**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



TRANG WEB QUẢN LÍ ĐĂNG KÍ HỌC PHẦN CHO SINH VIÊN TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TPHCM

***Nhóm XX - Sinh viên thực hiện***

1. Họ tên 1 – Mã SV
2. Họ tên 2 – Mã SV
3. Họ tên 3 – Mã SV

# MỤC LỤC

[MỤC LỤC 2](#_Toc166712722)

[CHƯƠNG 1 : GIỚI THIỆU 3](#_Toc166712723)

[1.1 Tổng quan 3](#_Toc166712724)

[1.2 Mục tiêu đề tài 4](#_Toc166712725)

[1.3 Phạm vi đề tài 5](#_Toc166712726)

[1.4 Mô tả yêu cầu chức năng 5](#_Toc166712727)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 7](#_Toc166712728)

[2.1 Các loại kiến trúc có thể áp dụng vào dự án? 7](#_Toc166712729)

[2.1.1 Kiến trúc Layer 7](#_Toc166712730)

[2.1.2 Kiến trúc Microservice 7](#_Toc166712731)

[2.2 Phân tích lựa chọn kiến trúc 7](#_Toc166712732)

[2.3 Các công nghệ sử dụng 7](#_Toc166712733)

[2.3.1 Java 7](#_Toc166712734)

[2.3.2 JWT (JSON Web Token) 7](#_Toc166712735)

[2.3.3 Spring boot 7](#_Toc166712736)

[2.3.4 Postman 7](#_Toc166712737)

[2.3.5 Apache Kafka 7](#_Toc166712738)

[2.3.6 Docker 7](#_Toc166712739)

[2.3.7 RDS của AWS (Amazon Relational Database Service) 7](#_Toc166712740)

[2.3.8 MariaDB 7](#_Toc166712741)

[2.3.9 Github 7](#_Toc166712742)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 8](#_Toc166712743)

[3.1 Phân tích yêu cầu bằng UML 8](#_Toc166712744)

[3.1.1 Usecase tổng quát 8](#_Toc166712745)

[3.1.2 Danh sách tác nhân và mô tả 8](#_Toc166712746)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 9](#_Toc166712747)

[3.1.4 Tình huống hoạt động 10](#_Toc166712748)

[3.2 Class diagram 10](#_Toc166712749)

[3.3 Deployment diagram 10](#_Toc166712750)

[CHƯƠNG 4 : HIỆN THỰC 11](#_Toc166712751)

[4.1 Cấu hình phần cứng, phần mềm 11](#_Toc166712752)

[4.2 Giao diện của hệ thống 11](#_Toc166712753)

[4.3 Kế hoạch và hiện thực kiểm thử hệ thống 11](#_Toc166712754)

[CHƯƠNG 5 : KẾT LUẬN 12](#_Toc166712755)

[5.1 Kết quả đạt được 12](#_Toc166712756)

[5.2 Hạn chế của đồ án 12](#_Toc166712757)

[5.3 Hướng phát triển 12](#_Toc166712758)

[TÀI LIỆU THAM KHẢO 13](#_Toc166712759)

# : GIỚI THIỆU

## Tổng quan

* + 1. **Giới thiệu hệ thống**

Hệ thống quản lý lớp học tín chỉ là một phần mềm trực tuyến hỗ trợ sinh viên trong quá trình đăng ký các học phần theo cơ chế tín chỉ. Hệ thống này giúp tự động hóa các quy trình quản lý đăng ký học phần, từ việc hiển thị danh sách các môn học có sẵn, kiểm tra điều kiện tiên quyết, xác nhận đăng ký, đến việc gửi thông báo qua email. Hệ thống không chỉ giúp giảm bớt khối lượng công việc thủ công cho nhân viên phòng đào tạo mà còn cung cấp một trải nghiệm mượt mà và thuận tiện cho sinh viên.

* + 1. **Bối cảnh**

Tại các trường đại học hiện nay, việc quản lý đăng ký học phần thường gặp nhiều khó khăn, đặc biệt là khi số lượng sinh viên và lớp học lớn. Các phương thức quản lý truyền thống, như đăng ký bằng giấy tờ hoặc qua các hệ thống không tự động hóa, thường dẫn đến tình trạng quá tải, sai sót và thiếu minh bạch. Việc triển khai hệ thống quản lý lớp học tín chỉ mang lại nhiều lợi ích như:

* **Tự động hóa và tối ưu hóa quy trình:** Hệ thống giúp tự động hóa nhiều công đoạn phức tạp trong quá trình đăng ký học phần, từ kiểm tra điều kiện tiên quyết đến xác nhận đăng ký.
* **Giảm thiểu sai sót:** Với hệ thống tự động, các lỗi phát sinh do nhập liệu thủ công sẽ được giảm thiểu.
* **Tiết kiệm thời gian:** Sinh viên và nhân viên phòng đào tạo có thể tiết kiệm được nhiều thời gian trong quá trình đăng ký và quản lý học phần.
* **Cải thiện trải nghiệm người dùng:** Sinh viên có thể dễ dàng tra cứu thông tin, đăng ký học phần và theo dõi tiến độ học tập của mình một cách nhanh chóng và tiện lợi.
  + 1. **Đối tượng sử dụng**

Hệ thống quản lý lớp học tín chỉ được thiết kế để phục vụ cho các đối tượng sau:

* **Sinh viên:** Là người trực tiếp sử dụng hệ thống để đăng ký các học phần, tra cứu thông tin về môn học, kiểm tra tiến độ học tập và đăng ký xét tốt nghiệp.
* **Giảng viên:** Sử dụng hệ thống để xem danh sách sinh viên đăng ký vào các lớp học của mình, cập nhật điểm và theo dõi tiến độ học tập của sinh viên.
* **Nhân viên phòng đào tạo:** Quản lý và theo dõi toàn bộ quy trình đăng ký học phần, cập nhật thông tin về môn học, quản lý danh sách dự bị và xử lý các yêu cầu đăng ký đặc biệt.
* **Ban quản lý nhà trường:** Sử dụng hệ thống để giám sát và điều chỉnh các hoạt động đăng ký học phần, đưa ra quyết định về việc mở thêm lớp hoặc điều chỉnh quy định liên quan đến quá trình học tập của sinh viên.

## Mục tiêu đề tài

* **Mục tiêu chính:** Xây dựng hệ thống quản lý lớp học tín chỉ hiệu quả, giúp sinh viên dễ dàng đăng ký học phần và theo dõi tiến độ học tập.
* **Mục tiêu cụ thể:**
* Tự động hóa quá trình đăng ký và quản lý học phần.
* Đảm bảo tính chính xác và minh bạch trong quá trình đăng ký và quản lý học phần.
* Tăng cường khả năng theo dõi tiến độ học tập của sinh viên.
* Cung cấp thông tin chính xác và kịp thời cho sinh viên về lịch học, điểm số và trạng thái đăng ký.

## Phạm vi đề tài

* **Phạm vi chức năng:** Hệ thống sẽ hỗ trợ quản lý các chức năng chính như đăng ký học phần, kiểm tra môn học tiên quyết, xác nhận đăng ký, gửi thông báo qua email, theo dõi tiến độ học tập và xét tốt nghiệp.
* **Đối tượng sử dụng:** Sinh viên, giảng viên, nhân viên quản lý của trường đại học.
* **Phạm vi kỹ thuật:** Triển khai trên nền tảng web với cơ sở dữ liệu quản lý thông tin sinh viên, học phần và lịch sử học tập. Sử dụng các công nghệ như Java, Spring Boot, Apache Kafka, Docker-compose, Postman, RDS của AWS, MariaDB.

## Mô tả yêu cầu chức năng

* **Quản lý đăng ký học phần:**
* Hiển thị danh sách các môn học có thể đăng ký trong mỗi học kỳ.
* Cho phép sinh viên chọn và đăng ký các môn học.
* Kiểm tra các điều kiện tiên quyết trước khi cho phép đăng ký.
* Giới hạn số tín chỉ tối đa mà sinh viên có thể đăng ký mỗi học kỳ.
* Yêu cầu sinh viên xác nhận đăng ký trước khi hoàn tất.
* Gửi thông báo qua email sau khi đăng ký thành công.
* Quản lý danh sách dự bị cho các lớp đã đầy.
* **Quản lý kết quả học tập:**
* Hiển thị số tín chỉ đã đạt, số môn đã học, điểm số của các môn học và điểm trung bình tích lũy.
* Cung cấp các tiện ích hỗ trợ khác như theo dõi tiến độ học tập và các thông tin liên quan.
* **Quản lý lịch học:**
* Hiển thị thời khóa biểu theo tuần cho sinh viên.
* **Xét tốt nghiệp:**
* Cho phép sinh viên đăng ký xét tốt nghiệp khi đủ điều kiện.
* Kiểm tra và xác nhận các tiêu chuẩn xét tốt nghiệp.
* Cấp bằng tốt nghiệp cho sinh viên đủ điều kiện và cập nhật danh sách cựu sinh viên.
* Công khai thông tin bằng cấp trên trang web của nhà trường.
* Lưu giữ thông tin cựu sinh viên để theo dõi và thực hiện các hoạt động khảo sát, theo dõi quá trình làm việc

# : CƠ SỞ LÝ THUYẾT

## Các loại kiến trúc có thể áp dụng vào dự án?

### Kiến trúc Layer

* **Ưu điểm:**
* Dễ hiểu và triển khai: Kiến trúc Layer phân chia các thành phần của hệ thống thành các tầng (layers) có trách nhiệm cụ thể. Điều này giúp dễ dàng hiểu cấu trúc của hệ thống và triển khai các chức năng một cách có tổ chức.
* Dễ bảo trì: Mỗi tầng trong kiến trúc Layer có trách nhiệm riêng, giúp dễ dàng bảo trì và cải tiến từng phần của hệ thống mà không làm ảnh hưởng đến các phần khác.
* Performance tốt: Trong môi trường nhỏ và đơn giản, kiến trúc Layer thường có performance tốt vì không gây overhead do giao tiếp giữa các thành phần.
* Phù hợp với các dự án vừa và nhỏ
* Ít tốn chi phí
* **Nhược điểm:**
* Coupling cao: Kiến trúc Layer có thể dẫn đến việc các tầng phụ thuộc chặt chẽ vào nhau, khiến cho việc thay đổi một phần có thể ảnh hưởng đến toàn bộ hệ thống.
* Khó mở rộng: Khi hệ thống phát triển và cần mở rộng, việc thêm mới chức năng có thể gặp khó khăn do cấu trúc Layer đã được xây dựng sẵn.
* Giới hạn khả năng tái sử dụng: Do sự phụ thuộc giữa các tầng, việc tái sử dụng các thành phần trong kiến trúc Layer có thể gặp khó khăn.

### Kiến trúc Microservice

* **Ưu điểm:**
* Tính linh hoạt và mở rộng cao: Mỗi dịch vụ (service) trong kiến trúc Microservice có thể phát triển, triển khai và mở rộng độc lập. Điều này giúp hệ thống dễ dàng mở rộng khi cần thiết.
* Tách biệt trách nhiệm: Mỗi dịch vụ chỉ chịu trách nhiệm về một phần cụ thể của hệ thống, giúp giảm coupling và tăng khả năng tái sử dụng.
* Phát triển độc lập: Các nhóm phát triển có thể làm việc độc lập trên các dịch vụ khác nhau, không cần phải lo lắng về ảnh hưởng đến các phần khác của hệ thống.
* **Nhược điểm:**
* Overhead giao tiếp: Việc giao tiếp giữa các dịch vụ trong kiến trúc Microservice có thể gây ra overhead, đặc biệt khi số lượng dịch vụ lớn.
* Chi phí cao
* Trùng lắp code giữa các service.
* Quản lý phức tạp: Hệ thống microservices đòi hỏi các công cụ và phương pháp quản lý phức tạp hơn để theo dõi và giám sát các dịch vụ riêng lẻ, bao gồm việc quản lý versioning, deployment, logging và monitoring.

## Phân tích lựa chọn kiến trúc

Theo yêu cầu của đề tài thì em chọn kiến trúc Micorservice để áp dụng vì:

* **Khả năng chịu tải cao:**
* ***Số lượng người truy cập lớn:*** Đây là trang web dùng để đăng ký học phần cho sinh viên của một trường đại học, do đó, vào những ngày đăng ký học phần, số lượng người truy cập sẽ rất lớn. Kiến trúc Layer truyền thống có thể gặp khó khăn khi xử lý một lượng lớn yêu cầu đồng thời, dễ dẫn đến quá tải hệ thống và sập toàn bộ trang web.
* ***Microservices giải quyết vấn đề này:*** Kiến trúc microservices giúp khắc phục nhược điểm này bằng cách tách biệt các dịch vụ. Mỗi dịch vụ sẽ có cơ sở dữ liệu riêng và có thể được triển khai độc lập, do đó, nếu một dịch vụ gặp lỗi, các dịch vụ khác vẫn hoạt động bình thường, giúp hệ thống tổng thể duy trì được tính ổn định.
* **Dễ mở rộng:**
* ***Tương lai mở rộng:*** Vì là một trang web cho trường đại học, việc mở rộng các dịch vụ khác trong tương lai là điều chắc chắn. Ví dụ, có thể cần thêm các dịch vụ quản lý điểm, quản lý thời khóa biểu, hoặc các dịch vụ hỗ trợ sinh viên khác.
* ***Microservices hỗ trợ mở rộng dễ dàng:*** Kiến trúc microservices cho phép dễ dàng thêm mới và mở rộng các dịch vụ mà không cần phải thay đổi cấu trúc hiện tại. Mỗi dịch vụ mới có thể được phát triển và triển khai độc lập, không ảnh hưởng đến các dịch vụ hiện có.
* **Quản lý lỗi và bảo trì:**
* ***Quản lý lỗi cụ thể:*** Khi một dịch vụ gặp sự cố, có thể quản lý và xử lý lỗi một cách cụ thể cho dịch vụ đó mà không cần lo lắng về tác động đến các dịch vụ khác. Điều này giúp giảm thiểu thời gian gián đoạn và tăng tính ổn định của hệ thống.
* ***Bảo trì và nâng cấp dễ dàng:*** Kiến trúc microservices giúp dễ dàng bảo trì và nâng cấp từng dịch vụ mà không ảnh hưởng đến toàn bộ hệ thống. Điều này là một lợi thế lớn trong việc duy trì và cải tiến hệ thống liên tục.
* **Phát triển độc lập và đa dạng công nghệ:**
* ***Nhóm phát triển đa năng:*** Với kiến trúc microservices, các nhóm phát triển có thể làm việc độc lập trên các dịch vụ khác nhau. Điều này cho phép tận dụng thế mạnh về ngôn ngữ và công nghệ của từng thành viên trong nhóm.
* ***Sử dụng công nghệ phù hợp:*** Mỗi dịch vụ có thể được phát triển bằng ngôn ngữ và công nghệ phù hợp nhất với yêu cầu của nó, điều này giúp tối ưu hóa hiệu suất và hiệu quả phát triển.
* Tóm lại, kiến trúc microservices là lựa chọn phù hợp cho hệ thống đăng ký học phần của trường đại học vì khả năng chịu tải cao, dễ mở rộng, quản lý lỗi tốt và hỗ trợ phát triển độc lập với nhiều công nghệ khác nhau. Điều này đảm bảo hệ thống hoạt động ổn định, dễ dàng nâng cấp và mở rộng trong tương lai.

## Các công nghệ sử dụng

### Java

1. **Java là gì?**

Java là một ngôn ngữ lập trình hướng đối tượng, đa nền tảng, và mạnh mẽ, được phát triển bởi Sun Microsystems (nay thuộc Oracle Corporation). Java được thiết kế để có thể chạy trên bất kỳ nền tảng nào có máy ảo Java (JVM).

1. **Lợi ích khi sử dụng Java?**

**Độc lập nền tảng:** "Viết một lần, chạy mọi nơi" nhờ JVM.

**Bảo mật cao:** Có cơ chế bảo mật mạnh mẽ để bảo vệ dữ liệu.

**Hiệu suất cao:** Tối ưu hóa hiệu suất qua nhiều cải tiến.

**Thư viện phong phú:** Có một hệ sinh thái thư viện và framework rộng lớn.

**Hỗ trợ mạnh mẽ:** Cộng đồng lập trình viên đông đảo và nhiều tài liệu hỗ trợ.

### JWT (JSON Web Token)

1. **JWT là gì?**

JWT là một tiêu chuẩn mở (RFC 7519) định dạng token trong JSON để truyền thông tin an toàn giữa các bên như là một đối tượng JSON. Thông tin có thể được xác thực và ký số bằng cách sử dụng các khóa bí mật hoặc cặp khóa công khai/tư nhân.

1. **Lợi ích khi sử dụng JWT?**

**Bảo mật:** Đảm bảo tính toàn vẹn và bảo mật của dữ liệu thông qua chữ ký số.

**Gọn nhẹ:** Token ngắn gọn, dễ truyền qua URL, HTTP Header.

**Dễ dàng phân phối:** JWT có thể được sử dụng trên nhiều nền tảng và ngôn ngữ lập trình.

**Xác thực người dùng:** JWT giúp xác thực người dùng và đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập tài nguyên.

**Truy cập tài nguyên:** JWT giúp ủy quyền truy cập tài nguyên, cho phép người dùng truy cập các tài nguyên mà họ được phép.

**Không trạng thái:** Server không cần lưu trữ thông tin đăng nhập hoặc trạng thái phiên làm việc, giảm tải cho server.

### Spring boot

1. **Spring Boot là gì?**

Spring Boot là một framework của Java dựa trên Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển ứng dụng Java, giúp tạo ra các ứng dụng độc lập, dễ triển khai mà không cần cấu hình nhiều.

1. **Lợi ích khi sử dụng Spring Boot?**

**Tạo ứng dụng nhanh:** Tích hợp nhiều cấu hình mặc định, giảm thiểu thời gian thiết lập.

**Độc lập:** Ứng dụng có thể chạy độc lập mà không cần container ứng dụng phức tạp.

Thư viện phong phú: Hỗ trợ nhiều thư viện và công nghệ khác nhau trong hệ sinh thái Spring.

**Dễ dàng triển khai:** Tạo ra các ứng dụng dễ triển khai trên các môi trường khác nhau.

### Postman

1. **Postman là gì?**

Postman là một công cụ API Client giúp phát triển, kiểm thử và tài liệu hóa các API. Nó cung cấp một giao diện thân thiện để gửi yêu cầu HTTP và nhận phản hồi.

1. **Lợi ích khi sử dụng Postman?**

**Dễ sử dụng:** Giao diện trực quan, thân thiện.

**Kiểm thử nhanh chóng:** Gửi yêu cầu và kiểm thử phản hồi một cách nhanh chóng.

**Tài liệu hóa API:** Hỗ trợ tài liệu hóa API tự động.

**Tích hợp CI/CD:** Dễ dàng tích hợp vào quy trình CI/CD để tự động kiểm thử API.

### Apache Kafka

1. **Apache Kafka là gì?**

Apache Kafka là một nền tảng xử lý dòng dữ liệu phân tán, được thiết kế để xử lý các dòng dữ liệu lớn theo thời gian thực. Nó được sử dụng để xây dựng các pipeline dữ liệu và ứng dụng streaming.

1. **Lợi ích khi sử dụng Apache Kafka?**

**Thông lượng cao:** Có thể xử lý hàng triệu sự kiện mỗi giây.

**Khả năng mở rộng:** Dễ dàng mở rộng bằng cách thêm thêm nhiều broker.

**Tính bền vững:** Dữ liệu được lưu trữ phân tán và bền vững.

**Khả năng tích hợp:** Tích hợp tốt với nhiều hệ thống và nền tảng khác.

### Docker

1. **Docker là gì?**

Docker là một nền tảng để phát triển, triển khai và chạy các ứng dụng bên trong các container. Container cung cấp một cách để đóng gói ứng dụng và các phụ thuộc của nó, giúp ứng dụng có thể chạy đồng nhất trên mọi môi trường.

1. **Lợi ích khi sử dụng Docker?**

**Nhất quán môi trường:** Đảm bảo ứng dụng chạy đồng nhất trên mọi môi trường.

**Hiệu quả tài nguyên:** Container sử dụng ít tài nguyên hơn so với máy ảo.

**Triển khai nhanh chóng:** Dễ dàng triển khai và mở rộng ứng dụng.

**Khả năng tích hợp:** Tích hợp dễ dàng với nhiều công cụ CI/CD và hệ sinh thái DevOps.

### RDS của AWS (Amazon Relational Database Service)

1. **RDS là gì?**

RDS là một dịch vụ quản lý cơ sở dữ liệu quan hệ của Amazon Web Services (AWS), cho phép người dùng thiết lập, vận hành và mở rộng cơ sở dữ liệu quan hệ trên đám mây.

1. **Lợi ích khi sử dụng RDS?**

**Quản lý tự động:** AWS quản lý các tác vụ bảo trì, sao lưu, và cập nhật phần mềm.

**Khả năng mở rộng:** Dễ dàng mở rộng quy mô cơ sở dữ liệu theo nhu cầu.

**Bảo mật:** Cung cấp các tính năng bảo mật như mã hóa dữ liệu, kiểm soát truy cập và kết nối an toàn.

**Độ sẵn sàng cao:** Tích hợp các tùy chọn sao lưu và phục hồi dữ liệu, đảm bảo độ sẵn sàng cao.

### MariaDB

1. **MariaDB là gì?**

MariaDB là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, phát triển từ mã nguồn của MySQL, cung cấp một sự thay thế an toàn, nhanh chóng và mạnh mẽ.

1. **Lợi ích khi sử dụng MariaDB?**

**Hiệu suất cao:** Tối ưu hóa cho hiệu suất và khả năng mở rộng.

**Mã nguồn mở:** Được phát triển và duy trì bởi cộng đồng mã nguồn mở.

**Tương thích MySQL:** Dễ dàng thay thế và tương thích với MySQL.

**Tính năng phong phú:** Cung cấp nhiều tính năng nâng cao như engine lưu trữ đa dạng, khả năng xử lý song song và bảo mật.

### Github

1. **GitHub là gì?**

GitHub là một nền tảng lưu trữ mã nguồn dựa trên Git, cung cấp các công cụ để phát triển phần mềm, quản lý dự án và cộng tác giữa các nhà phát triển.

1. **Lợi ích khi sử dụng GitHub?**

**Quản lý mã nguồn:** Lưu trữ và quản lý mã nguồn dự án hiệu quả.

**Cộng tác dễ dàng:** Hỗ trợ các tính năng cộng tác như pull request, code review và issue tracking.

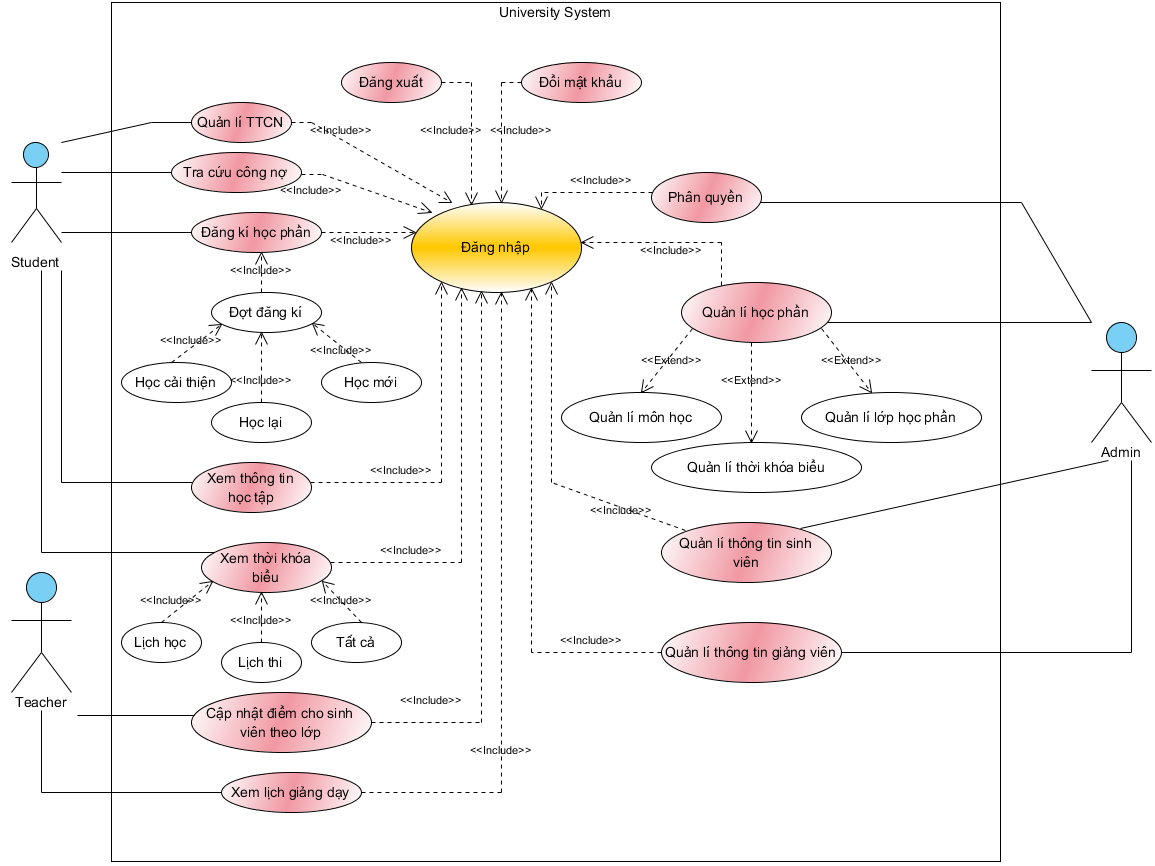
**CI/CD tích hợp:** Hỗ trợ tích hợp liên tục và triển khai liên tục (CI/CD).

**Cộng đồng lớn:** Cộng đồng nhà phát triển rộng lớn và nhiều tài nguyên hỗ trợ.

# : 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 |
| Student | Mô tả: Sinh viên là người sử dụng hệ thống để đăng ký các học phần, xem thông tin về các môn học, kiểm tra lịch học, quản lí thông tin cá nhân và tra cứu công nợ. |
| Teacher | Mô tả: Giáo viên là người sử dụng hệ thống để quản lí các lớp học phần mà mình giảng dạy (nhập điểm), và xem lịch giảng dạy của mình. |
| Admin | Mô tả: Quản trị viên là người quản lí hệ thống, có quyền truy cập và quản lí các thông tin cơ bản như tạo, sửa, xóa và cấp quyền người dùng, quản lí danh sách các môn học và lớp học phần, quản lí thời khóa biểu, xử lý các vấn đề kỹ thuật và bảo mật. |

Bảng 3‑1 Danh sách tác nhân và mô tả

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

|  |  |
| --- | --- |
| ID | Tên Use case |
| UC01 |  |
| UC02 |  |
| UC03 |  |
| UC04 |  |
| UC05 |  |
| UC06 |  |
|  |  |
|  |  |
| … |  |

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

### Tình huống hoạt động

Đặc tả use case.

Activity diagram.

Sequence diagram cho từng tình huống.

## Class diagram

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

## Deployment diagram

A diagram of a computer

Description automatically generated

# : HIỆN THỰC

## Cấu hình phần cứng, phần mềm

Ghi rõ thông tin sử dụng các dịch vụ nào của AWS để thực hiện đồ án.

Cấu hình phần cứng, phần mềm của Server (nếu có).

Cấu hình phần cứng, phần mềm của Client trong hệ thống (nếu có).

## Giao diện của hệ thống

Màn hình chức năng đã thực hiện.

## Kế hoạch và hiện thực kiểm thử hệ thống

# : KẾT LUẬN

## Kết quả đạt được

**5.1.1 Hoàn thành chức năng cơ bản của sinh viên:**

- Đăng nhập

- Xem thông tin cá nhân

- Đăng kí học phần

- Xem lịch học

- Xem kết quả học tập

- Giao diện người dùng thân thiện, dễ sử dụng.

**5.1.2 Đảm bảo tính bảo mật:**

- Mã hóa mật khẩu

- Gửi thông báo về mail mỗi khi đăng kí học phần.

**5.1.3 Tối ưu hóa hiệu suất:**

- Tải nhanh, mượt mà.

- Hỗ trợ nhiều người dùng truy cập.

## Hạn chế của đồ án

* Chưa có dịch vụ tra cứu công nợ.
* Chưa có thanh toán trực tuyến sau khi đăng kí học phần.
* Thiếu phần mở rộng cho giảng viên, phụ huynh cũng như admin của trường.

## Hướng phát triển

* Sau khi đăng kí học phần thành công sinh viên có thể kiểm tra công nợ và thanh toán trực tuyến thông qua ngân hàng hoặc các dịch vụ khác (momo, zalopay...).
* Ngoài tài khoản dành riêng cho sinh viên thì còn có tài khoản của phụ huynh, giảng viên và admin của trường.
* Phụ huynh có thể đăng nhập vào để kiểm tra kết quả học tập của con mình, cũng như kiểm tra công nợ và thanh toán tiền học phí.
* Giảng viên thì có thể xem lịch giảng dạy theo lớp được khoa phân công và nhập điểm cho sinh viên của lớp mà mình đảm nhận.
* Admin có thể quản lí các môn học để đăng kí học phần (thêm, sửa, xóa, xem), quản lí thời khóa biểu cho từng môn học (khi sinh viên đăng kí học phần sẽ có phần chọn lịch học), đồng thời có thể cập nhật thời khóa biểu mới cho sinh viên và giảng viên khi có các sự thay đổi (lễ, hội nghị, sự kiện, giảng viên báo bận). Quản lí thông tin sinh viên, giảng viên, phụ huynh.

# TÀI LIỆU THAM KHẢO

Các tài liệu Tiếng Anh

1. Kevin Faaborg & Sandro Pasquali - Mastering Node.js Second Edition – December 2017
2. David Herron, Diogo Resende & Valentin Bojinov - Node.js Complete Reference Guide – December 2018

Các tài liệu từ Internet

1. <https://nodejs.org/en/docs/>
2. <https://expressjs.com/en/guide/routing.html>