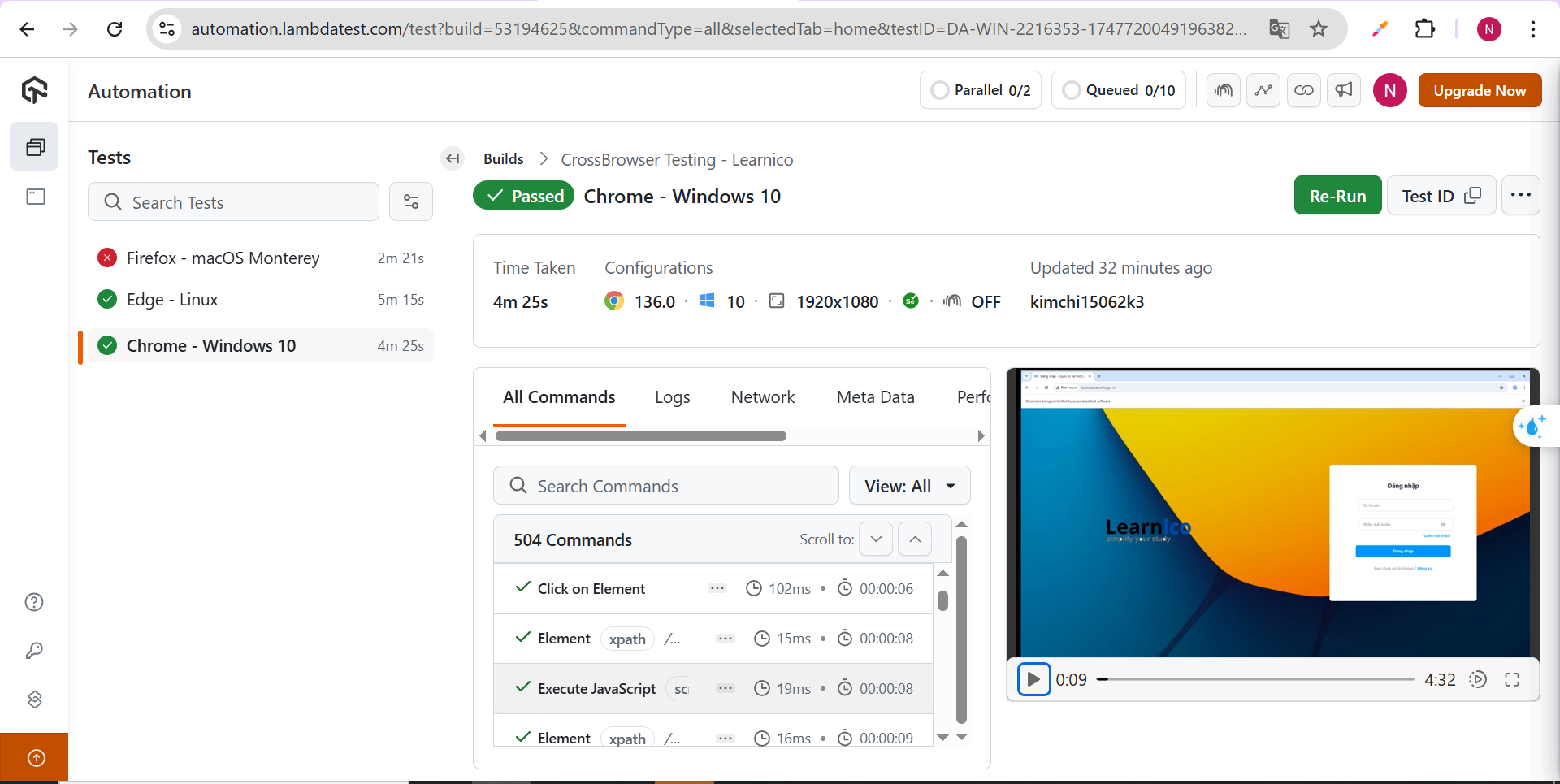


Hình 3.18: Kết quả test được hiển thị lên file html

### Xây dựng script cho kiểm thử tương thích sử dụng lambdatest



Hình 3.19: Xây dựng script cho kiểm thử tương thích sử dụng lambdatest



Hình 3.20: Kết quả sau khi chạy testcase kiểm thử tương thích trên lambdatest

## Kết quả kiểm thử

Bảng 3.14: Bảng tổng hợp kết quả kiểm thử chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Tổng số TC** | **Số TC pass** | **Số TC fail** |
| Đăng Nhập | 6 | 5 | 1 |
| Quản lý bài giảng | 20 | 19 | 1 |
| Danh mục bài giảng | 13 | 13 | 0 |
| Loại bài giảng | 13 | 13 | 0 |
| **TỔNG** | **52** | **50** | **2** |

Bảng 3.15: Bảng tổng hợp kết quả kiểm thử tương thích

|  |  |  |  |
| --- | --- | --- | --- |
| **Trình duyệt + Hệ điều hành** | **Tổng số TC** | **Số TC pass** | **Số TC fail** |
| Chrome + Windows 10 | 1 | 1 | 0 |
| Edge + Linux | 1 | 1 | 0 |
| Firefox + macOS Monterey | 1 | 0 | 1 |
| **TỔNG** | **3** | **1** | **1** |

Bảng 3.16: Bảng tổng hợp lỗi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tiêu đề** | **Các bước tái hiện** | **Chức năng** | **Testcase ID** |
| 1 | Hiển thị sai thông báo lỗi | 1.Nhập tên tài khoản và mật khẩu của tài khoản chưa được kích hoạt  2. Click button Đăng nhập | Đăng Nhập | DK\_6 |
| 2 | Không hiển thị đúng dữ liệu trên popup Cập nhật quản lý bài giảng ở trường Thời lượng | 1.Click vào button thêm mới  2. Nhập dữ liệu vào các trường bắt buôc  3. Lưu > Lưu  4. Click tìm kiếm  5. Nhập tên bài giảng vừa thêm mới > Enter  6. Click vào icon cập nhật trên dòng dữ liệu vừa thêm mới  7. So sánh dữ liệu vừa thêm mới và dữ liệu vừa ở màn cập nhật | Quản lý bài giảng | QLBG\_S\_1 |
| 3 | Kiểm thử toàn hệ thống trên Firefox + macOS Monterey | 1. Truy cập http://learnico.id.vn  2. Đăng nhập với tài khoản đúng/sai  3. Thêm/sửa/xóa danh mục bài giảng  4. Thêm/sửa/xóa loại bài giảng  5. Thêm/sửa/xóa bài giảng  6. Quan sát giao diện, bố cục | Kiểm thử tương thích | KTTT\_3 |

# KẾT LUẬN

Kiểm thử phần mềm hiện nay vẫn là một vấn đề hết sức quan trọng đối với các tổ chức phát triển phần mềm. Trong quá trình thực hiện đề tài này, mặc dù còn những hạn chế nhất định về thời gian, kinh nghiệm thực tế và kiến thức chuyên môn, em đã cố gắng tiếp cận vấn đề một cách có hệ thống, từ việc nghiên cứu lý thuyết cho đến ứng dụng thực tế vào kiểm thử hệ thống phần mềm học trực tuyến Learnico.

Sau một thời gian thực hiện khóa luận dưới sự hướng dẫn của thầy Hoàng Quang Huy báo cáo của em đã thực hiện tốt được các mục tiêu đề ra và đạt được những kết quả sau:

* Tìm hiểu và trình bày tổng quan về kiểm thử phần mềm, vai trò của kiểm thử trong chu trình phát triển phần mềm.
* Phân tích các phương pháp kiểm thử phổ biến như kiểm thử hộp trắng, kiểm thử hộp đen, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử hồi quy, v.v.
* Thiết kế các kịch bản kiểm thử (testcase) chi tiết dựa trên yêu cầu hệ thống.
* Ứng dụng công cụ Selenium WebDriver để xây dựng các kịch bản kiểm thử tự động cho một số chức năng tiêu biểu của hệ thống học trực tuyến Learnico như: chức năng đăng nhập, quản lý bài giảng, danh mục bài giảng và loại bài giảng.
* Triển khai kiểm thử tương thích trình duyệt (Cross-browser Testing) với sự hỗ trợ từ nền tảng LambdaTest. Qua đó, đảm bảo rằng các chức năng của hệ thống Learnico hoạt động ổn định và đồng nhất trên các trình duyệt phổ biến như Chrome, Firefox, Edge, Safari… và trên nhiều hệ điều hành khác nhau như Windows, macOS, Linux.
* Thực hiện ghi nhận kết quả kiểm thử, đánh giá chất lượng của hệ thống thông qua việc chụp màn hình, quay video quá trình kiểm thử, đồng thời tạo báo cáo tự động để theo dõi và tổng hợp kết quả kiểm thử.

Tuy nhiên, do thời gian nghiên cứu và triển khai có hạn, đề tài vẫn còn tồn tại một số điểm hạn chế nhất định như:

* Chưa tiến hành kiểm thử hiệu năng hệ thống với số lượng lớn người dùng truy cập đồng thời để đánh giá khả năng xử lý tải và kiểm tra tính ổn định trong môi trường thực tế.
* Phạm vi kiểm thử còn giới hạn, chủ yếu tập trung vào một số module chức năng cốt lõi mà chưa bao quát toàn bộ hệ thống Learnico.
* Các kỹ thuật kiểm thử nâng cao như kiểm thử bảo mật, hoặc tích hợp với các công cụ quản lý lỗi chuyên nghiệp vẫn chưa được triển khai trong khuôn khổ đề tài này.

Trong thời gian tới, em sẽ tiếp tục trau dồi kiến thức chuyên môn, mở rộng phạm vi nghiên cứu sang các chủ đề nâng cao như kiểm thử phi chức năng, kiểm thử API, kiểm thử trên thiết bị di động để có thể vận dụng vào kiểm thử các ứng dụng lớn hơn trong thực tế công việc trong tương lai nhằm góp một phần vào công cuộc nâng cao hiệu quả kiểm thử phần mềm ở Việt Nam.

# TÀI LIỆU THAM KHẢO

[1] BrowserStack. (n.d.). Automation testing tutorial. <https://www.browserstack.com/guide/automation-testing-tutorial>

[2] Khoa Công nghệ Thông tin, (n.d.), Tài liệu hướng dẫn làm báo cáo tốt nghiệp

[3] LambdaTest, (n.d.), Getting started with LambdaTest automation. LambdaTest. <https://www.lambdatest.com/support/docs/getting-started-with-lambdatest-automation/>

[4] NXB Đại học Công nghiệp Hà Nội, (n.d.), Tài liệu môn học Kiểm thử phần mềm.

[5]Selenium, (n.d.), Overview, Selenium, <https://www.selenium.dev/documentation/overview/>

[6]Wikipedia, (n.d.), Kiểm thử phần mềm, [https://vi.wikipedia.org/wiki/Kiểm\_thử\_phần\_mềm](https://vi.wikipedia.org/wiki/Ki%E1%BB%83m_th%E1%BB%AD_ph%E1%BA%A7n_m%E1%BB%81m)