



**TÊN ĐỀ TÀI: ỨNG DỤNG ĐĂNG KÝ HỌC PHẦN**

**MÔN: KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

***Lớp HP: DHKTPM16A***

***Nhóm 06 - Sinh viên thực hiện:***

Đỗ Quốc Tuấn - 20079191

Phan Hoài An - 20012781

Trần Thị Minh Huyền - 20105231

**MỤC LỤC**

[**CHƯƠNG 1: GIỚI THIỆU**](#_heading=h.57hwh7zg7hzl) **3**

[1.1 Tổng quan](#_heading=h.3dy6vkm) 3

[1.2 Kiến trúc và lý do](#_heading=h.1t3h5sf) 3

[1.3 Ưu điểm](#_heading=h.4d34og8) 3

[1.4 Nhược điểm](#_heading=h.r3vkbhaq4c81) 4

[1.5 Mô tả yêu cầu chức năng](#_heading=h.2s8eyo1) 4

[1.4.1. Yêu cầu chức năng](#_heading=h.5fprhuufkddl) 4

[1.4.2. Yêu cầu phi chức năng](#_heading=h.u7ku9y5qvolo) 5

[**CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ**](#_heading=h.1v1yuxt) **6**

[2.1 Sơ đồ kiến trúc](#_heading=h.2u6wntf) 6

[2.2 Use case Diagram](#_heading=h.z2jdnqyzc2yo) 7

[2.3 Class diagram](#_heading=h.1egqt2p) 8

[2.4 Deployment diagram](#_heading=h.2dlolyb) 8

[2.6 Component diagram](#_heading=h.p7mr5fj8kv2v) 9

[**CHƯƠNG 3: HIỆN THỰC**](#_heading=h.3cqmetx) **10**

[3.1 Đăng ký học phần](#_heading=h.1rvwp1q) 10

[3.2 Thông tin sinh viên](#_heading=h.1rvwp1q) 13

[**CHƯƠNG 4: KẾT LUẬN**](#_heading=h.2r0uhxc) **15**

* 1. [Kết quả đạt được](#_heading=h.1664s55) 15

# CHƯƠNG 1: GIỚI THIỆU

## Tổng quan

Hệ thống quản lý lớp học tín chỉ của trường đại học được thiết kế để đáp ứng nhu cầu đăng ký học phần theo cơ chế tín chỉ của sinh viên. Mỗi học kỳ, sinh viên có thể chọn đăng ký các môn học từ danh sách được nhà trường cung cấp, tuân theo các quy định về số tín chỉ tối đa và các yêu cầu tiên quyết của môn học.

## Kiến trúc và lý do

Microservices Architecture

Phân rã dịch vụ: Hệ thống quản lý lớp học tín chỉ yêu cầu nhiều chức năng độc lập như quản lý người dùng, quản lý môn học, quản lý đăng ký, và xử lý thông tin cá nhân. Mỗi một trong những chức năng này có thể được phát triển và triển khai độc lập dưới dạng microservices. Có thể chia thành nhiều team mỗi team đảm nhận phát triển một dịch vụ giúp tăng tốc thời gian hoàn thành dự án.

Khả năng mở rộng và chịu tải: Trong thời gian bình thường thì hệ thống đăng ký môn học rất ít người sử dụng nhưng trong thời gian đăng ký môn học, hệ thống có thể phải chịu tải rất lớn do tất cả sinh viên cùng truy cập đồng thời. Sử dụng microservices cho phép mở rộng các thành phần riêng lẻ theo nhu cầu thực tế. Ví dụ: khi đến thời gian đăng ký môn học thì cần mở rộng nhiều service đăng ký học phần giúp hệ thống không bị quá tải.

Nâng cấp và bảo trì: Với microservices, việc cập nhật hoặc sửa chữa một phần của hệ thống không ảnh hưởng đến các phần khác, giúp giảm đáng kể thời gian và chi phí bảo trì. Ví dụ: trong thời gian bảo trì service đăng ký học phần thì sinh viên vẫn có thể vào service sinh viên để xem thông tin lịch học, bảng điểm,...

## Ưu điểm

**Improved Scalability**: khả năng mở rộng là một trong những ưu điểm quan trọng nhất của kiến trúc microservices. Theo kiến trúc monolithic việc tăng số lượng người dùng quá cao sẽ ảnh hưởng đến tất cả ứng dụng. Kiến trúc microservices giúp giải quyết vấn đề này. Ví dụ: khi đến thời gian đăng ký học phần, tất cả sinh viên sẽ đăng nhập cùng lúc vào hệ thống đăng ký học phần, điều này gây ra quá tải tại thời điểm đó. Lúc này service đăng ký học phần sẽ được tăng lên để đáp ứng được nhu cầu thay đổi, khi đã hết thời gian đăng ký học phần thì giảm service đăng ký học phần lại để tiết kiệm chi phí.

**Better Fault Isolation for More Resilient Applications**: với kiến trúc microservices, khi một service lỗi thì ít có khả năng tác động tới các service khác do mỗi service chạy độc lập với nhau. Tuy nhiên đối với các hệ thống lớn các service có nhiều phần phụ thuộc với nhau, vì vậy các developer phải tránh tình trạng shutdown các service phụ thuộc lẫn nhau. Cần áp dụng [Circuit Breakers](https://www.c-sharpcorner.com/article/microservice-circuit-breaker-pattern/) để giải quyết vấn đề quá tải đối với dịch vụ.

**Programming Language and Technology Agnostic**: sử dụng kiến trúc microservices các developer có thể chọn bất kì ngôn ngữ nào để tận dụng những ưu điểm cho các yêu cầu thực tế. Các service cũng có thể triển khai trên bất kỳ nền tảng nào. Từ đó giúp các developer có thể áp dụng các giải pháp mới vào dự án một cách dễ dàng hơn. Ví dụ: nếu hệ thống cần tích hợp thêm chức năng thống kê điểm số của sinh viên từ đó sử dụng các mô hình machine learning để dự đoán nếu điểm số như vậy thì tương lai sẽ tốt nghiệp trong bao lâu và xếp loại gì thì cần sử dụng ngôn ngữ python để dễ dàng phát triển.

**Better Data Security and Compliance**: vì mỗi service đảm nhận một nhiệm vụ riêng cho nên dữ liệu cũng sẽ được chia nhỏ cho từng service, điều này giúp bảo mật dữ liệu tốt hơn, nếu như cơ sở dữ liệu của một service bị rò rỉ thì không ảnh hưởng đến service khác. Cách xử lý này giúp cải thiện việc bảo mật tốt hơn là một database cho toàn bộ ứng dụng.

**Support for DevOp**s: là một tập hợp các quy trình tự động hóa để triển khai phần mềm một cách nhanh chóng và hiệu quả. Các service hoạt động độc lập với nhau nên có khả năng tự động hóa quy trình triển khai rất tốt, vì vậy có thể triển khai các service riêng lẻ mà không làm ảnh hưởng đến các phần khác của ứng dụng.

## Nhược điểm

Khi triển khai, quản lý, testing phát triển chức năng gồm nhiều service liên quan sẽ phức tạp hơn kiến trúc monolithic rất nhiều.Vì phải giao tiếp qua lại giữa các service qua internet.

Do chia nhiều service khác nhau nên phải dùng các dịch vụ như messaging hay RPC để giao tiếp nên có thể có nhiều sự cố như kết nối chậm hoặc thông điệp không gửi được tới đích.

Tốn nhiều tài nguyên, dẫn đến chi phí vận hành cao hơn.

Xác định và giải quyết các vấn đề trong kiến trúc microservices cũng phức tạp tương tự như việc tổng hợp các file theo dõi và log. Khi một yêu cầu thất bại đi qua nhiều microservice được lưu trữ trong các môi trường chạy riêng biệt, việc xác định chính xác vị trí và nguyên nhân của sự cố có thể rất khó khăn.

Khó khăn trong việc đảm bảo dữ liệu nhất quán giữa các service. Việc đồng bộ hóa và quản lý trạng thái giữa các service là một thách thức đòi hỏi các developer phải am hiểu sâu sắc về hệ thống để đảm bảo dữ liệu nhất quán.

## Mô tả yêu cầu chức năng

### 1.5.1. Yêu cầu chức năng

* Đăng kí học phần
* Số tín chỉ đã đạt
* Số môn đã học
* Thời khóa buổi
* Thông tin cá nhân

### 1.5.2. Yêu cầu phi chức năng

- Đảm bảo các vấn đề cơ bản về bảo mật, xác thực

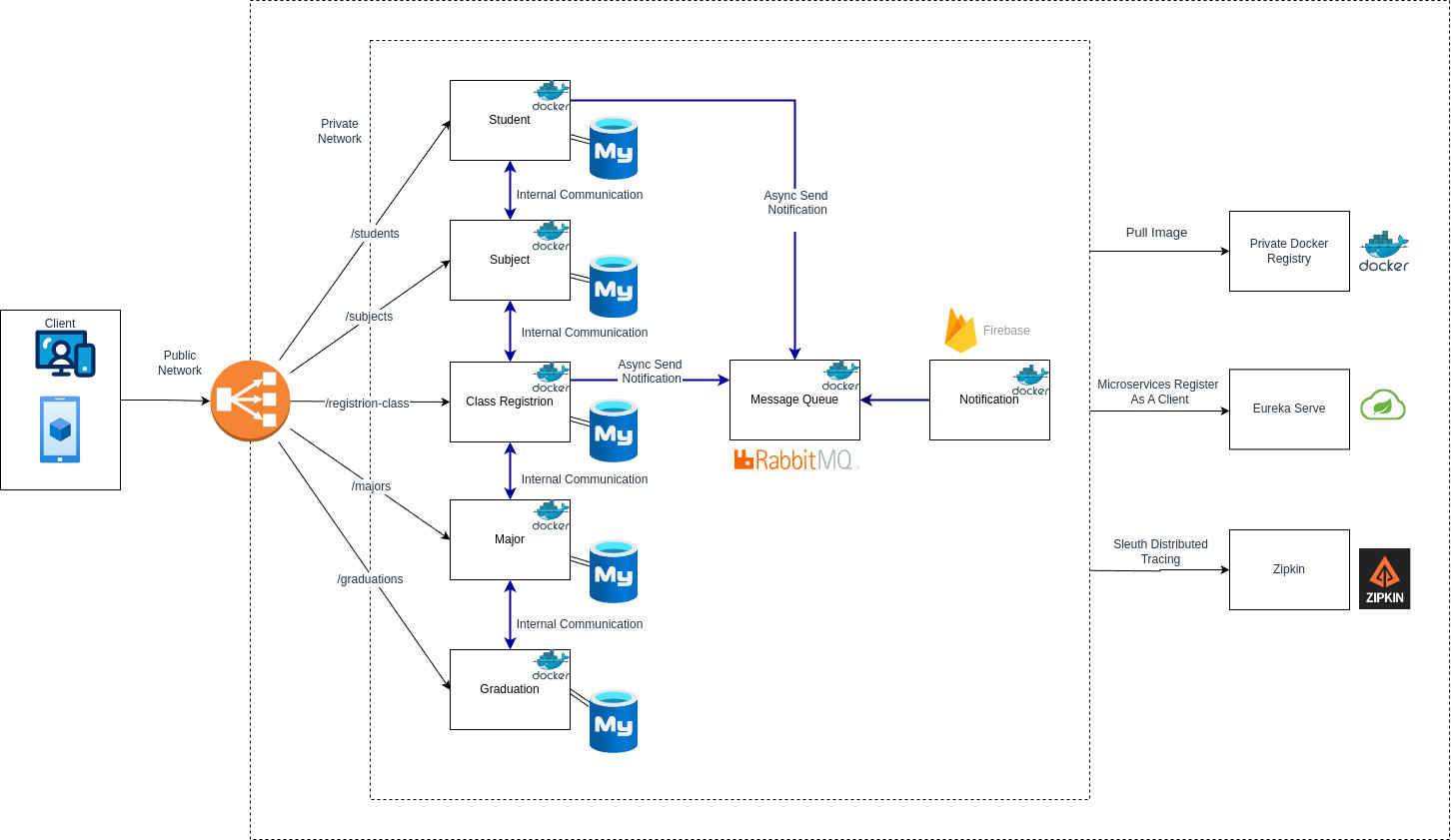
- Dung lượng lưu trữ lớn

- Hệ thống chạy ổn định

- Chịu được lượng truy cập cùng lúc 30.000 user

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ

## 2.1 Sơ đồ kiến trúc



Hệ thống là một mô hình kiến trúc dựa trên microservices được triển khai để quản lý các hoạt động đào tạo sinh viên của một trường đại học. Các service chính của hệ thống bao gồm: Quản lý Sinh viên, Môn học, Đăng ký Lớp học, Chuyên ngành, và Tốt nghiệp. Mỗi service được đóng gói trong Docker và có cơ sở dữ liệu MySQL riêng, đảm bảo tính độc lập và dễ dàng mở rộng.

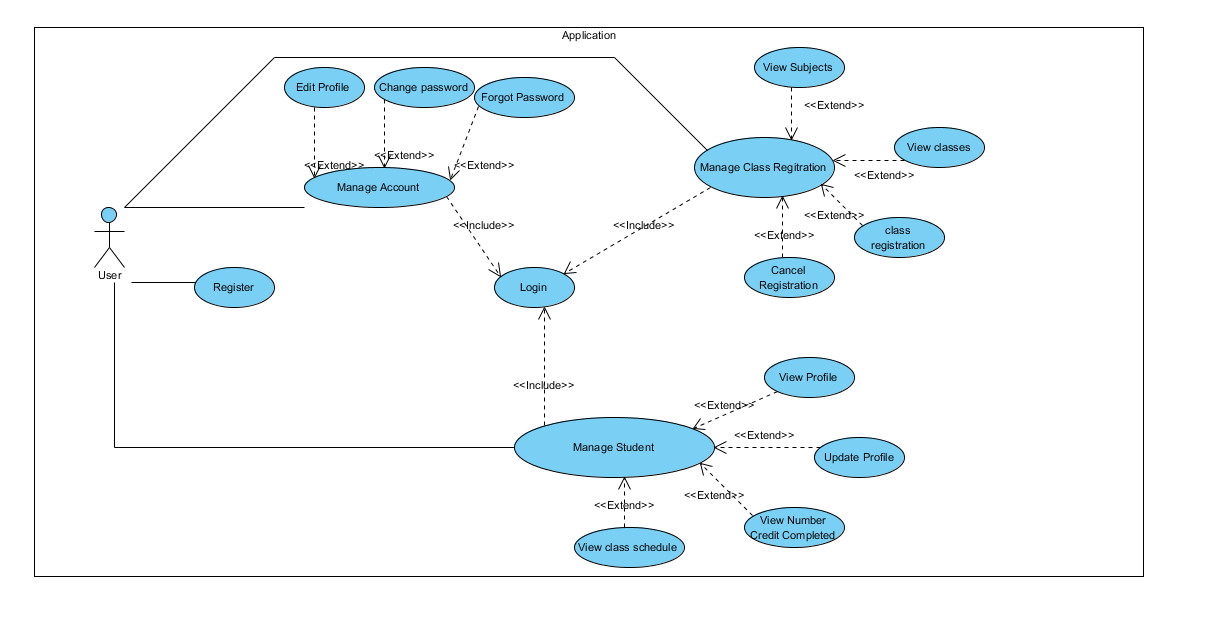
Tất cả các service đều được quản lý thông qua một mạng riêng, với giao tiếp nội bộ được bảo vệ. Các service giao tiếp với nhau qua các API nội bộ để thực hiện các nhiệm vụ như cập nhật thông tin sinh viên, đăng ký môn học, và các hoạt động khác liên quan đến quản lý học tập.

Thông báo đến người dùng được xử lý thông qua RabbitMQ, một hệ thống hàng đợi thông điệp, cho phép gửi thông báo bất đồng bộ đến các sinh viên, ví dụ như thông báo về việc đăng ký lớp học thành công hay các thông tin cập nhật từ nhà trường. Các thông báo này được gửi đi qua Firebase để đảm bảo rằng chúng được phân phối một cách hiệu quả và đáng tin cậy đến các thiết bị di động của người dùng.

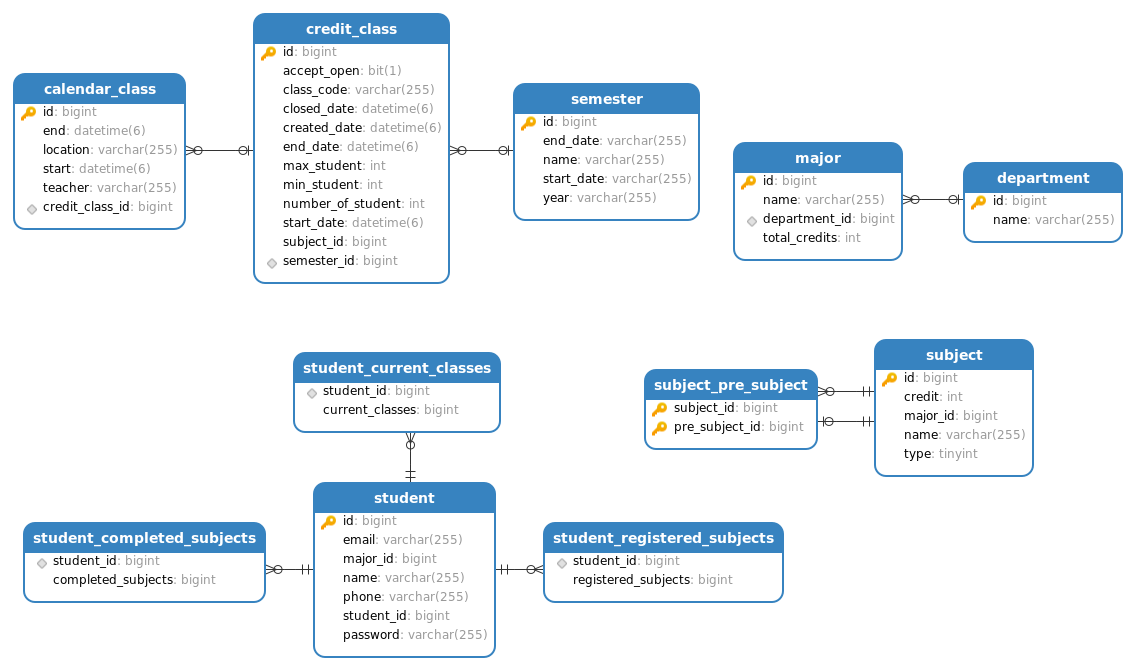
Ngoài ra, hệ thống còn tích hợp Eureka Server để quản lý đăng ký và phát hiện các microservices và Zipkin để theo dõi phân tán, giúp theo dõi và phân tích các luồng giao tiếp và xử lý trong hệ thống. Điều này giúp nhận diện và giải quyết nhanh chóng các vấn đề về hiệu suất hay lỗi trong hệ thống.

Kiến trúc này không chỉ đảm bảo tính linh hoạt và khả năng mở rộng của hệ thống mà còn giúp duy trì hiệu suất ổn định và đáng tin cậy, điều cần thiết cho một môi trường giáo dục đòi hỏi sự liên tục và không gián đoạn trong việc cung cấp dịch vụ đến người dùng cuối.

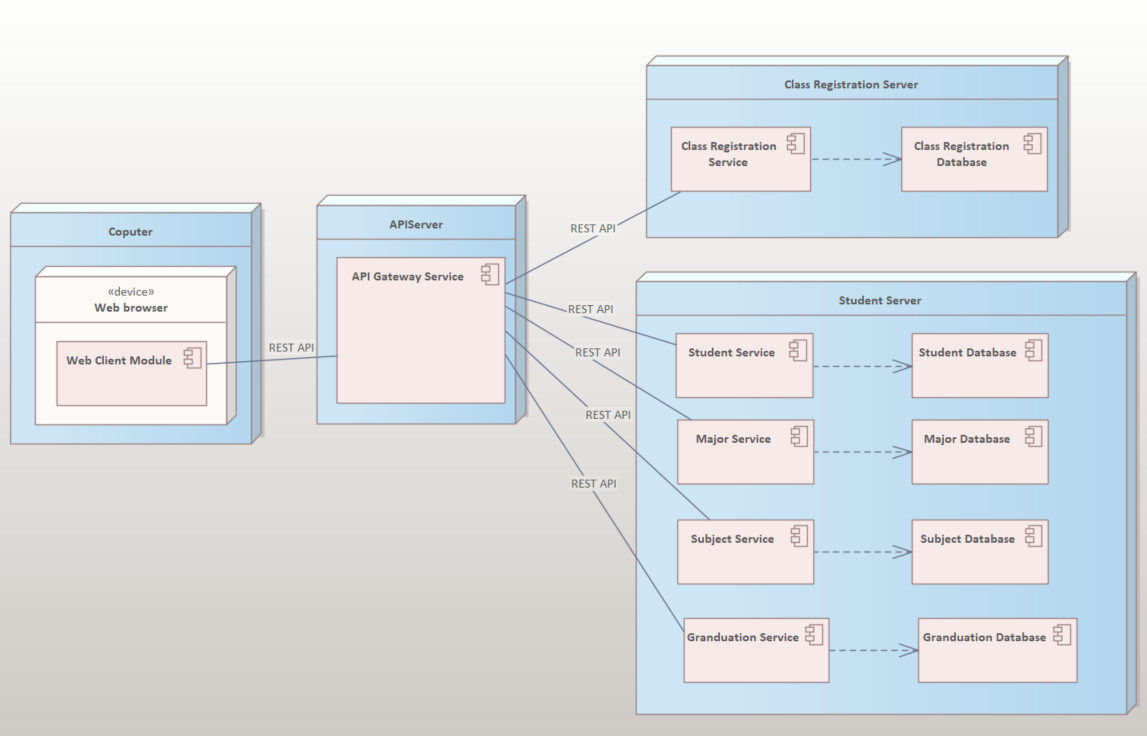
## 2.2 Use case Diagram



## 2.3 Class diagram

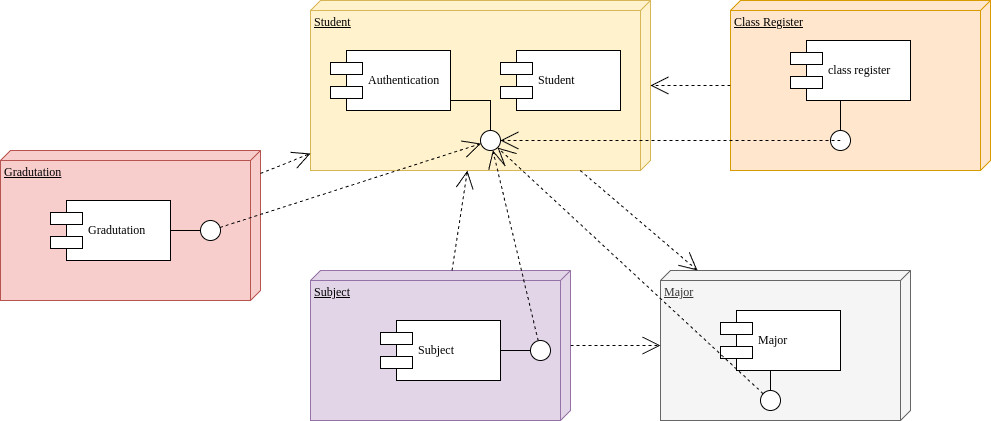


## 2.4 Deployment diagram



## 

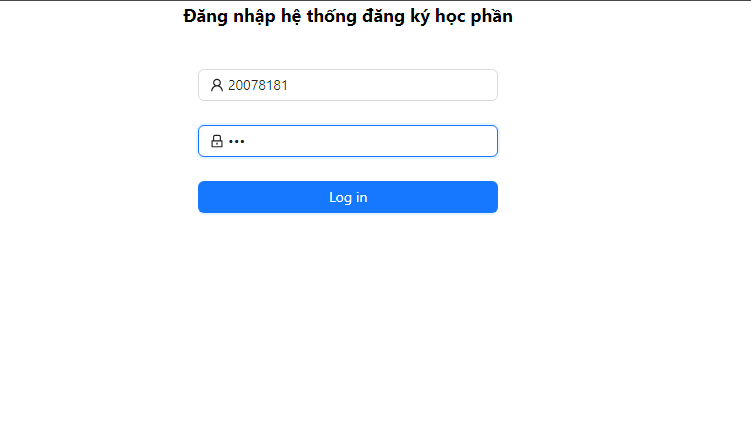
## 2.6 Component diagram

****

# CHƯƠNG 3: HIỆN THỰC

## 3.1 Đăng ký học phần

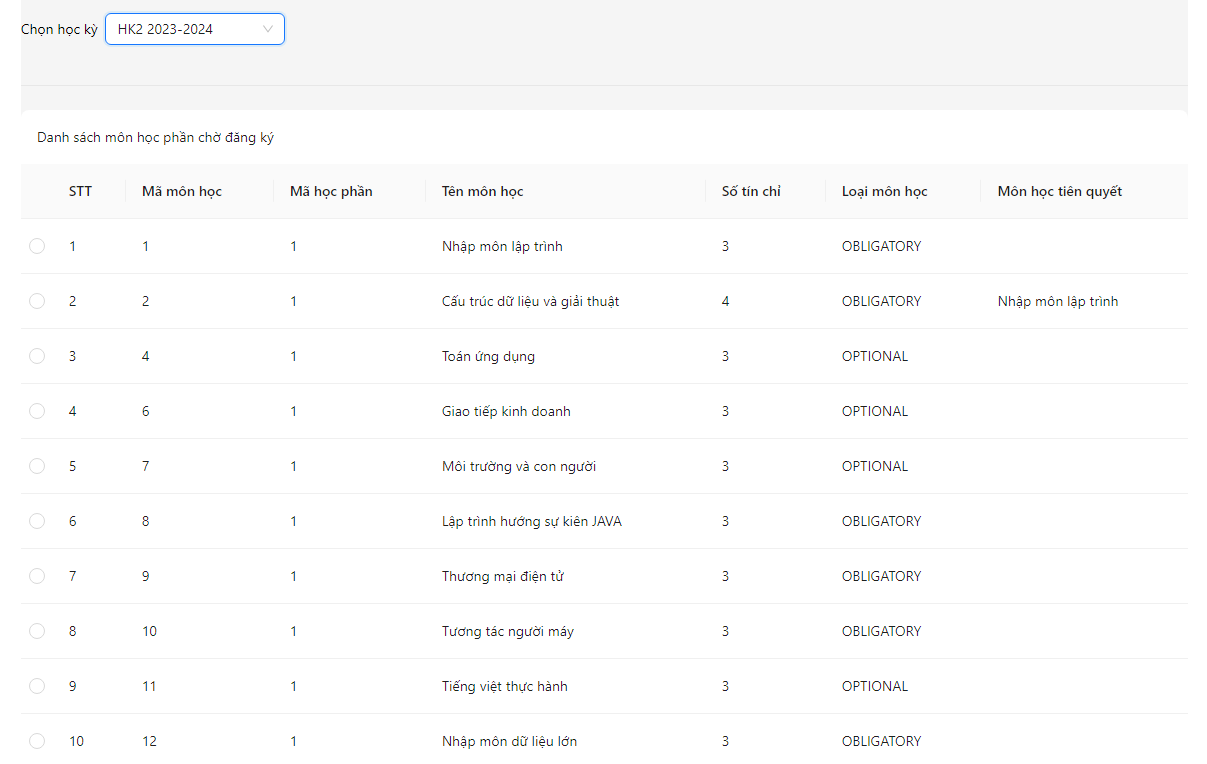
Đăng nhập



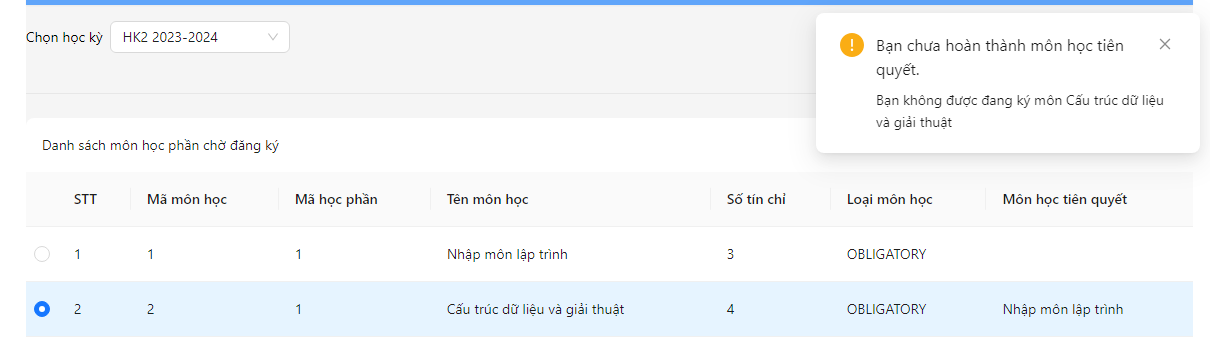
Chọn học kỳ để đăng ký môn học



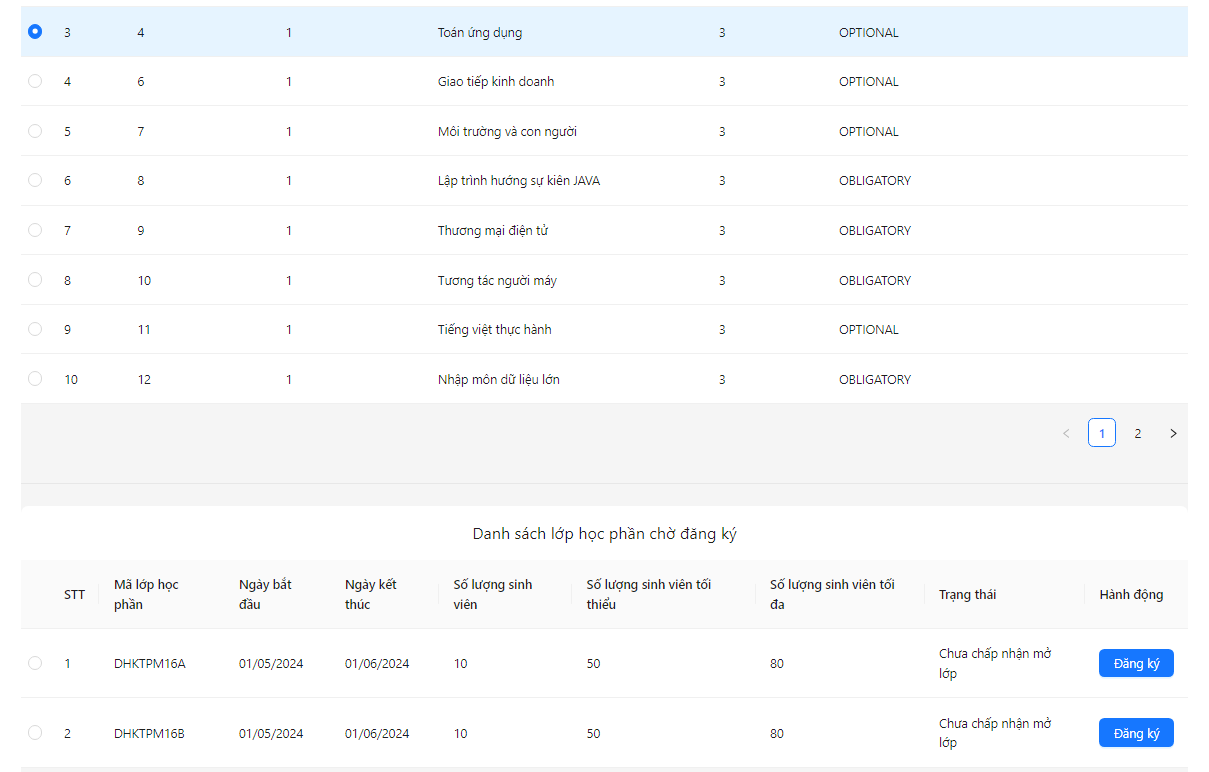
Danh sách các môn mở trong học kỳ đó



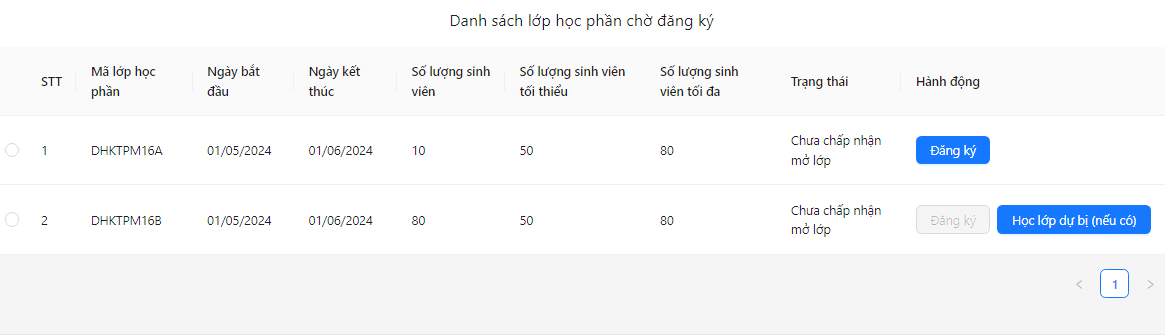
Không đăng ký được khi chưa hoàn thành môn học trước



Hiển thị danh sách các lớp đang mở của 1 môn học



Nếu lớp đầy thì k đăng ký được



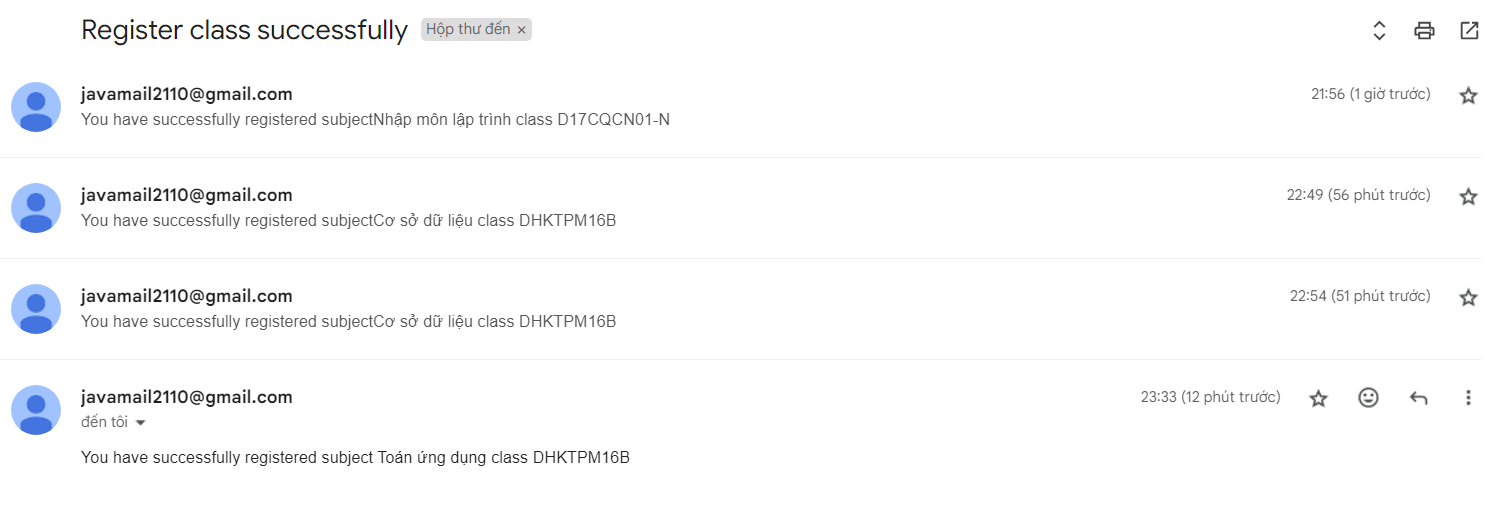
Nếu đăng ký mà vượt quá 30 tín chỉ 1 học kỳ cũng không được



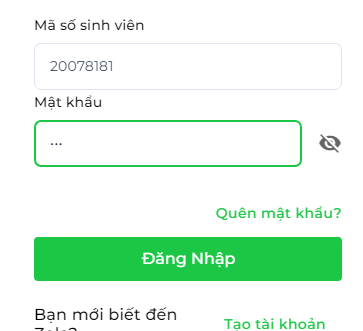
Đăng ký thành công hiển thị ở danh sách đã đăng ký



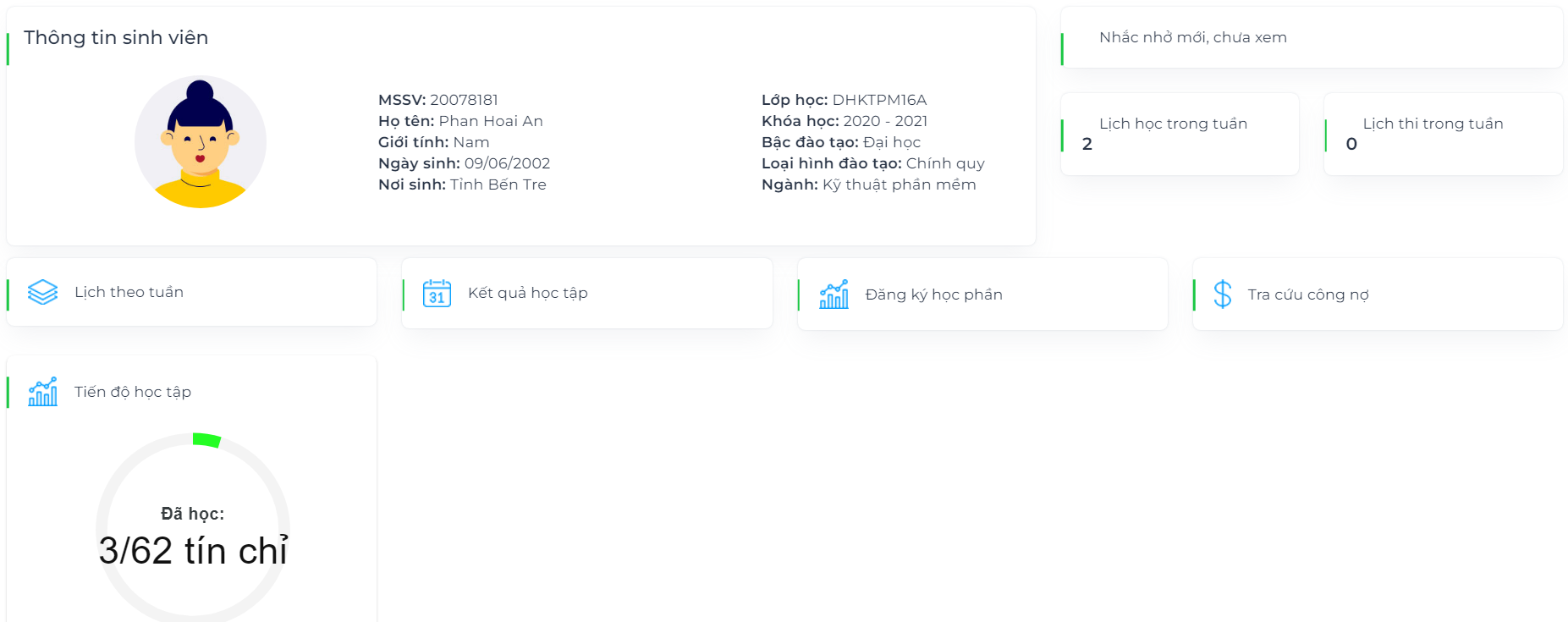
Khi đăng ký thành công sẽ có mail thông báo



## 3.2 Thông tin sinh viên

Đăng nhập

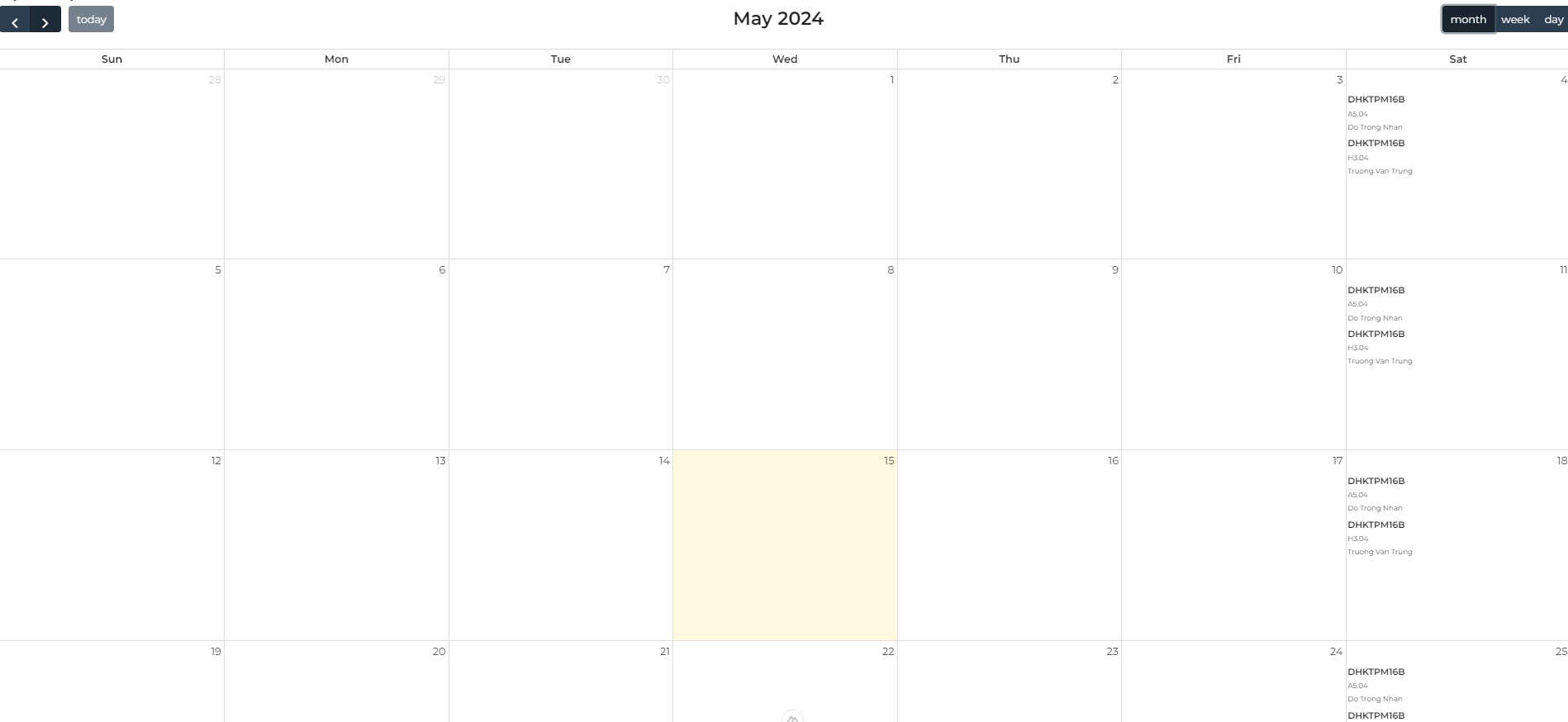
Thông tin sinh viên



Hiển thị lịch học theo tuần



Hiển thị lịch học theo tháng



# CHƯƠNG 4: KẾT LUẬN

## 4.1 Kết quả đạt được

* Đăng nhập.
* Tạo tài khoản sinh viên
* Đăng lý học phần
* Thông tin sinh viên
* Lịch học sinh viên
* Tiến độ học tập của sinh viên