BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

**KHOA khoa CÔNG NGHỆ THÔNG TIN**



**MÔN: KIẾN TRÚC PHẦN MỀM**

**ĐỀ CƯƠNG NGHIÊN CỨU**

***Đề tài:***

**ĐĂNG KÝ HỌC PHẦN**

Nhóm 22 – Sinh viên thực hiện

1. Lê Hữu Bằng – 20049601
2. Nguyễn Quốc Bảo – 21107451
3. Võ Mạnh Hiếu – 21097401

Thành phố Hồ Chí Minh, 19 tháng 05 năm 2024

**MỤC LỤC**

[CHƯƠNG 1 : GIỚI THIỆU 1](#_Toc166926494)

[1.1 Mô tả yêu cầu cơ bản của hệ thống 1](#_Toc166926495)

[1.2 Lý do chọn Layered Architecture: 2](#_Toc166926496)

[1.3 Giải thuyết để chọn Layered Architecture: 2](#_Toc166926497)

[1.4 Layered Architecture có những thuận lợi và khó khăn: 3](#_Toc166926498)

[CHƯƠNG 2 : MÔ HÌNH MINH HOẠ CHO KIẾN TRÚC 5](#_Toc166926499)

[2.1 Mô hình liên kết 5](#_Toc166926500)

[2.2 Mô hình kiến trúc phân lớp: (Layered Architecture) 5](#_Toc166926501)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 6](#_Toc166926502)

[3.1 Phân tích yêu cầu bằng UML 6](#_Toc166926503)

[3.1.1 Usecase tổng quát 6](#_Toc166926504)

[3.1.2 Danh sách tác nhân và mô tả 6](#_Toc166926505)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 6](#_Toc166926506)

[3.2 Class diagram 16](#_Toc166926507)

[3.3 Deployment diagram 17](#_Toc166926508)

[3.4 Package diagram 18](#_Toc166926509)

[CHƯƠNG 4 : HIỆN THỰC 19](#_Toc166926511)

[4.1 Cấu hình phần cứng, phần mềm 19](#_Toc166926512)

[4.2 Giao diện của hệ thống 19](#_Toc166926513)

[CHƯƠNG 5 : KẾT LUẬN 23](#_Toc166926514)

[5.1 Kết quả đạt được: 23](#_Toc166926515)

[5.2 Hạn chế của đồ án: 23](#_Toc166926516)

[5.3 Hướng phát triển: 23](#_Toc166926517)

[TÀI LIỆU THAM KHẢO 24](#_Toc166926518)

# : GIỚI THIỆU

## Mô tả yêu cầu cơ bản của hệ thống

Một trường đại học cần quản lý hệ thống đăng ký học phần của sinh viên theo theo cơ chế tín chỉ. Các sinh viên sẽ được biên chế ở các khoa theo mỗi ngành học của mình đăng ký.Trong mỗi học kỳ, nhà trường sẽ chọn một số môn học của từng ngành học cho sinh viên đăng ký học. Sinh viên đăng ký theo học vào một lớp nhất định (lớp tín chỉ) với số sinh viên được đăng ký tối đa được qui định cho từng lớp.Môn học được mở có thể là môn học tự chọn Mỗi môn học có một hoặc nhiều môn học tiên quyết. Sinh viên sẽ không được đăng ký môn học mà có môn tiên quyết mình chưa học. Mỗi học kỳ sinh viên được đăng ký tối đa 30 tín chỉ. Nếu quá số này, hệ thống sẽ không cho đăng ký.

Để phòng ngừa trường hợp sinh viên đăng ký xong rồi hủy bỏ, nhà trường yêu cầu sinh viên phải xác nhận trước khi đăng ký. Vào ngày mở đăng ký, sinh viên sẽ đăng nhập vào hệ thống và sẽ nhìn thấy danh sách các môn học mà mình có khả năng đăng ký. Sinh viên chọn các môn học và tiến hành đăng ký. Trường hợp các lớp đã đầy, sinh viên sẽ được đưa vào một danh sách dự bị để nhà trường cân nhắc có mở thêm lớp hay không. Nếu không mở thêm lớp, sinh viên sẽ bị hủy đăng ký môn đó.

Sau khi đăng ký thành công, một email thông báo sẽ được gửi cho sinh viên xác nhận việc đăng ký và nhận quyết định đóng học phí.

Ngoài ra, hệ thống còn có khả năng cho phép sinh viên xem thông tin học tập của mình (số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy…), thời khóa biểu theo tuần và các tiện ích khác. Sau khi số tín chỉ đã đạt theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp. Nếu mọi tiêu chuẩn đều thỏa mãn, sinh viên sẽ được cấp bằng tốt nghiệp và sẽ được đưa vào danh sách các cựu sinh viên. Thông tin về bằng cấp sẽ được công khai trên trang web của nhà trường. Thông tin của cựu sinh viên sẽ được lưu giữ để theo dõi quá trình làm việc (nếu sinh viên đồng ý), làm các cuộc survey, cũng như nhiều hoạt động khác.

## Lý do chọn Layered Architecture:

Layered là kiến trúc phổ biến được sử dụng rộng rãi trong các ứng dụng phần mềm, phù hợp với ứng dụng xác định rõ ràng về mặt chức năng của các lớp. Tính modun hóa cho từng lớp chức năng, giúp kiểm thử và bảo trì dễ dàng hơn. Đồng thời có thể tái sử dụng ở những phần khác của dự án.

Kiến trúc layered có thể giúp cải thiện bảo mật phần mềm bằng cách cô lập các lớp khỏi nhau.Điều này giúp hạn chế tác động của lỗi bảo mật trong một lớp đối với các lớp khác.Kiến trúc layered cho phép thay đổi dễ dàng các thành phần bên trong của một lớp mà không ảnh hưởng đến giao diện bên ngoài của nó. Ví dụ, lớp quản lý lớp học có thể được thay đổi để sử dụng một cơ sở dữ liệu mới mà không cần thay đổi cách thức đăng ký lớp học của sinh viên.

## Giải thuyết để chọn Layered Architecture:

 Hệ thống cần quản lý 10.000 sinh viên, 20 khoa, 50 ngành học và 100 môn học.

**Giả thuyết:**

* Số lượng sinh viên đăng ký mỗi học kỳ trung bình là 5.000 sinh viên.
* Mỗi sinh viên đăng ký trung bình 5 môn học mỗi học kỳ.
* Mỗi môn học có sĩ số tối đa 45 sinh viên.

**Phân tích:**

* Hệ thống cần quản lý lượng dữ liệu lớn về sinh viên, khoa, ngành học, môn học, lịch học, điểm số, v.v.
* Hệ thống cần xử lý đồng thời nhiều yêu cầu đăng ký học phần từ sinh viên.
* Hệ thống cần đảm bảo tính bảo mật dữ liệu của sinh viên.

Hệ thống cần tích hợp với các hệ thống khác của nhà trường như hệ thống quản lý tài chính, hệ thống quản lý thư viện.

**Giả thuyết:** Hệ thống cần trao đổi dữ liệu với các hệ thống khác như thông tin sinh viên, điểm số, lịch học.

**Phân tích:**

* Hệ thống cần có khả năng tương tác với các hệ thống khác một cách hiệu quả.
* Việc tích hợp dữ liệu từ các hệ thống khác giúp nâng cao hiệu quả hoạt động của hệ thống quản lý lớp học tín chỉ.

## Layered Architecture có những thuận lợi và khó khăn:

**Thuận lợi của Layered Architecture:**

* Đơn giản và rõ ràng: Kiến trúc phân lớp dễ hiểu và dễ triển khai, đặc biệt đối với các đội ngũ phát triển mới hoặc có kinh nghiệm ít. Các lớp được phân chia rõ ràng với các nhiệm vụ cụ thể, giúp tăng cường khả năng quản lý và bảo trì.
* Tách biệt các mối quan tâm: Mỗi lớp có một chức năng riêng biệt, giúp dễ dàng quản lý và phát triển từng phần một cách độc lập.
* Khả năng mở rộng: Hệ thống có thể mở rộng bằng cách thêm các lớp mới hoặc mở rộng các lớp hiện có mà không ảnh hưởng lớn đến các phần khác của hệ thống.
* Dễ bảo trì: Sự tách biệt rõ ràng giữa các lớp giúp dễ dàng thay đổi, sửa chữa hoặc cập nhật một lớp mà không ảnh hưởng đến các lớp khác.

**Khó khăn của Layered Architecture:**

* Hiệu suất: Việc đi qua nhiều lớp có thể làm tăng độ trễ và giảm hiệu suất của hệ thống, đặc biệt khi các lớp không được tối ưu hóa tốt.
* Phức tạp trong việc thay đổi: Khi có sự thay đổi lớn ở một lớp nào đó, có thể phải thực hiện thay đổi tương ứng ở các lớp khác, làm tăng độ phức tạp và công sức phát triển.
* Khó khăn trong việc áp dụng tính năng mới: Khi hệ thống yêu cầu thêm tính năng mới, có thể phải thay đổi hoặc thêm mới nhiều lớp, điều này đôi khi gây ra sự phức tạp và tốn kém thời gian.

**Giá của chọn việc Layered Architecture:**

* Hiệu suất và độ trễ

Việc xử lý qua nhiều lớp có thể dẫn đến độ trễ cao hơn so với các kiến trúc khác. Mỗi lớp thêm vào độ trễ khi xử lý yêu cầu, từ lớp giao diện người dùng, qua lớp logic nghiệp vụ, đến lớp truy cập dữ liệu và ngược lại.

Hiệu suất tổng thể có thể bị ảnh hưởng nếu không tối ưu hóa kỹ lưỡng từng lớp.

* Phức tạp trong việc thay đổi và bảo trì

Thay đổi một lớp có thể yêu cầu thay đổi ở các lớp khác, gây ra sự phức tạp và tốn kém thời gian.

Khi cần cập nhật hoặc thêm tính năng mới, việc phải thay đổi nhiều lớp có thể gây ra lỗi và khó khăn trong việc kiểm thử.

* Độ phức tạp khi triển khai và quản lý

Triển khai và quản lý hệ thống phân lớp yêu cầu kiến thức và kỹ năng để đảm bảo các lớp được tách biệt và giao tiếp đúng cách.

Phải quản lý các kết nối và phụ thuộc giữa các lớp, có thể phức tạp khi hệ thống lớn lên.

* Tính linh hoạt và khả năng mở rộng

Tính linh hoạt và khả năng mở rộng hạn chế so với các kiến trúc hiện đại như Microservices. Khi một lớp cần mở rộng, phải đảm bảo tất cả các lớp liên quan được cập nhật phù hợp.

Khả năng mở rộng tốt hơn Monolithic Architecture nhưng không linh hoạt và dễ dàng như Microservices.

# : MÔ HÌNH MINH HOẠ CHO KIẾN TRÚC

## Mô hình liên kết

## Mô hình kiến trúc phân lớp: (Layered Architecture)

A diagram of a software process

Description automatically generated with medium confidence

# : PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

### Usecase tổng quát

***A diagram of a diagram

Description automatically generated***

### 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 |
| Sinh viên | Sinh viên sử dụng hệ thống để đăng ký lớp học phần, xem kết quả học tập và xem lịch học |
| Giáo vụ | Người tạo lớp học phần cho sinh viên đăng ký |

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 ký học phần |
| UC02 | Đăng nhập |
| UC03 | Xem kết quả học tập |
| UC04 | Xem chi tiết học phần |
| UC05 | Hủy học phần |
| UC06 | Tạo lớp học phần |
| UC07 | Xem lịch học |

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

**UC01: Đăng ký học phần**

1. Activity diagram.

A diagram of a flowchart

Description automatically generated

1. Sequence diagram

A diagram of a project

Description automatically generated

**UC02: Đăng nhập**

1. Activity diagram.

A diagram of a flowchart

Description automatically generated

1. Sequence diagram

A diagram of a diagram

Description automatically generated

**UC03: Xem kết quả học tập**

1. Activity diagram.

A screenshot of a computer

Description automatically generated

1. Sequence diagram

A diagram of a diagram

Description automatically generated

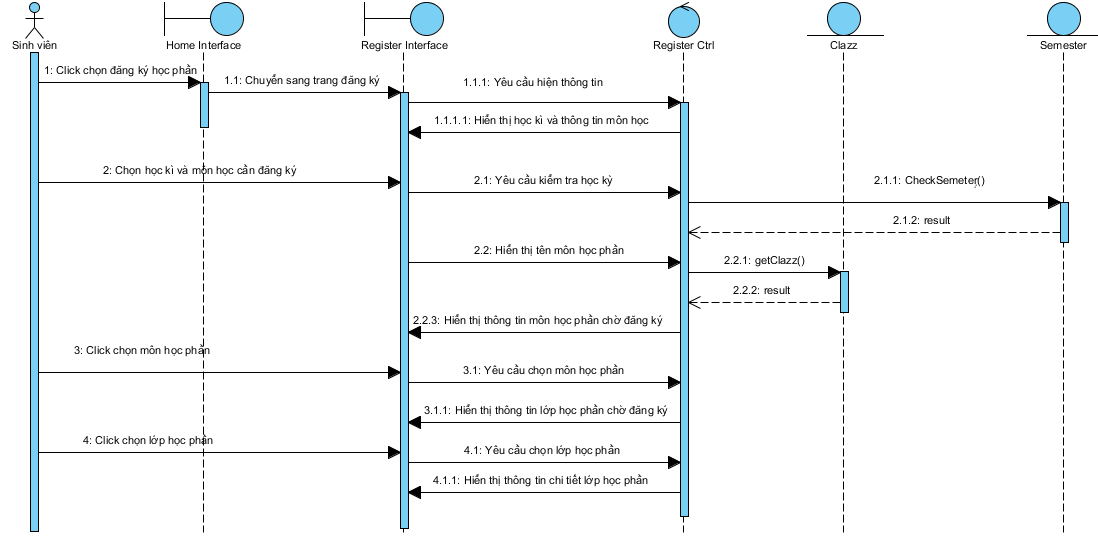
**UC04: Xem chi tiết học phần**

1. Activity diagram.

A diagram of a diagram

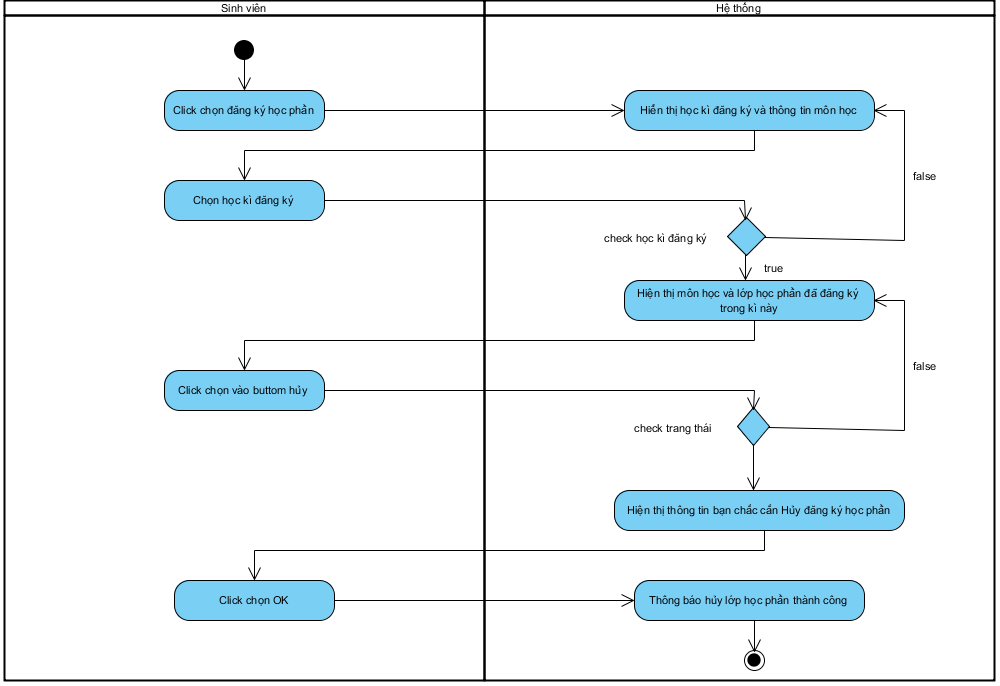
Description automatically generated with medium confidence

1. Sequence diagram

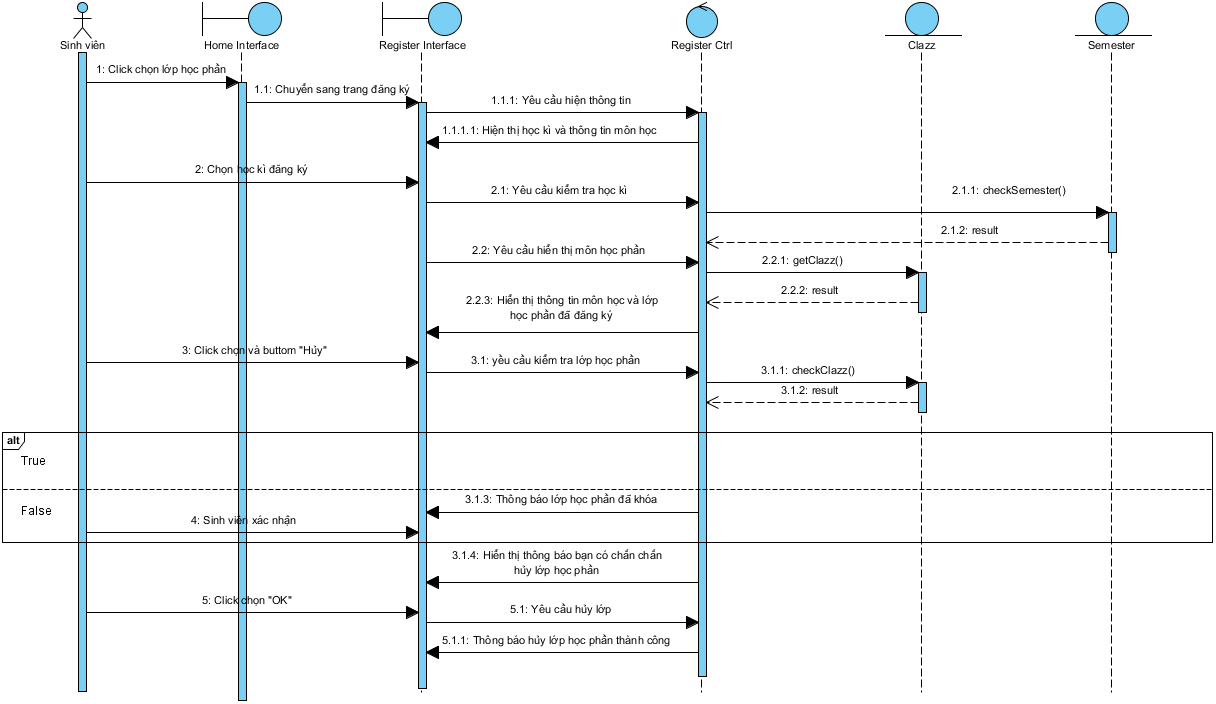


**UC05: Hủy học phần**

1. Activity diagram.



1. Sequence diagram



**UC06: Tạo lớp học phần**

1. Activity diagram.

A diagram of a chat

Description automatically generated with medium confidence

1. Sequence diagram

A diagram of a project

Description automatically generated

**UC07: Xem lịch học**

1. Activity diagram.

A diagram of a flowchart

Description automatically generated

1. Sequence diagram

A diagram of a diagram

Description automatically generated

## Class diagram

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

A white background with many black text

Description automatically generated with medium confidence

## Deployment diagram

Mô hình triển khai hệ thống.

A diagram of a computer software

Description automatically generated

## Package diagram

## A computer screen shot of a diagram Description automatically generated

# : HIỆN THỰC

## Cấu hình phần cứng, phần mềm

a. Máy Chủ (Server):

* CPU: Sử dụng CPU với ít nhất 4 lõi và hỗ trợ đa luồng để xử lý các yêu cầu đồng thời từ các layered.
* RAM: Đảm bảo có đủ bộ nhớ RAM để chạy các ứng dụng Java và các dịch vụ Mariadb một cách mượt mà. Tối thiểu 8GB RAM là một lựa chọn phù hợp.

b. Các Dịch Vụ Google Sử Dụng:

* Flaticon: Để lấy các icon
* Google Chrome: Để chạy giao diện fontend

## Giao diện của hệ thống

Màn hình chức năng đã thực hiện:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

# : KẾT LUẬN

## Kết quả đạt được:

 Hoàn thành một số chức năng cơ bản như:

* Đăng ký học phần
* Xem lịch học
* Xem kết quả học tập
* Hủy học phần
* Đăng nhập
* Xem thông tin sinh viên

## Hạn chế của đồ án:

* Giáo vụ khoa chưa tạo được lớp học phần
* Chưa xử lý được tác vụ đồng thời
* Tính bảo mật chưa cao

## Hướng phát triển:

* Sử dụng ReactJS, Angular  cho ứng dụng web để tăng trải nghiệm người dùng với các tính năng và mượt mà
* Sử dụng React Native hoặc Flutter để phát triển cho ứng dụng di động cho cả Android và IOS
* Thiết lập CI/CD giúp hệ thống trong trạng thái ổn định và dễ dàng triển khai các bản cập nhật
* Tập trung thiết kế giao diện người dùng thân thiện, dễ sử dụng
* Sử dụng OAuth2 hoặc OpenID Connect: để xác thực người dùng và quản lý truy cập

# TÀI LIỆU THAM KHẢO

Các tài liệu từ Internet

1. Spring. Spring Document Guide [Online]. Từ <https://spring.io/quickstart>
2. Ant Design. Ant Design UI components Document [Online]. Từ <https://ant.design/components/overview/>
3. Ant Design. Ant Design UI components Document [Online]. Từ <https://ant.design/components/overview/>