**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**TÊN ĐỀ TÀI MÔN KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

***Nhóm XX - Sinh viên thực hiện***

1. Nguyễn Thành Doanh - 20054911
2. Dương Tuấn Kiệt - 20001545
3. Vũ Thế Kiệt - 20001565

# MỤC LỤC

[MỤC LỤC 1](#_Toc14693502)

[DANH MỤC CÁC HÌNH VẼ 1](#_Toc14693503)

[DANH MỤC CÁC BẢNG BIỂU 1](#_Toc14693504)

[CHƯƠNG 1 : GIỚI THIỆU 1](#_Toc14693505)

[1.1 Tổng quan 1](#_Toc14693506)

[1.2 Mục tiêu đề tài 1](#_Toc14693507)

[1.3 Phạm vi đề tài 1](#_Toc14693508)

[1.4 Mô tả yêu cầu chức năng 1](#_Toc14693509)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 1](#_Toc14693510)

[2.1 Model-View-Controller Architecture 1](#_Toc14693511)

[2.2 Spring boot 1](#_Toc14693512)

[2.3 Docker 1](#_Toc14693513)

[2.4 HeiSQL 1](#_Toc14693513)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 1](#_Toc14693514)

[3.1 Phân tích yêu cầu bằng UML 1](#_Toc14693515)

[3.1.1 Usecase tổng quát 1](#_Toc14693516)

[3.1.2 Danh sách tác nhân và mô tả 1](#_Toc14693517)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 1](#_Toc14693518)

[3.1.4 Tình huống hoạt động 1](#_Toc14693519)

[3.2 Class diagram 1](#_Toc14693520)

[3.3 Cấu trúc project 1](#_Toc14693521)

[3.4 Mô hình MVC 1](#_Toc14693521)

[3.5 Deployment diagram 1](#_Toc14693521)

[CHƯƠNG 4 : HIỆN THỰC 1](#_Toc14693522)

[4.1 Cấu hình phần cứng, phần mềm 1](#_Toc14693523)

[4.2 Giao diện của hệ thống 1](#_Toc14693524)

[CHƯƠNG 5 : KẾT LUẬN 1](#_Toc14693525)

[5.1 Kết quả đạt được 1](#_Toc14693526)

[5.2 Hạn chế của đồ án 1](#_Toc14693527)

[5.3 Hướng phát triển 1](#_Toc14693528)

[TÀI LIỆU THAM KHẢO 1](#_Toc14693529)

# DANH MỤC CÁC HÌNH VẼ

[Hình 2‑1 Node.js development 1](#_Toc14693413)

# DANH MỤC CÁC BẢNG BIỂU

[Bảng 3‑1 Danh sách tác nhân và mô tả 1](#_Toc14682192)

[Bảng 3‑2 Danh sách các tình huống trong hệ thống 1](#_Toc14682193)

**LỜI MỞ ĐẦU**

Trong thế giới hiện nay công nghệ thông tin ngày càng phát triển vượt bậc và ngày càng đạt được thành tựu to lớn trong việc phát triển kinh tế. Trên hầu hết tất cả lĩnh vực thì đều có mặt ngành công nghệ thông tin trong đó, nó đã trở thành một phần thiết yếu trong cuộc sống. Nhờ có công nghệ thông tin mà kinh tế có thể phát triển và đời sống con người được nâng cao, xã hội ngày càng tiên tiến. Việc tận dụng công nghệ thông tin vào mọi ngành nghề đã không còn là một việc quá xa lạ.

Đăng ký học phần là một trong những nhiệm vụ quan trọng của bất kỳ tổ chức giáo dục nào. Việc thu thập, lưu trữ và xử lý thông tin về sinh viên đòi hỏi sự chính xác, hiệu quả và dễ dàng truy cập để hỗ trợ quá trình quản lý và đưa ra các quyết định phù hợp.

Với sự phát triển không ngừng của công nghệ thông tin, việc sử dụng các công cụ và kỹ thuật công nghệ mới trong việc đăng ký học phần đã trở nên cần thiết. Trong đó, sự kết hợp giữa ngôn ngữ lập trình Java, và cơ sở dữ liệu Mariadb Workbench là một phương pháp phổ biến và mạnh mẽ để tạo ra một hệ thống đăng ký học phần linh hoạt, mạnh mẽ và dễ dàng mở rộng.

Với đề tài này giúp chúng ta củng cố lại những kiến thức về công nghệ. Đồ án thực hiện dựa trên những kiến thức đã học và tìm kiếm trên Internet. Do kiến thức và trình độ còn non kém nên chúng em chưa hoàn thành đầy đủ các công tác quản lý. Trong quá trình thực hiện nếu có sai sót mong thầy bỏ qua.

Chúng em xin cam đoan đồ án: “Đăng ký học phần” là công trình nghiên cứu của bản thân. Những phần sử dụng tài liệu tham khảo trong luận văn đã được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong luận văn là hoàn toàn trung thực, nếu sai chúng em xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của bộ môn và nhà trường đề ra.

# : GIỚI THIỆU

## Tổng quan

Đăng ký học phần là một phần quan trọng trong hệ thống giáo dục, đặc biệt là trong các cơ sở giáo dục lớn với số lượng sinh viên đông đảo. Để quản lý và theo dõi thông tin của sinh viên một cách hiệu quả, việc xây dựng các hệ thống đăng ký học phần đã trở thành một xu hướng không thể phủ nhận.

Chương này giới thiệu tổng quan về đề tài đăng ký học phần, nhằm cung cấp một cái nhìn tổng quan về nội dung và mục tiêu của nghiên cứu. Chúng ta sẽ tìm hiểu về tình hình hiện tại của đăng ký học phần, các vấn đề và thách thức mà các tổ chức giáo dục đang phải đối mặt, cũng như ý nghĩa và lợi ích của việc xây dựng một hệ thống đăng ký học phần hiệu quả.

Chương này cũng sẽ trình bày về phạm vi và phương pháp nghiên cứu được sử dụng để đạt được mục tiêu của đề tài. Bằng việc nắm bắt các khía cạnh quan trọng của đăng ký học phần và áp dụng các phương pháp nghiên cứu phù hợp, chúng ta hy vọng sẽ đưa ra các giải pháp và gợi ý để nâng cao hiệu quả việc đăng ký học phần trong các tổ chức giáo dục.

Mục tiêu của chương này là cung cấp một cái nhìn tổng quan về đề tài đăng ký học phần và tạo nền tảng cho các phần tiếp theo của báo cáo nghiên cứu. Bằng cách hiểu rõ về tình hình hiện tại, mục tiêu, phạm vi và phương pháp nghiên cứu, chúng ta sẽ có cơ sở để tiếp cận và nghiên cứu chi tiết hơn về các khía cạnh cụ thể của đăng ký học phần và xây dựng một hệ thống đăng ký học phần hiệu quả.

## Mục tiêu đề tài

Tối ưu hóa khả năng mở rộng: Phân chia hệ thống thành các dịch vụ nhỏ độc lập giúp tăng khả năng mở rộng và tối ưu hóa hiệu suất của hệ thống.

Tăng tính linh hoạt: Cho phép mỗi dịch vụ microservice được phát triển và triển khai độc lập, giảm thiểu ảnh hưởng của các thay đổi và cập nhật.

Cải thiện trải nghiệm người dùng: Tạo ra một giao diện dễ sử dụng cho sinh viên và giảng viên để đăng ký học phần, quản lý thời khóa biểu và tra cứu thông tin học tập.

Tăng cường tính bảo mật: Phân tách hệ thống thành các dịch vụ nhỏ giúp tăng cường bảo mật bằng cách giảm thiểu rủi ro từ các lỗ hổng an ninh.

Tối ưu hóa quy trình đăng ký học phần: Tự động kiểm tra điều kiện tiên quyết và quản lý số lượng sinh viên đăng ký vào mỗi lớp học, giúp quy trình đăng ký trở nên nhanh chóng và thuận tiện.

## Phạm vi đề tài

Xây dựng hệ thống quản lý lớp học tín chỉ: Phát triển một hệ thống toàn diện để quản lý thông tin về sinh viên, môn học, và lớp học tín chỉ, bao gồm các tính năng đăng ký học phần, quản lý thời khóa biểu, và tra cứu thông tin học tập.

Áp dụng kiến trúc microservice: Sử dụng kiến trúc microservice để phân chia hệ thống thành các dịch vụ nhỏ độc lập, tối ưu hóa hiệu suất và tính linh hoạt của hệ thống.

Tích hợp hệ thống: Kết nối các dịch vụ microservice với nhau và với các hệ thống và dịch vụ bên ngoài (ví dụ: hệ thống quản lý sinh viên, hệ thống thanh toán) để tạo ra một hệ thống hoàn chỉnh và tích hợp.

Phát triển giao diện người dùng: Xây dựng giao diện người dùng thân thiện và dễ sử dụng cho sinh viên và giảng viên, cung cấp các tính năng như đăng ký học phần, quản lý thời khóa biểu, và tra cứu thông tin học tập.

Bảo mật và bảo vệ dữ liệu: Đảm bảo tính bảo mật và bảo vệ dữ liệu thông qua các biện pháp bảo mật như xác thực, phân quyền và mã hóa dữ liệu.

## Mô tả yêu cầu chức năng

* Đăng ký học phần:

+ Sinh viên có thể đăng ký học phần thông qua giao diện người dùng.

+ Hệ thống tự động kiểm tra các điều kiện tiên quyết trước khi cho phép sinh viên đăng ký.

+ Hạn chế số lượng sinh viên đăng ký vào mỗi lớp học tín chỉ để đảm bảo không quá tải.

* Quản lý thời khóa biểu:

+ Sinh viên và giảng viên có thể xem và quản lý thời khóa biểu của mình.

+ Hệ thống cập nhật thời khóa biểu tự động khi có thay đổi trong đăng ký học phần.

+ Tra cứu thông tin học tập:

+ Sinh viên có thể tra cứu thông tin về các môn học đã đăng ký, điểm số, và tình trạng học tập.

+ Giảng viên có thể tra cứu thông tin về các lớp học mình giảng dạy và điểm số của sinh viên.

* Xác nhận và quản lý đăng ký:

+ Sinh viên phải xác nhận đăng ký học phần trước khi hệ thống chấp nhận đăng ký của họ.

+ Sinh viên có thể quản lý và chỉnh sửa đăng ký của mình trước thời hạn đăng ký.

* Quản lý thông tin sinh viên:

+ Hệ thống lưu trữ và quản lý thông tin cá nhân của sinh viên, bao gồm hồ sơ học tập, điểm số, và tiến trình học tập.

* Tính năng quản trị:

+ Quản trị viên có thể quản lý danh sách môn học, lớp học, thông tin sinh viên, và các thiết lập hệ thống khác.

+ Hệ thống cung cấp giao diện quản trị dễ sử dụng cho việc quản lý hệ thống và xử lý các vấn đề kỹ thuật.

# : CƠ SỞ LÝ THUYẾT

## Model-View-Controller Architecture:

* Khái niêm: Model-View-Controller (MVC) là một mẫu kiến ​​trúc/thiết kế tách ứng dụng thành ba thành phần logic chính Model, View và Controller
* Đặc điểm:

+ Nó cung cấp sự phân tách rõ ràng giữa logic nghiệp vụ, logic giao diện người dùng và logic đầu vào.

+ Nó cung cấp toàn quyền kiểm soát HTML và URL của bạn, giúp bạn dễ dàng thiết kế kiến ​​trúc ứng dụng web.

+ Đây là một thành phần ánh xạ URL mạnh mẽ mà chúng tôi có thể sử dụng để xây dựng các ứng dụng có URL dễ hiểu và có thể tìm kiếm được.

+ Nó hỗ trợ Phát triển dựa trên thử nghiệm (TDD).

## Spring boot:

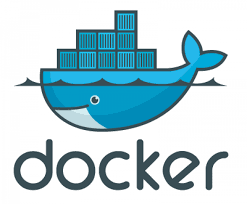
* Khái niệm: Spring Boot là một dự án phát triển bởi JAV (ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.



Hình 2‑1 Spring boot

## Docker:

* Khái niệm: Docker là một nền tảng cho developers và system admin để develop, deploy và run application với container. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Khi cần deploy lên bất kỳ server nào chỉ cần run container của Docker thì application của bạn sẽ được khởi chạy ngay lập tức.



## HeiSQL:

* Khái niệm: HeidiSQL là một công cụ quản lý database tất cả trong một. Có thể dùng HeidiSQL để kết nối từ xa tới database được tạo trên Hostinger.



# : PHÂN TÍCH VÀ THIẾT KẾ

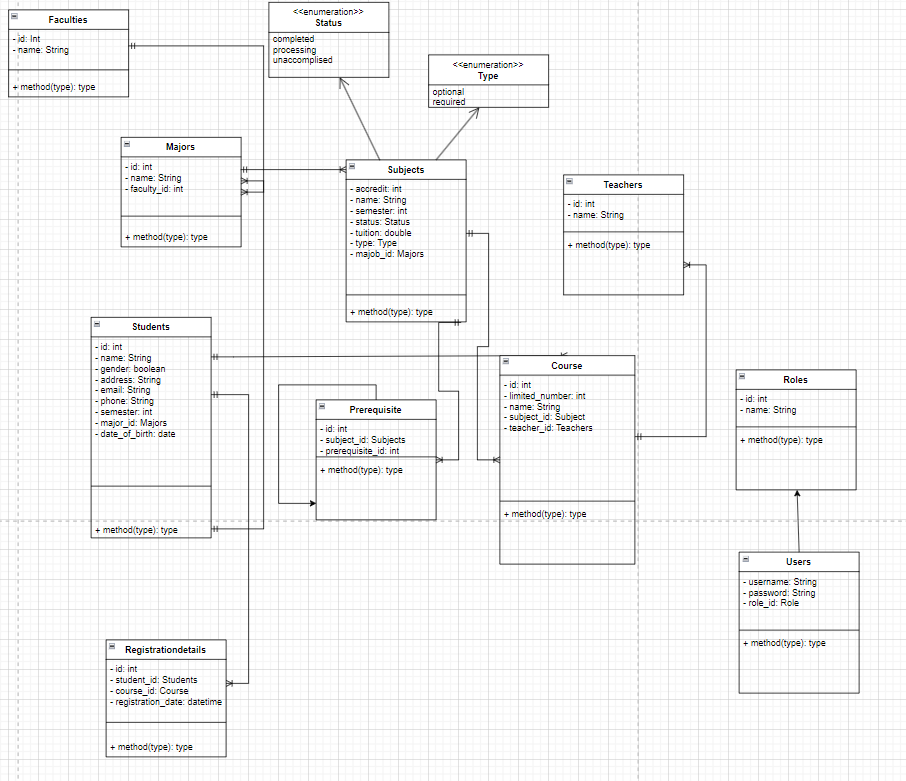
## Phân tích yêu cầu bằng UML

### Usecase tổng quát



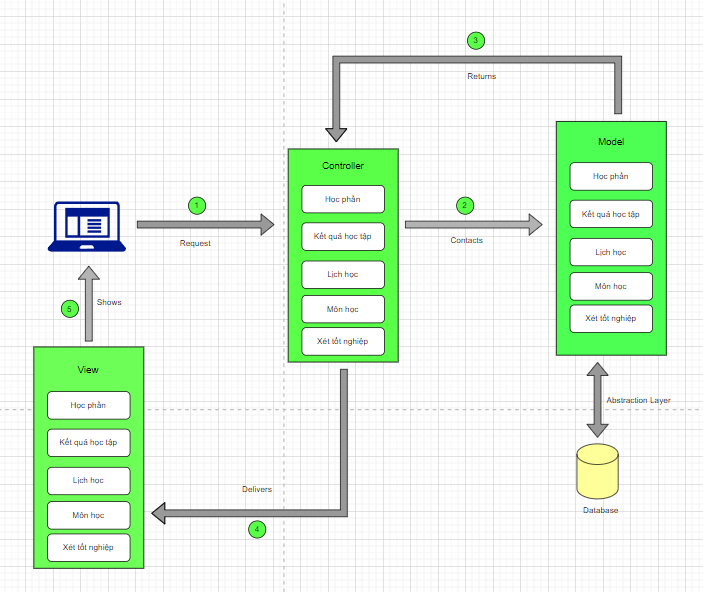
## Class diagram

Mô hình lớp (Mô hình lớp đầy đủ + Mô hình lớp bao gồm các lớp Entity)



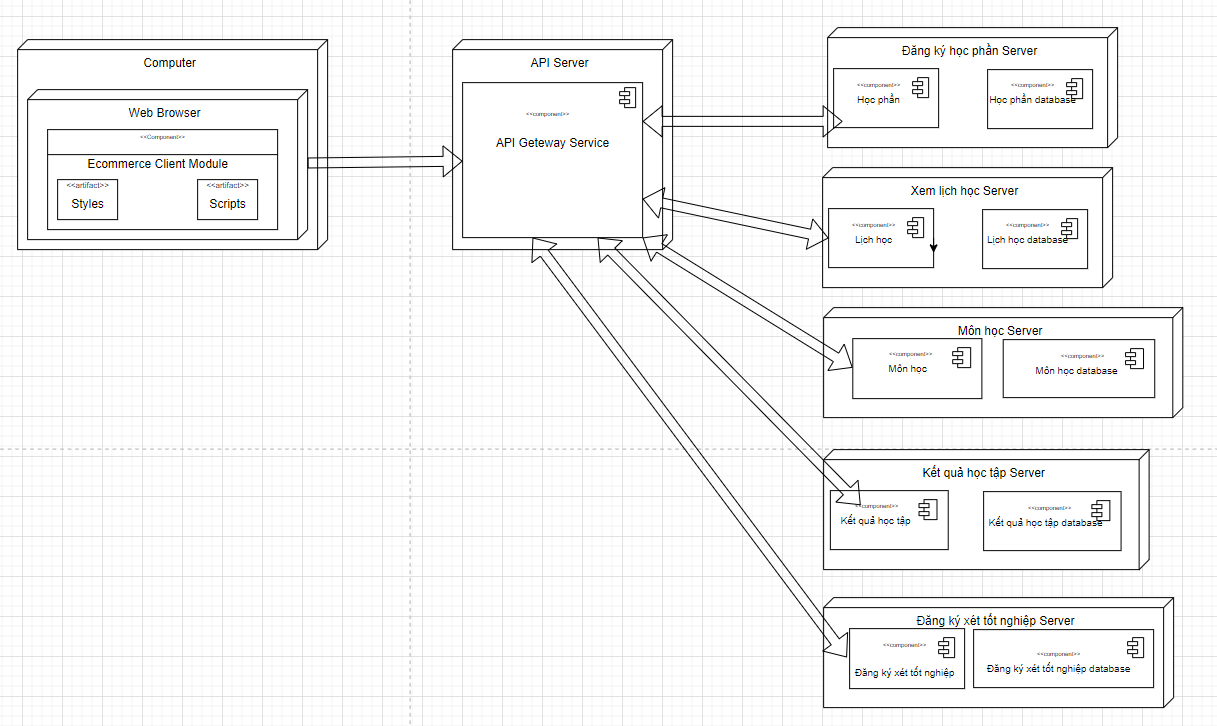
**Mô hình MVC:**

**-** Mô hình MVC(Model-View-Control):



## Deployment diagram:

Mô hình triển khai hệ thống.



### Danh sách tác nhân và mô tả

Danh sách tác nhân và mô tả xem ở Bảng 3-1.

|  |  |
| --- | --- |
| Tác nhân | Mô tả tác nhân |
| User | User được xác định thông qua tài khoản đăng nhập, bao gồm mã số sinh viên, mật khẩu, user có thể đăng ký học phần, hủy học phần mình đã đăng ký, xem danh sách môn mình đã đăng ký, xem lịch học, xem điểm, đóng học phí bằng cách thanh toán trực tiếp trên trang web & đăng ký xét tốt nghiệp. |

Bảng 3‑1 Danh sách tác nhân và mô tả

### Danh sách các tình huống hoạt động (Use cases)

|  |  |
| --- | --- |
| ID | Tên Use case |
| UC01 | Đăng nhập |
| UC02 | Đăng ký học phần mới |
| UC03 | Hủy học phần vừa đăng ký |
| UC04 | Xem danh sách môn học đã đăng ký học một kỳ |
| UC05 | Xem lịch học |
| UC06 | Xem điểm |
| UC07 | Đóng học phí Onlline |
| UC08 | Đăng ký xét tốt nghiệp |

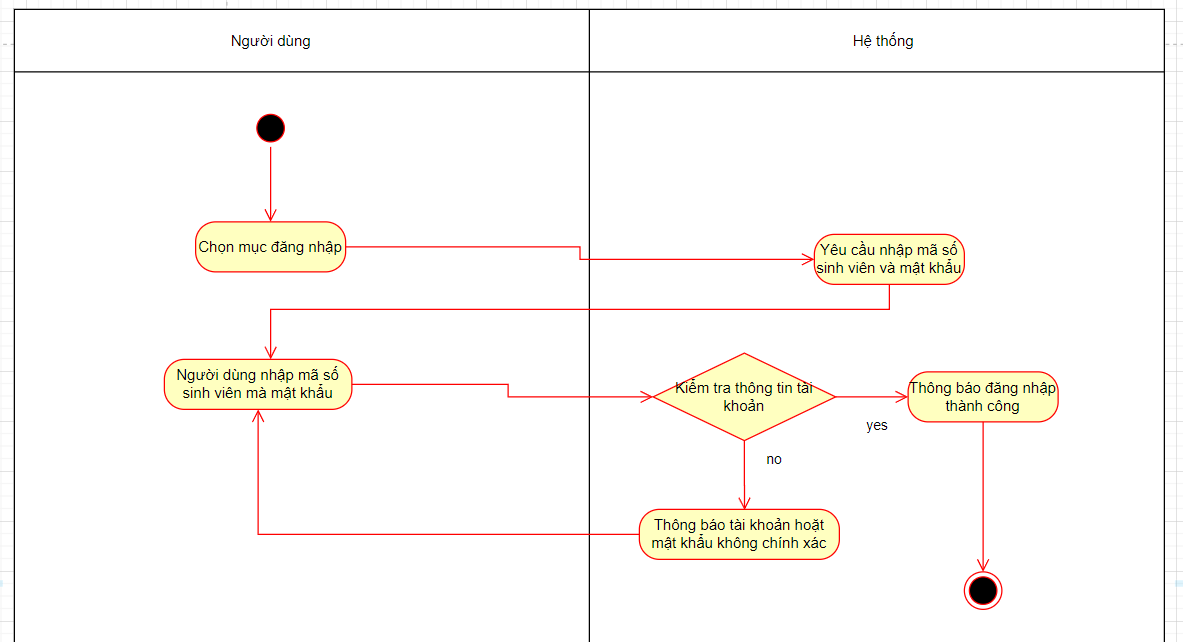
Bảng 3‑2 Danh sách các tình huống trong hệ thống

### Tình huống hoạt động

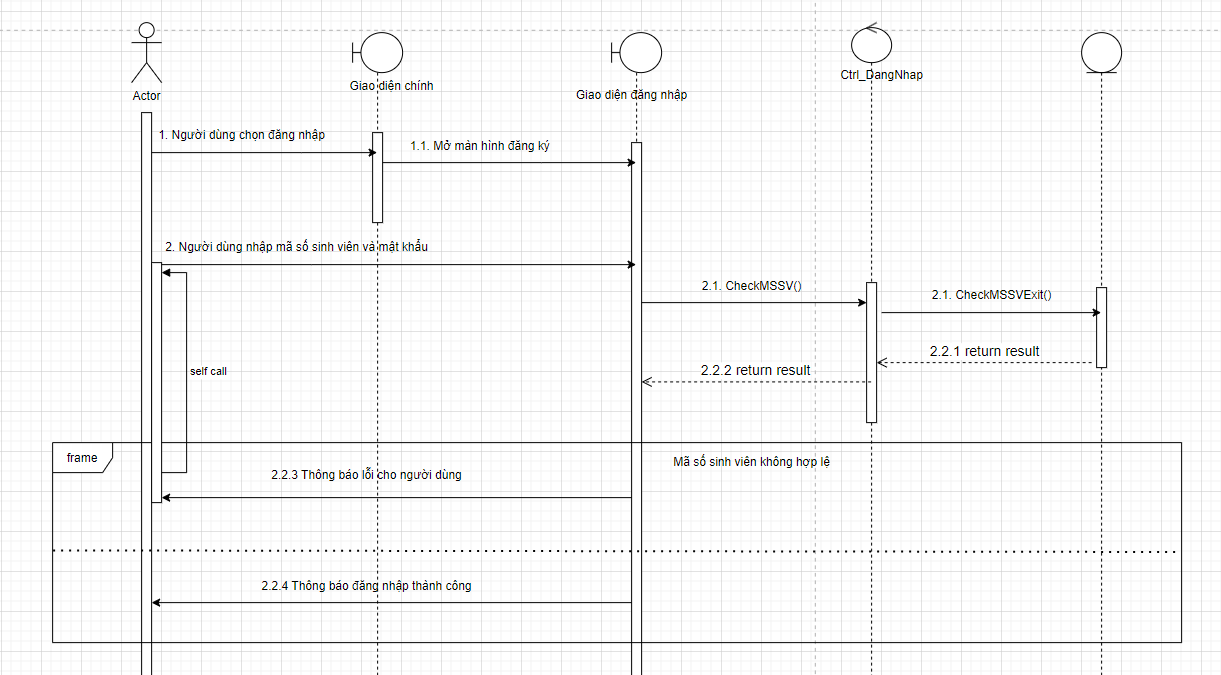
Đặc tả use case.

|  |  |  |
| --- | --- | --- |
| **UC001\_Đăng Nhập** | | |
| **Mục đích** | Cho phép người dùng đăng nhập tài khoản | |
| **Mô tả** | Usecase này mô tả cách người dùng đăng nhập tài khoản | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng sở hữu tài khoản | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn mục đăng nhập | | 2. Hệ thống yêu cầu người dùng nhập mã số sinh viên và mật khẩu |
| 3. Người dùng nhập mã số sinh viên và mật khẩu bấm đăng nhập | | 4.Hệ thống kiểm tra thông tin tài khoản |
|  | | 5.Hệ thống thông báo đăng nhập thành công |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |
|  | | 4.1 Nếu mã số sinh viên hoặc mật khẩu không hợp lệ thì hệ thống báo lỗi. |
| 4.2 Người dùng quay về bước 3 | |  |

Activity diagram.

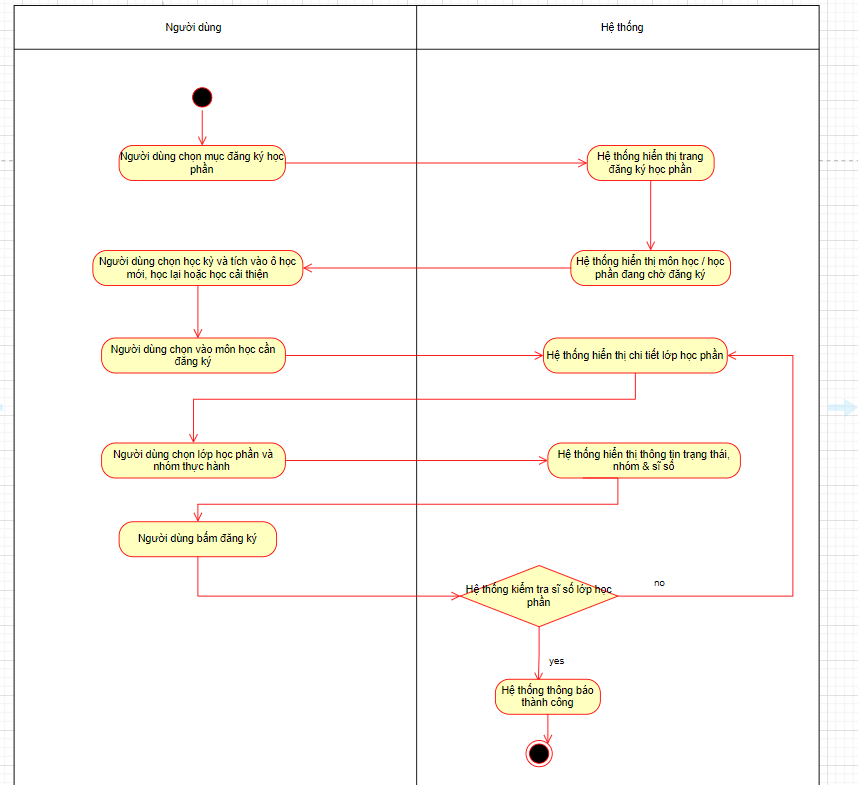


Sequence diagram cho từng tình huống.

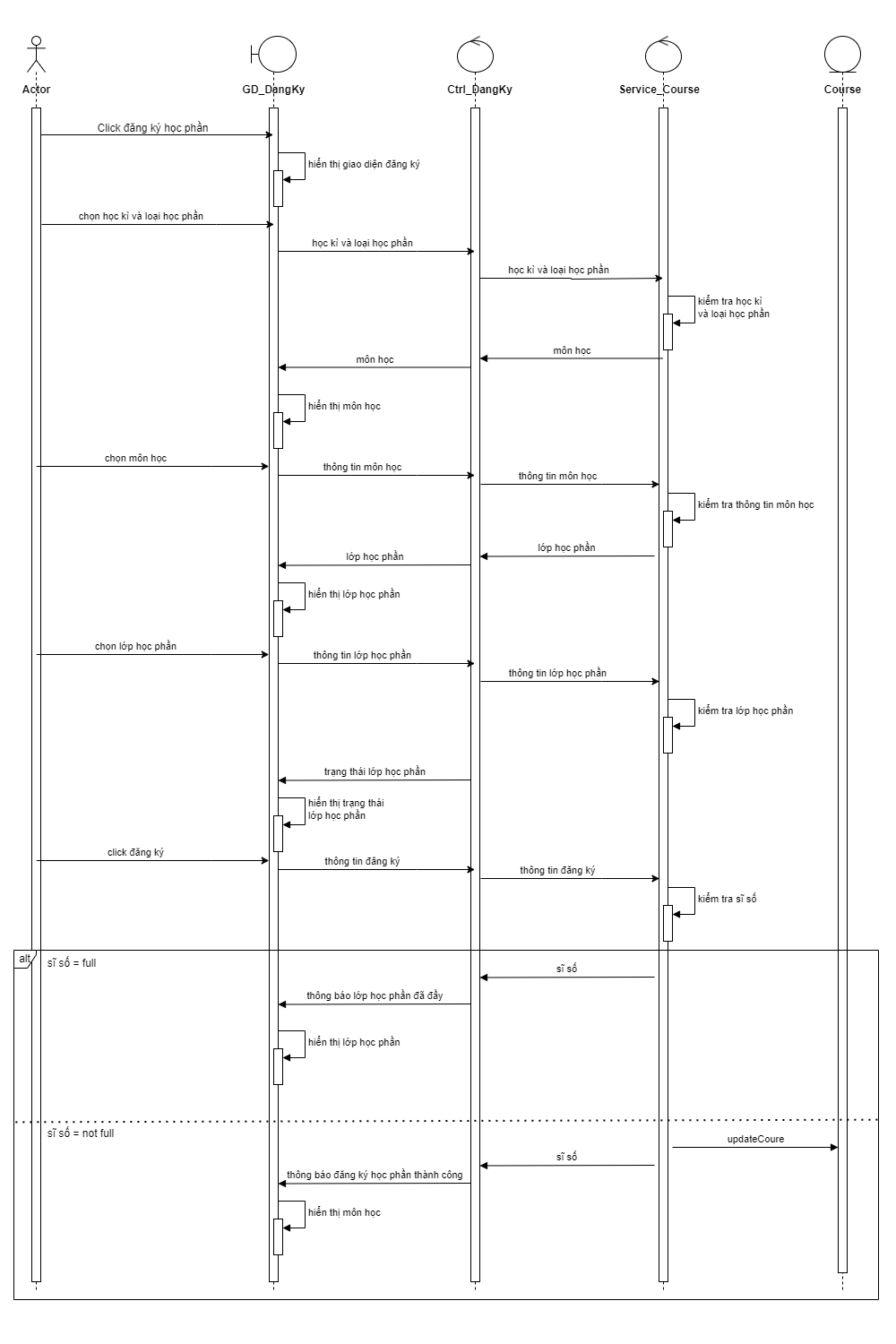


|  |  |  |
| --- | --- | --- |
| **UC002\_Đăng ký học phần mới** | | |
| **Mục đích** | Cho phép người dùng đăng kí học phần | |
| **Mô tả** | Usecase này mô tả cách người dùng đăng kí học phần mới | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Phải đăng ký học môn tiên quyết trước | |
| **Điều kiện sau** | Người dùng đăng kí học phần thành công | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn mục đăng kí học phần | | 2. Hệ thống hiển thị trang đăng ký học phần |
| 3. Người dùng chọn học kỳ và tích vào ô học mới, học lại hoặc học cải thiện | | 4. Hệ thống hiển thị môn học/ học phần đang chờ đăng ký |
| 5. Người dùng chọn vào môn học cần đăng ký | | 6. Hệ thống hiển thị chi tiết lớp học phần |
| 7. Người dùng chọn lớp học phần và nhóm thực hành | | 8.Hệ thống hiển thị thông tin trạng thái, nhóm & sĩ số |
| 9.Người dùng bấm đăng ký | | 10. Hệ thống kiểm tra sĩ số lớp học phần |
|  | | 11.Hệ thống thông báo đăng ký thành công |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |
|  | | 10.1 Nếu lớp học phần bị đầy, hệ thống sẽ thông báo lớp học phần này đã đủ số lượng học viên, không được phép đăng ký vào lớp này**.** |
| 10.2 Người dùng quay về bước 6 | |  |

Activity diagram.

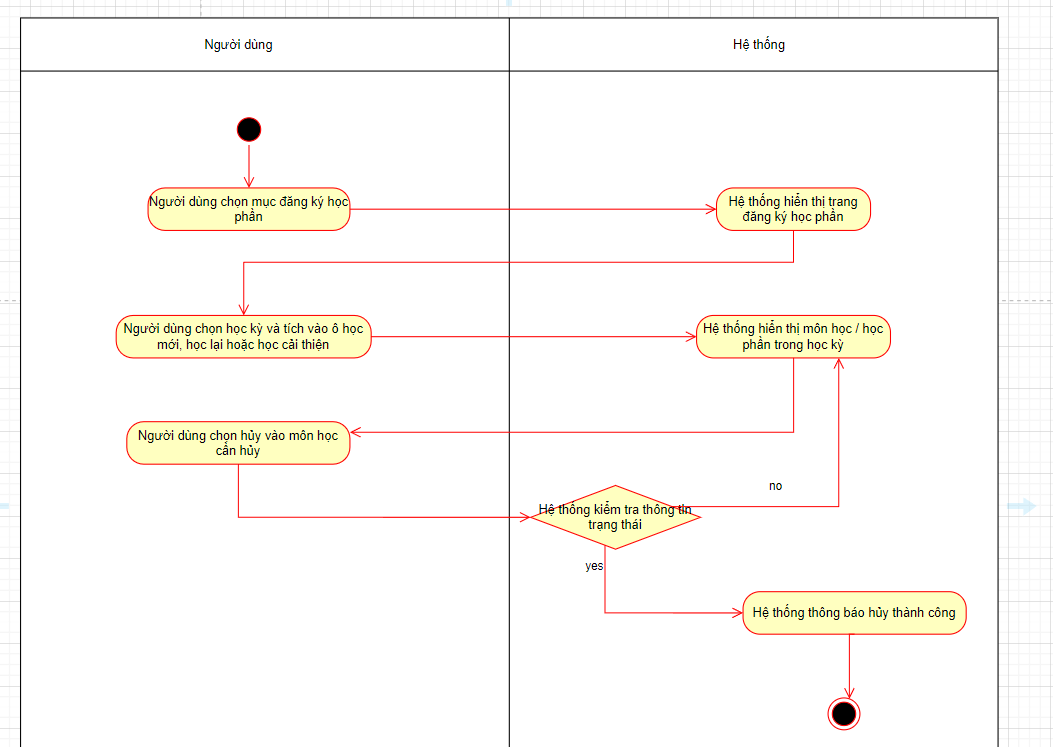


Sequence diagram cho từng tình huống.

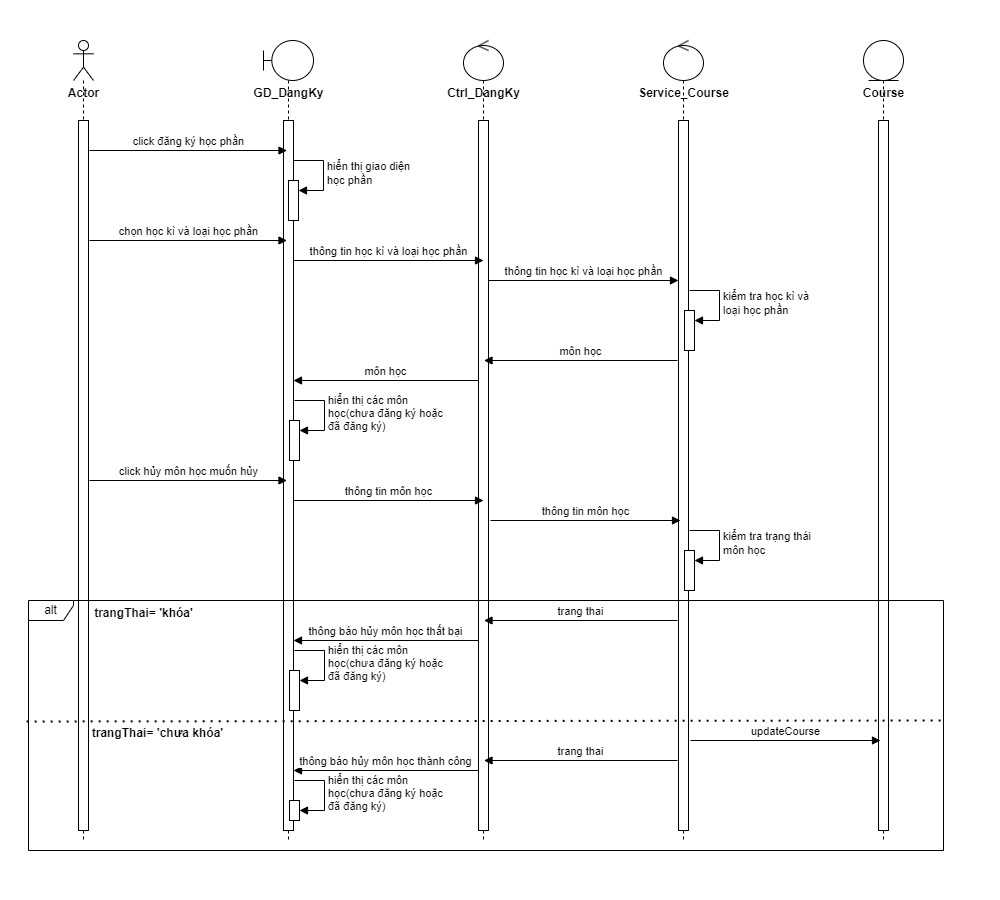


|  |  |  |
| --- | --- | --- |
| **UC003\_Hủy học phần** | | |
| **Mục đích** | Cho phép người dùng hủy học phần vừa đăng ký | |
| **Mô tả** | Usecase này mô tả cách người dùng hủy học phần vừa đăng ký | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn mục đăng kí học phần | | 2. Hệ thống hiển thị trang đăng ký học phần |
| 3. Người dùng chọn học kỳ và tích vào ô học mới, học lại hoặc học cải thiện | | 4. Hệ thống hiển thị môn học/ học phần đang chờ đăng ký |
| 5. Người dùng chọn hủy vào môn học cần hủy | | 6. Hệ thống kiểm tra thông tin trạng thái |
|  | | 7.Hệ thống thông báo hủy thành công |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |
|  | | 7.1 Nếu trạng thái lớp học phần đã khóa, hiển thị thông báo hủy thất bại |
| 7.2 Người dùng quay về bước 4 | |  |

Activity diagram.

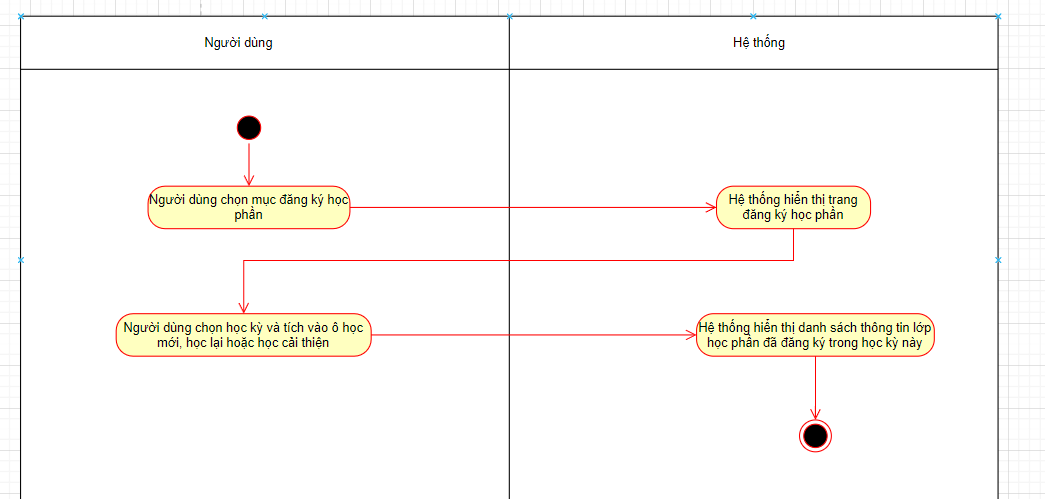


Sequence diagram cho từng tình huống.

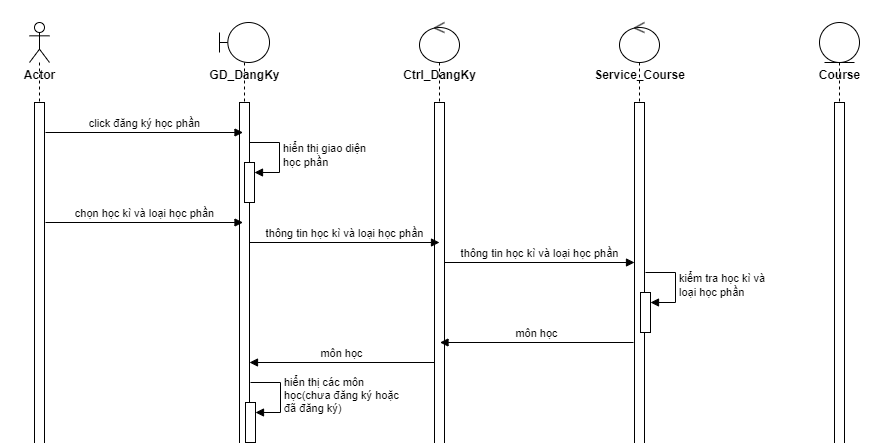


|  |  |  |
| --- | --- | --- |
| **UC004\_Xem danh sách môn học đã đăng ký trong học kỳ** | | |
| **Mục đích** | Cho phép người dùng xem danh sách môn học đã đăng ký trong học kỳ | |
| **Mô tả** | Usecase này mô tả cách người dùng xem danh sách môn học đã đăng ký trong học kỳ | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn mục đăng kí học phần | | 2. Hệ thống hiển thị trang đăng ký học phần |
| 3. Người dùng chọn học kỳ và tích vào ô học mới, học lại hoặc học cải thiện | | 4. Hệ thống hiển thị danh sách thông tin lớp học phần đã đăng ký trong học kỳ này |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |

Activity diagram.

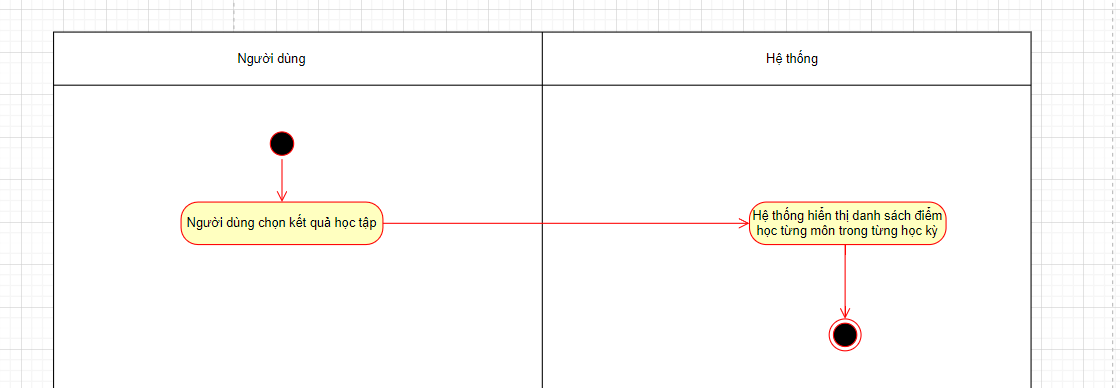


Sequence diagram cho từng tình huống.

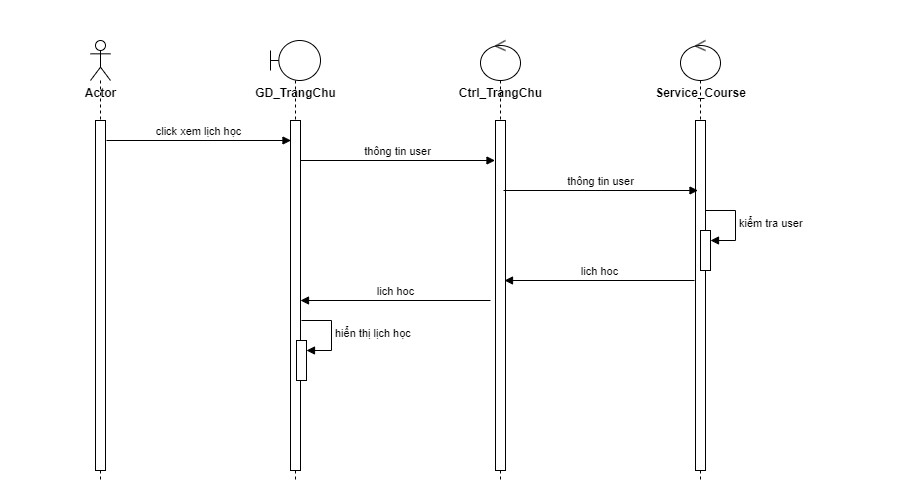


|  |  |  |
| --- | --- | --- |
| **UC005\_Xem lịch học** | | |
| **Mục đích** | Cho phép người dùng xem lịch học | |
| **Mô tả** | Usecase này mô tả cách người dùng xem lịch học | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn lịch học theo tuần | | 2. Hệ thống hiện thị danh sách lịch học theo tuần |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |

* Sơ đồ Activiti:

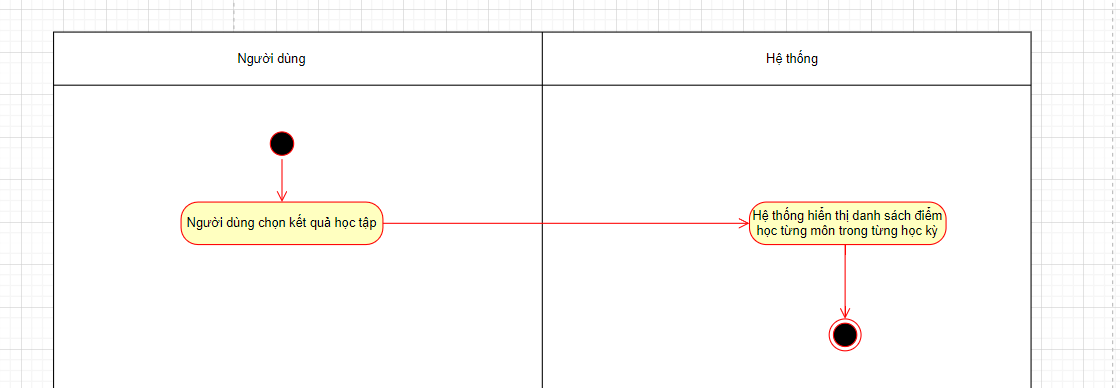


* Sequence diagram cho từng tình huống.

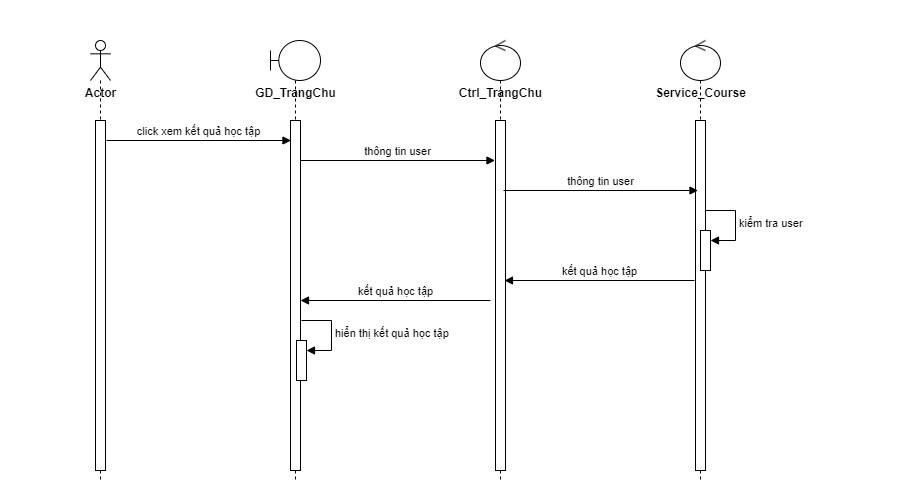


|  |  |  |
| --- | --- | --- |
| **UC006\_Xem điểm** | | |
| **Mục đích** | Cho phép người dùng xem điểm | |
| **Mô tả** | Usecase này mô tả cách người dùng xem điểm cá nhân | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng bấm chọn kết quả học tập | | 2. Hệ thống hiển thị danh sách điểm học trong từng học kỳ |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |

* Sơ đồ Activiti:

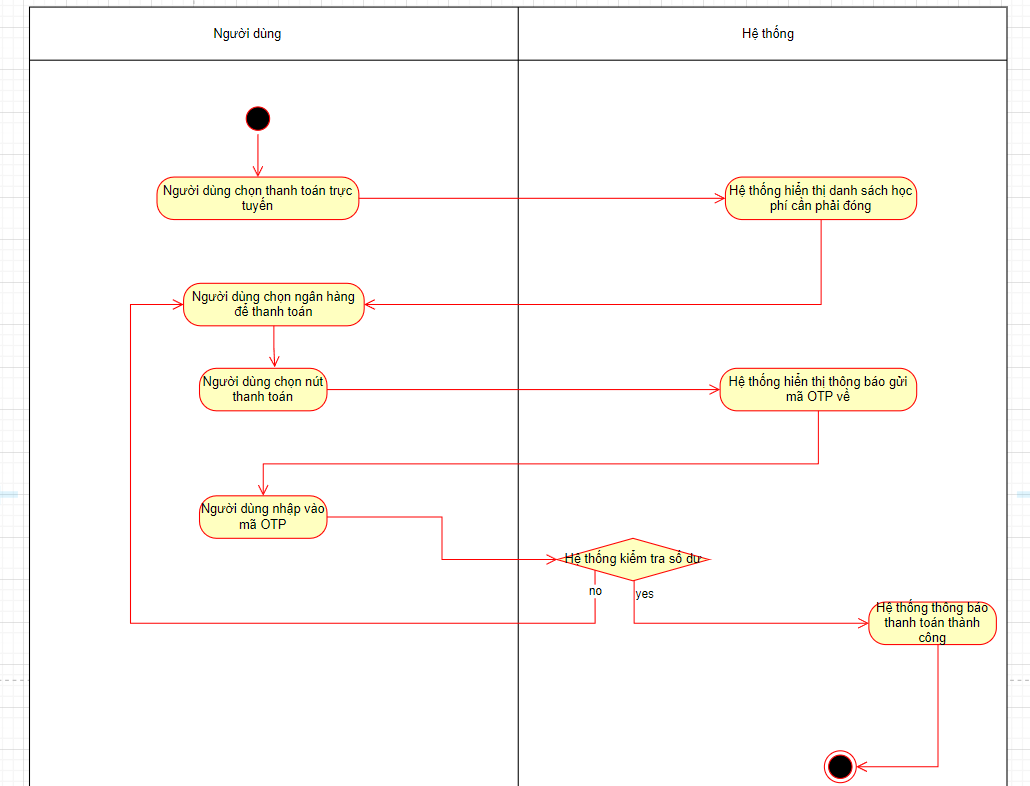


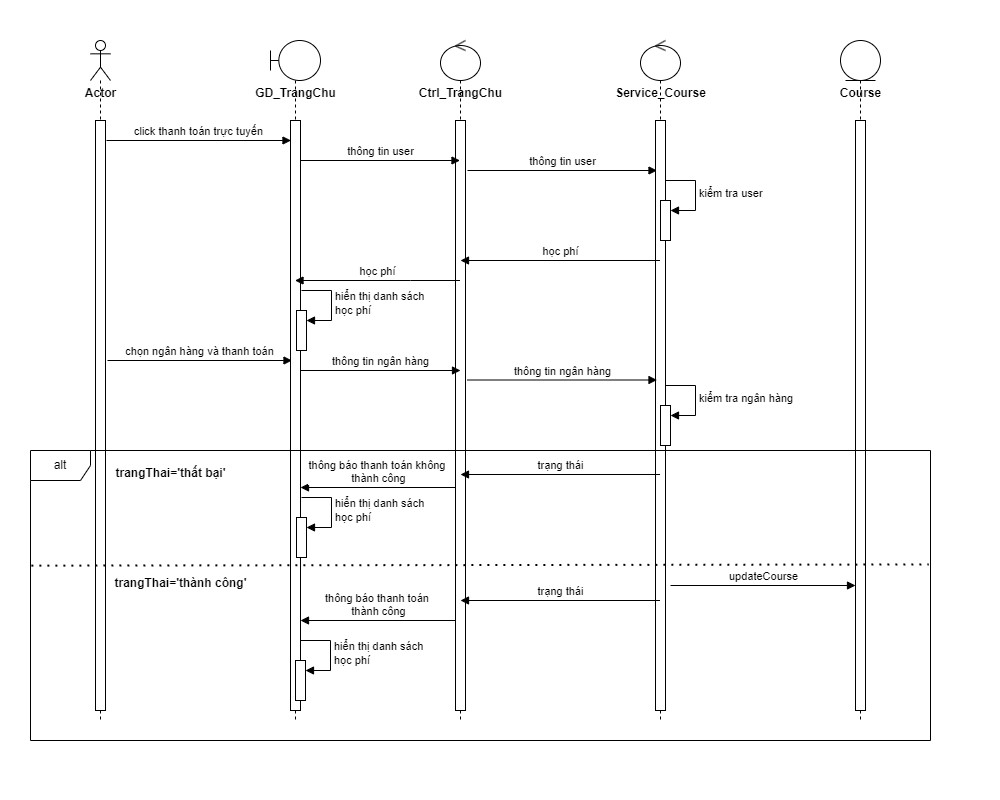
* Sequence diagram cho từng tình huống.



|  |  |  |
| --- | --- | --- |
| **UC007\_Đóng học phí Onlline** | | |
| **Mục đích** | Cho phép người dùng thực hiện đóng học phí trực tuyến | |
| **Mô tả** | Usecase này mô tả cách người dùng đóng học phí trực tuyến | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng bấm chọn thanh toán trực tuyến | | 2. Hệ thống hiện thị danh sách các khoản học phí cần phải đóng |
| 3. Người dùng chọn ngân hàng để thanh toán | |  |
| 4. Người dùng chọn nút thanh toán | | 5. Hệ thống hiển thị thông báo gửi mã OTP về |
| 6. Người dùng nhập vào mã OTP | | 7. Hệ thống kiểm tra số dư |
|  | | 8. Hệ thống thông báo thanh toán thành công |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |
|  | | 7.1 Nếu số dư không đủ thì sẽ thông báo thanh toán thất bại |
| 7.2 Người dùng quay về bước 3 | |  |

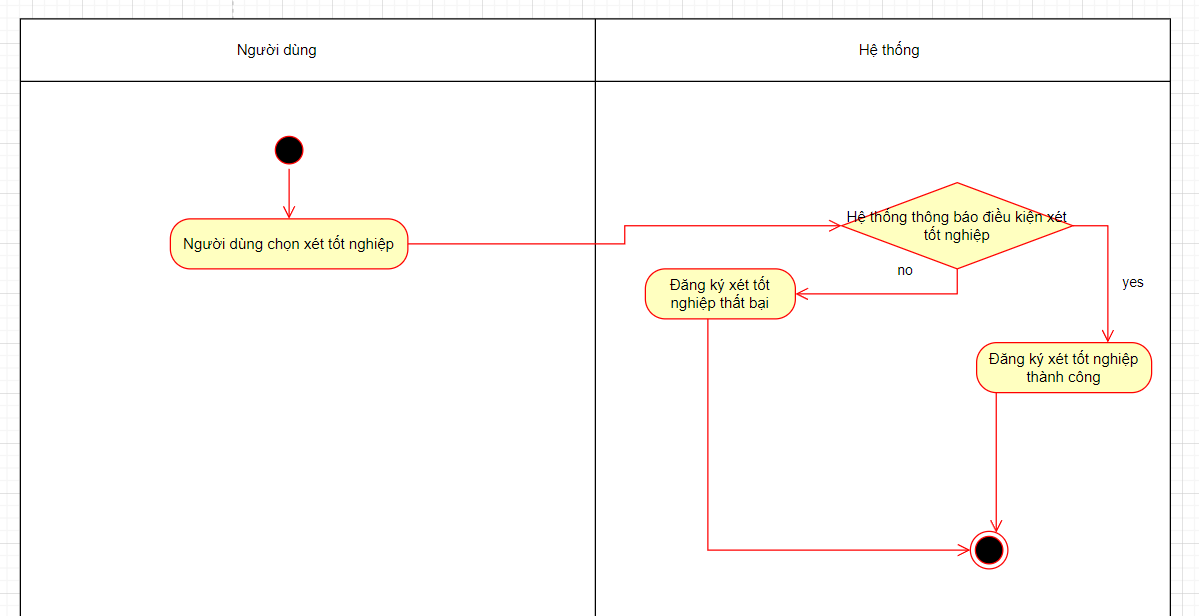
* Sơ đồ Activiti:





|  |  |  |
| --- | --- | --- |
| **UC008\_** Đăng ký xét tốt nghiệp | | |
| **Mục đích** | Cho phép người dùng đăng ký xét tốt nghiệp | |
| **Mô tả** | Usecase này mô tả cách người dùng xét tốt nghiệp | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đã có tài khoản & đủ điều kiện xét tốt nghiệp | |
| **Điều kiện sau** | Không | |
| **Luồng sự kiện chính** | | |
| **Actor** | | **System** |
| 1. Người dùng chọn mục xét tốt nghiệp | | 2. Hệ thống thông báo đủ điều xét tốt nghiệp |
| **Luồng sự kiện phụ** | | |
| **Actor** | | **System** |
|  | | 2.1 Nếu không đủ điều kiện xét, hệ thống thông báo đăng ký xét tốt nghiệp thất bại |

* Sơ đồ Activiti:



# : HIỆN THỰC

## Cấu hình phần cứng, phần mềm

4.1.1 Cấu hình phần cứng

Để triển khai hệ thống quản lý lớp học tín chỉ theo mô hình microservice, chúng ta cần một cơ sở hạ tầng phần cứng đủ mạnh để đảm bảo hiệu suất và khả năng mở rộng của hệ thống. Dưới đây là cấu hình phần cứng đề xuất cho hệ thống:

1. Máy chủ ứng dụng:

CPU: 16 cores

RAM: 64 GB

Ổ cứng: SSD 1 TB

Mạng: Kết nối mạng tốc độ cao (1 Gbps hoặc cao hơn)

1. Máy chủ cơ sở dữ liệu:

CPU: 16 cores

RAM: 128 GB

Ổ cứng: SSD 2 TB (có khả năng mở rộng)

Mạng: Kết nối mạng tốc độ cao (1 Gbps hoặc cao hơn)

1. Máy chủ container (Kubernetes nodes):

CPU: 8 cores mỗi node

RAM: 32 GB mỗi node

Ổ cứng: SSD 500 GB mỗi node

Mạng: Kết nối mạng tốc độ cao (1 Gbps hoặc cao hơn)

Số lượng nodes: Tùy thuộc vào quy mô và yêu cầu mở rộng của hệ thống (tối thiểu 3 nodes để đảm bảo khả năng chịu lỗi và cân bằng tải)

1. Máy chủ lưu trữ và sao lưu:

CPU: 8 cores

RAM: 32 GB

Ổ cứng: HDD 4 TB (có khả năng mở rộng)

Mạng: Kết nối mạng tốc độ cao (1 Gbps hoặc cao hơn)

* + 1. Cấu hình phần mềm

1. Hệ điều hành:

Ubuntu Server 20.04 LTS: Hệ điều hành ổn định và được hỗ trợ lâu dài, phù hợp cho các máy chủ ứng dụng và cơ sở dữ liệu.

1. Java Development Kit (JDK):

OpenJDK 11: Phiên bản JDK được hỗ trợ lâu dài, tương thích tốt với Spring Boot.

1. Spring Boot và Spring Cloud:

Spring Boot 2.5.x: Phiên bản mới nhất với nhiều tính năng và cải tiến.

Spring Cloud 2020.x: Hỗ trợ đầy đủ các thành phần cần thiết cho xây dựng microservices.

1. Containerization và Orchestration:

Docker: Sử dụng Docker để container hóa các microservice.

Kubernetes: Sử dụng Kubernetes để quản lý các container, đảm bảo khả năng mở rộng và chịu lỗi.

1. Cơ sở dữ liệu:

MySQL 8.0: Hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ và phổ biến.

Spring Data JPA: Framework để tương tác với cơ sở dữ liệu MySQL.

1. Message Broker:

RabbitMQ: Hệ thống message broker để quản lý giao tiếp giữa các microservice.

1. Giám sát và quản lý log:

Prometheus: Công cụ giám sát và cảnh báo.

Grafana: Công cụ hiển thị dữ liệu giám sát.

ELK Stack (Elasticsearch, Logstash, Kibana): Bộ công cụ quản lý và phân tích log.

1. Hệ thống CI/CD:

Jenkins: Công cụ tự động hóa xây dựng và triển khai.

GitHub Actions: Công cụ tích hợp và triển khai liên tục.

Bảo mật:

Spring Security: Framework bảo mật để quản lý xác thực và ủy quyền.

OAuth2/JWT: Công nghệ xác thực và ủy quyền hiện đại.

1. Email Service:

SMTP Server: Cấu hình SMTP server để gửi email thông báo.

JavaMail: Thư viện Java để gửi email.

***4.1.3 Các công cụ phát triển và quản lý***

1. IDE:

IntelliJ IDEA: Môi trường phát triển tích hợp mạnh mẽ cho Java và Spring Boot.

2. Quản lý mã nguồn:

Git: Hệ thống quản lý mã nguồn phân tán.

GitHub/GitLab: Nền tảng lưu trữ mã nguồn và hỗ trợ CI/CD.

3. Quản lý dự án:

Jira: Công cụ quản lý dự án và theo dõi vấn đề.

Confluence: Công cụ tài liệu và hợp tác.

4. Các công cụ kiểm thử:

JUnit: Framework kiểm thử đơn vị cho Java.

Mockito: Framework mô phỏng đối tượng để kiểm thử đơn vị.

Postman: Công cụ kiểm thử API.

1. Quản lý cấu hình:

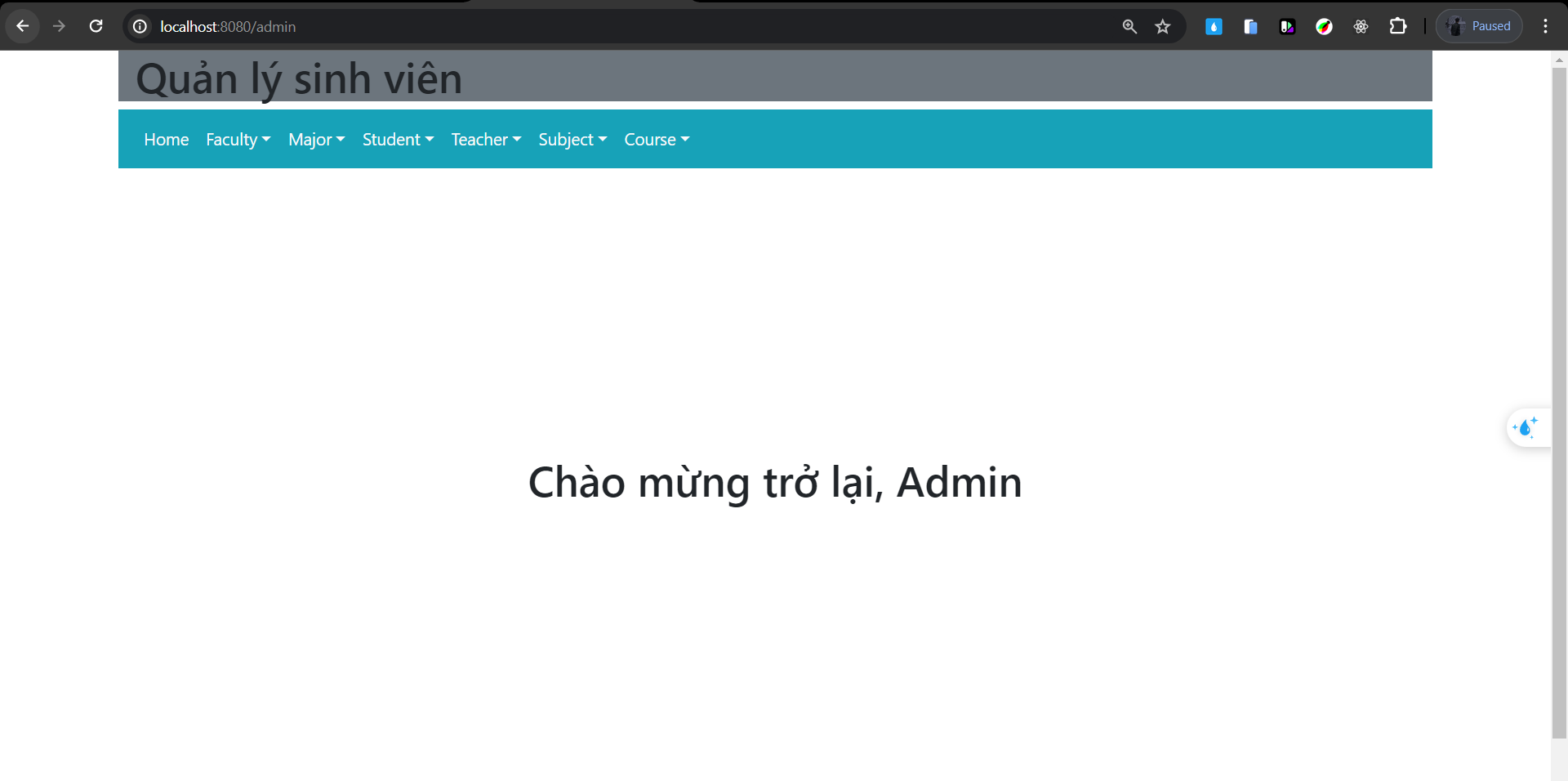
Spring Cloud Config: Quản lý cấu hình phân tán cho các microservice.

Consul/ZooKeeper: Các dịch vụ đăng ký và khám phá cấu hình.

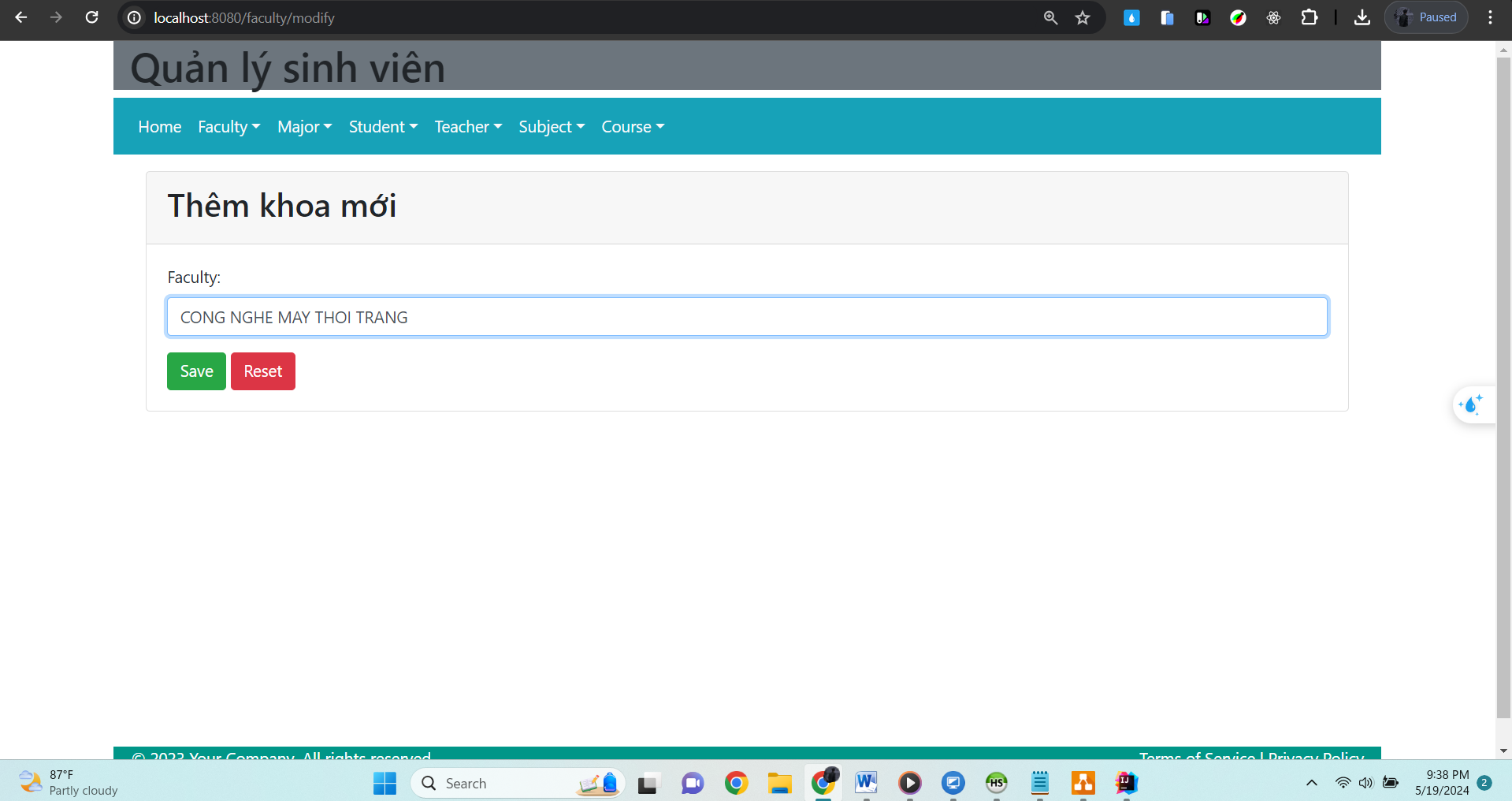
## Giao diện của hệ thống

Màn hình chức năng đã thực hiện.

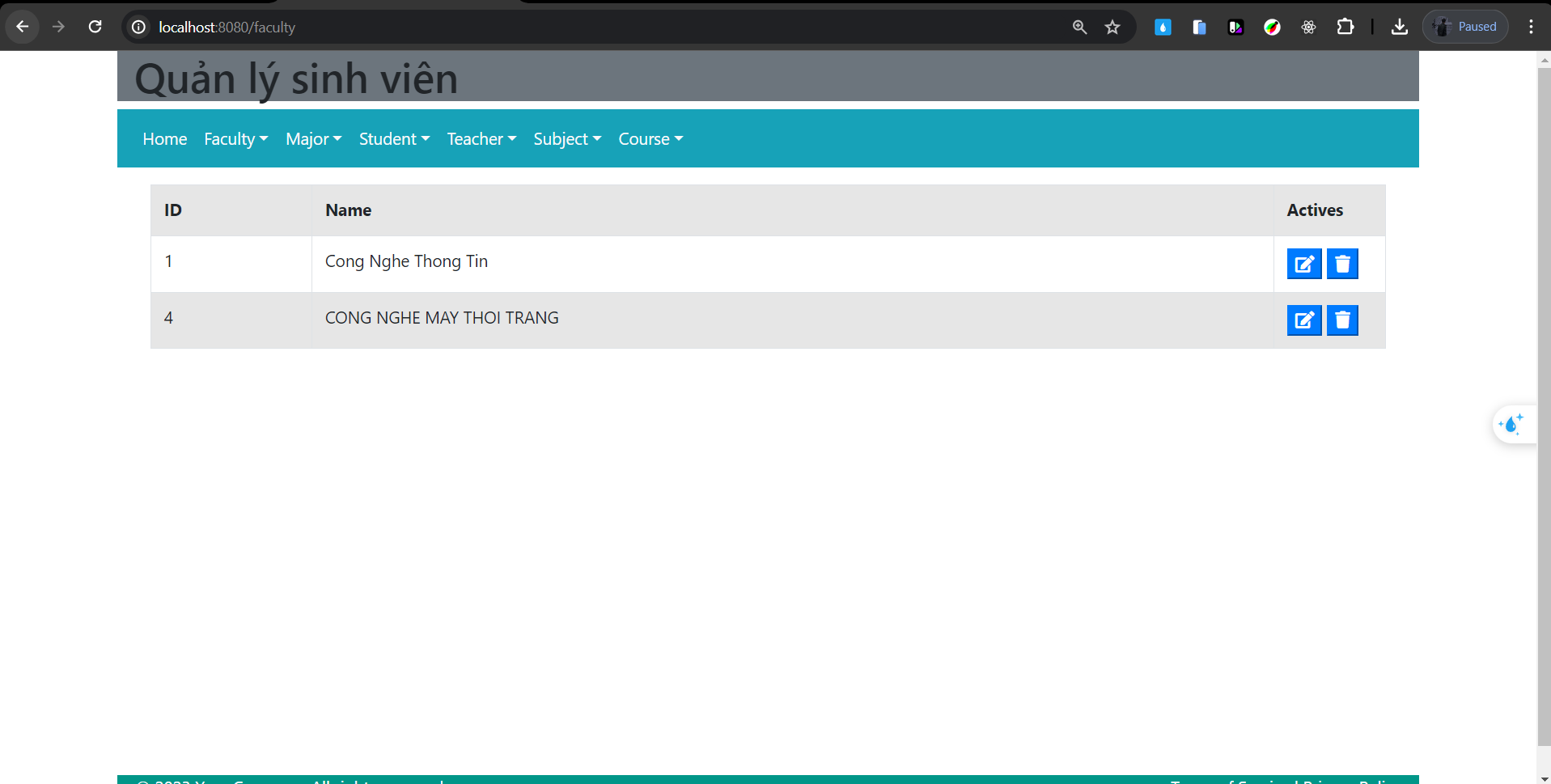
* Bên Admin:
* Giao diện Home:



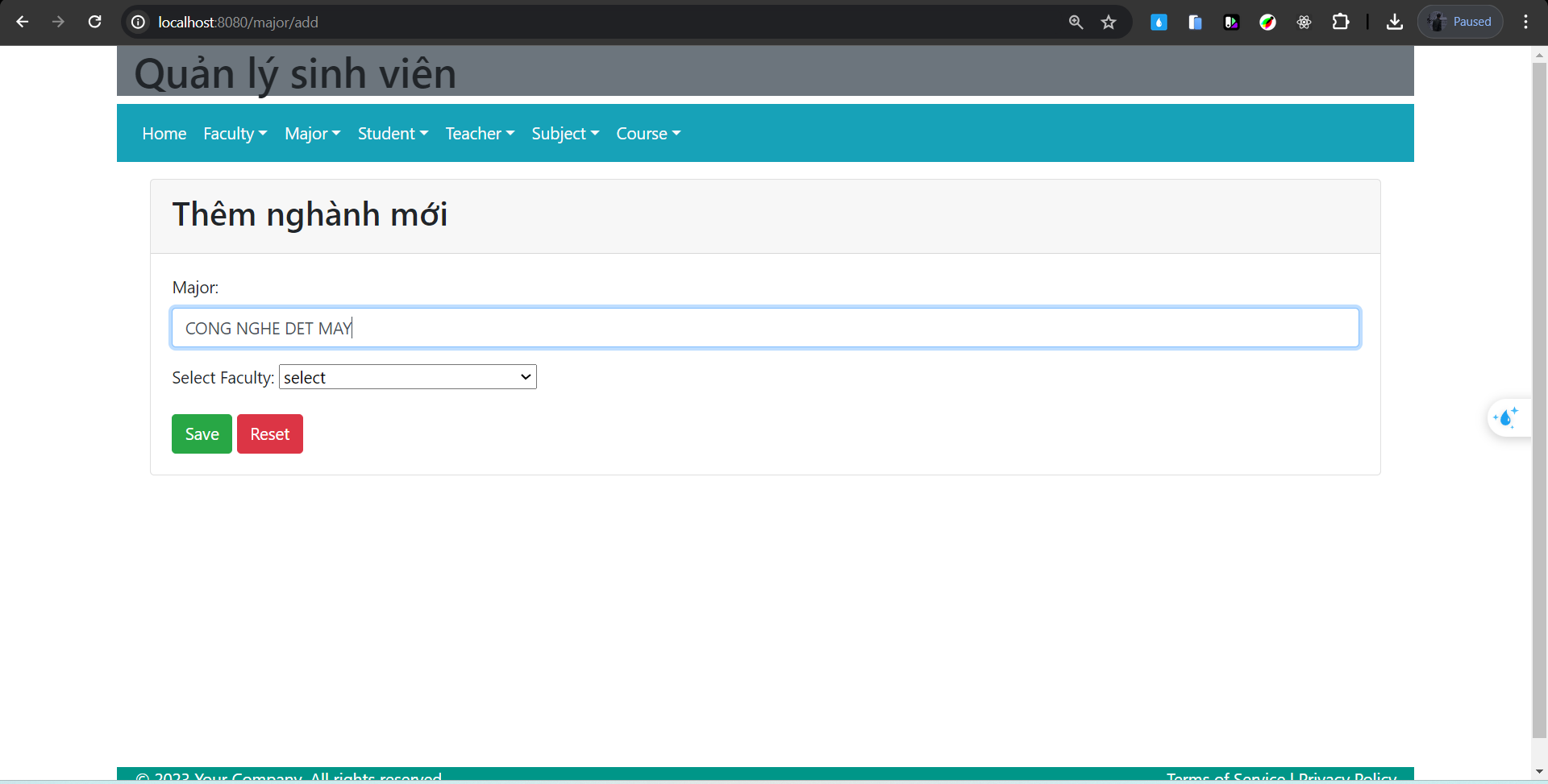
* Thêm Khoa mới



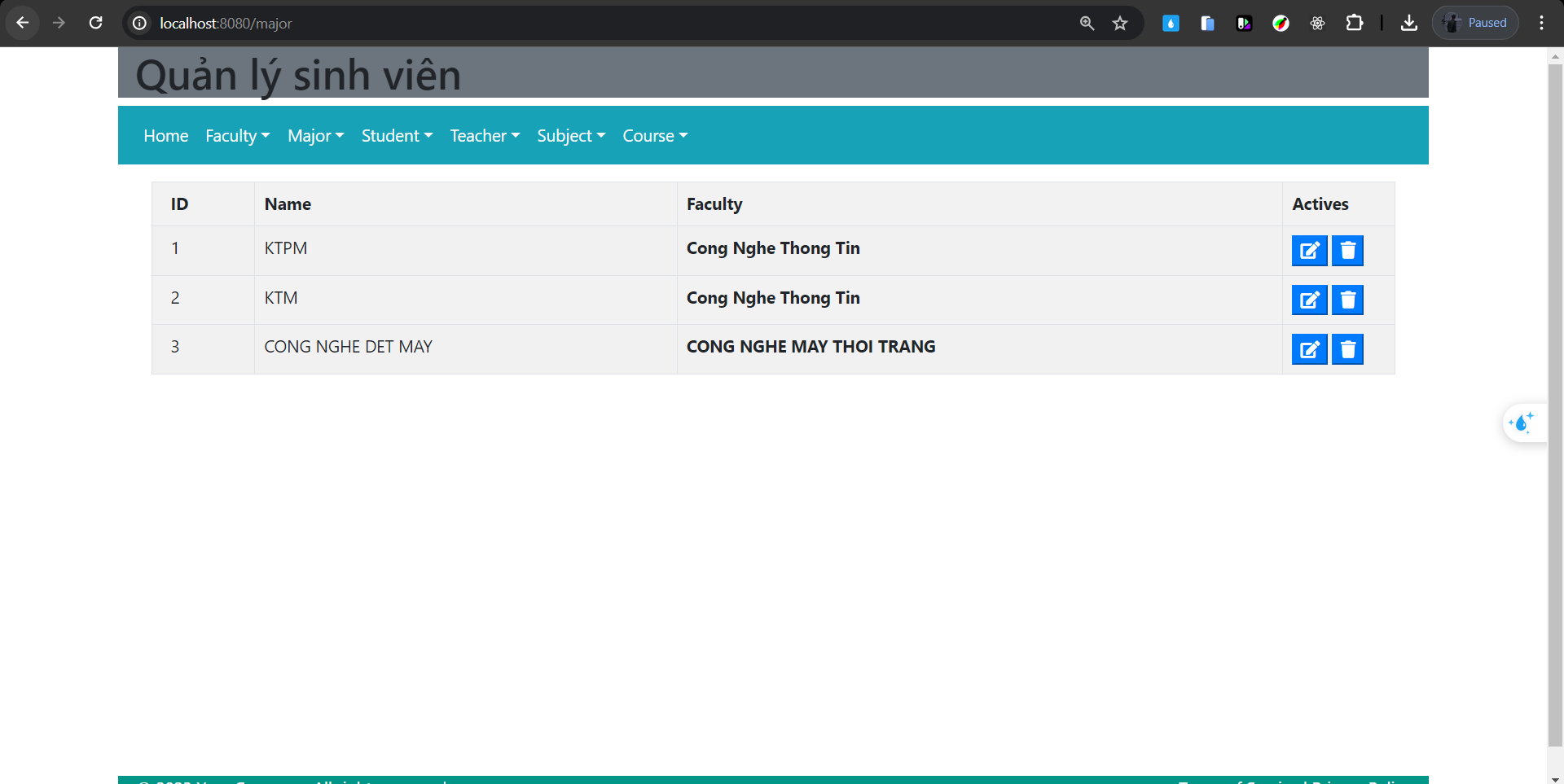
* Danh sách khoa:



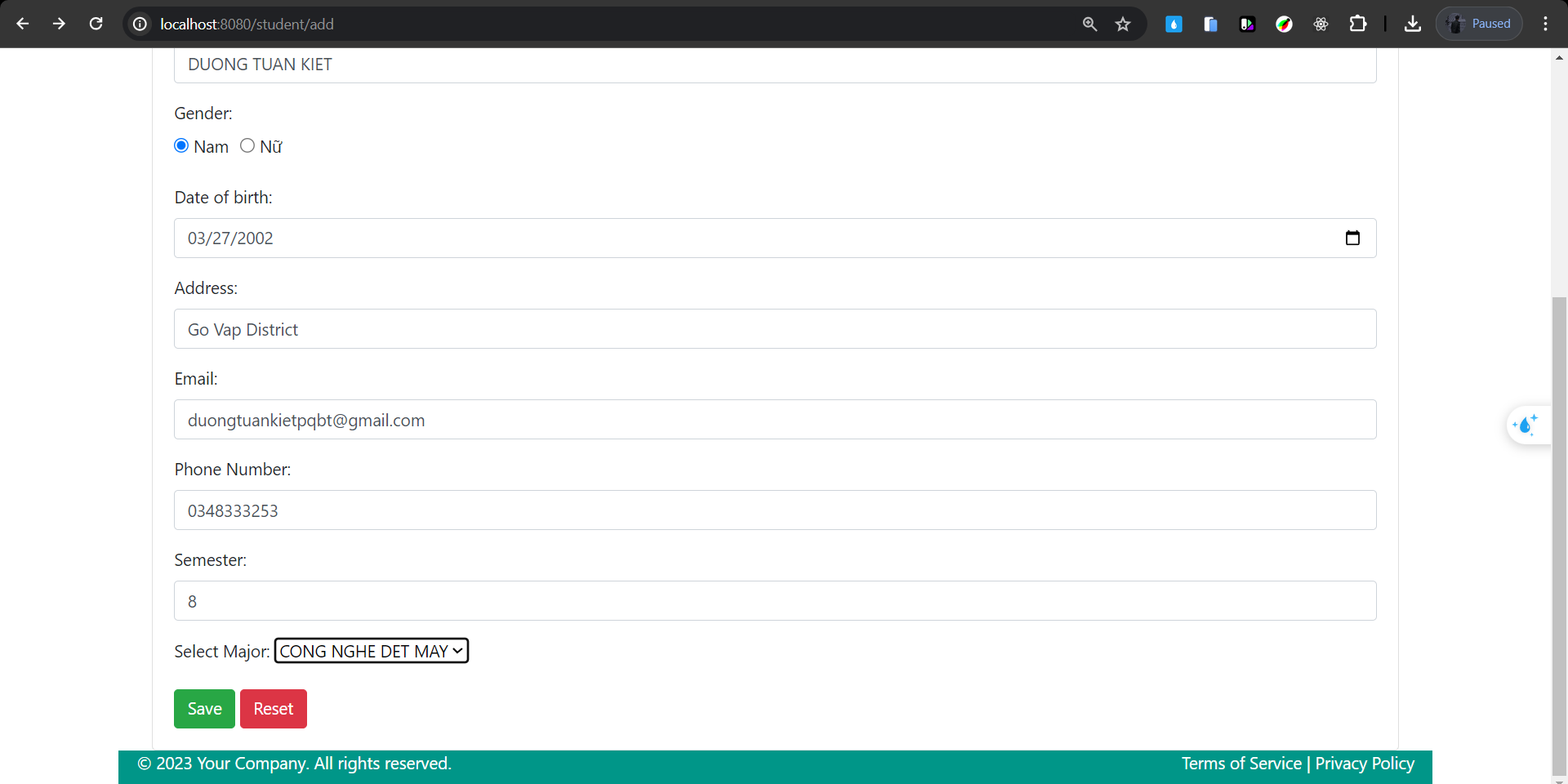
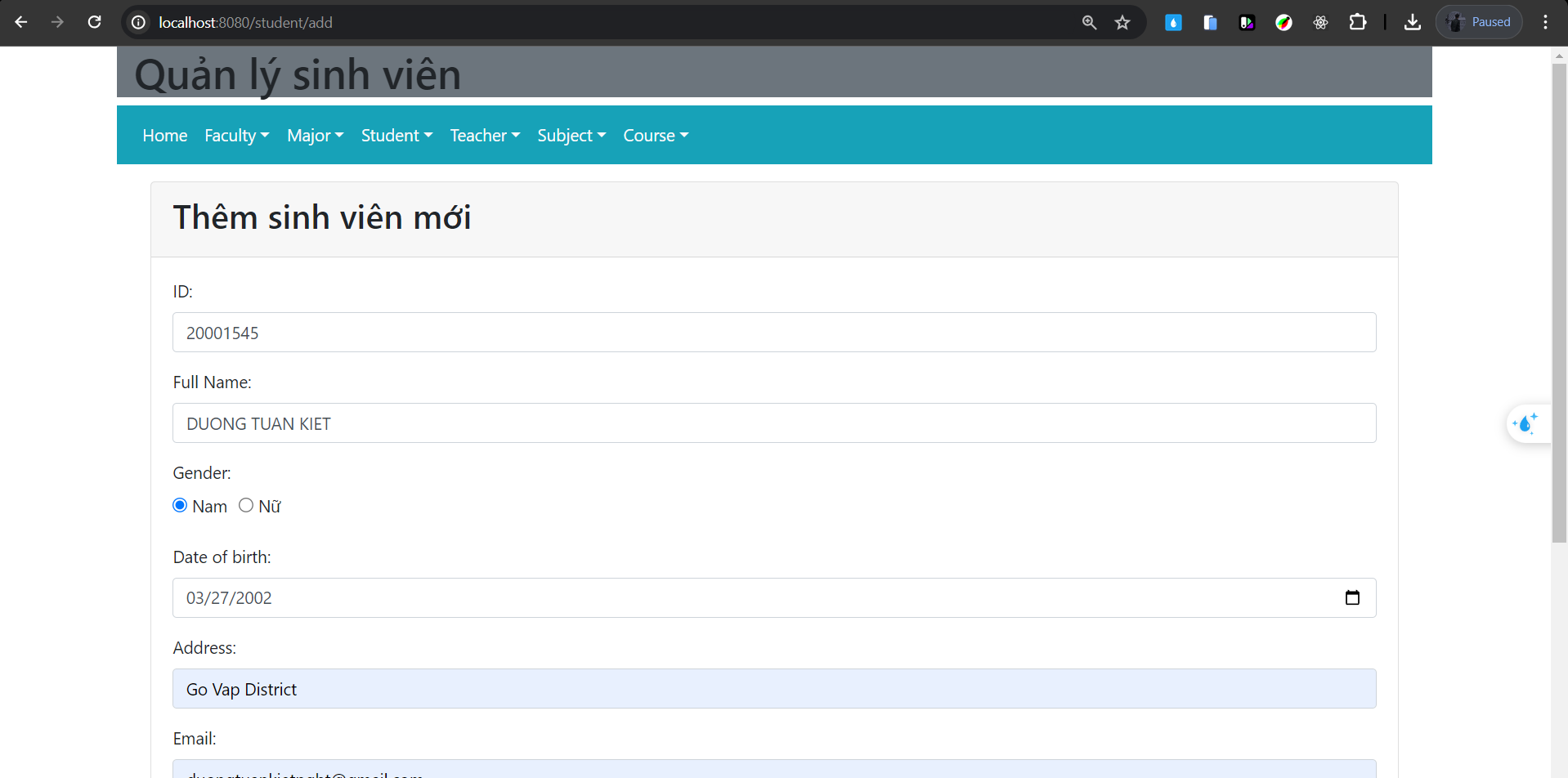
* Thêm ngành mới:



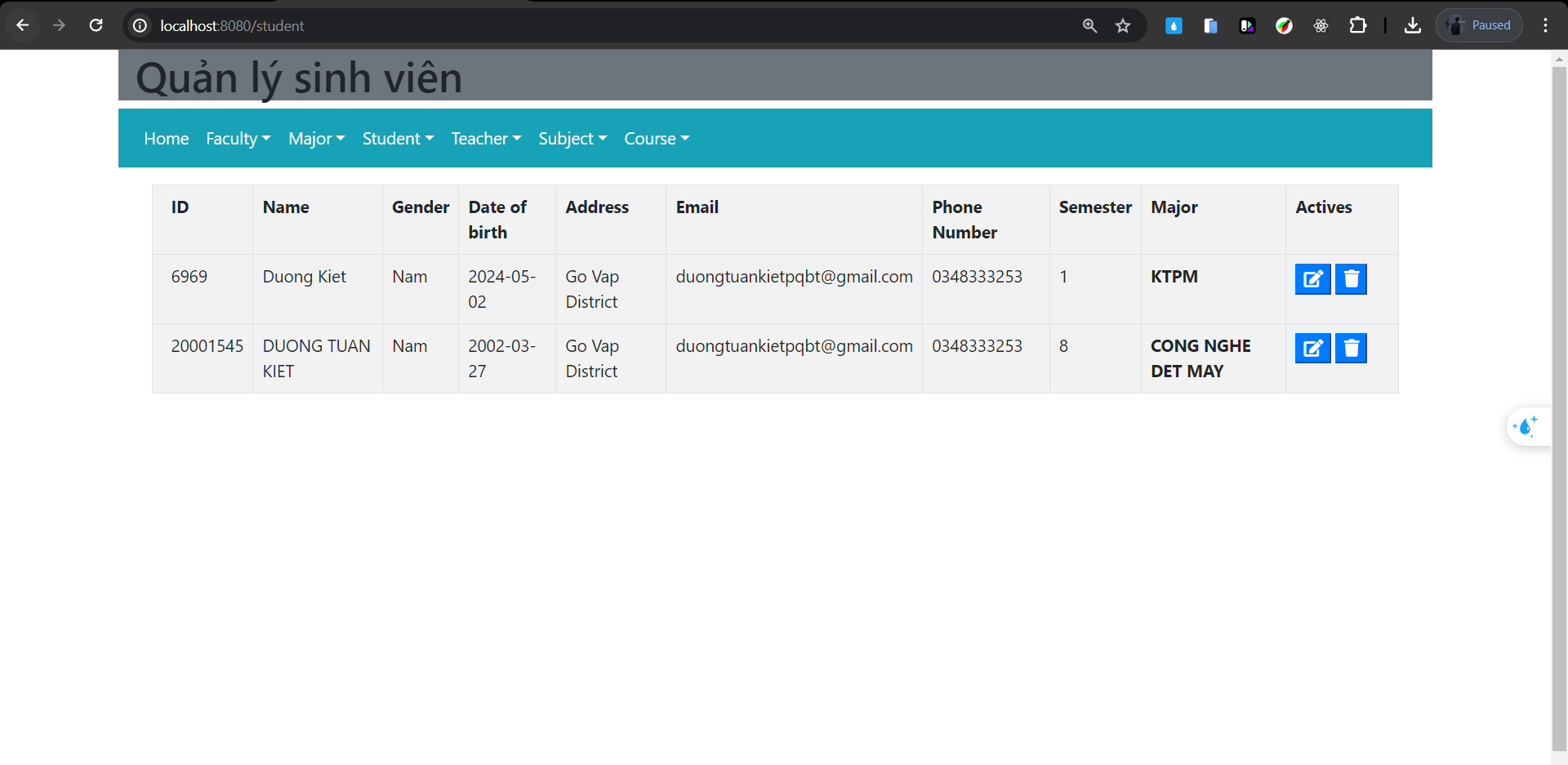
* Danh sách ngành:



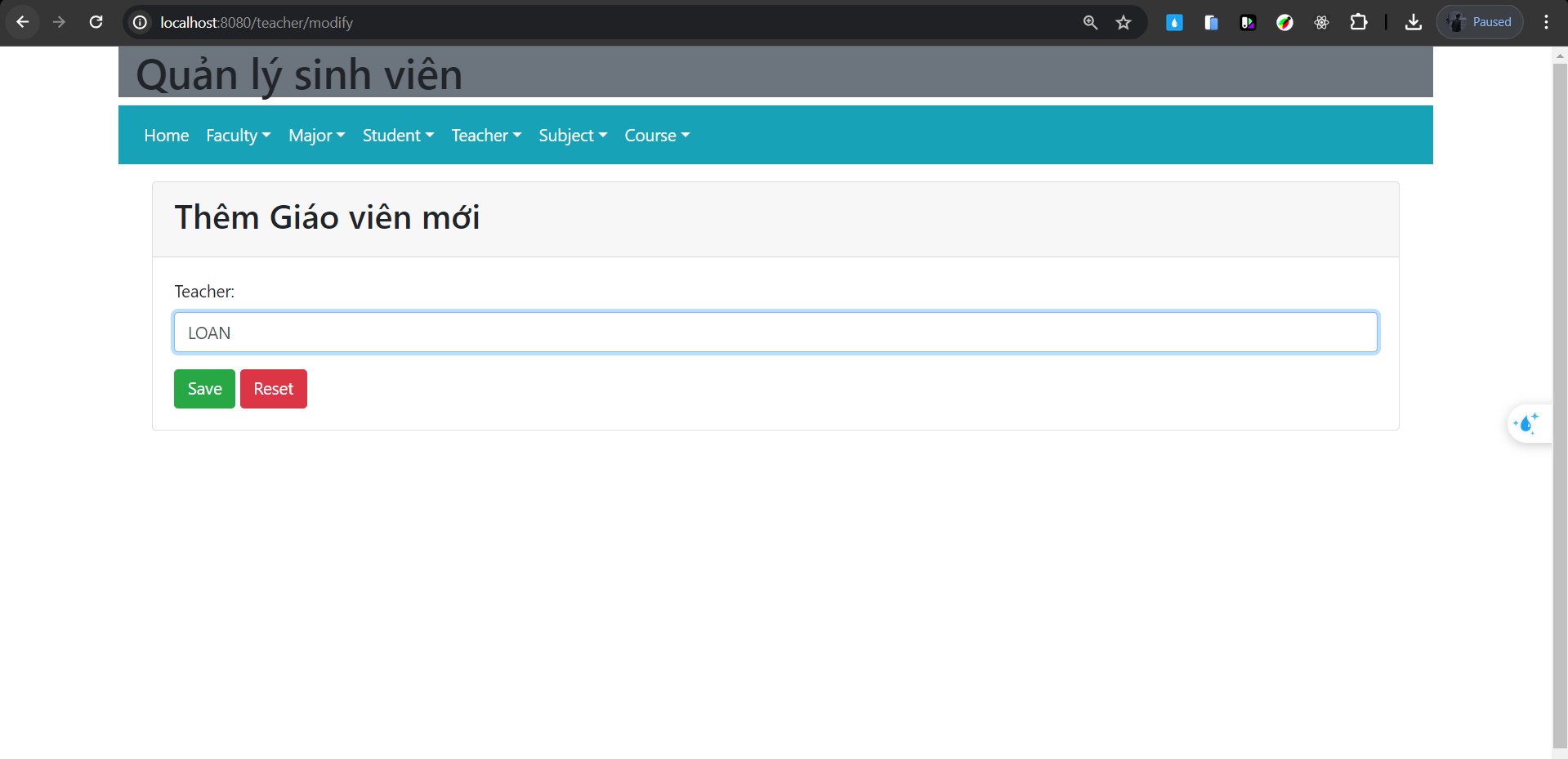
* Thêm sinh viên mới:



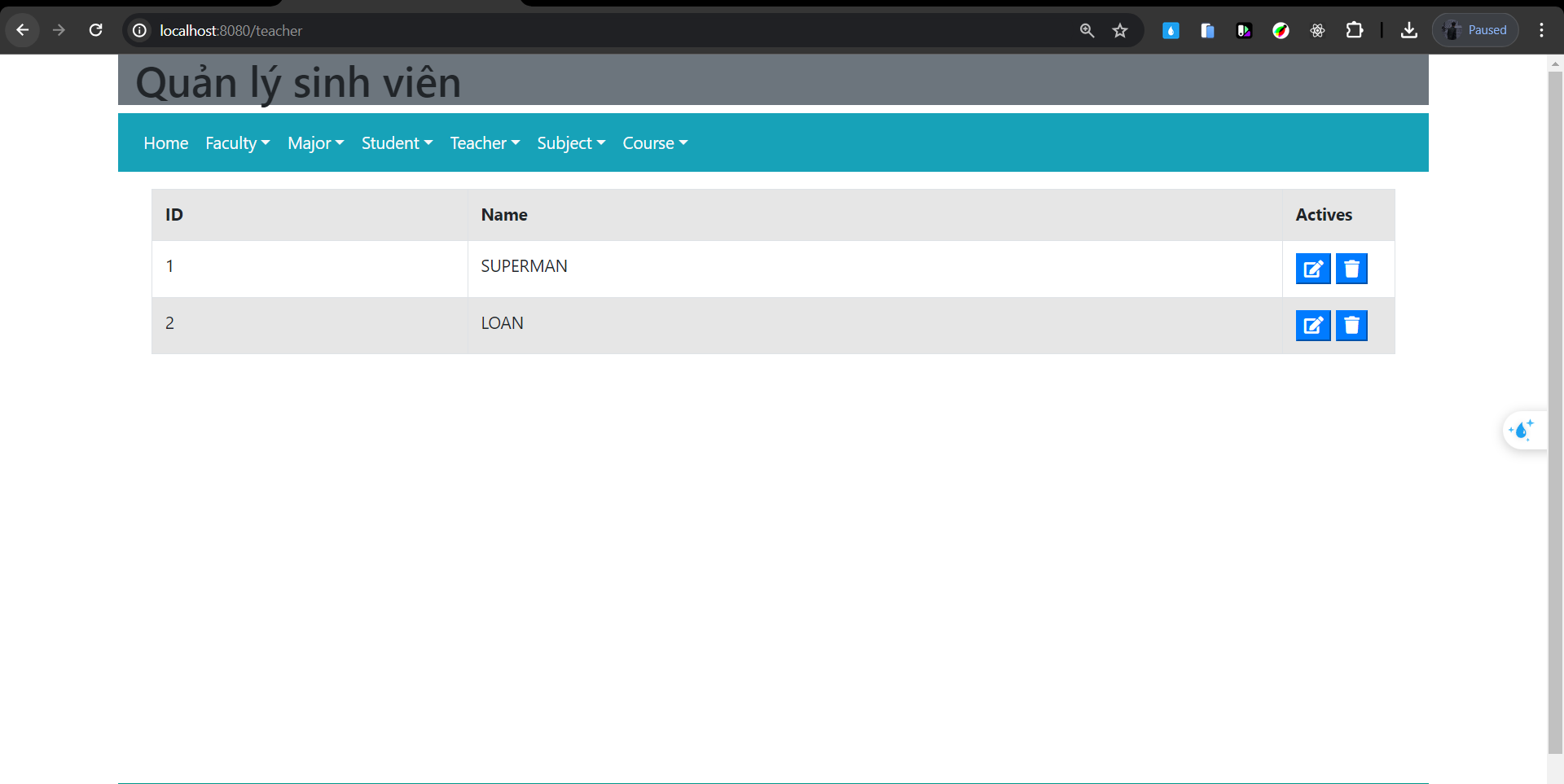
* Danh sách sinh viên vừa thêm:



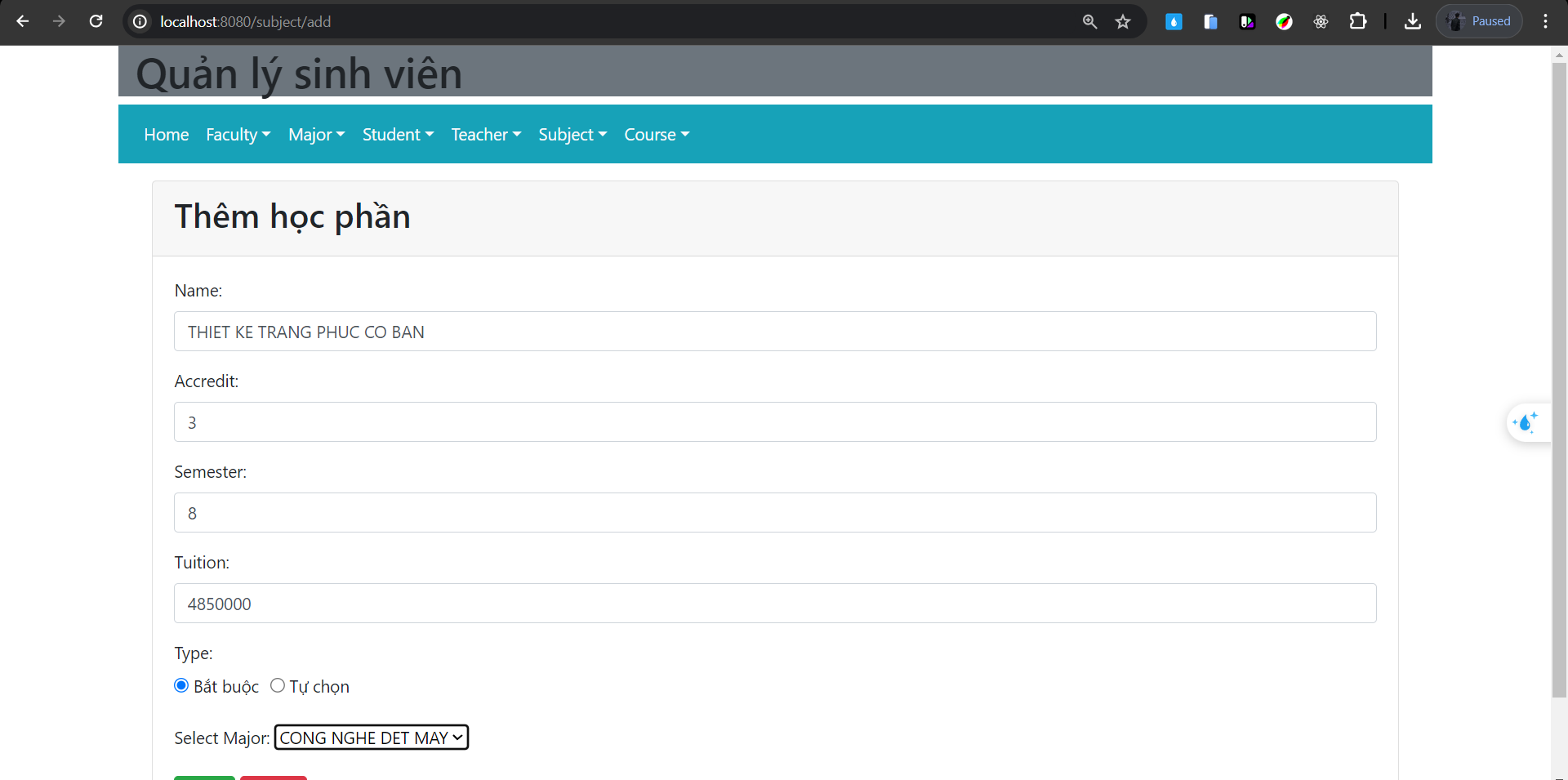
* Thêm 1 giáo viên mới:



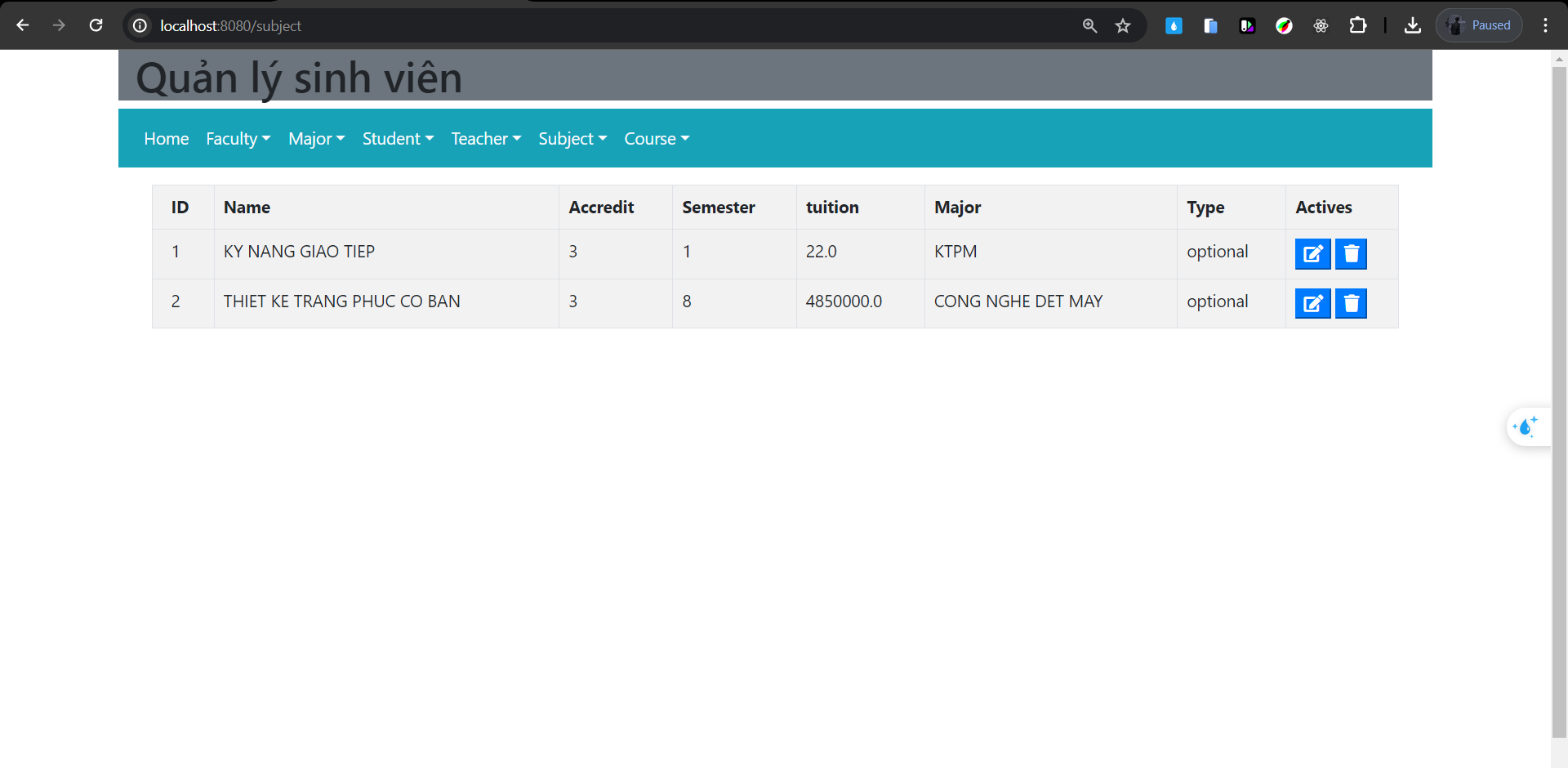
* Danh sách giáo viên vừa thêm:



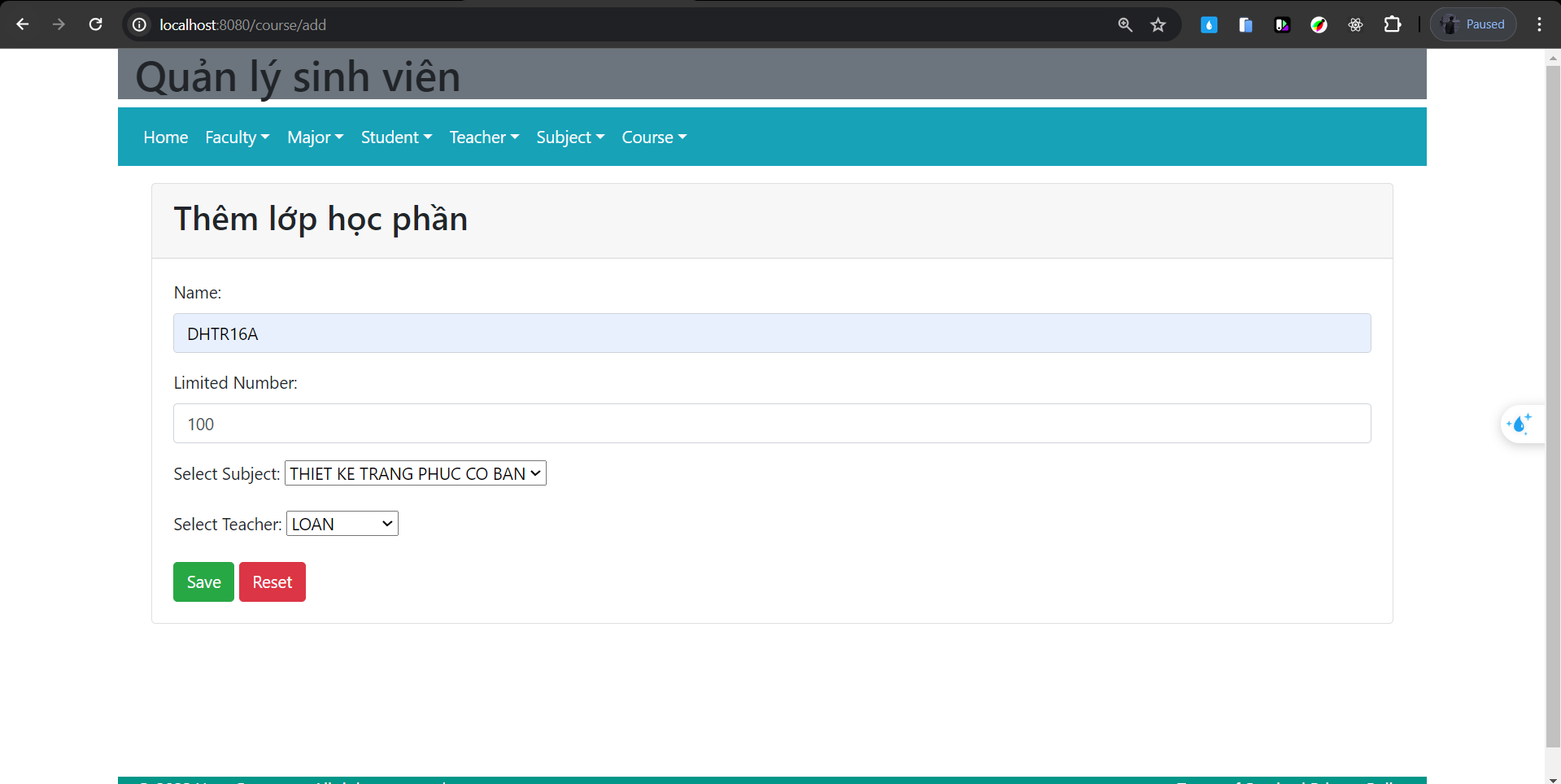
* Thêm học phần:



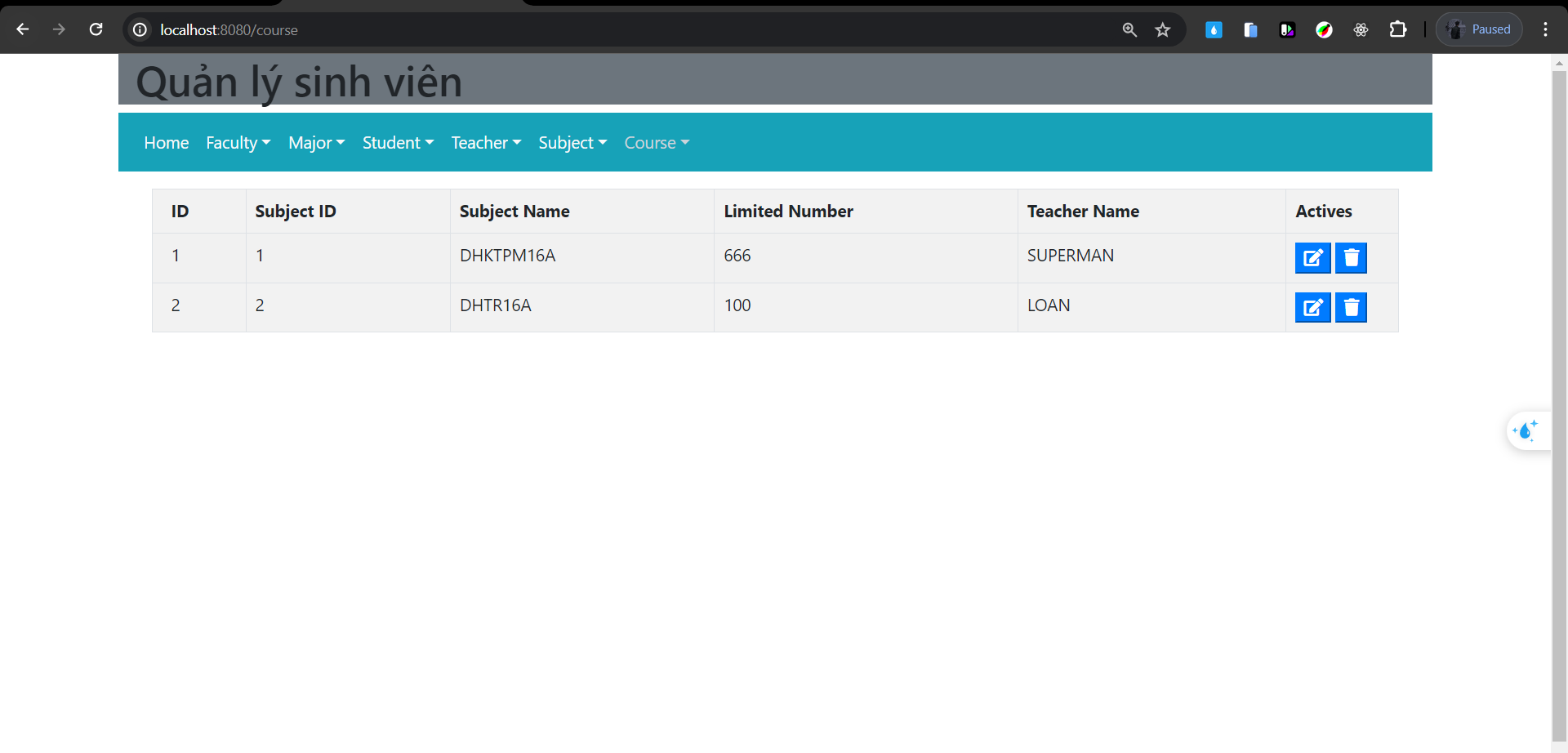
* Danh sách học phần :



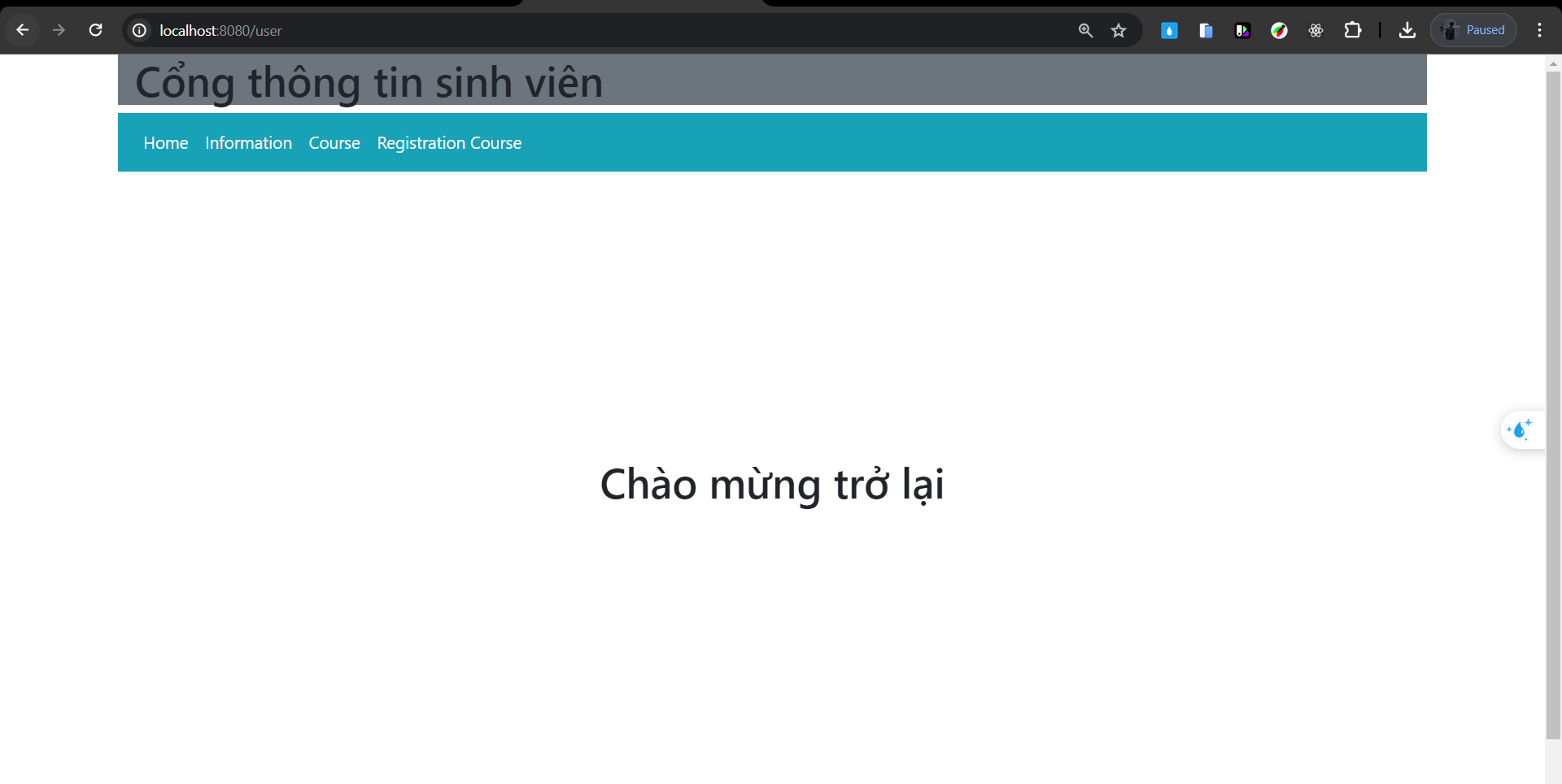
* Thêm lớp học phần:



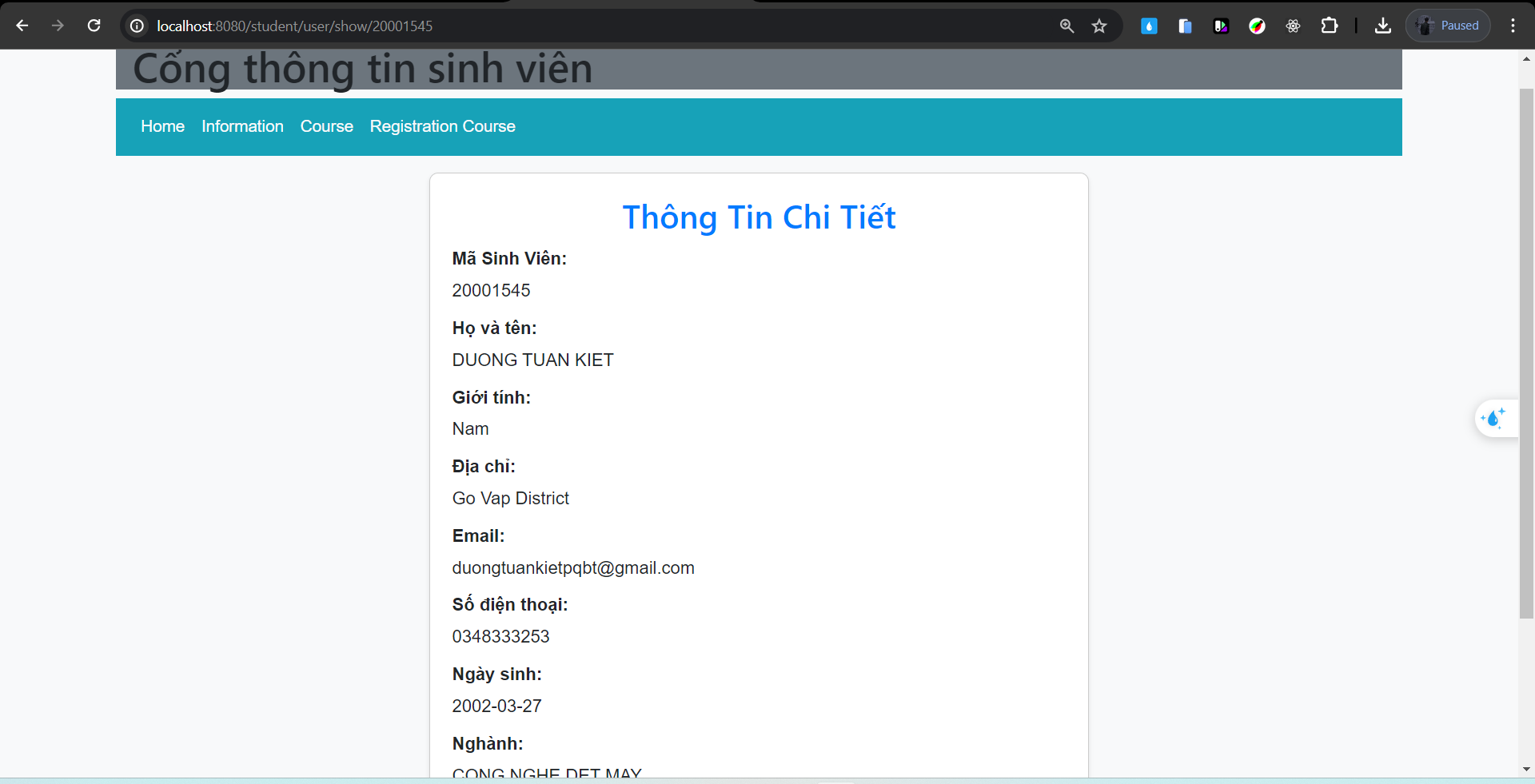
* Danh sách lớp học phần:



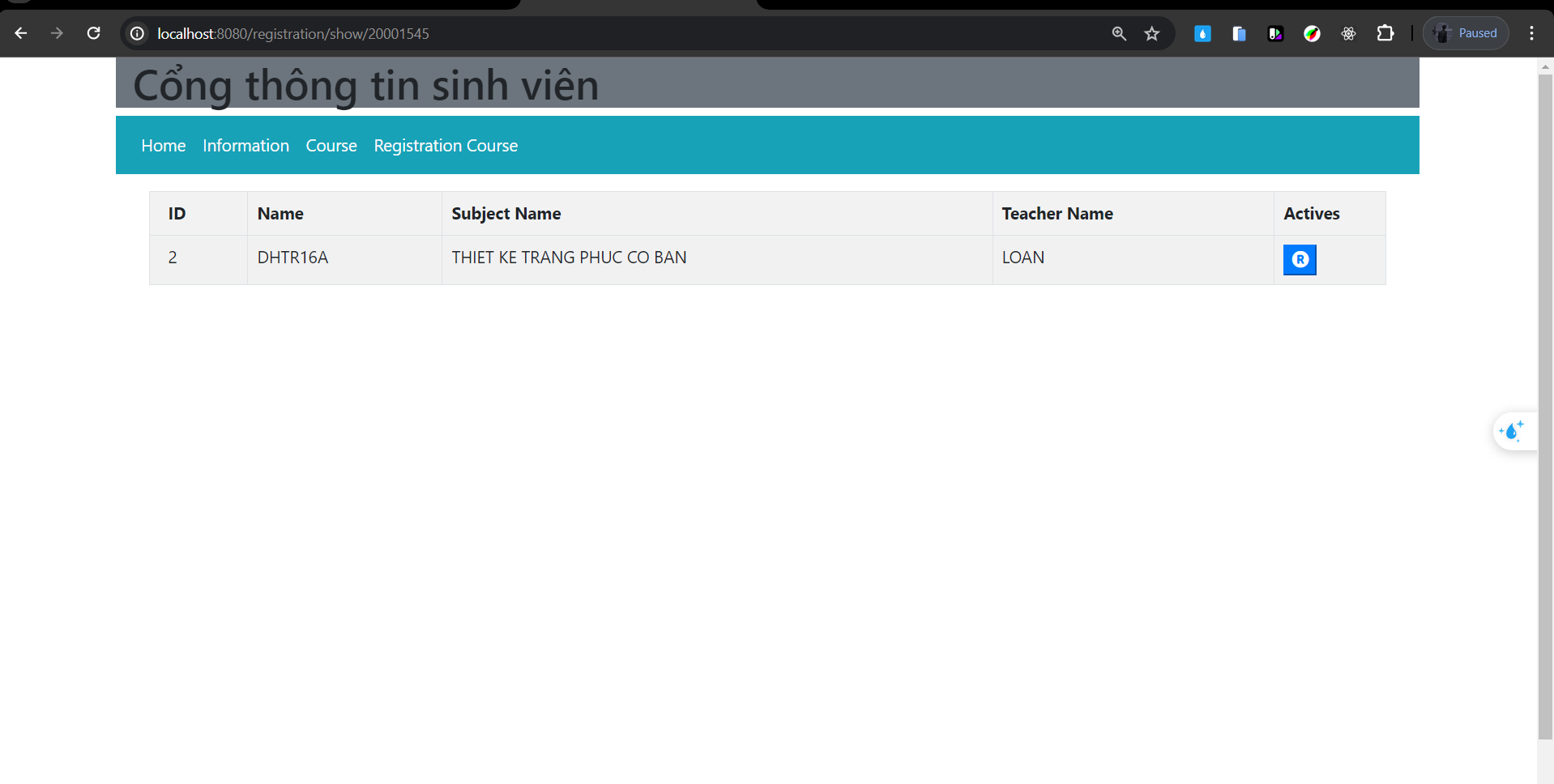
* Bên Sinh Viên:
* Home:

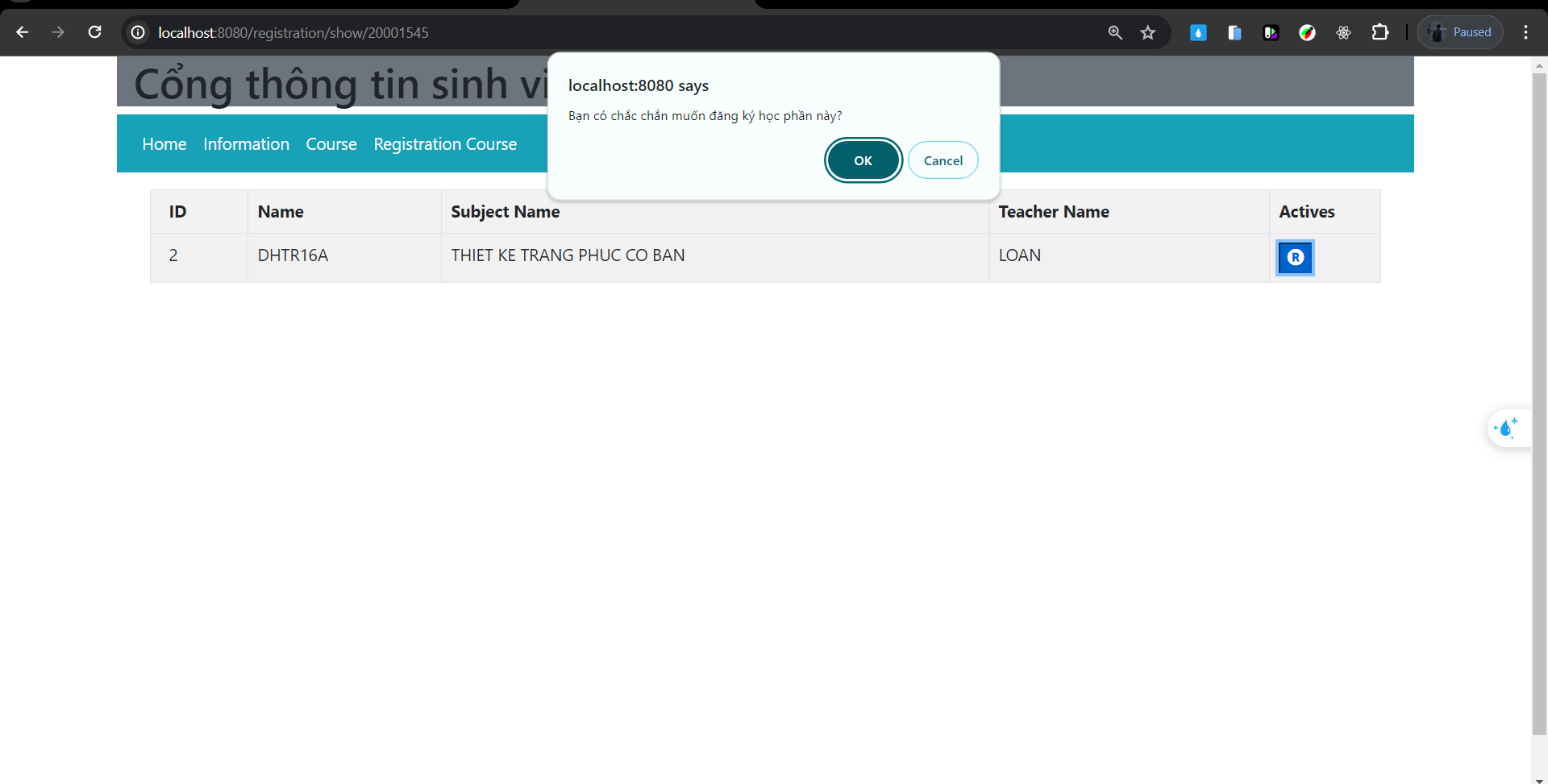


* Thông tin chi tiết:

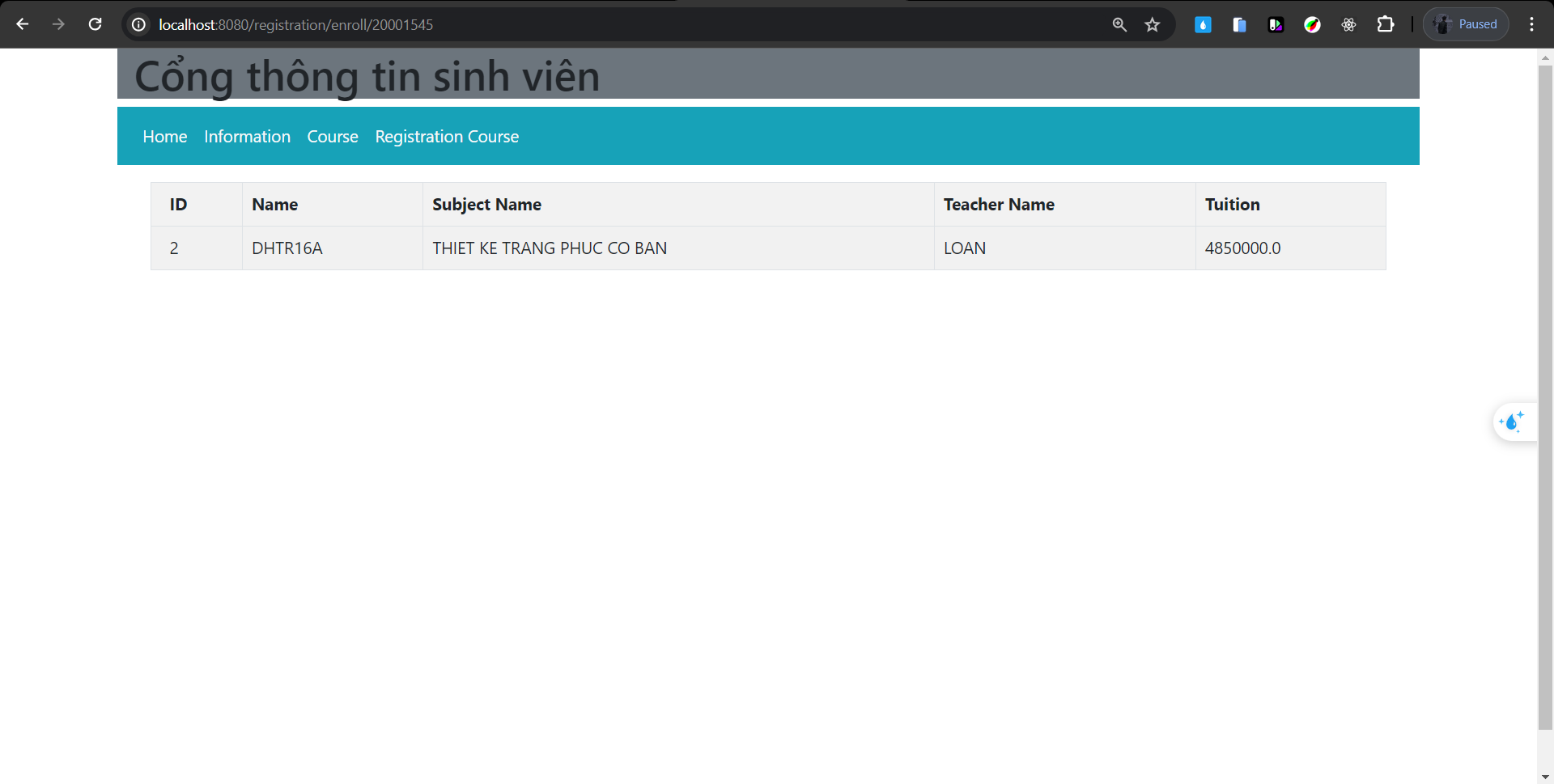


* Đăng ký hóa học:





* Đăng ký khóa học thành công:



## Kế hoạch và hiện thực kiểm thử hệ thống

### Kế hoạch kiểm thử

### Kiểm thử hệ thống

# : KẾT LUẬN

## Kết quả đạt được

1. Đăng nhập:

• Đăng nhập: Cho phép người dùng nhập thông tin đăng nhập (mã số sinh viên và mật khẩu) để truy cập vào tài khoản của họ.

2. Phân chia chức năng theo microservice:

Hệ thống được chia thành các microservice độc lập, mỗi service quản lý một khía cạnh cụ thể của hệ thống: quản lý thông tin sinh viên, quản lý môn học và lớp tín chỉ, quản lý đăng ký học phần, gửi thông báo, và quản lý thông tin học tập.

Sự phân chia này giúp cải thiện khả năng mở rộng và bảo trì hệ thống, cho phép phát triển và triển khai các phần của hệ thống một cách độc lập.

3. Tích hợp Spring Boot và kiến trúc MVC:

Sử dụng Spring Boot để xây dựng từng microservice với cấu trúc rõ ràng theo mô hình MVC (Model-View-Controller).

Các thành phần chính của MVC như Model, Repository, Service, và Controller được triển khai một cách nhất quán trong từng microservice.

4.Chức năng quản lý sinh viên và cựu sinh viên:

Microservice User Service đã cung cấp đầy đủ chức năng quản lý thông tin cá nhân của sinh viên, bao gồm đăng ký, cập nhật và truy xuất thông tin sinh viên.

Hệ thống còn hỗ trợ quản lý thông tin cựu sinh viên để theo dõi quá trình làm việc và thực hiện các hoạt động liên quan.

5.Quản lý môn học và lớp tín chỉ:

Microservice Course Service cho phép quản lý thông tin chi tiết về môn học, bao gồm môn học tiên quyết và danh sách các lớp tín chỉ.

Sinh viên có thể xem danh sách các môn học khả dụng, các môn tiên quyết và đăng ký vào các lớp tín chỉ phù hợp.

6.Xử lý đăng ký và danh sách chờ:

Microservice Enrollment Service xử lý quá trình đăng ký môn học của sinh viên, kiểm tra giới hạn tín chỉ và quản lý danh sách chờ cho các lớp đã đầy.

Hệ thống đảm bảo sinh viên không thể đăng ký quá số tín chỉ cho phép trong mỗi học kỳ.

7. Thông báo và xác nhận đăng ký:

Microservice Notification Service gửi email thông báo xác nhận đăng ký thành công và thông báo quyết định đóng học phí.

Hệ thống đảm bảo sinh viên nhận được thông tin kịp thời và chính xác về trạng thái đăng ký của mình.

8. Quản lý thông tin học tập và xét tốt nghiệp:

Microservice Transcript Service cung cấp chức năng quản lý và hiển thị thông tin học tập của sinh viên, bao gồm số tín chỉ đã đạt, điểm từng môn học, và điểm trung bình tích lũy.

Hệ thống hỗ trợ quá trình xét tốt nghiệp, cấp bằng và cập nhật danh sách cựu sinh viên khi sinh viên đạt đủ các tiêu chuẩn.

9. Tích hợp và triển khai dễ dàng:

Việc sử dụng Spring Boot giúp cho việc cấu hình và triển khai các microservice trở nên dễ dàng và nhanh chóng.

Hệ thống có thể dễ dàng mở rộng để phục vụ số lượng sinh viên và lớp học lớn hơn trong tương lai.

## Hạn chế của đồ án

1. Độ phức tạp trong việc triển khai và quản lý microservice:

Việc chia nhỏ hệ thống thành nhiều microservice tạo ra sự phức tạp trong việc triển khai, quản lý và giám sát các dịch vụ.

Yêu cầu kỹ thuật và công cụ quản lý microservice như Docker, Kubernetes, và các giải pháp giám sát như Prometheus và Grafana để đảm bảo hệ thống hoạt động trơn tru.

1. Giao tiếp giữa các microservice:

Giao tiếp giữa các microservice chủ yếu dựa vào HTTP REST API, có thể gây ra vấn đề về hiệu suất và độ trễ khi số lượng yêu cầu tăng cao.

Việc quản lý và xử lý lỗi trong giao tiếp giữa các dịch vụ cần được cải thiện để đảm bảo tính ổn định của hệ thống.

1. Đồng bộ hóa và nhất quán dữ liệu:

Việc đảm bảo tính nhất quán dữ liệu giữa các microservice là một thách thức lớn, đặc biệt khi có các giao dịch phân tán.

Hệ thống hiện tại có thể gặp khó khăn trong việc xử lý các tình huống phức tạp liên quan đến đồng bộ hóa dữ liệu và xử lý sự kiện bất đồng bộ.

1. Bảo mật và quyền riêng tư:

Hệ thống cần phải cải thiện các biện pháp bảo mật để bảo vệ thông tin cá nhân và học tập của sinh viên khỏi các mối đe dọa tiềm ẩn.

Quản lý quyền truy cập và xác thực người dùng cần được xem xét kỹ lưỡng để ngăn chặn các truy cập trái phép.

1. Tính năng và giao diện người dùng:

Giao diện người dùng của hệ thống hiện tại có thể chưa thân thiện và trực quan đối với tất cả người dùng.

Cần cải tiến giao diện và trải nghiệm người dùng để đảm bảo mọi chức năng được sử dụng một cách hiệu quả và dễ dàng.

1. Khả năng mở rộng và hiệu suất:

Mặc dù hệ thống đã được thiết kế để có khả năng mở rộng, nhưng trong thực tế, khi số lượng người dùng và dữ liệu tăng cao, có thể gặp vấn đề về hiệu suất và tải hệ thống.

Cần thực hiện các bài kiểm tra tải và tối ưu hóa hiệu suất để đảm bảo hệ thống hoạt động mượt mà dưới áp lực cao.

1. Quản lý lỗi và khắc phục sự cố:

Hệ thống cần có cơ chế quản lý lỗi và khắc phục sự cố mạnh mẽ hơn để giảm thiểu thời gian chết và đảm bảo dịch vụ liên tục.

Cần xây dựng các chiến lược phục hồi sau sự cố và sao lưu dữ liệu định kỳ để bảo vệ hệ thống trước các tình huống không mong muốn.

1. Tích hợp với các hệ thống khác:

Hệ thống hiện tại có thể gặp khó khăn trong việc tích hợp với các hệ thống quản lý khác của nhà trường (như hệ thống tài chính, quản lý nhân sự, thư viện, v.v.).

Cần cải thiện khả năng tích hợp và tương tác với các hệ thống bên ngoài để tạo ra một hệ sinh thái quản lý giáo dục hoàn chỉnh.

## Hướng phát triển

1. Tối ưu hóa hiệu suất và khả năng mở rộng:

Cải tiến giao tiếp giữa các microservice: Sử dụng các giao thức và công nghệ giao tiếp tiên tiến hơn như gRPC để cải thiện hiệu suất và giảm độ trễ.

Tối ưu hóa cơ sở dữ liệu: Xem xét việc sử dụng các giải pháp cơ sở dữ liệu phân tán và tối ưu hóa các truy vấn để cải thiện hiệu suất truy cập và xử lý dữ liệu.

Cân bằng tải và tự động mở rộng: Áp dụng các kỹ thuật cân bằng tải và tự động mở rộng (auto-scaling) để đảm bảo hệ thống có thể đáp ứng được yêu cầu khi lượng người dùng tăng cao.

1. Cải thiện giao diện và trải nghiệm người dùng:

Thiết kế giao diện người dùng thân thiện: Nâng cấp giao diện người dùng để trực quan hơn, dễ sử dụng hơn, đặc biệt là đối với sinh viên và giảng viên.

Phản hồi người dùng: Thu thập và phân tích phản hồi từ người dùng để liên tục cải thiện trải nghiệm người dùng.

1. Tăng cường bảo mật:

Xác thực và ủy quyền: Sử dụng các phương pháp xác thực và ủy quyền mạnh mẽ hơn như OAuth2, JWT để bảo vệ hệ thống.

Mã hóa dữ liệu: Mã hóa dữ liệu cả trong quá trình truyền tải và lưu trữ để đảm bảo an toàn thông tin cá nhân và học tập của sinh viên.

Kiểm tra bảo mật thường xuyên: Thực hiện các kiểm tra bảo mật định kỳ để phát hiện và khắc phục các lỗ hổng bảo mật kịp thời.

1. Tích hợp với các hệ thống khác:

Tích hợp hệ thống tài chính: Kết nối với hệ thống tài chính của nhà trường để tự động hóa quá trình thanh toán học phí và các khoản phí khác.

Tích hợp hệ thống quản lý nhân sự: Kết nối với hệ thống quản lý nhân sự để đồng bộ hóa thông tin giảng viên và nhân viên.

Tích hợp thư viện và hệ thống học liệu: Kết nối với thư viện và hệ thống quản lý học liệu để sinh viên có thể truy cập tài liệu học tập dễ dàng.

1. Phát triển các tính năng mới:

Hỗ trợ học tập trực tuyến: Phát triển thêm các tính năng hỗ trợ học tập trực tuyến như lớp học ảo, tài liệu học tập trực tuyến, và các công cụ hỗ trợ giảng dạy khác.

Quản lý dự án và thực tập: Thêm chức năng quản lý dự án học tập và thực tập để theo dõi tiến độ và kết quả thực tập của sinh viên.

Công cụ hỗ trợ học tập cá nhân hóa: Sử dụng trí tuệ nhân tạo và machine learning để phân tích dữ liệu học tập và cung cấp các gợi ý học tập cá nhân hóa cho từng sinh viên.

1. Nâng cao quản lý và phân tích dữ liệu:

Báo cáo và thống kê: Phát triển các công cụ báo cáo và thống kê để hỗ trợ quản lý nhà trường trong việc đưa ra quyết định.

Phân tích dữ liệu học tập: Sử dụng các kỹ thuật phân tích dữ liệu để theo dõi và cải thiện hiệu quả học tập của sinh viên.

1. Tăng cường hỗ trợ và phản hồi từ người dùng:

Hệ thống hỗ trợ trực tuyến: Xây dựng hệ thống hỗ trợ trực tuyến để giải đáp thắc mắc và hỗ trợ người dùng một cách nhanh chóng và hiệu quả.

Khảo sát và phản hồi: Triển khai các khảo sát và thu thập phản hồi từ người dùng để liên tục cải thiện hệ thống.

1. Đào tạo và nâng cao năng lực sử dụng hệ thống:

Tài liệu hướng dẫn sử dụng: Cung cấp tài liệu hướng dẫn sử dụng chi tiết và dễ hiểu cho sinh viên, giảng viên và nhân viên nhà trường.

Chương trình đào tạo: Tổ chức các chương trình đào tạo và workshop để nâng cao năng lực sử dụng hệ thống cho người dùng.

# TÀI LIỆU THAM KHẢO

Các tài liệu Tiếng Anh

1. "Spring Boot Reference Documentation." Accessed May 2024. https://docs.spring.io/spring-boot/docs/current/reference/html/.
2. Richardson, Chris. "Microservices Patterns: With examples in Java." Manning Publications, 2018.
3. Newman, Sam. "Building Microservices: Designing Fine-Grained Systems." O'Reilly Media, 2015.
4. Carnell, John. "Spring Microservices in Action." Manning Publications, 2017.
5. Richardson, Leonard, and Sam Ruby. "RESTful Web Services." O'Reilly Media, 2007.
6. Burns, Brendan, Joe Beda, and Kelsey Hightower. "Kubernetes: Up & Running: Dive into the Future of Infrastructure." O'Reilly Media, 2017.
7. Bloch, Joshua. "Effective Java." Addison-Wesley, 2018.
8. Kleppmann, Martin. "Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems." O'Reilly Media, 2017.
9. Subramanian, Dinesh. "Mastering Spring Boot 2.0: Build modern, cloud-native, and distributed systems using Spring Boot." Packt Publishing, 2017.
10. Rahman, Magnus Larsson. "Hands-On Microservices with Spring Boot and Spring Cloud: Build and deploy Java microservices using Spring Cloud, Istio, and Kubernetes." Packt Publishing, 2019.
11. Cosmina, Felipe Gutierrez. "Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices." Apress, 2018.
12. Walls, Craig. "Spring in Action." Manning Publications, 2018.
13. Newman, Sam. "Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith." O'Reilly Media, 2019.

Các tài liệu từ Internet

1. <https://docs.spring.io/spring-cloud/docs/current/reference/html/>
2. <https://restfulapi.net/>
3. <https://kubernetes.io/docs/>
4. <https://docs.docker.com/>
5. <https://microservices.io/>
6. <https://www.baeldung.com/spring-tutorial>
7. <https://stackoverflow.com/questions/tagged/spring-boot>
8. <https://spring.io/guides>
9. <https://github.com/spring-projects>
10. <https://dzone.com/spring-boot>
11. <https://medium.com/tag/spring-boot>
12. <https://www.thoughtworks.com/insights/blog/spring-boot-practice>
13. <https://dev.to/t/springboot>
14. <https://www.youtube.com/c/SpringFrameworkGuru>
15. https://docs.aws.amazon.com/