**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



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

***Nhóm 34 - 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)

[1.5 Mô hình MVC (Model-View-Controller) là một lựa chọn hợp lý cho hệ thống quản lý lớp họ***c*** 1](#_Toc14693509)

[1.6 Một số tình huống trong đăng ký học phần và lý do chọn kiến trúc 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.

***1.5 Mô hình MVC (Model-View-Controller) là một lựa chọn hợp lý cho hệ thống quản lý lớp học tín chỉ vì các lý do sau:***

1. Phân tách rõ ràng giữa các thành phần:

- Model: Quản lý dữ liệu và logic liên quan đến dữ liệu. Trong hệ thống này, Model sẽ quản lý các đối tượng như sinh viên, môn học, lớp học, đăng ký học phần, và thông tin học tập.

- View: Quản lý giao diện người dùng. View sẽ hiển thị các trang như danh sách môn học, form đăng ký, thông tin học tập, thời khóa biểu, và thông báo.

- Controller: Xử lý các yêu cầu từ người dùng, gọi Model để lấy dữ liệu và sau đó chọn View để hiển thị kết quả. Controller sẽ xử lý các thao tác như đăng nhập, đăng ký học phần, xác nhận đăng ký, và cập nhật thông tin học tập.

2. Dễ bảo trì và mở rộng:

- Phân tách rõ ràng giữa các thành phần giúp việc bảo trì hệ thống dễ dàng hơn. Khi cần cập nhật hoặc thay đổi giao diện, chỉ cần làm việc với View mà không ảnh hưởng đến logic xử lý trong Controller hay dữ liệu trong Model.

- Khi cần mở rộng hệ thống, ví dụ như thêm chức năng mới, có thể thêm vào một cách dễ dàng mà không làm phức tạp thêm mã nguồn hiện tại.

3. Tái sử dụng mã nguồn: Các thành phần Model và Controller có thể được tái sử dụng cho các View khác nhau. Ví dụ, logic kiểm tra điều kiện đăng ký môn học có thể được sử dụng lại cho nhiều giao diện người dùng khác nhau.

4. Đồng bộ hóa và nhất quán dữ liệu: Model là nơi quản lý dữ liệu tập trung, giúp đảm bảo tính nhất quán và đồng bộ hóa dữ liệu trong toàn hệ thống.

5. Hỗ trợ cho phát triển theo nhóm: Mô hình MVC cho phép các lập trình viên làm việc độc lập trên các thành phần khác nhau mà không gây xung đột. Ví dụ, một nhóm có thể làm việc trên View, trong khi nhóm khác làm việc trên Controller và Model.

***1.6 Một số tình huống trong đăng ký học phần và lý do chọn kiến trúc***

***-*** Hệ thống đăng ký học phần được xây dựng bởi những người chưa nhiều có kinh nghiệm => MVC là kiến trúc khá phổ biến được đưa vào giảng dạy trong học tập vì tính chặt chẽ, tách biệt giữa các phần, có thể dễ dàng thực hiện các công việc như kiểm tra, rà soát lỗi phần mềm. Ví dụ hệ thống phát triển bởi sinh viên chưa có nhiều tiếp xúc với các kiến trúc phức tạp hơn yêu cầu nhiều cài đặt và vận hành

- Hệ thống sử dụng các công nghệ hiện đại => MVC dễ dàng tích hợp với các công nghệ hiện đại như các framework frontend (Angular, React, Vue.js) và các dịch vụ backend (RESTful API, GraphQL). Điều này giúp bạn xây dựng ứng dụng web mạnh mẽ, hiện đại và dễ duy trì. Ví dụ các công nghệ liên tục ra mắt khiến các công nghệ cũ trở nên lỗi thời việc có thể dễ dàng tích hợp công nghệ giúp giảm gánh nặng lập trình.

- Hệ thống cần tính bảo mật che giấu thông tin => Bằng cách tách biệt rõ ràng các thành phần của ứng dụng, MVC giúp giảm thiểu rủi ro bảo mật. Ví dụ, Controller có thể xử lý kiểm tra và xác thực người dùng trước khi cho phép truy cập vào các chức năng nhạy cảm, trong khi View chỉ tập trung vào hiển thị thông tin mà không xử lý logic nhạy cảm. Ví dụ sinh viên không thể biết điểm của người khác cũng không thể biết lịch học hay công nợ người khác.

- Hệ thống cần phát triển trên đa nền tảng => MVC giúp dễ dàng phát triển ứng dụng đa nền tảng. Bạn có thể sử dụng cùng một Model và Controller cho nhiều View khác nhau, bao gồm web, mobile, và desktop, giúp tiết kiệm thời gian và công sức trong quá trình phát triển. Ví dụ hệ thống phát triển trên nhiều nền tảng phù hợp với nhiều kiểu người dùng khác nhau.

- Hệ thống yêu cầu kết hợp với các kiến trúc khác => MVC dễ dàng tích hợp với các kiến trúc dịch vụ như microservices. Bạn có thể xây dựng các dịch vụ độc lập xử lý các phần khác nhau của ứng dụng và tích hợp chúng thông qua API, giúp tăng cường tính mô-đun và khả năng mở rộng. Ví dụ mvc là 1 service của kiến trúc hệ thống microservice.

# : 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:

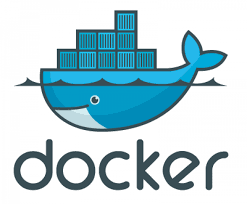
* Khái niệm: Spring Boot là một dự án phát triển bởi JAV (ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.



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



### 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 |

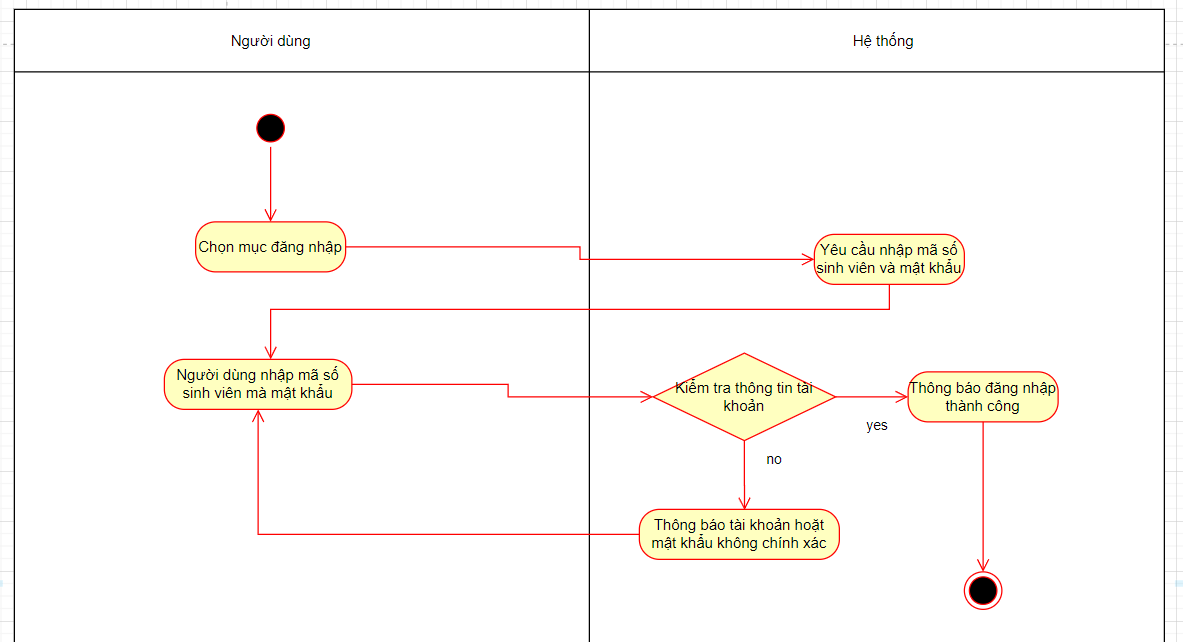
Bảng 3‑2 Danh sách các tình huống trong hệ thống

### Tình huống hoạt động

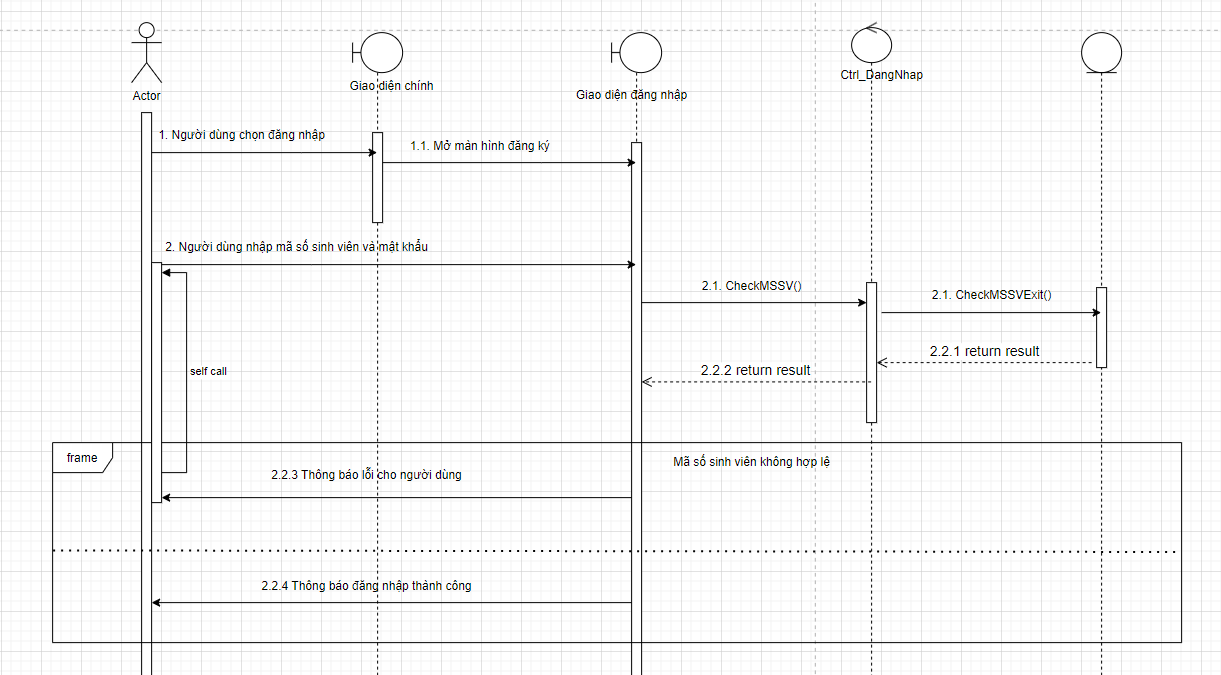
Đặc tả use case.

|  |  |  |
| --- | --- | --- |
| **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.

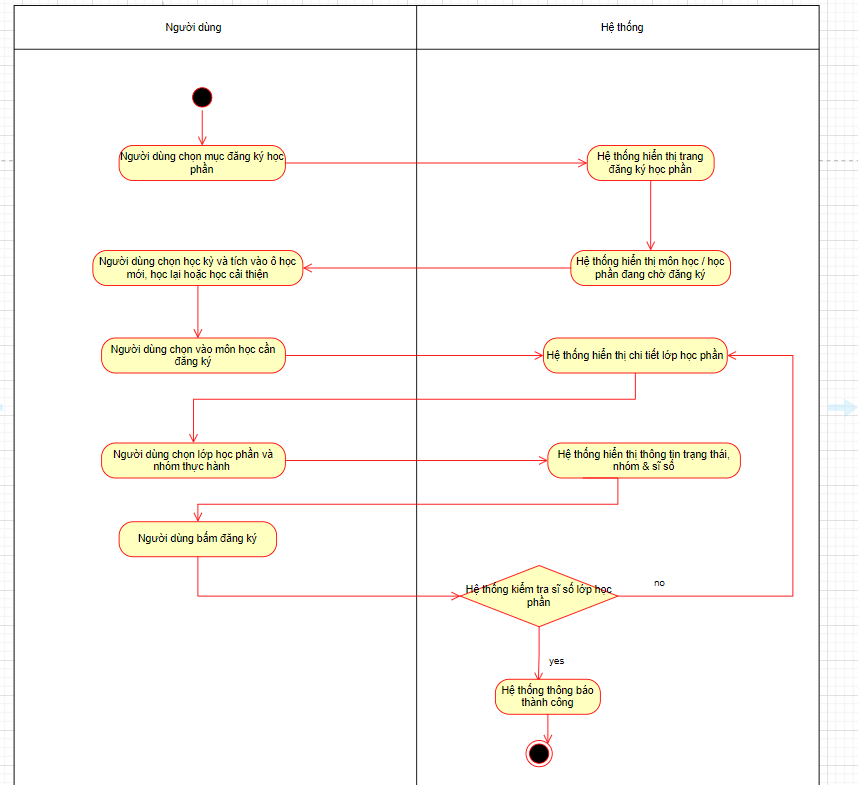


Sequence diagram cho từng tình huống.

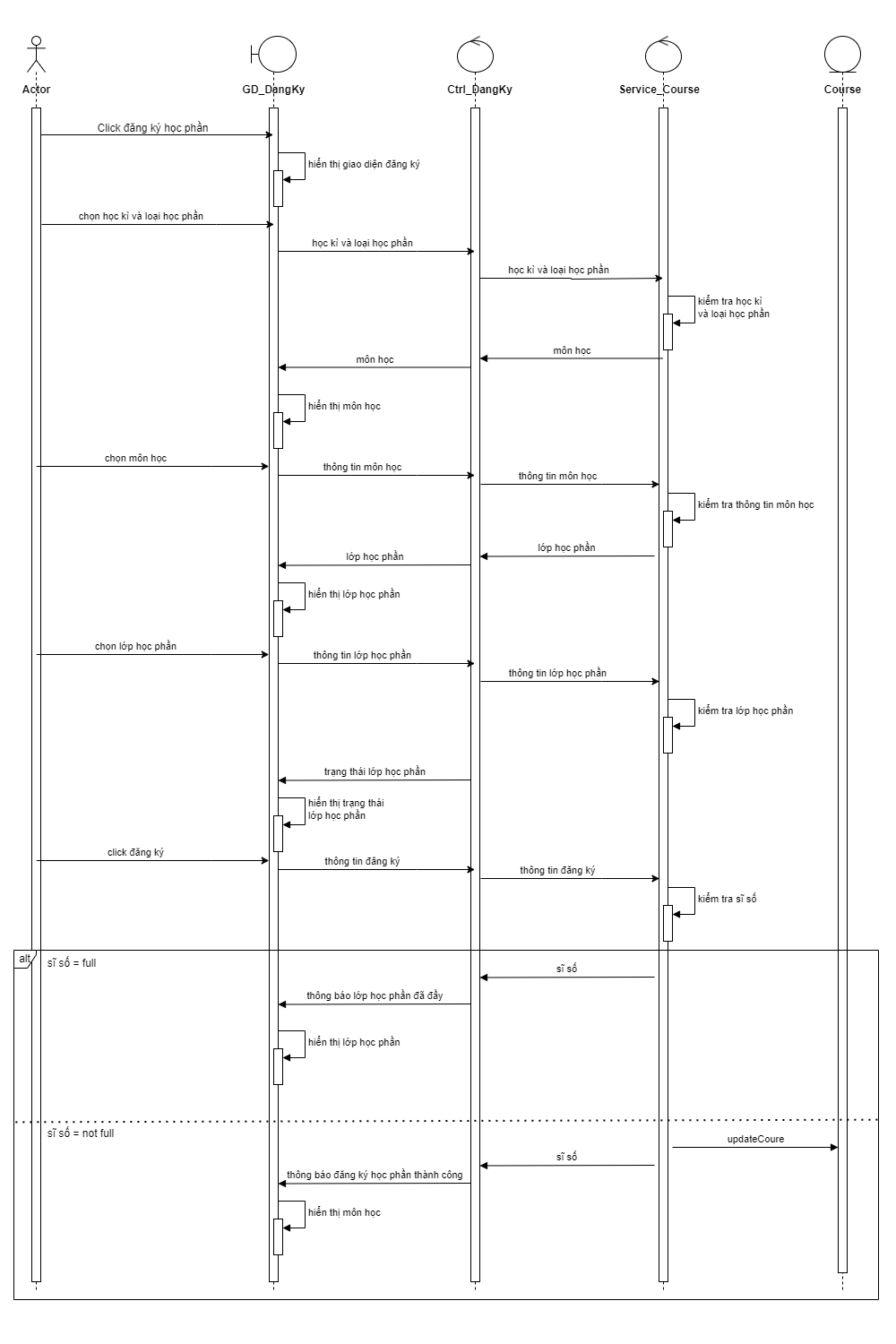


|  |  |  |
| --- | --- | --- |
| **UC002\_Đăng ký học phần mới** | | |
| **Mục đích** | Cho phép người dùng đăng kí học phần | |
| **Mô tả** | Usecase này mô tả cách người dùng đăng kí học phần mới | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Phải đăng ký học môn tiên quyết trước | |
| **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.

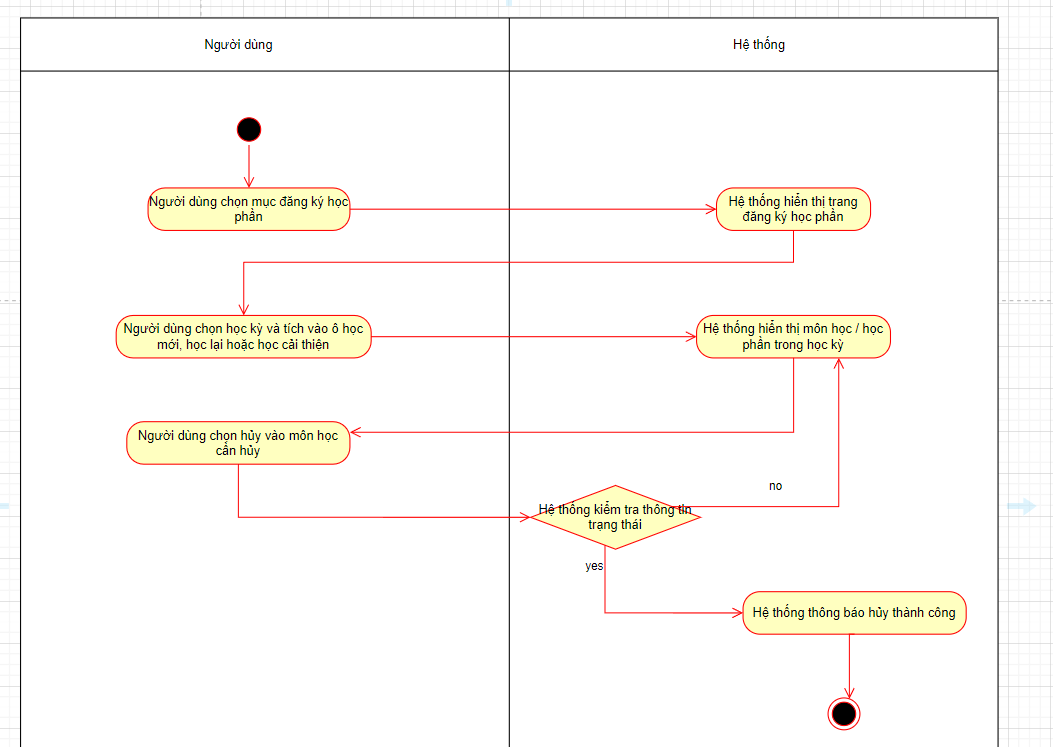


Sequence diagram cho từng tình huống.

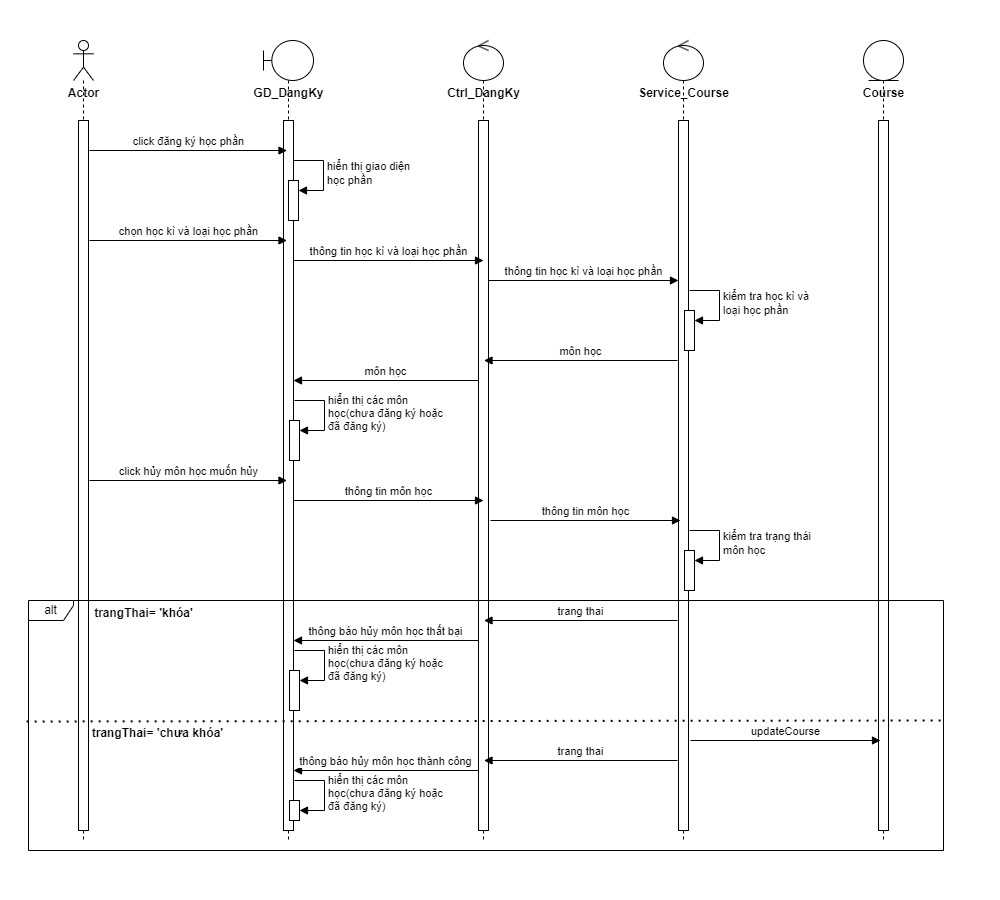


|  |  |  |
| --- | --- | --- |
| **UC003\_Hủy học phần** | | |
| **Mục đích** | Cho phép người dùng hủy học phần vừa đăng ký | |
| **Mô tả** | Usecase này mô tả cách người dùng hủy học phần vừa đăng ký | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá 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.

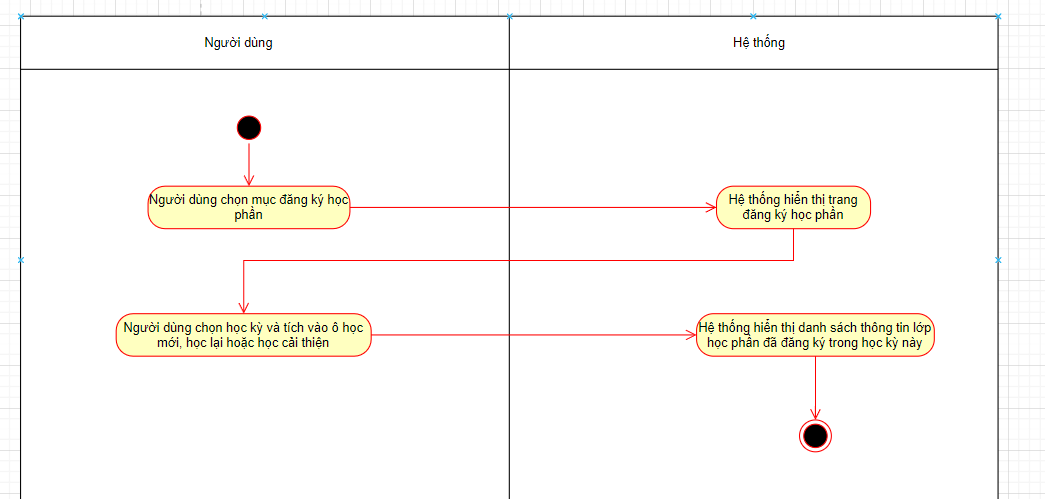


Sequence diagram cho từng tình huống.

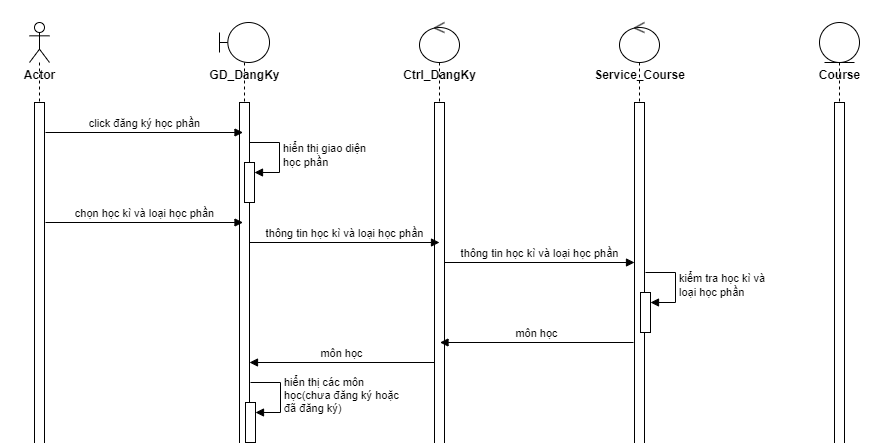


|  |  |  |
| --- | --- | --- |
| **UC004\_Xem danh sách môn học đã đăng ký trong học kỳ** | | |
| **Mục đích** | Cho phép người dùng xem danh sách môn học đã đăng ký trong học kỳ | |
| **Mô tả** | Usecase này mô tả cách người dùng xem danh sách môn học đã đăng ký trong học kỳ | |
| **Tác nhân** | Người 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.

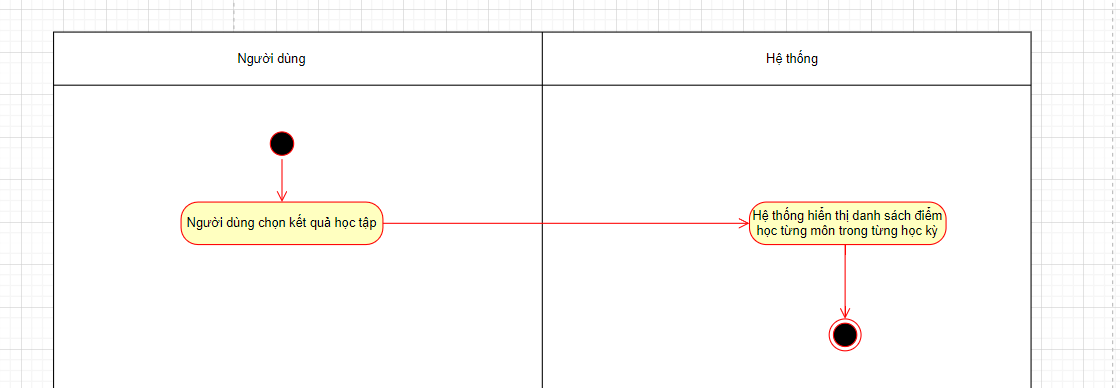


Sequence diagram cho từng tình huống.

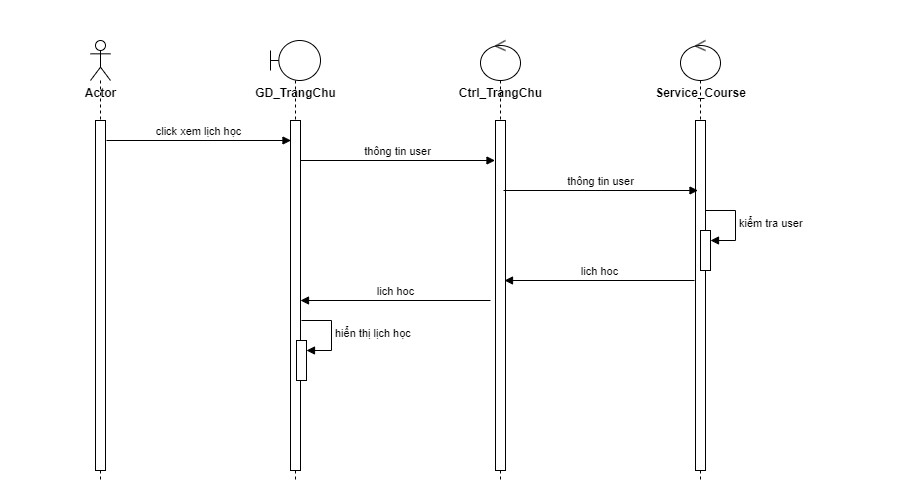


|  |  |  |
| --- | --- | --- |
| **UC005\_Xem lịch học** | | |
| **Mục đích** | Cho phép người dùng xem lịch học | |
| **Mô tả** | Usecase này mô tả cách người dùng xem lịch học | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | 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:

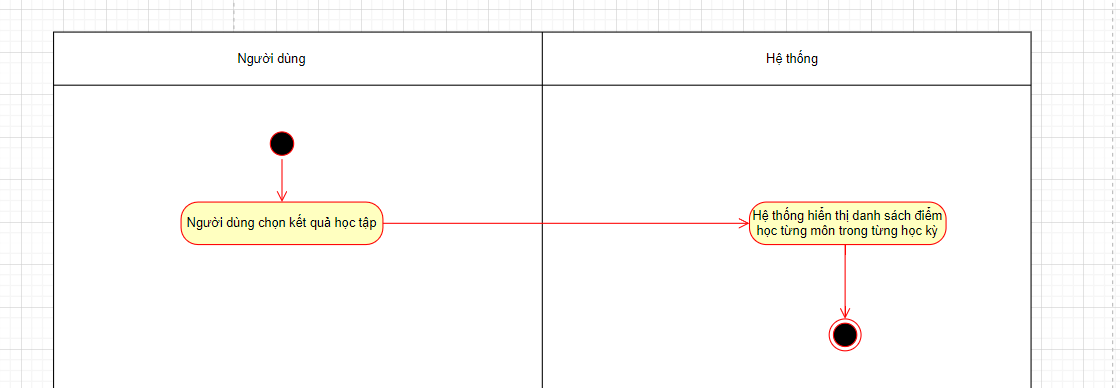


* Sequence diagram cho từng tình huống.

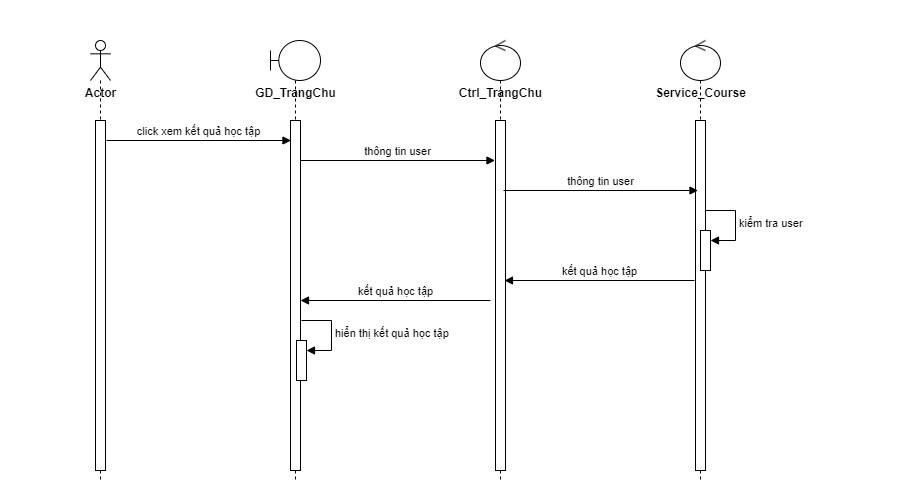


|  |  |  |
| --- | --- | --- |
| **UC006\_Xem điểm** | | |
| **Mục đích** | Cho phép người dùng xem điểm | |
| **Mô tả** | Usecase này mô tả cách người dùng xem điểm cá nhân | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đang đăng nhập tài khoản cá nhân | |
| **Điều kiện sau** | 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:

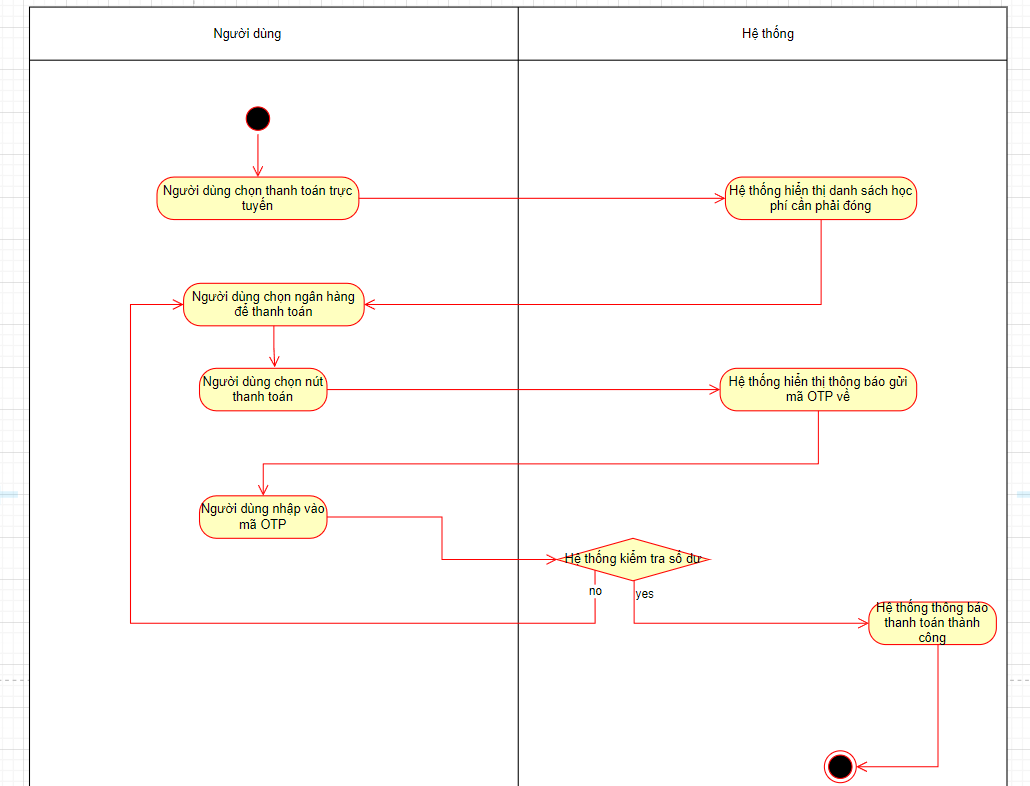


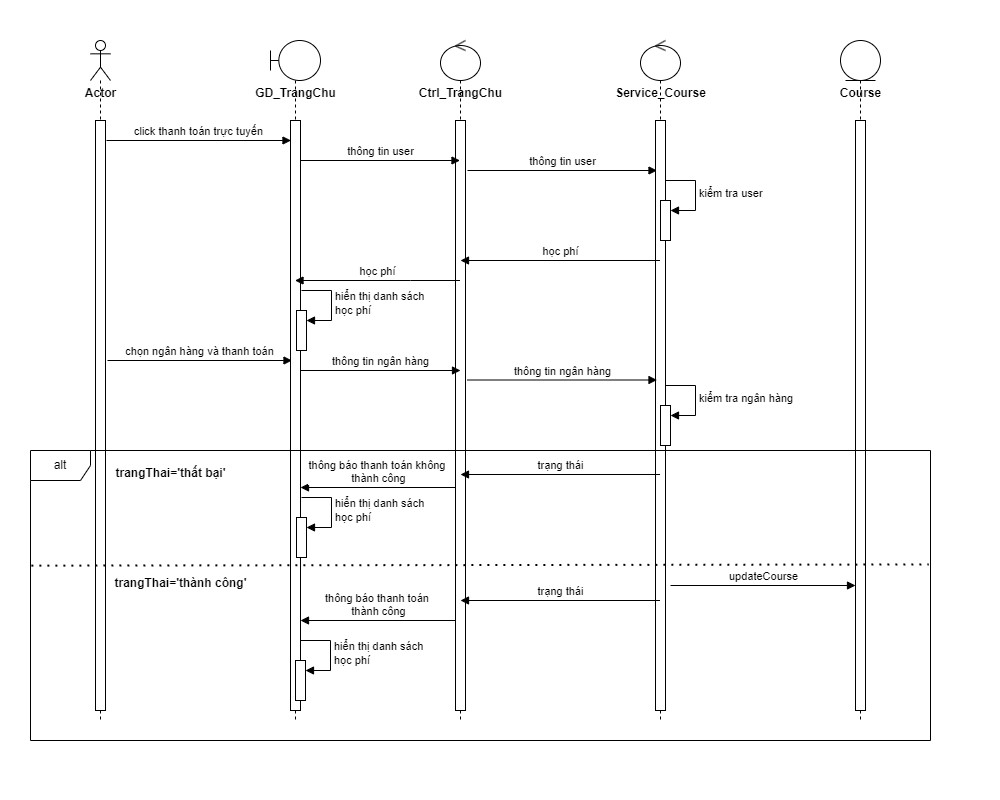
* Sequence diagram cho từng tình huống.



|  |  |  |
| --- | --- | --- |
| **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 | |  |

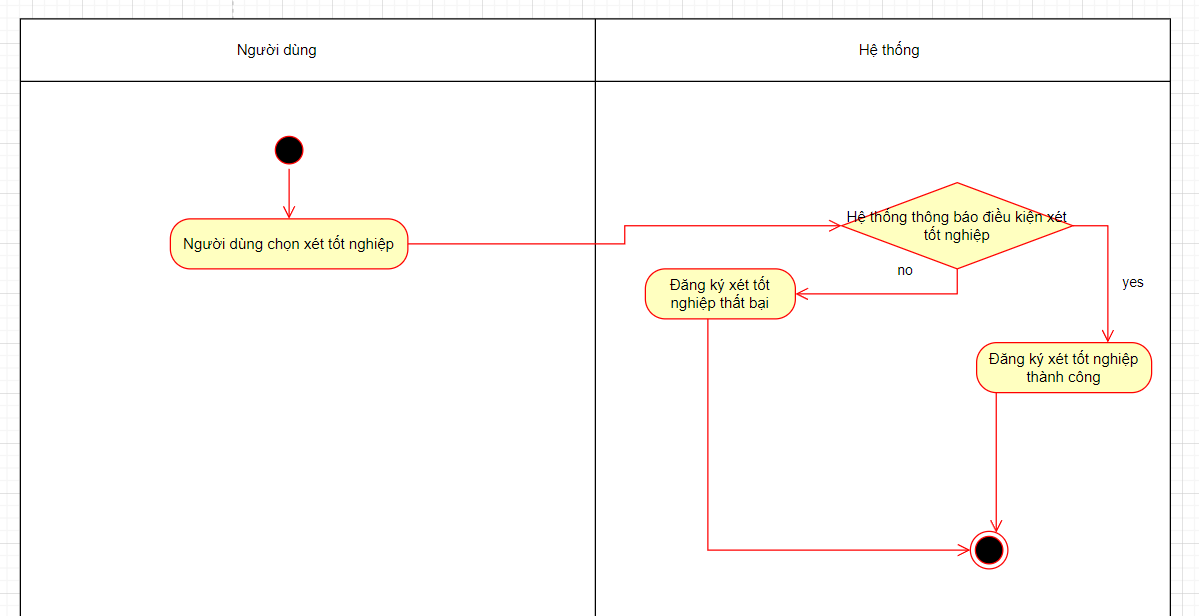
* Sơ đồ Activiti:





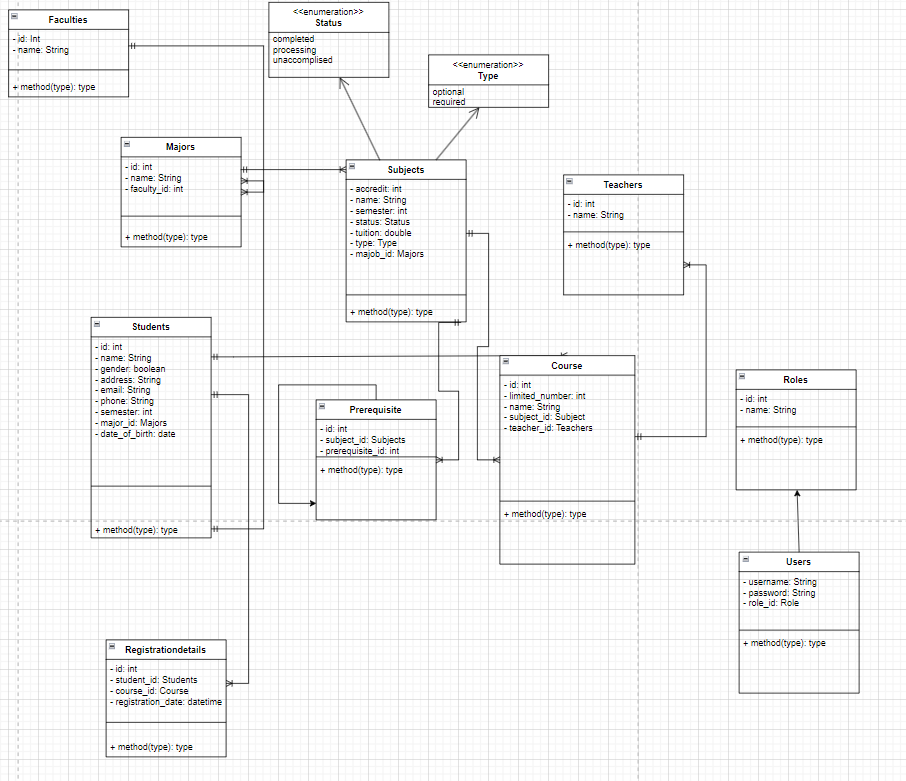
|  |  |  |
| --- | --- | --- |
| **UC008\_** Đăng ký xét tốt nghiệp | | |
| **Mục đích** | Cho phép người dùng đăng ký xét tốt nghiệp | |
| **Mô tả** | Usecase này mô tả cách người dùng xét tốt nghiệp | |
| **Tác nhân** | Người dùng | |
| **Điều kiện trước** | Người dùng đã có tài khoản & đủ điều kiện 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:

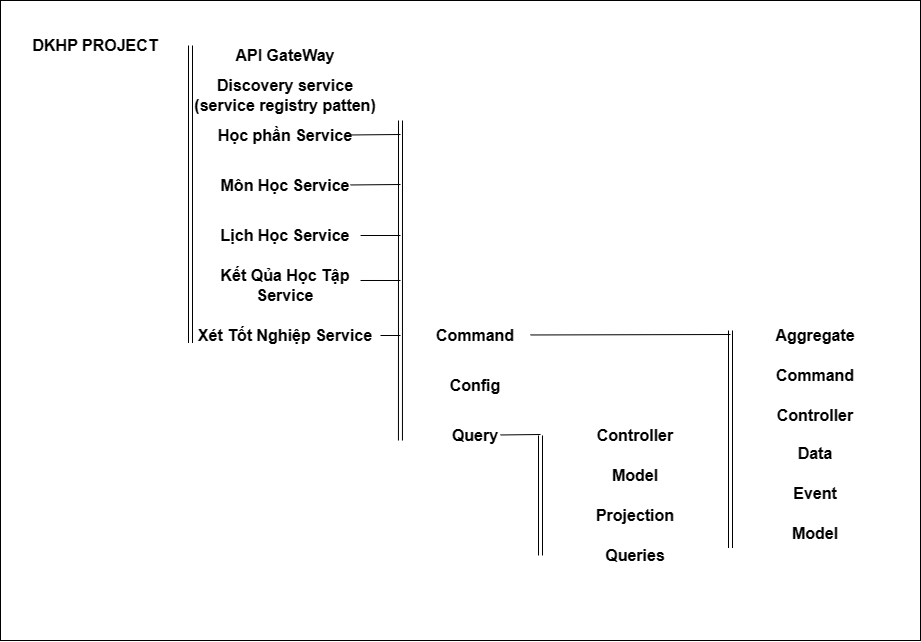


## Class diagram

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

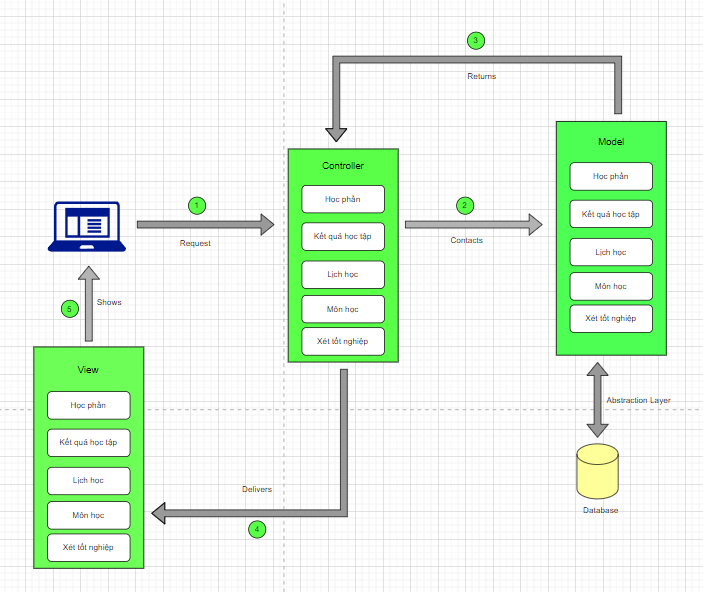


**3.3 Cấu trúc project:**



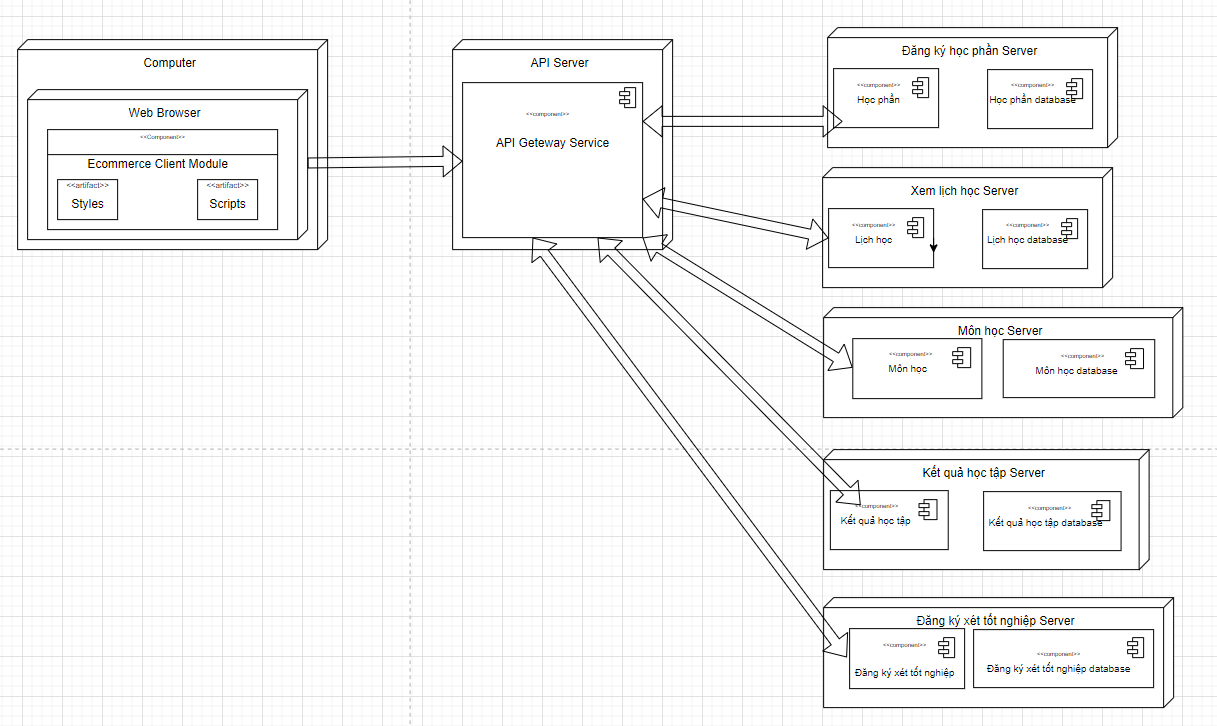
**3.4 Mô hình MVC:**

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



## 3.5 Deployment diagram:

Mô hình triển khai hệ thống.



# : 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:

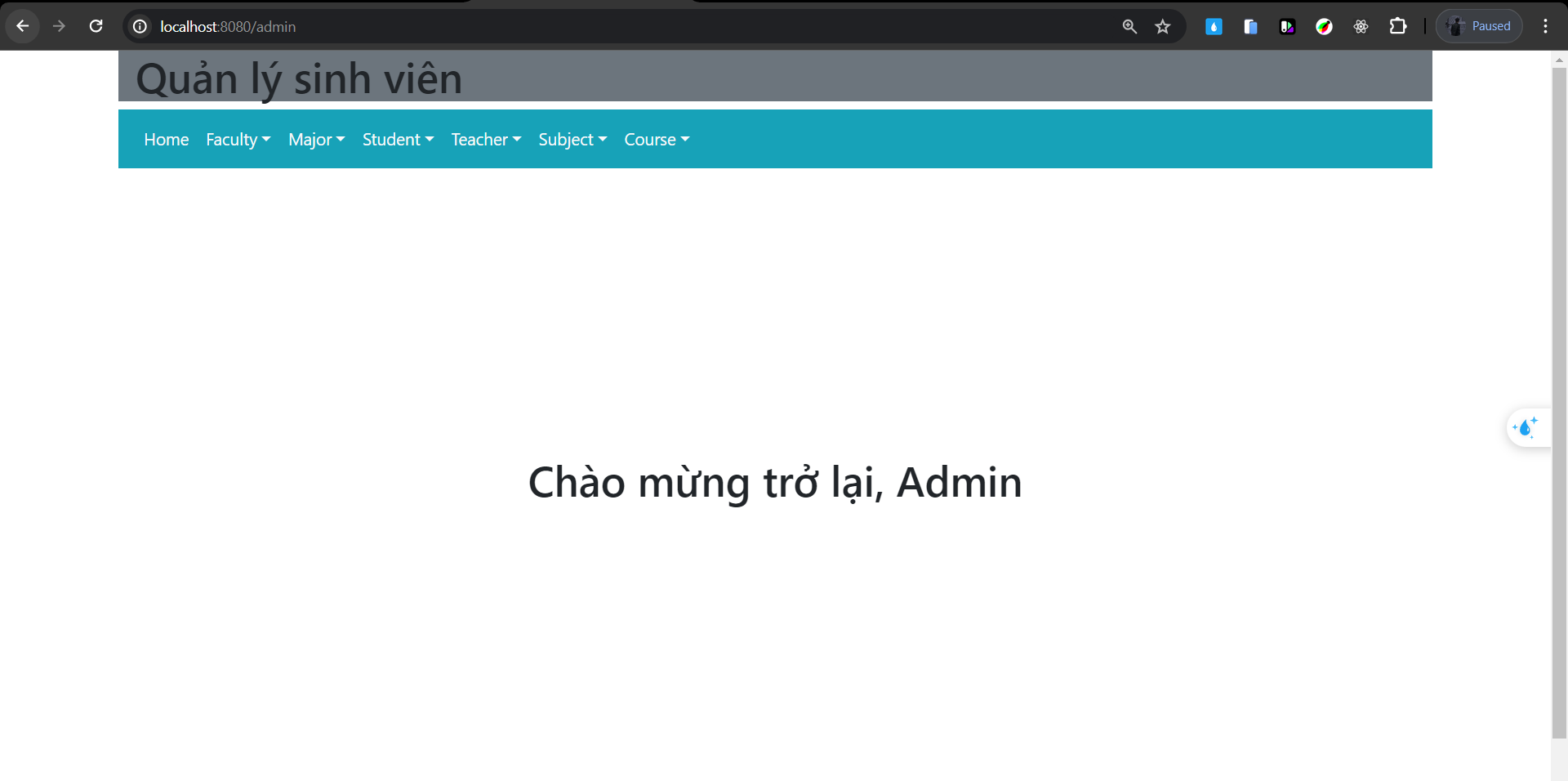
Spring Cloud Config: Quản lý cấu hình phân tán cho các microservice.

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

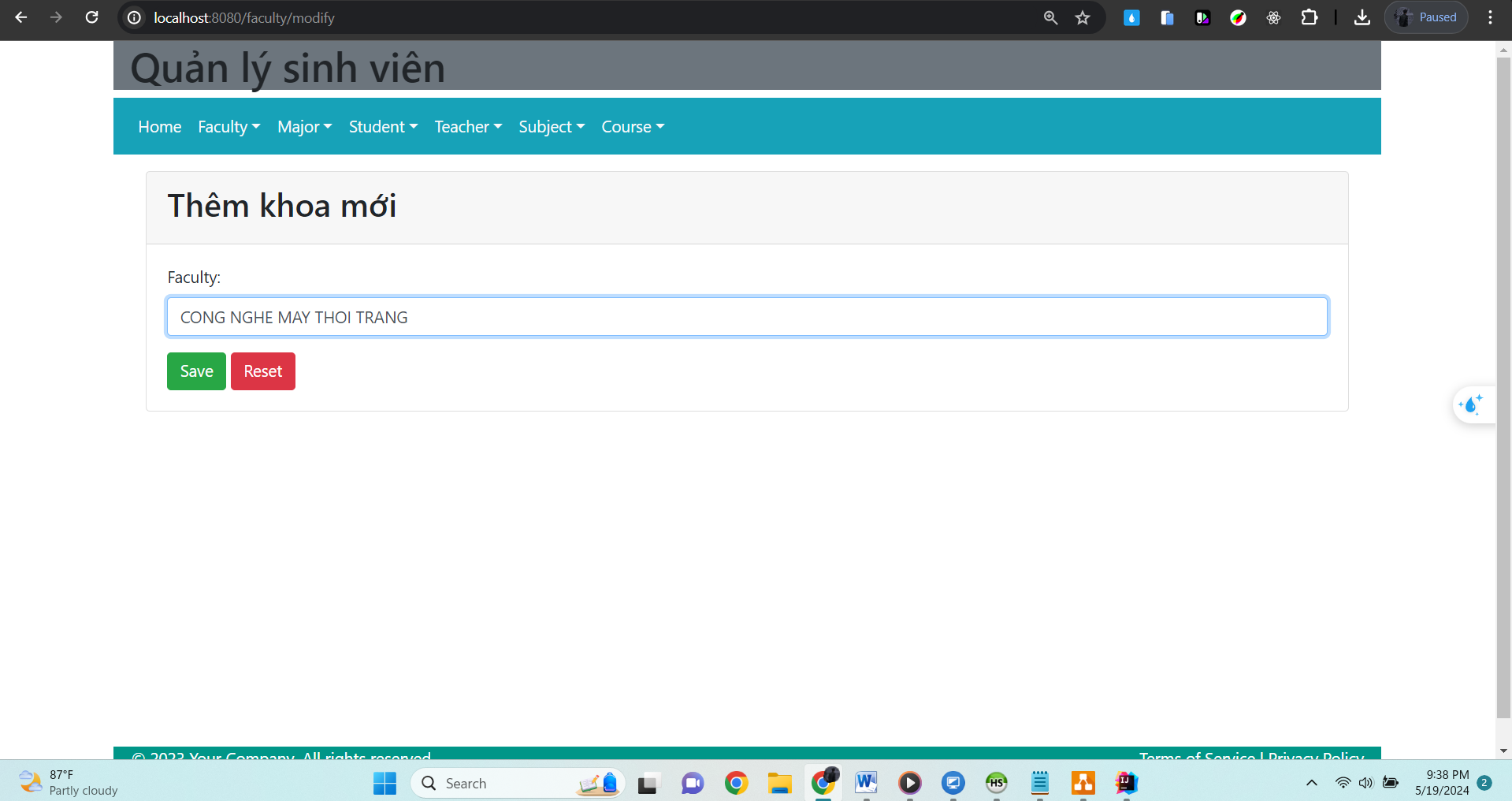
## Giao diện của hệ thống

Màn hình chức năng đã thực hiện.

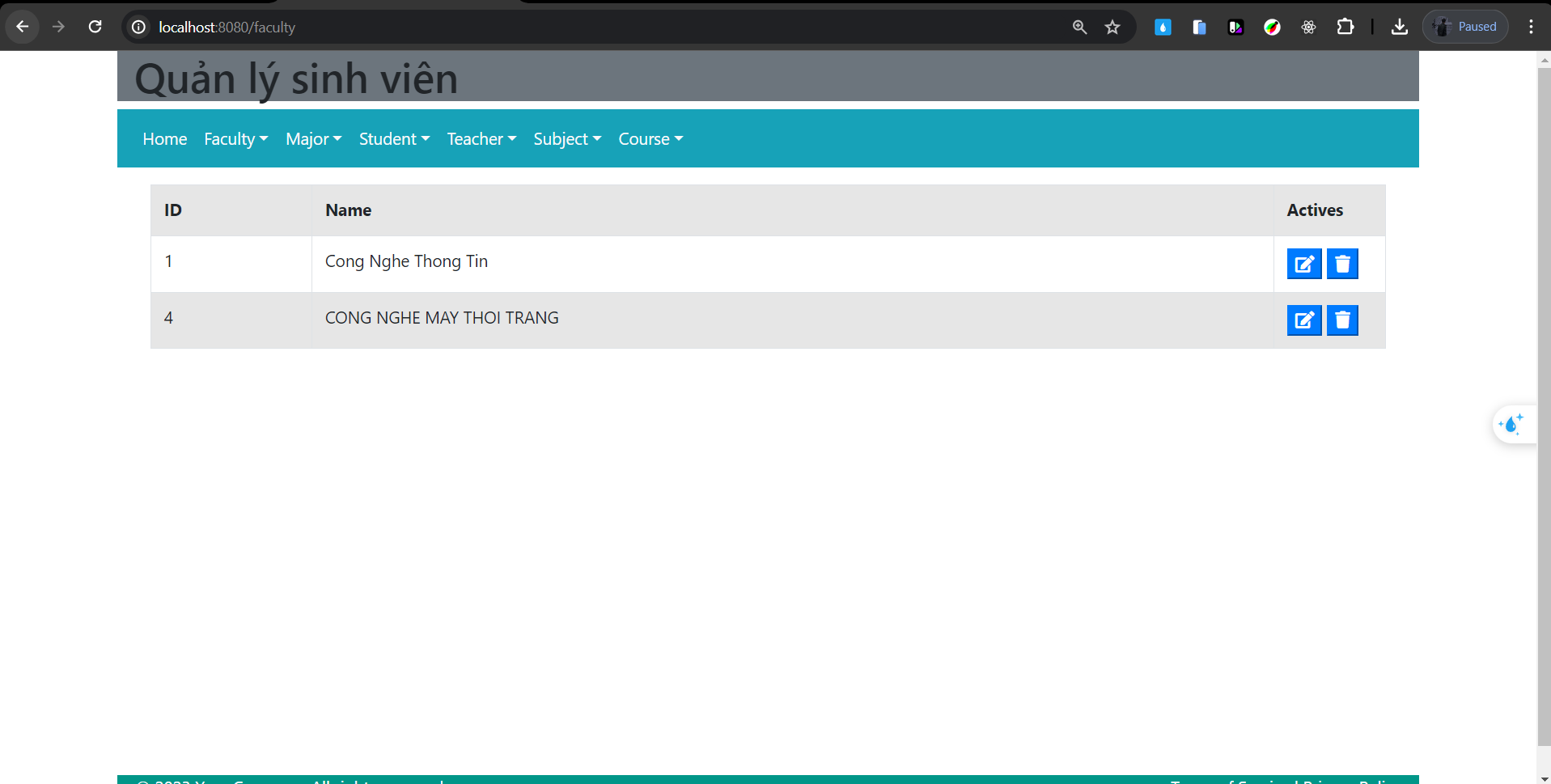
* Bên Admin:
* Giao diện Home:



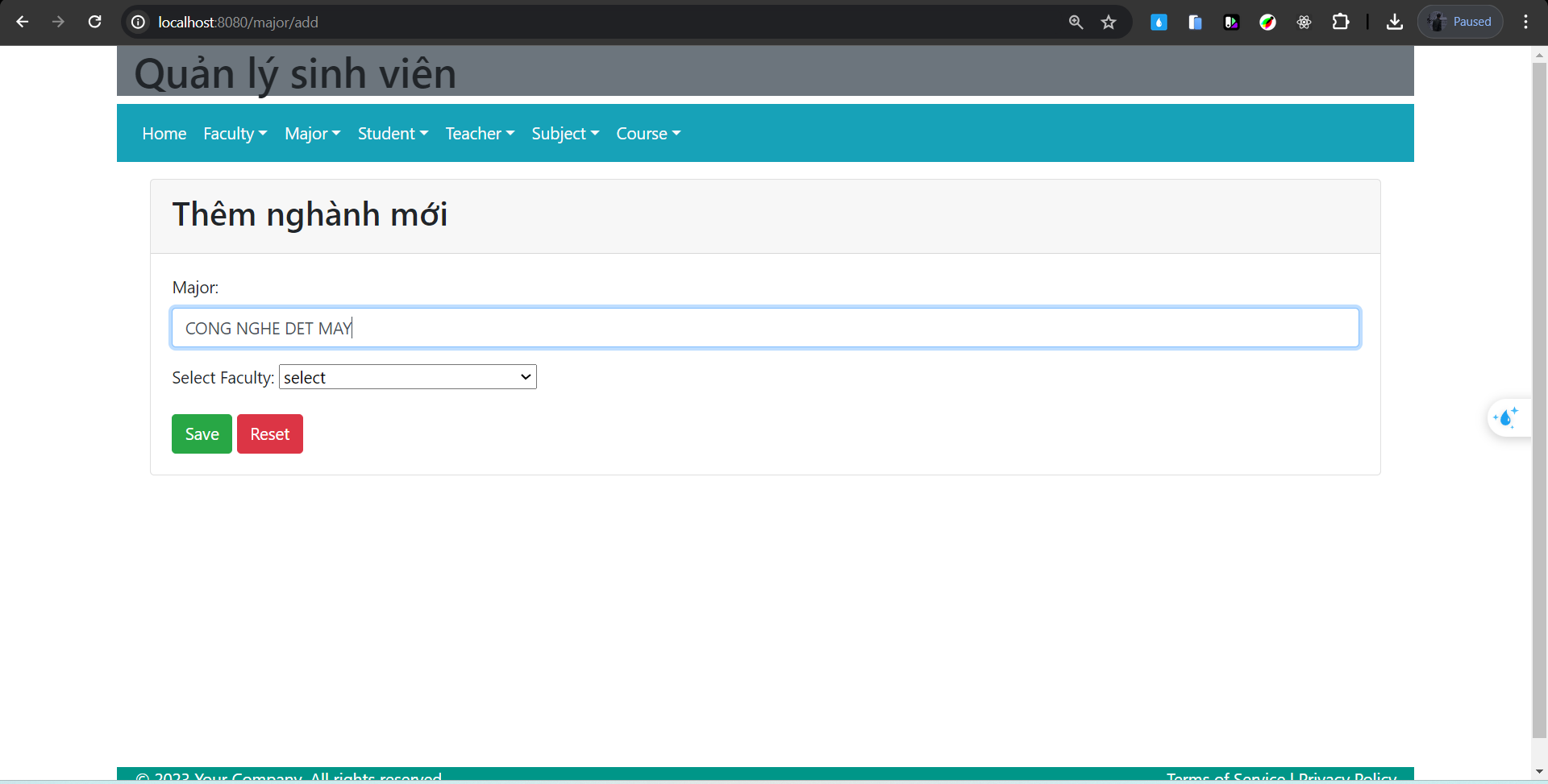
* Thêm Khoa mới



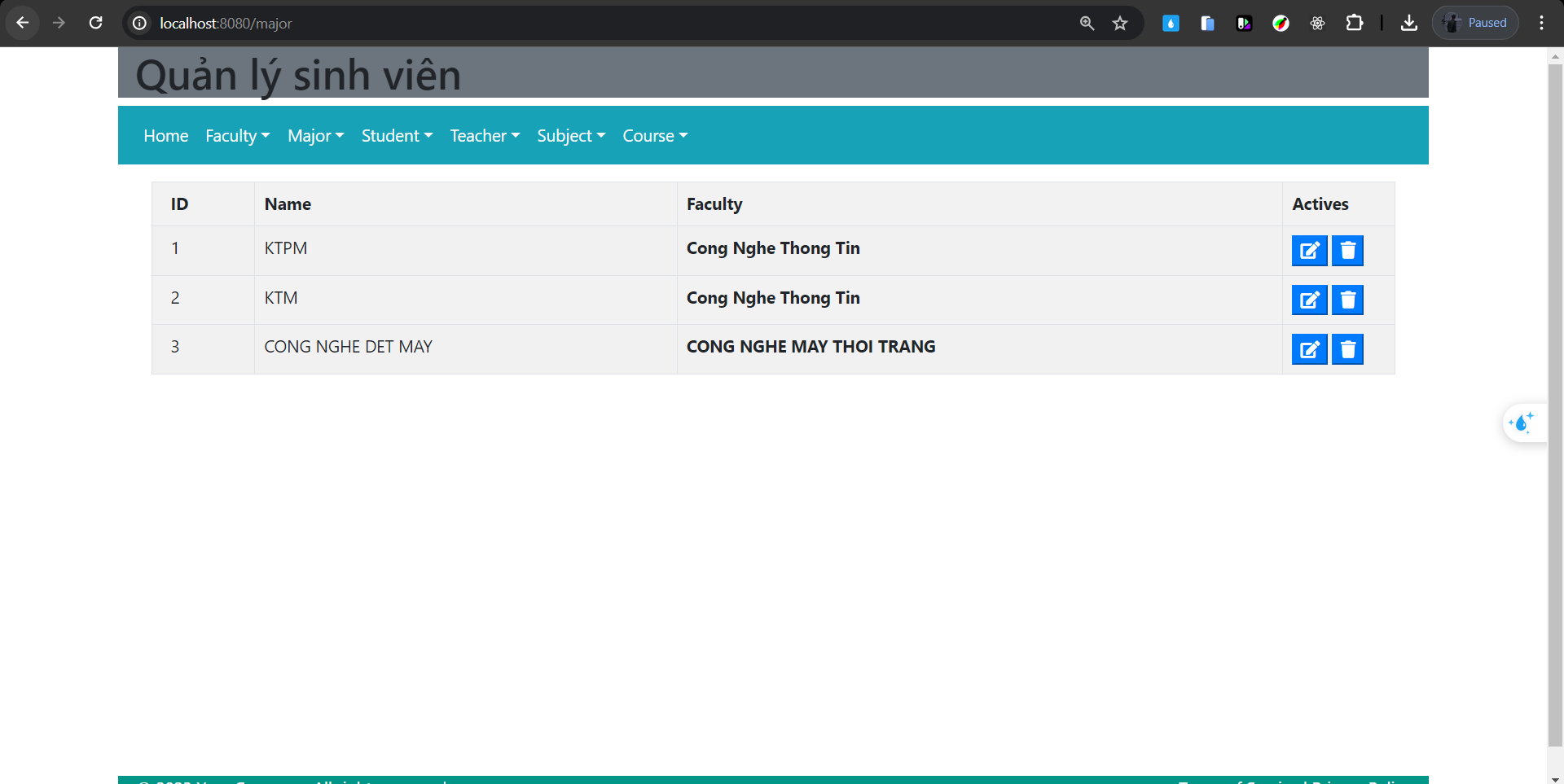
* Danh sách khoa:



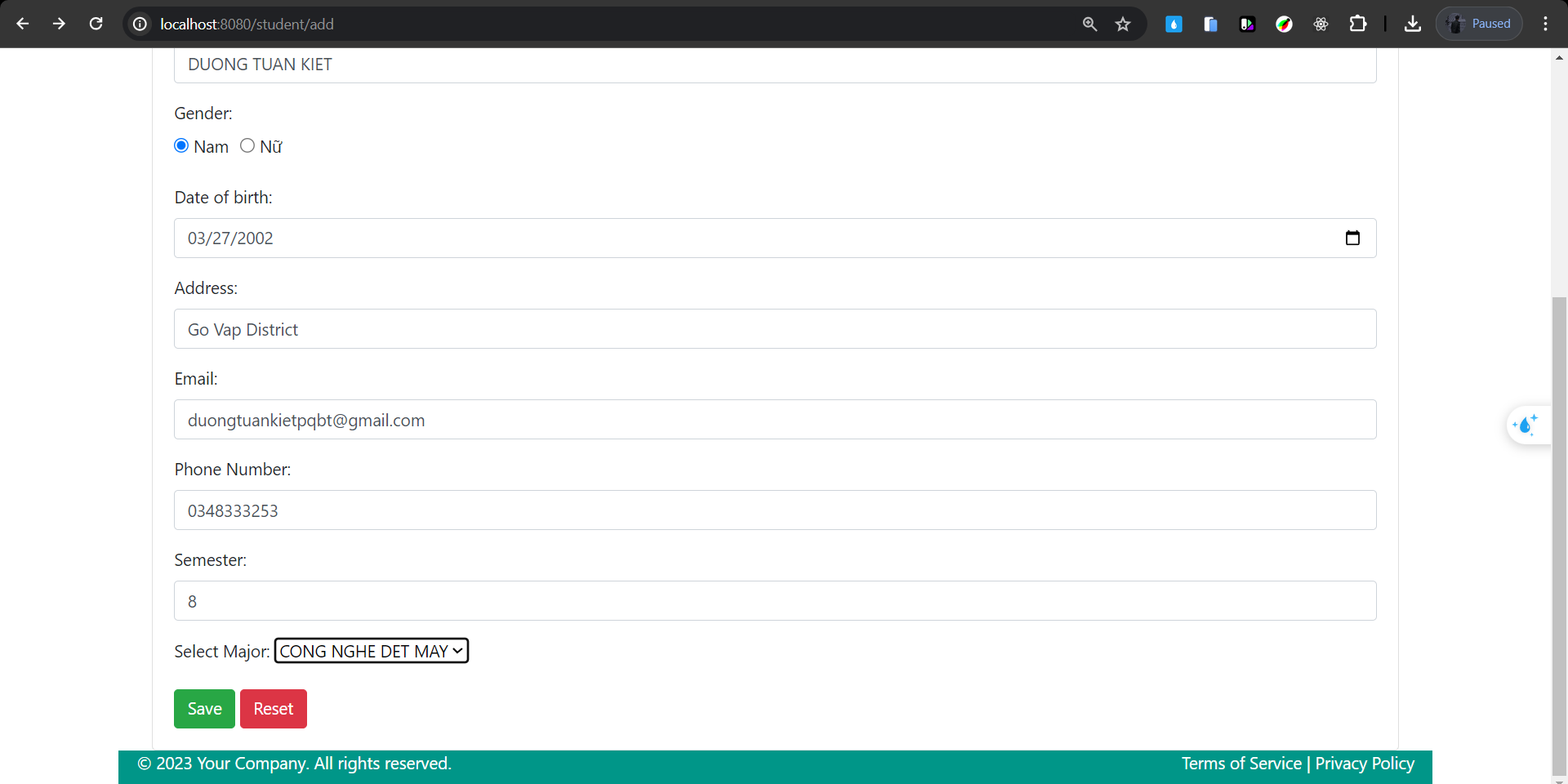
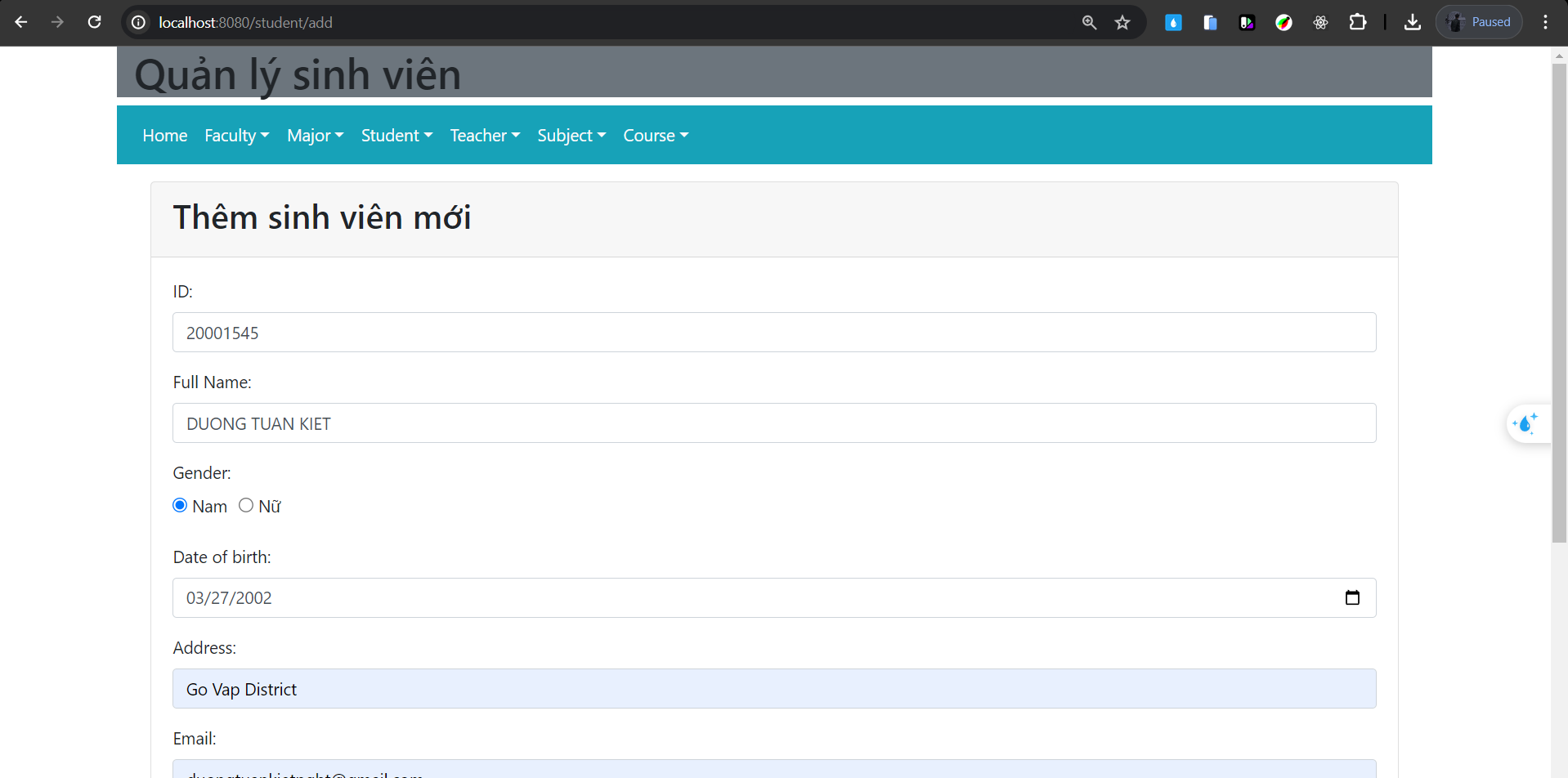
* Thêm ngành mới:



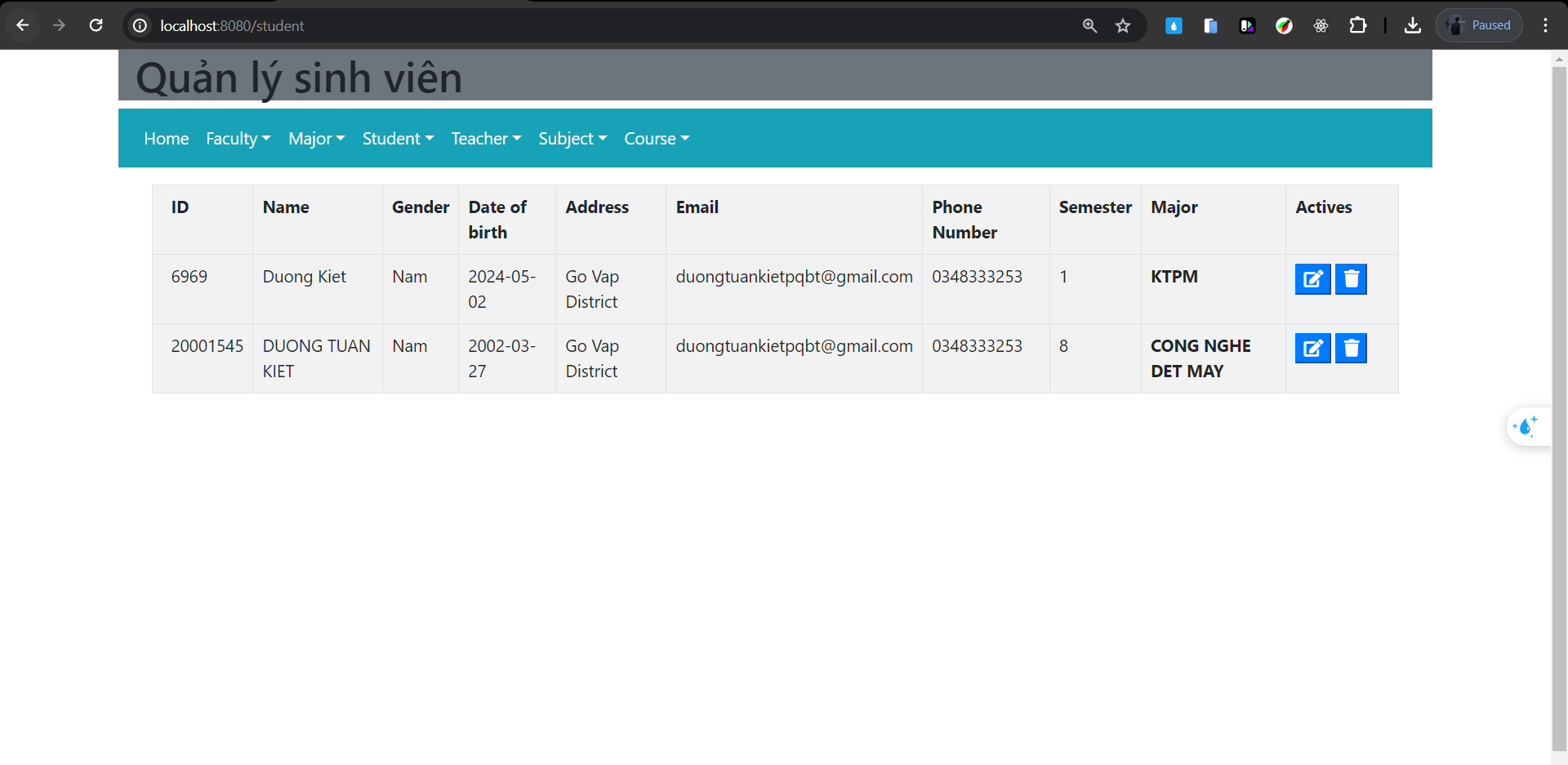
* Danh sách ngành:



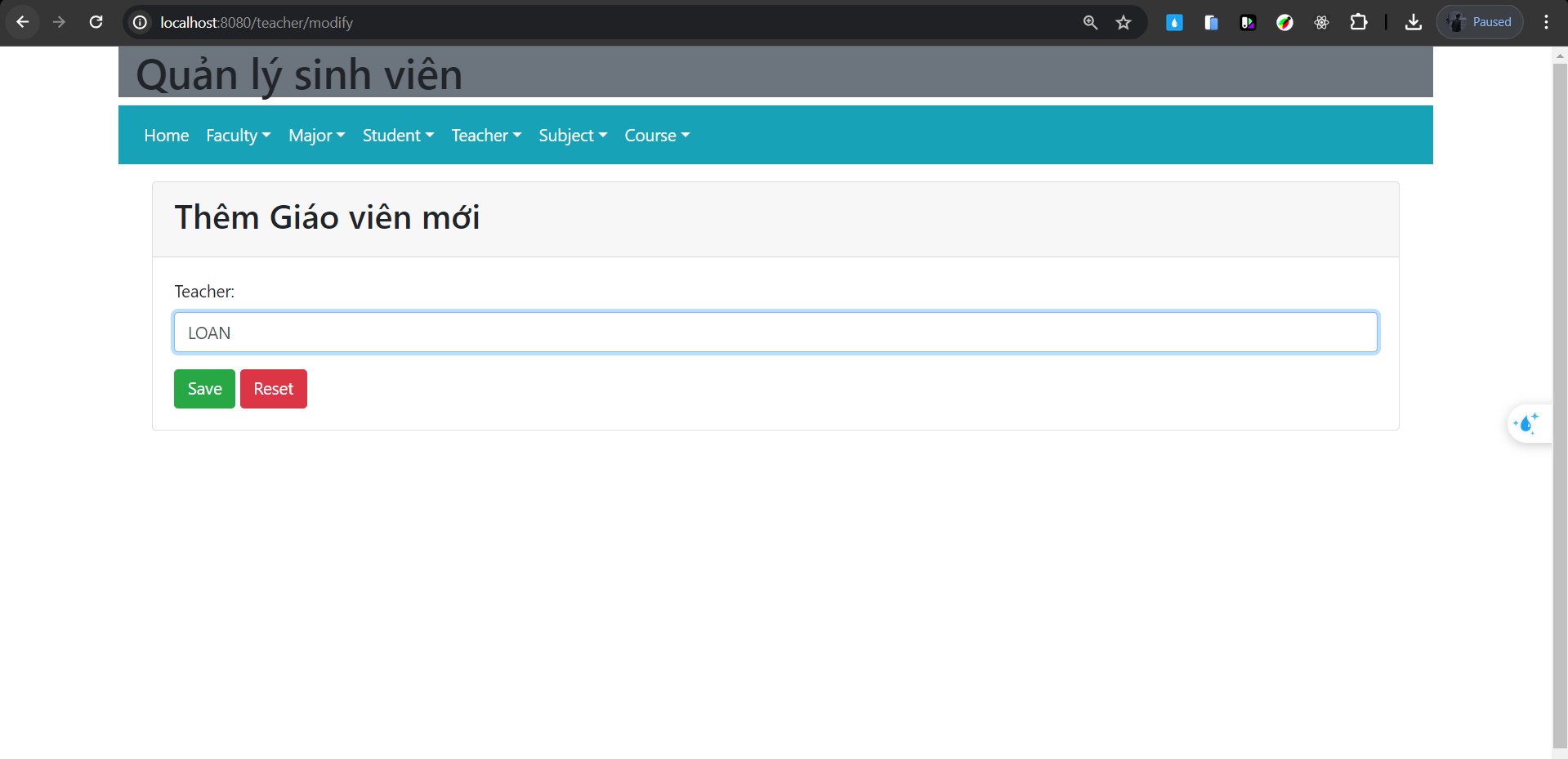
* Thêm sinh viên mới:



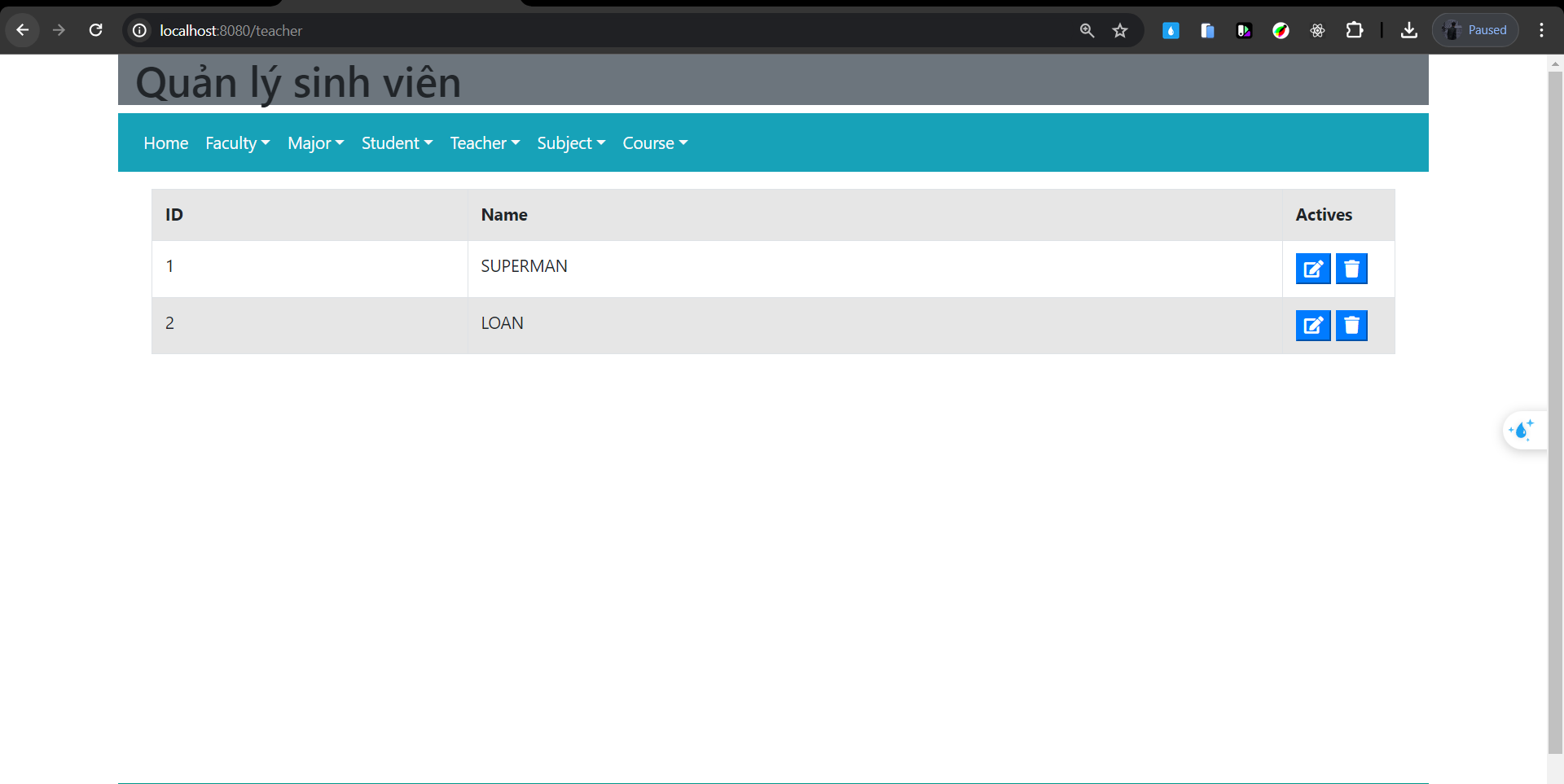
* Danh sách sinh viên vừa thêm:



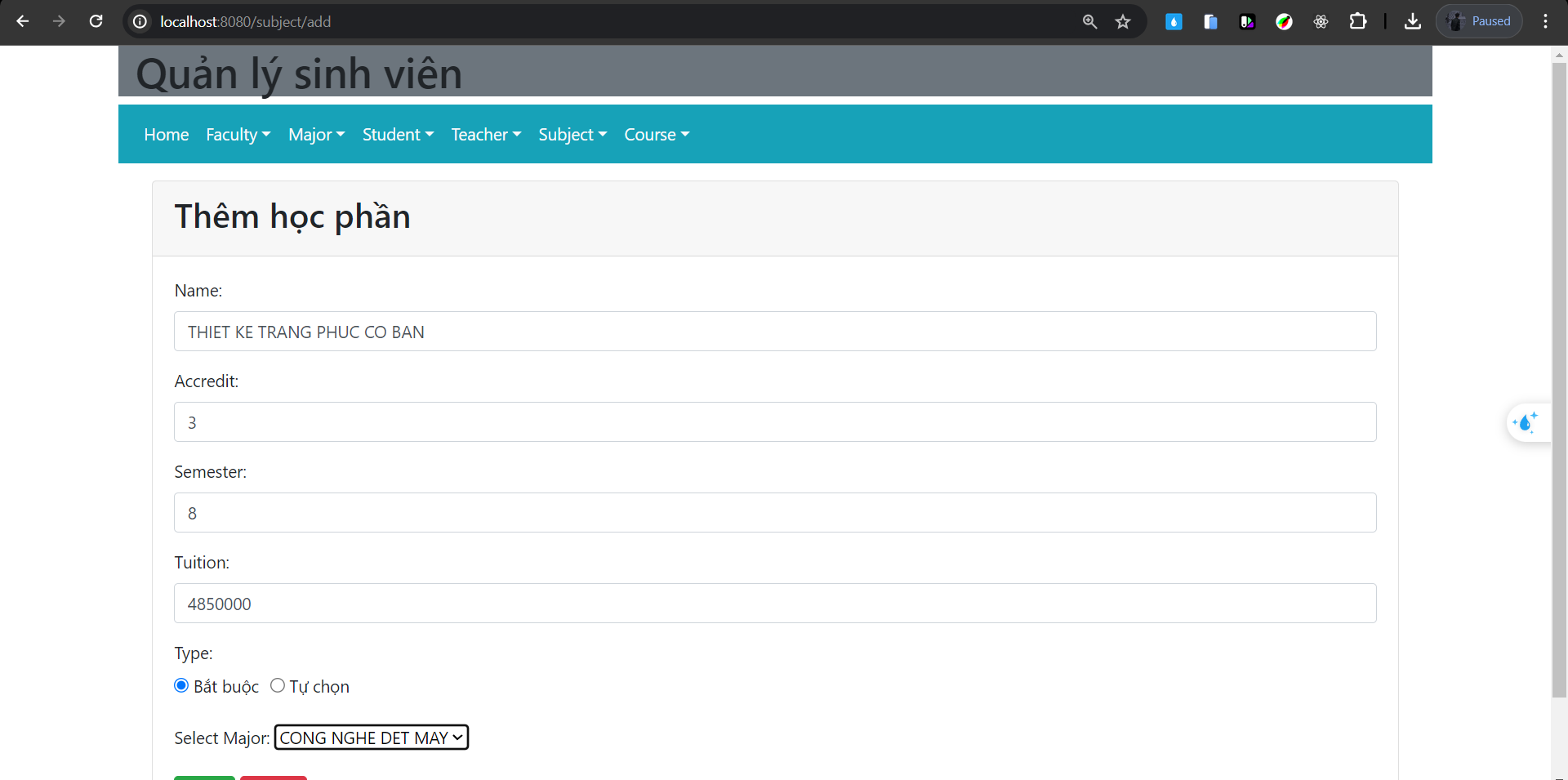
* Thêm 1 giáo viên mới:



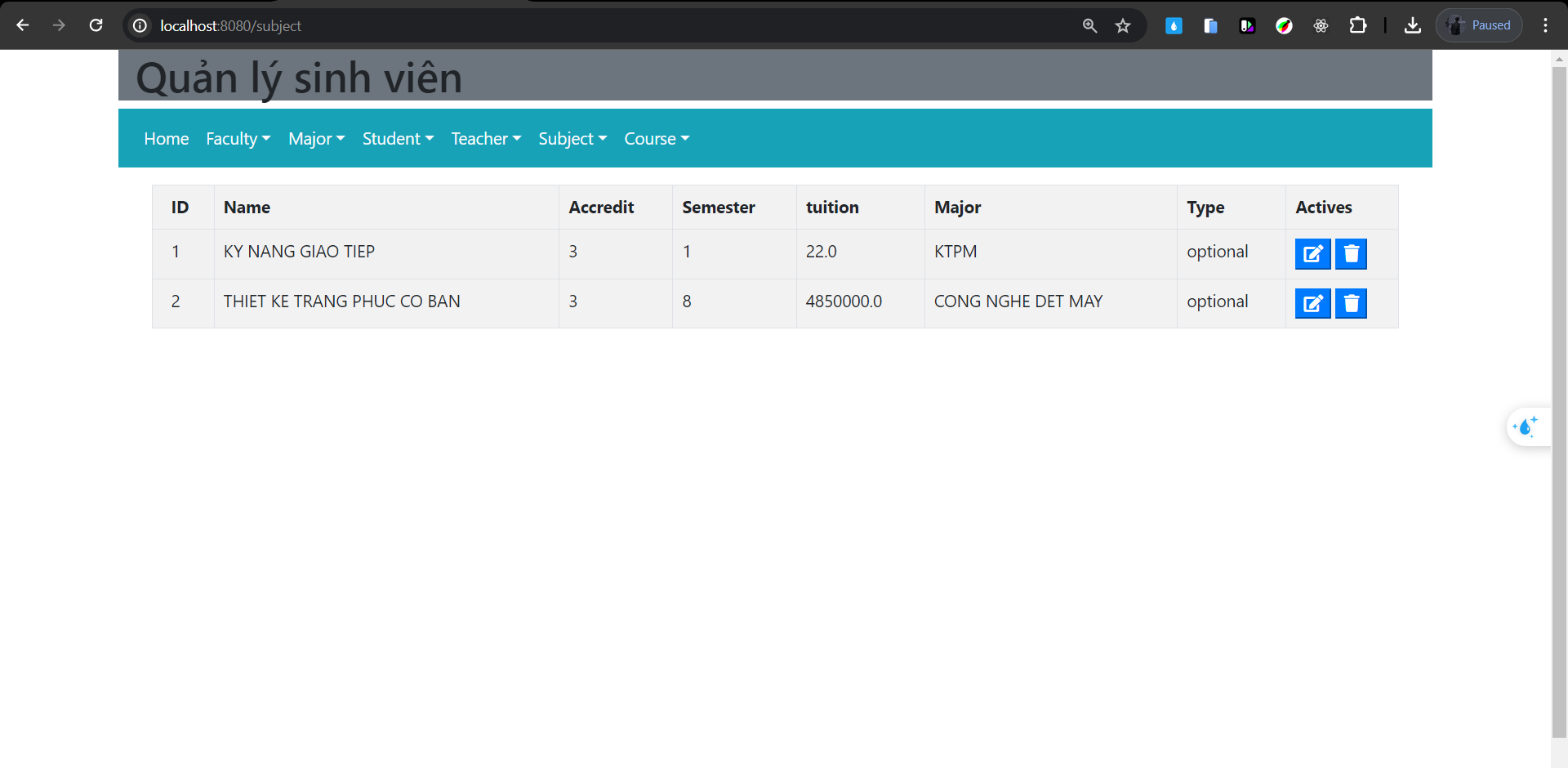
* Danh sách giáo viên vừa thêm:



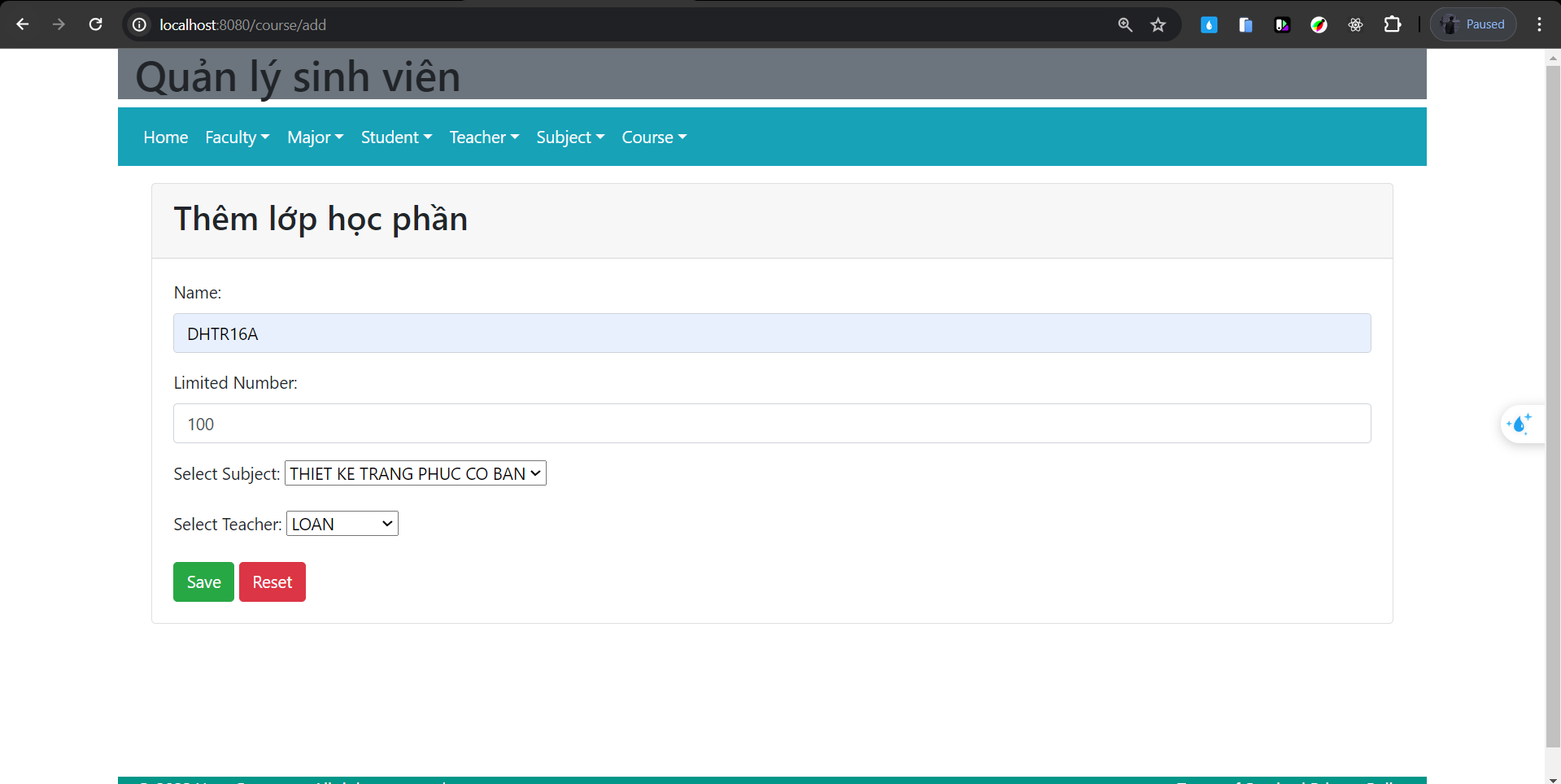
* Thêm học phần:



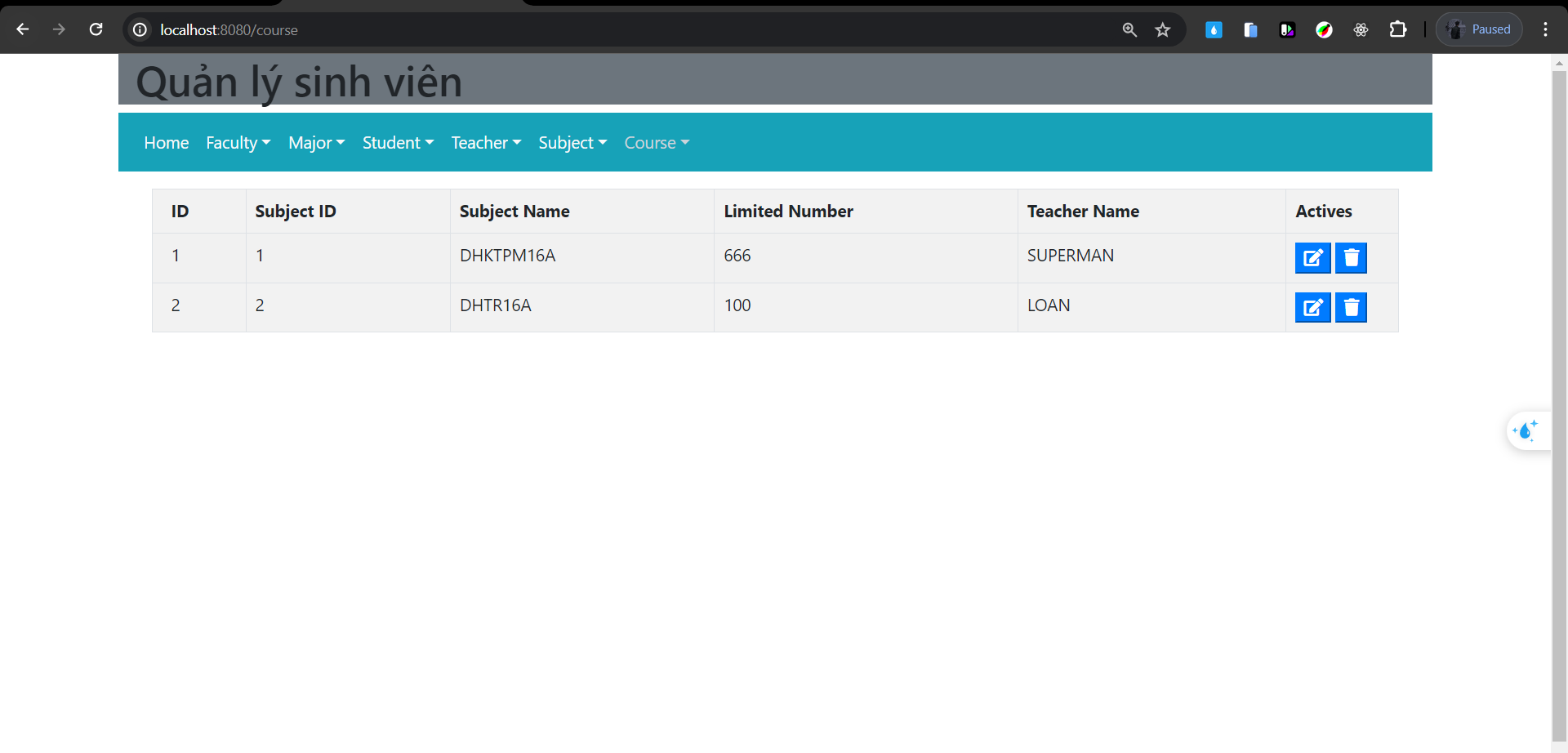
* Danh sách học phần :



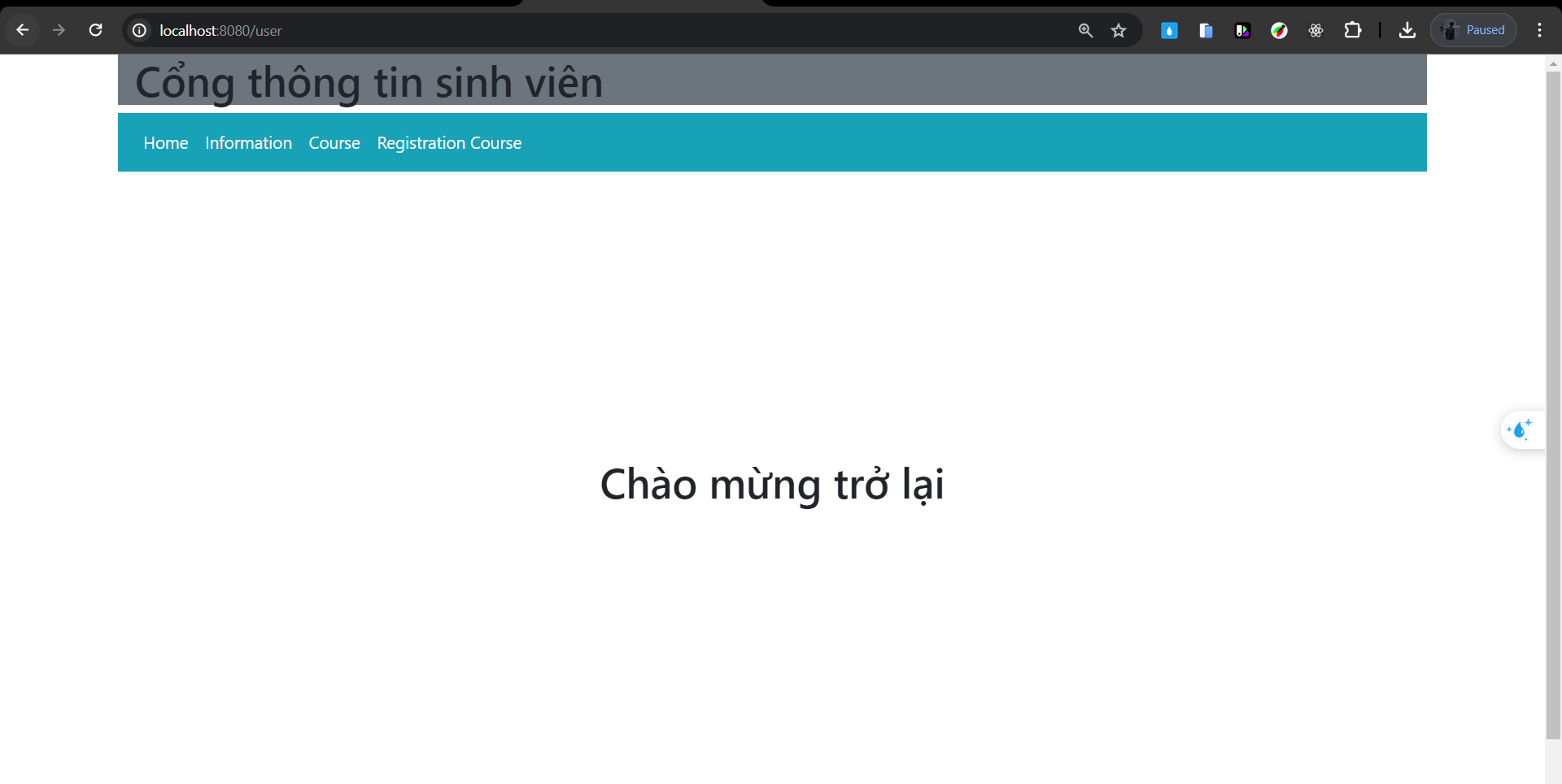
* Thêm lớp học phần:



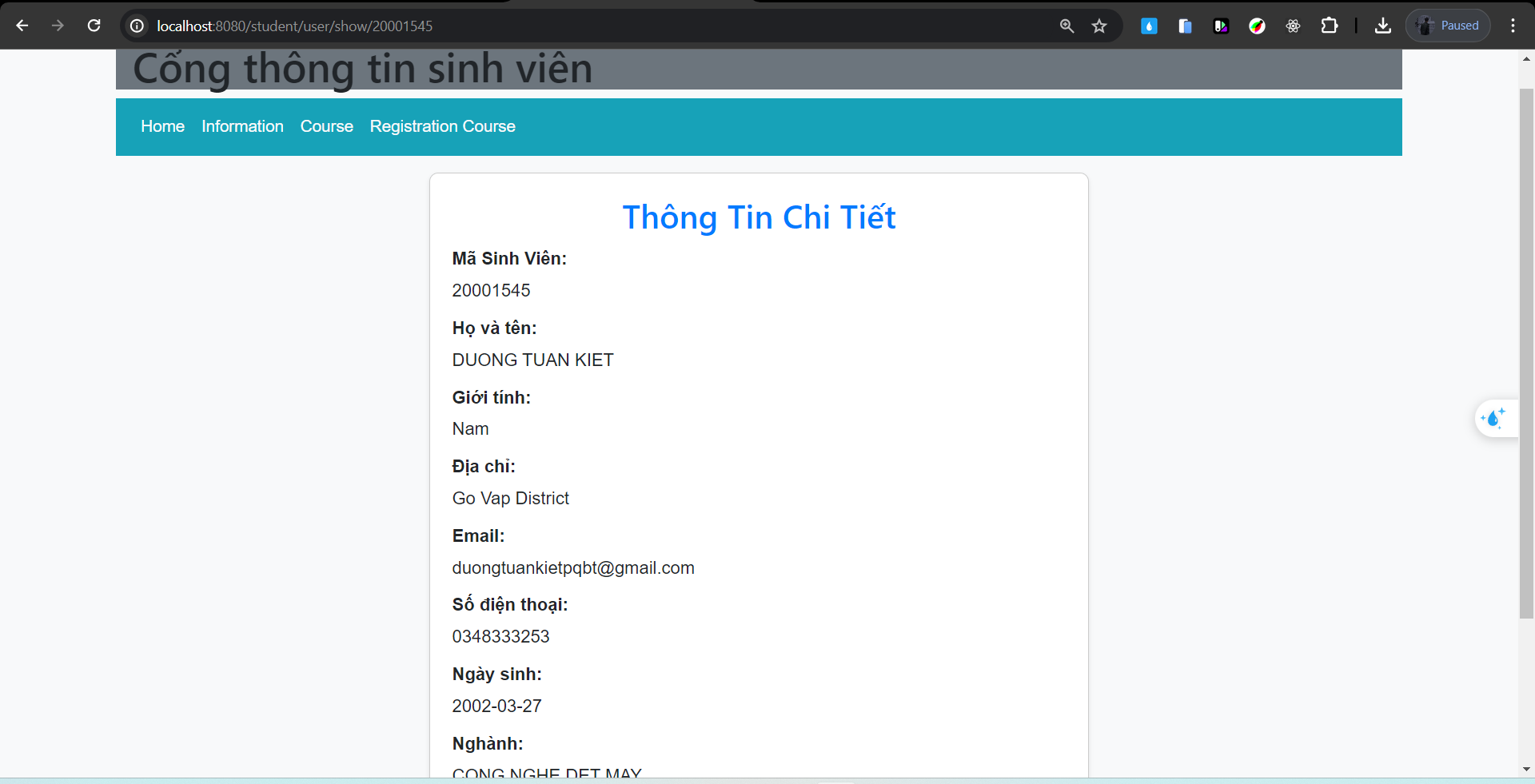
* Danh sách lớp học phần:



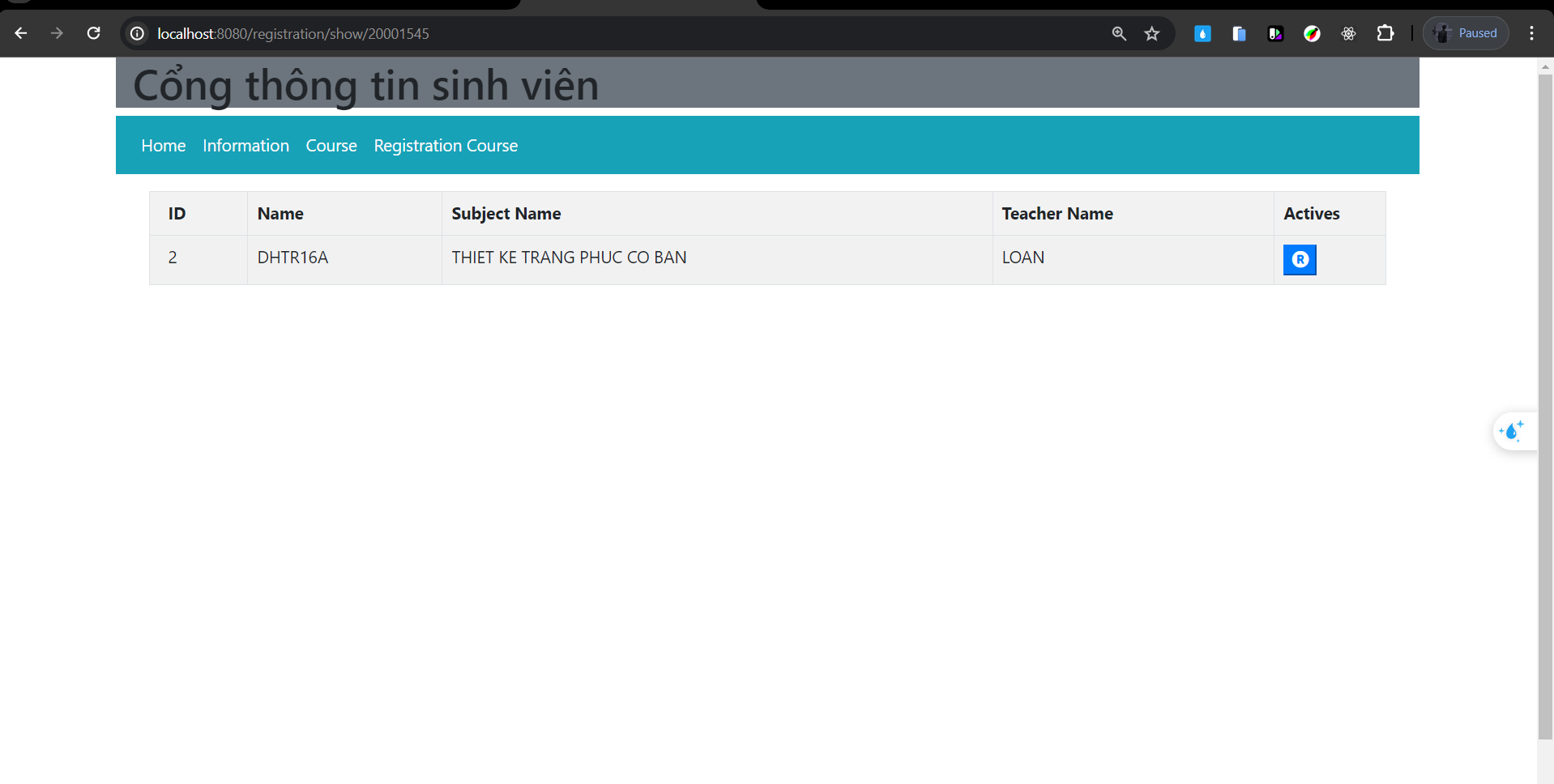
* Bên Sinh Viên:
* Home:

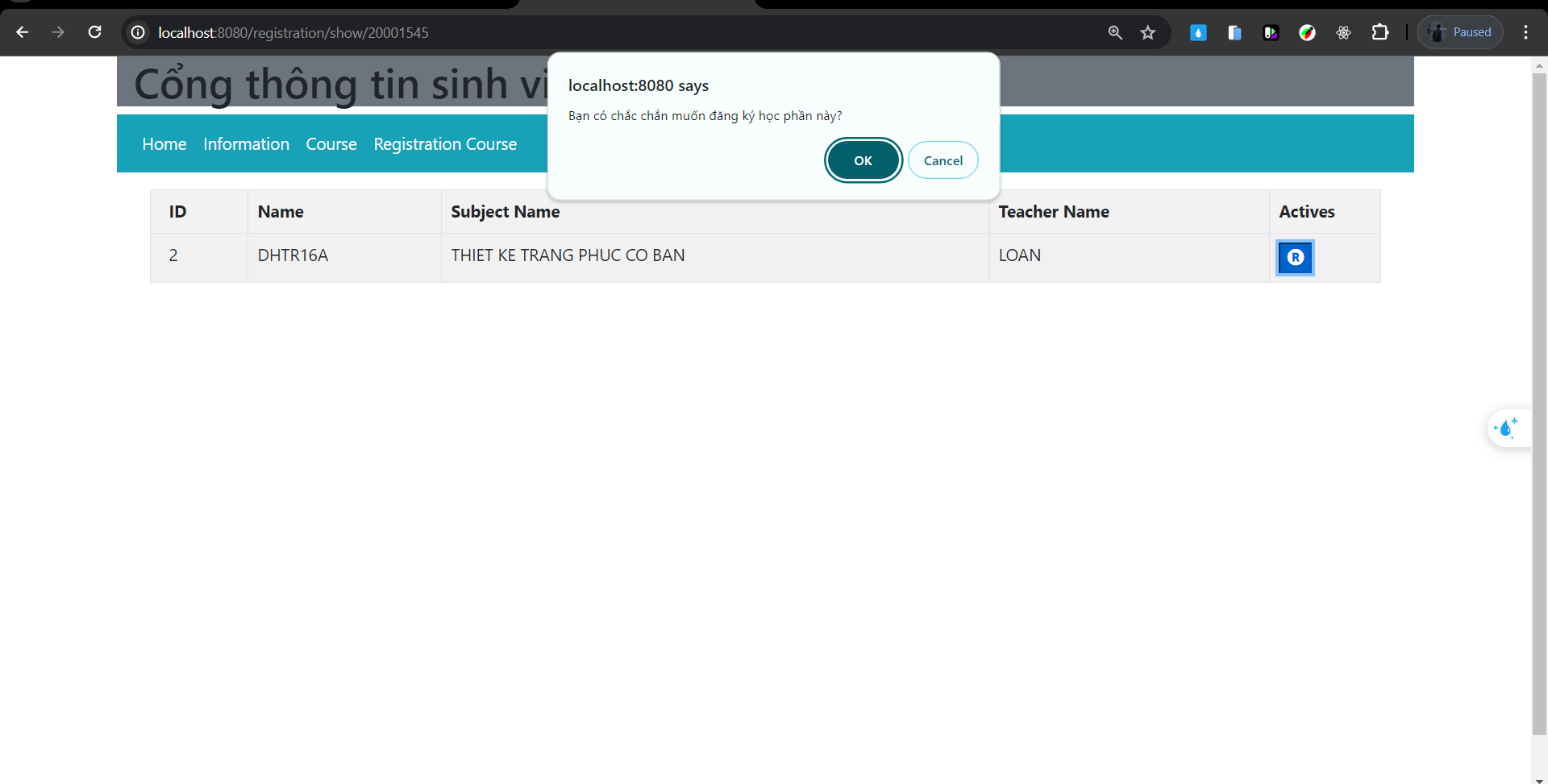


* Thông tin chi tiết:

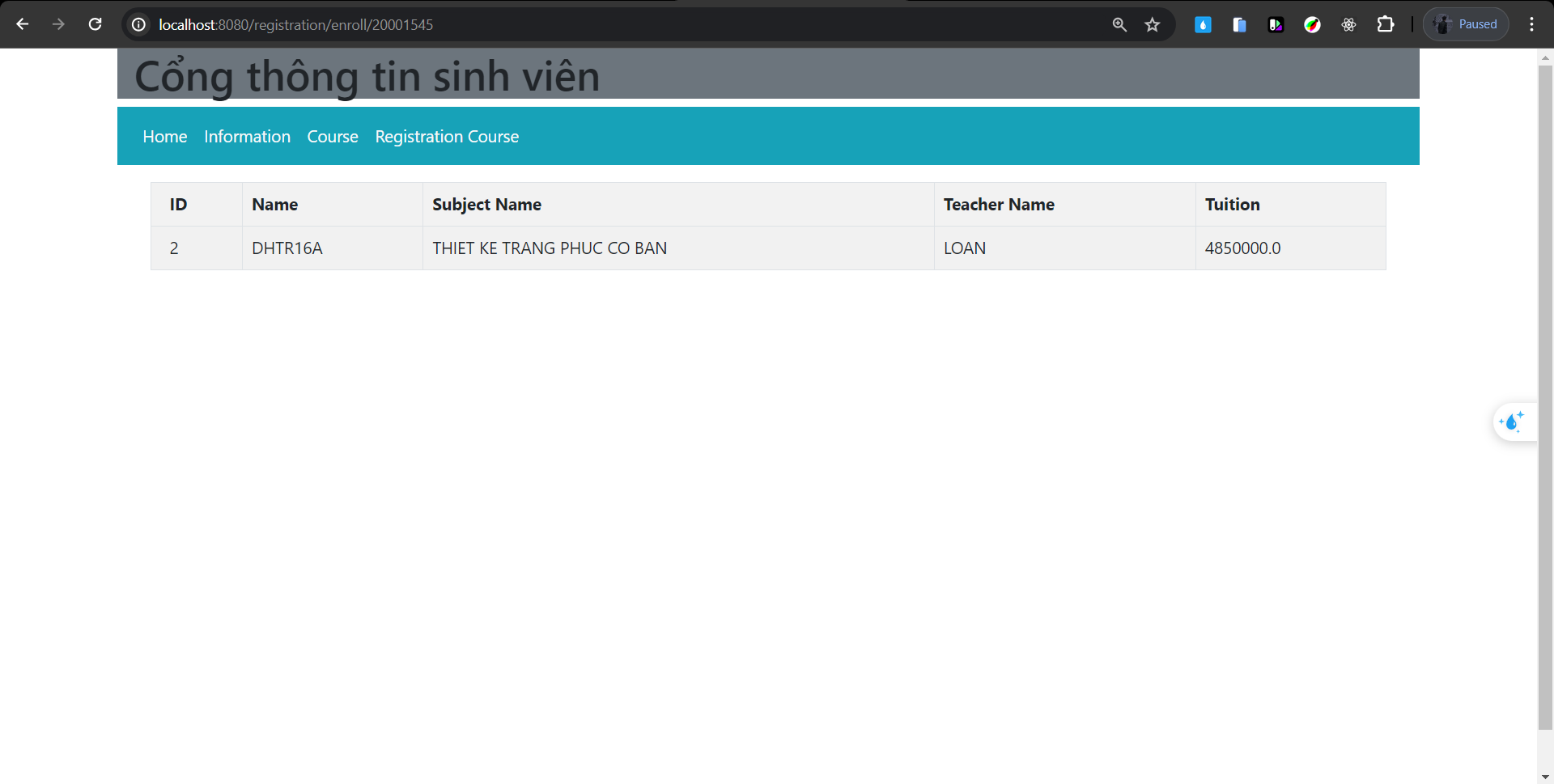


* Đăng ký hóa học:





* 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.

1. Hoàn thành mô hình dữ liệu (Model):

Xây dựng đầy đủ các entity cần thiết như Student, Department, Course, Class, Enrollment.

Thiết lập các mối quan hệ giữa các entity để quản lý thông tin sinh viên, môn học, lớp học và đăng ký học phần.

Triển khai các lớp xử lý nghiệp vụ như RegistrationManager, NotificationManager và GraduationManager, đảm bảo việc kiểm tra điều kiện tiên quyết, quản lý số tín chỉ tối đa, gửi email thông báo, và xét tốt nghiệp.

1. Phát triển giao diện người dùng (View):

Thiết kế giao diện bảng điều khiển cho sinh viên, giúp họ dễ dàng xem thông tin cá nhân, số tín chỉ đã đạt, danh sách môn học có thể đăng ký, và thời khóa biểu.

Xây dựng các giao diện chi tiết về môn học và lớp học, hỗ trợ sinh viên trong việc lựa chọn và đăng ký môn học phù hợp.

Phát triển giao diện xác nhận đăng ký, giúp sinh viên xác nhận đăng ký học phần một cách dễ dàng và rõ ràng.

Thiết kế giao diện đăng ký xét tốt nghiệp, giúp sinh viên kiểm tra điều kiện và đăng ký xét tốt nghiệp thuận tiện.

1. Triển khai các controller (Controller):

Phát triển các controller cho sinh viên và quản trị viên, xử lý các yêu cầu từ người dùng, cập nhật model và chọn view phù hợp để hiển thị kết quả.

Đảm bảo các controller hoạt động chính xác và hiệu quả, hỗ trợ tốt quá trình đăng ký môn học, kiểm tra trạng thái đăng ký, và xét tốt nghiệp.

1. Xây dựng luồng hoạt động của hệ thống:

Thiết kế và triển khai các luồng hoạt động quan trọng như đăng ký môn học, xác nhận đăng ký, và xét tốt nghiệp.

Đảm bảo các luồng hoạt động diễn ra mượt mà, chính xác và phù hợp với yêu cầu nghiệp vụ của nhà trường.

1. Đảm bảo tính bảo mật và ổn định của hệ thống:

Thiết lập các cơ chế bảo mật để bảo vệ thông tin sinh viên và dữ liệu hệ thống.

Kiểm tra và tối ưu hóa hệ thống để đảm bảo hoạt động ổn định và hiệu quả.

## Hướng phát triển

1. Tối ưu hóa trải nghiệm người dùng:

Giao diện thân thiện và dễ sử dụng: Tiếp tục cải thiện giao diện người dùng để trở nên trực quan và dễ sử dụng hơn, đặc biệt là đối với các sinh viên mới.

Ứng dụng di động: Phát triển phiên bản di động của hệ thống để sinh viên có thể đăng ký môn học, kiểm tra thông tin học tập và nhận thông báo mọi lúc, mọi nơi.

1. Tích hợp hệ thống thanh toán học phí:

Cổng thanh toán trực tuyến: Tích hợp các cổng thanh toán như thẻ tín dụng, ví điện tử, và ngân hàng trực tuyến để sinh viên có thể thanh toán học phí một cách dễ dàng và nhanh chóng sau khi đăng ký môn học.

1. Nâng cao tính năng quản lý học tập:

Hệ thống đánh giá và phản hồi: Tích hợp tính năng đánh giá và phản hồi của sinh viên về các môn học và giảng viên, giúp nhà trường cải thiện chất lượng giảng dạy.

Tư vấn học tập tự động: Sử dụng trí tuệ nhân tạo (AI) để tư vấn lộ trình học tập cá nhân hóa cho sinh viên dựa trên thành tích học tập và mục tiêu nghề nghiệp.

1. Mở rộng chức năng hỗ trợ sinh viên:

Cố vấn học tập trực tuyến: Tích hợp tính năng hỗ trợ cố vấn học tập trực tuyến, giúp sinh viên dễ dàng tiếp cận và nhận tư vấn từ các cố vấn học tập.

Thông báo và nhắc nhở: Phát triển hệ thống thông báo và nhắc nhở sinh viên về các sự kiện quan trọng như hạn đăng ký môn học, hạn thanh toán học phí, và lịch thi.

1. Quản lý và phân tích dữ liệu:

Báo cáo và phân tích dữ liệu: Phát triển các công cụ báo cáo và phân tích dữ liệu học tập, giúp nhà trường đưa ra các quyết định quản lý và chiến lược giáo dục dựa trên dữ liệu thực tế.

Hệ thống dự báo: Sử dụng công nghệ học máy (machine learning) để dự báo xu hướng đăng ký môn học, giúp nhà trường lập kế hoạch mở lớp học hiệu quả.

1. Tích hợp với các hệ thống khác:

Tích hợp với hệ thống quản lý nhân sự và tài chính: Đảm bảo sự liên kết giữa hệ thống quản lý lớp học tín chỉ và các hệ thống quản lý nhân sự, tài chính của nhà trường, giúp quản lý đồng bộ và hiệu quả hơn.

Liên kết với các nền tảng học tập trực tuyến: Tích hợp với các nền tảng học tập trực tuyến để sinh viên có thể tiếp cận các khóa học bổ trợ và tài liệu học tập phong phú hơn.

1. Cải thiện tính bảo mật và khả năng mở rộng:

Nâng cao bảo mật: Tiếp tục nâng cao các biện pháp bảo mật để bảo vệ dữ liệu sinh viên và thông tin hệ thống trước các mối đe dọa mạng.

Khả năng mở rộng: Thiết kế hệ thống với khả năng mở rộng cao để có thể dễ dàng bổ sung các tính năng mới và mở rộng quy mô khi cần thiết.

1. Phát triển cộng đồng cựu sinh viên:

Cổng thông tin cựu sinh viên: Xây dựng cổng thông tin cho cựu sinh viên, cung cấp các thông tin về việc làm, các hoạt động mạng lưới, và các sự kiện kết nối.

Khảo sát và phân tích nghề nghiệp: Thực hiện các khảo sát và phân tích về quá trình làm việc của cựu sinh viên, giúp nhà trường điều chỉnh chương trình học đáp ứng nhu cầu thị trường lao động.

# 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/