**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**TÊN ĐỀ TÀI:**

**HỆ THỐNG QUẢN LÝ**

**LỚP HỌC TÍN CHỈ**

***Nhóm 12 Sinh viên thực hiện***

|  |  |  |
| --- | --- | --- |
| **Họ Tên** | **MSSV** | **Ghi chú** |
| **Lê Quang Lộc** | **20084661** |  |
| **Hà Đức Anh** | **20087301** |  |

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

|  |
| --- |
| *Ngày ....... tháng ........ năm .........*  **GIẢNG VIÊN HƯỚNG DẪN**  *(Ký và ghi rõ họ tên)* |

# KHẢO SÁT HIỆN TRẠNG

## Đặt vấn đề

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. Nhà trường cũng sẽ cung cấp tài khoản đăng nhập cho sinh viên, giáo viên, nhân viên mới của nhà trường. Nhân viên nhà trường sẽ có nhiệm vụ quản lý thông tin của sinh viên, giáo viên, lớp học phần,... và một số chức năng khác.

Mỗi năm khoa sẽ tạo trương trình khung mới cho sinh viên khóa mới của từng ngành học. Trương trình khung bao gồm các môn học cũng như một số điều kiện cần thiết để sinh viên có thể tốt nghiệp.

Trong mỗi học kỳ, khoa sẽ tạo lớp học phần dựa trên các môn học theo nhu cầu của sinh viên. Số lượng sinh viên mỗi lớp học phần tối đa là 40 sinh viên.

Môn học có thể là môn học tự chọn hoặc môn học bắt buộc. Mỗi môn học có một hoặc nhiều môn học tiên quyết. Sinh viên chỉ có thể đăng ký môn học nếu đã học qua hết (không cần đậu ) các môn tiên quyết. Sinh viên không thể đăng ký nếu trùng thời khóa biểu với các lớp đã đăng ký trước đó.

Đối với môn tự chọn, chúng sẽ được chia thành các nhóm học phần tự chọn. Sinh viên sẽ đăng ký môn học sao cho đủ số tín chỉ yêu cầu của từng nhóm học phần. Sinh viên có thể đăng ký nhiều hơn số môn trong học phần tự chọn nhưng chỉ tính tối đa theo số tín chỉ yêu cầu của học phần tự chọn của kỳ đó. Nếu sinh viên rớt môn học tự chọn thì vẫn có thể chọn môn học khác trong cùng học phần tự chọn với số tín chỉ tương đương mà không cần học lại môn đã rớt.

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.

Trong thời gian 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. Sinh viên vẫn có thể xóa đăng ký môn học nếu vẫn còn trong thời gian đăng ký.

Sau khi hết thời gian đăng ký, các lớp học phần đủ điều kiện mở lớp sẽ được khóa lại và sinh viên sẽ có thể đóng học phí. Trường hợp lớp học không đủ số lượng sinh viên tối thiểu thì lớp học phần sẽ bị hủy.

Nhân viên nhà trường có thể thêm, xóa (không thể xóa nếu sinh viên đã học phí) sinh viên khỏi lớp học phần ngay cả khi lớp học phần đã khóa chức năng đăng ký của sinh viên.

Sau khi hết thời gian đóng học phí, sinh viên chưa đóng học phí sẽ bị xóa khỏi lớp học. Sinh viên có thể xin hoãn đóng học phí. Lúc này trạng thái của sinh viên sẽ là “hoãn học phí”. Đến ngày thi giữa kỳ nếu sinh viên vẫn chưa đóng đủ học phí sẽ không có tên trong danh sách thi. Đến ngày thi cuối kỳ sinh viên không có điểm thi giữa kỳ cũng sẽ không có tên trong danh sách thi. Nhân viên nhà trường hoặc giáo viện của lớp đó sẽ có thểin danh sách thi.

Trong quá trình học thì nhân viên của nhà trường cũng có thể thay đổi lịch học từng tuần như thêm, xóa ngày học, thay đổi phòng học, thêm ngày nghỉ (tất cả lớp học sẽ được nghỉ vào ngày này),... Lưu ý là vẫn phải đảm bảo đủ số tiết học yêu cầu của lớp học phần đó để có thể kết thúc môn.

Nhà trường cũng yêu cầu hệ thống phải có chức năng nhập điểm cho từng lớp học phần. Giáo viên chỉ có thể nhập điểm của lớp học phần mình đang dạy. Mọi hoạt động liên quan đến bảng điểm phải được ghi lại.

Hệ thống cũng phải có chức năng cho phép sinh viên, giáo viên xem thông tin cá nhân, lớp học phần mà mình đang học hoặc giảng dạy, xem lịch học, lịch thi,...

Sau khi nhập điểm đầy đủ thì nhà trường có thể kết thúc lớp học phần đó. Sau khi kết thúc lớp học phần thì không cho phép chỉnh sửa thông tin bảng điểm. Đồng thời, sinh viên sẽ phải làm khảo sát để đánh giá chất lượng dạy học của giáo viên của những lớp học phần mà mình đã học kỳ đó.

Hệ thống có khả năng cho phép sinh viên xem tiến độ 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, lịch thi thay đổi thông tin cá nhân và các tiện ích khác.

Hệ thống cũng cần có một số chức năng sau: cho phép phụ huynh theo dõi quá trình học tập của sinh viên, chức năng đóng học phí online, chức năng tính điểm rèn luyện.

Sau khi số tín chỉ đã đạt theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp. Nếu mọi yêu cầu 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.

Ngoài ra hệ thống phải đảm bảo các yêu cầu phi chức năng như:

- Bảo mật dữ liệu và hệ thống, xác thực người dùng, quản lý quyền truy cập và bảo vệ hệ thống khỏi các mối đe dọa bên ngoài là những yêu cầu không thể thiếu

- Đáp ứng các tiêu chuẩn về hiệu suất và thời gian phản hồi nhằm đảm bảo trải nghiệm người dùng.

- Có khả năng hoạt động liên tục mà không bị gián đoạn quá nhiều

- Dễ bảo trì, cập nhật và nâng cấp để duy trì tính đồng nhất và khả năng tương tích với các phiên bản mới.

- Phải có đầy đủ tài liệu hướng dẫn sử dụng, hỗ trợ kỹ thuật và đào tạo người dùng

- Hệ thống phải được thiết kế để tiết kiệm năng lượng, giảm thiểu dữ liệu không cần thiết và tối ưu hòa tài nguyên nhằm bảo vệ môi trường cũng như tiết kiệm chi phí

- Giao diện phải dễ sử dụng và thân thiện với người dùng

- Tài liệu hỗ trợ phải đầy đủ

- Để đảm bảo độ bảo mật của hệ thống nhà trường có dự định sẽ trang bị 1 server riêng để lưu trữ dữ liệu.

## Mục tiêu và chức năng của hệ thống

Mục tiêu là tạo ra một hệ thống quản lý lớp học tín chỉ giúp cho nhà trường có thể quản lý, số hóa toàn bộ thông tin về quá trình học tập của sinh viên, cũng như cung cấp 1 kênh thông tin cho sinh viên và giáo viên trong suốt quá trình học tập tại trường

## Mô tả yêu cầu chức năng

### Đối với yêu cầu chức năng:

* Quản lý sinh viên: Đăng ký, chỉnh sửa thông tin, danh sách lớp học.
* Quản lý giảng viên: Thêm, sửa, xóa thông tin giảng viên, phân công giảng dạy.
* Quản lý môn học: Thêm mới, chỉnh sửa thông tin môn học, phân lớp.
* Quản lý lớp học: Đăng ký lớp học, theo dõi tiến độ, điểm danh, quản lý điểm số.
* Thông báo: Gửi thông báo, email nhắc nhở. Báo cáo: Thống kê, xuất báo cáo.

### **Đối với yêu cầu phi chức năng:**

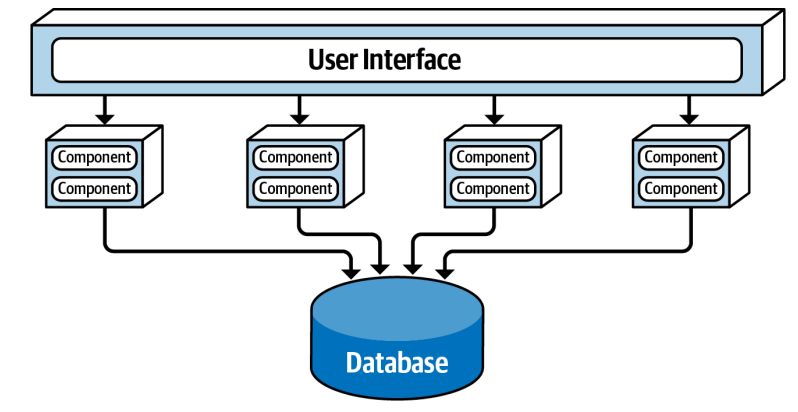
Yêu cầu phi chức năng là các yêu cầu mô tả các đặc tính và hành vi không trực tiếp liên quan đến chức năng cụ thể của hệ thống, nhưng rất quan trọng để đảm bảo hệ thống hoạt động với hiệu suất tốt, có khả năng mở rộng, luôn luôn sẵn sàng, có khả năng bảo mật, bào trì, tương thích với nhiều hệ điều hành, trình duyệt web, có tính linh hoạt cao, có khả năng phục hồi dữ liệu khi hệ thống gặp sự cố, phải có tài liệu hướng dẫn sử dụng chi tiết.

## Định hướng giải quyết vấn đề

Sau khi phân tích và thảo luận tất cả các yêu cầu của hệ thống, nhà phát triển đã quyết định xây dựng hệ thống dựa trên kiểu kiến trúc service-based.

### **Giới thiệu về kiến trúc service-based**

Kiến trúc dựa trên dịch vụ là một sự kết hợp giữa kiểu kiến trúc microservices và được coi là một trong những kiến trúc phổ biến nhất và thực tế nhất, chủ yếu nhờ vào tính linh hoạt trong kiến trúc của nó. Mặc dù kiến trúc dựa trên dịch vụ là một kiến trúc phân tán (distributed architecture), nhưng nó không có cùng mức độ phức tạp và chi phí như các kiến trúc phân tán khác, như kiến trúc microservices hoặc kiến trúc hướng sự kiện (event-driven architecture), khiến nó trở thành lựa chọn phổ biến cho nhiều ứng dụng liên quan đến doanh nghiệp.



Hình 1.4.1 Mô hình cơ bản của kiến trúc service-based

### Lý do lựa chọn kiến trúc service-based**:**

- Hệ thống cần ít nhất 3 giao diện dành cho: sinh viên, giáo viên và nhân viên nhà trường

- Hệ thống cần có 1 cơ sở dữ liệu nguyên khối

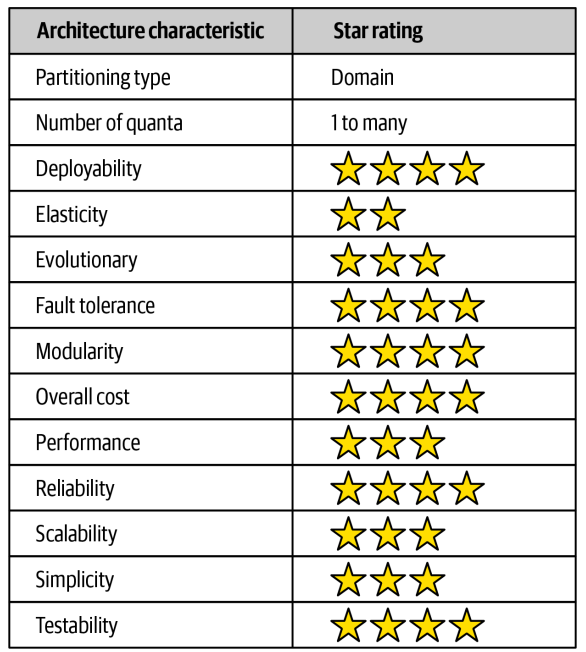
- Hệ thống cần có khả năng mở rộng về sau nhưng số lượng chỉ nằm trong khoảng từ

4-12 dịch vụ

- Hệ thống cần có khả năng hoạt động ngay cả khi 1 dịch vụ không hoạt động

- Hệ thống cần có chi phí vừa phải

### Ưu điểm và nhược điểm của kiến trúc service-based



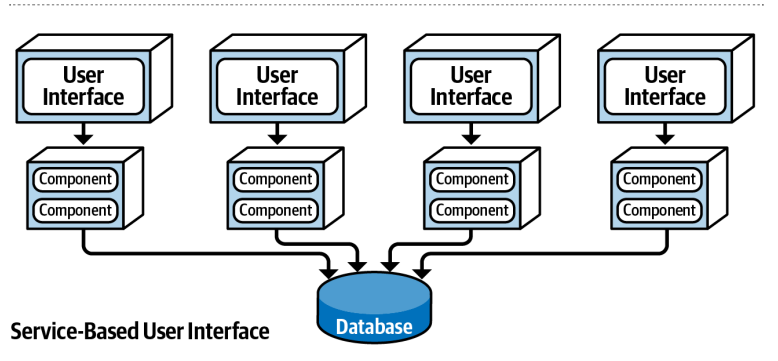
Hình 1.3.3 Đánh giá các đặc trưng của kiến trúc dựa trên dịch vụ

# : CƠ SỞ LÝ THUYẾT

## Thiết kế của hệ thống

### **Các yêu cầu đặt ra cho hệ thống:**

Hệ thống sẻ được thiết kế dựa trên mô hình kiến trúc Service-Based User Interface với 3 User Interface là 3 trang web riêng biệt dành cho sinh viên, giảng viên và nhân viên nhà trường. Hệ thống sẽ sử dụng một server riêng để đảm bảo tính bảo mật của hệ thống.



2.1 Mô hình kiến trúc Service-Based User Interface

### **Đối với yêu cầu phi chức năng:**

**Hiệu suất (Performance):**

* **Thời gian phản hồi của dịch vụ:** Mỗi dịch vụ phải có thời gian phản hồi dưới 1000 mili giây trong điều kiện tải thông thường.
* **Thông lượng dịch vụ:** Mỗi dịch vụ phải có khả năng xử lý 2000 yêu cầu mỗi giây.
* **Thời gian xử lý giao dịch:** Tổng thời gian xử lý một giao dịch qua nhiều dịch vụ không vượt quá 5 giây.

**Khả năng mở rộng (Scalability):**

* **Mở rộng theo chiều ngang:** Các dịch vụ phải có khả năng mở rộng theo chiều ngang bằng cách thêm các instance mới để đáp ứng tăng trưởng tải.
* **Tự động cân bằng tải:** Hệ thống phải tự động cân bằng tải giữa các instance của dịch vụ để tối ưu hóa hiệu suất.
* Hệ thống phải hỗ trợ tối thiểu 1000 người dùng đồng thời mà không ảnh hưởng đến hiệu suất.
* Hệ thống phải có khả năng mở rộng để đáp ứng số lượng người dùng tăng trưởng 10% hàng năm.

**Tính sẵn sàng (Availability):**

* **Sẵn sàng cao:** Mỗi dịch vụ phải được thiết kế để đạt tỷ lệ sẵn sàng 99.9%, với cơ chế tự động phát hiện và khôi phục từ lỗi.
* **Khả năng chịu lỗi:** Hệ thống phải có khả năng chuyển hướng yêu cầu đến các instance dịch vụ khác trong trường hợp một instance gặp sự cố.
* Hệ thống phải có tỷ lệ hoạt động ít nhất 99.9% trong khoảng thời gian 1 giây.
* Thời gian gián đoạn không quá 60 phút mỗi tháng.

**Khả năng bảo trì (Maintainability):**

* Mã nguồn phải được viết rõ ràng và dễ hiểu để các nhà phát triển có thể dễ dàng bảo trì và cập nhật.
* **Triển khai và cập nhật dễ dàng:** Mỗi dịch vụ phải có khả năng triển khai và cập nhật độc lập mà không ảnh hưởng đến các dịch vụ khác.
* **Theo dõi và ghi log:** Mỗi dịch vụ phải cung cấp cơ chế ghi log và theo dõi chi tiết để hỗ trợ bảo trì và gỡ lỗi.

**Bảo mật (Security):**

* **Xác thực và phân quyền:** Mỗi dịch vụ phải yêu cầu xác thực và phân quyền để đảm bảo chỉ người dùng hợp lệ mới có thể truy cập.
* **Mã hóa:** Dữ liệu truyền giữa các dịch vụ phải được mã hóa để bảo vệ tính toàn vẹn và bảo mật.

**Khả năng sử dụng (Usability):**

* **Giao diện API rõ ràng:** Các dịch vụ phải cung cấp API rõ ràng, nhất quán và dễ sử dụng cho các nhà phát triển.
* **Tài liệu API đầy đủ:** Cung cấp tài liệu chi tiết cho API để hỗ trợ các nhà phát triển tích hợp và sử dụng các dịch vụ.
* Giao diện người dùng phải thân thiện và dễ sử dụng đối với người dùng mới.
* Hệ thống phải hỗ trợ ít nhất 2 ngôn ngữ (việt – anh) để phục vụ người dùng quốc tế.

**Khả năng tương thích (Compatibility):**

* Tích hợp dịch vụ: Các dịch vụ phải hỗ trợ các tiêu chuẩn giao tiếp phổ biến (REST, gRPC, GraphQL) để dễ dàng tích hợp với các dịch vụ khác.
* Hệ thống phải tương thích với các trình duyệt phổ biến như Chrome, Cốc cốc và Edge.
* Độc lập nền tảng: Các dịch vụ phải hoạt động trên các nền tảng và môi trường khác nhau (Windows, Linux, container).

**Tính linh hoạt (Flexibility):**

* **Tùy chỉnh dễ dàng:** Mỗi dịch vụ phải dễ dàng tùy chỉnh để đáp ứng các yêu cầu cụ thể của người dùng.
* Hỗ trợ các trường hợp sử dụng khác nhau: Các dịch vụ phải hỗ trợ nhiều trường hợp sử dụng khác nhau thông qua cấu hình linh hoạt.
* Hệ thống phải cho phép tùy chỉnh giao diện người dùng và các chức năng để phù hợp với nhu cầu khác nhau của người dùng.
* Hệ thống phải hỗ trợ tích hợp với các ứng dụng bên thứ ba qua API.

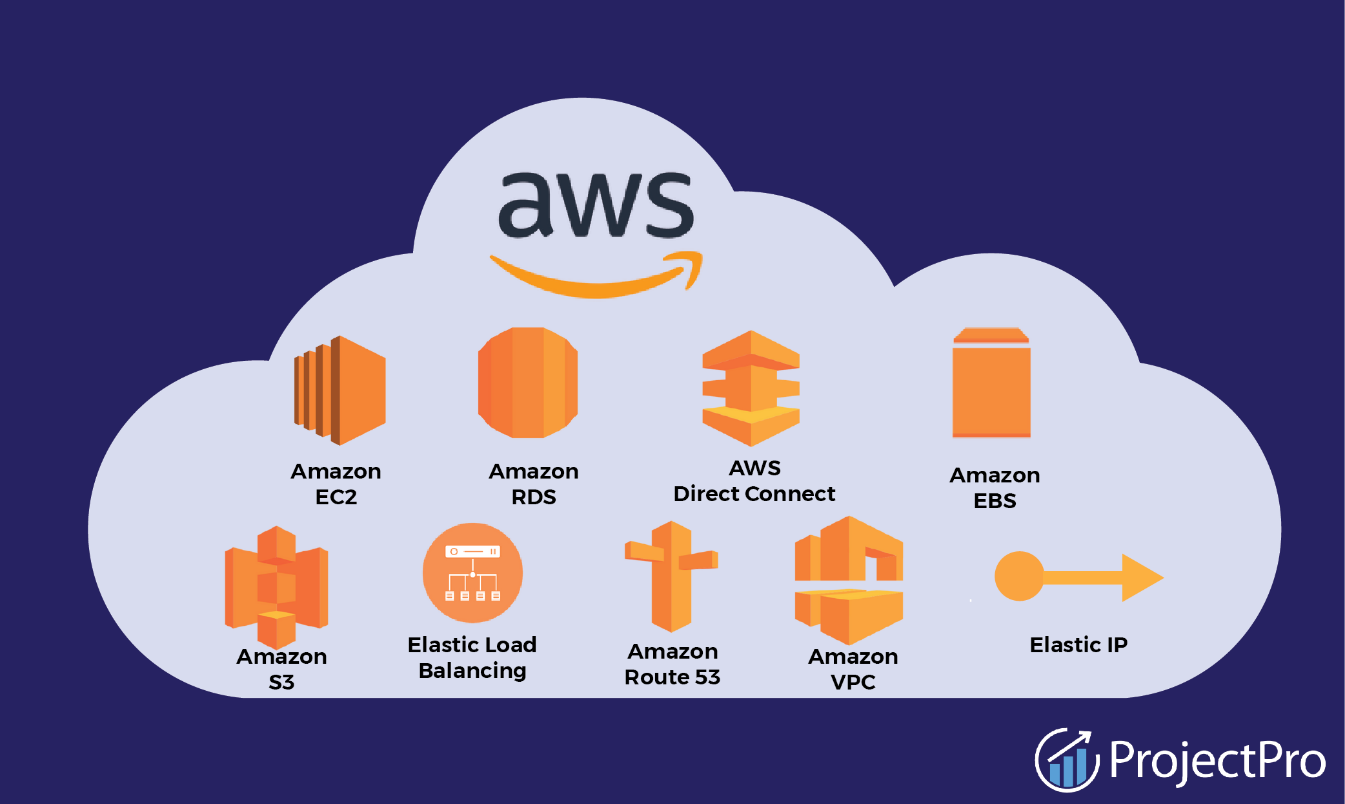
**Khả năng phục hồi (Recoverability):**

* **Tự động khôi phục:** Mỗi dịch vụ phải có khả năng tự động khôi phục sau sự cố và duy trì tính liên tục của dịch vụ.
* **Sao lưu và khôi phục dữ liệu:** Cơ chế sao lưu và khôi phục dữ liệu phải được thực hiện định kỳ để bảo vệ dữ liệu quan trọng.

**Khả năng quản lý (Manageability):**

* **Giám sát và cảnh báo:** Các dịch vụ phải được giám sát liên tục với hệ thống cảnh báo để nhanh chóng phản ứng với các vấn đề.
* **Quản lý cấu hình:** Các dịch vụ phải có cơ chế quản lý cấu hình tập trung để dễ dàng quản lý và thay đổi cấu hình.
* Hệ thống phải có báo cáo chi tiết về sử dụng và hoạt động để hỗ trợ trong việc phân tích và tối ưu hóa.

## Cloud Computing với AWS (IAM users và S3)



Hình 2- 1 Amazon web service

1. Tổng quan

Amazon Web Services (AWS) là giải pháp đám mây toàn diện và được sử dụng rộng rãi nhất, cung cấp trên 200 dịch vụ đầy đủ tính năng từ các trung tâm dữ liệu trên toàn thế giới. Hàng triệu khách hàng – bao gồm các công ty khởi nghiệp tăng trưởng nhanh nhất, các tập đoàn lớn nhất cũng như các cơ quan hàng đầu của chính phủ – đều tin tưởng vào AWS để giảm chi phí, trở nên linh hoạt hơn và đổi mới nhanh hơn.

1. Dịch vụ đã sử dụng

* **S3**: là dịch vụ lưu trữ đối tượng được xây dựng để lưu trữ và truy xuất bất kỳ lượng dữ liệu nào từ bất cứ nơi nào. Đây là dịch vụ lưu trữ đơn giản có độ bền, độ sẵn có, hiệu suất, tính bảo mật dẫn đầu ngành và khả năng thay đổi quy mô gần như không giới hạn với chi phí cực kỳ thấp.
* **IAM users:** là một chủ thể được người dùng tạo trong aws và được sử dụng để đại diện cho một người hoặc một service tương tác với AWS. IAM user có thể được xác thực bằng name và password hoặc sử dụng với aws api, aws cli bằng các credentials được tạo. IAM groups là một đại diện cho tập hợp các user.

## Node.js



Hình 2- 2 Node.js

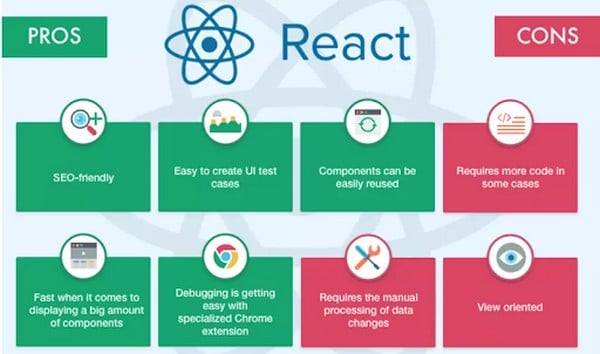
1. Tổng quan:

* **Node.js** : là một platform (nền tảng) độc lập được xây dựng trên môi trường Javascript Runtime - trình thông dịch ngôn ngữ Javascript cực nhanh trên trình duyệt Chrome. Dựa trên Node.js, chúng ta có thể xây dựng các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.Các ứng dụng được tạo ra bởi Node.js có tốc độ xử lý nhanh, realtime thời gian thực.Node.js sẽ thích hợp cho các sản phẩm có lưu lượng truy cập lớn, cần sự thay đổi về công nghệ và cần được mở rộng hoặc tạo ra các dự án Startup một cách nhanh nhất có thể.

1. Ưu điểm:

* Các ứng dụng Nodejs được viết bằng javascript, ngôn ngữ này là một ngôn ngữ khá thông dụng. Theo tác giả của ngôn ngữ Javascript, Ryan Dahl: “Javascript có những đặc tính mà làm cho nó rất khác biệt so với các ngôn ngữ lập trình động còn lại, cụ thể là nó không có khái niệm về đa luồng, tất cả là đơn luồng và hướng sự kiện.”
* Nodejs chạy đa nền tảng phía Server, sử dụng kiến trúc hướng sự kiện Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.
* Có thể chạy ứng dụng Nodejs ở bất kỳ đâu trên máy Mac – Window – Linux, hơn nữa cộng đồng Nodejs rất lớn và hoàn toàn miễn phí. Các bạn có thể thấy cộng đồng Nodejs lớn như thế nào tại đây, các package đều hoàn toàn free.
* Các ứng dụng NodeJS đáp ứng tốt thời gian thực và chạy đa nền tảng, đa thiết bị.

## React.js



Hình 2- 3 React.js

1. Tổng quan:

* ReactJS được hiểu là một thư viện mã nguồn mở chứa nhiều JavaScript và người tạo ra ReactJS chính là ông trùm với cái tên quen thuộc Facebook, được ra đời vào năm 2013. Mục đích của việc tạo ReactJS là tạo ra các ứng dụng web nhanh, hiệu quả và hấp dẫn với nỗ lực viết mã tối thiểu. Mục tiêu chính của ReactJS là bất kỳ trang web nào sử dụng ReactJS phải mượt mà, nhanh chóng, khả năng mở rộng cao và dễ thực hiện.Rất nhiều người sử dụng React làm V trong MVC

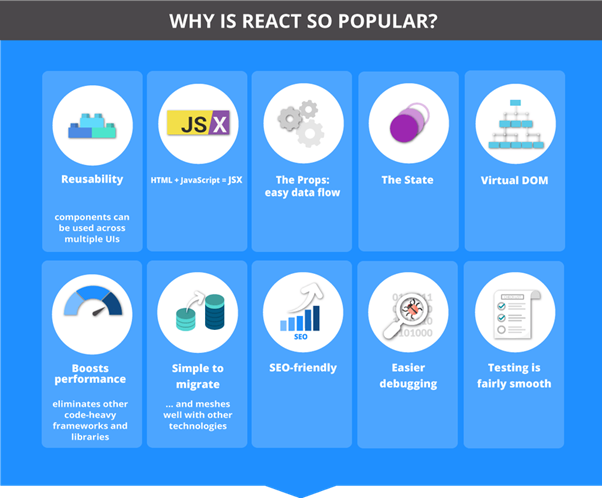
1. Ưu điểm:

* Tạo cho chính bản thân ReactJS một dom ảo, đây là nơi các component được tồn tại trên đó. Tạo một dom như thế này sẽ cải thiện hiệu suất rất nhiều.
* Có thể được sử dụng trên máy khách và máy chủ cũng như với các khuôn

khổ khác.

* Các mẫu thành phần và dữ liệu cải thiện khả năng đọc, giúp duy trì các ứng dụng lớn hơn.

## React native



Hình 2- 4 React Native

1. Tổng quan:

* React Native là một framework mã nguồn mở được sáng tạo bởi Facebook. Nó được sử dụng để phát triển ứng dụng di động Android, iOS, Web và UWP bằng cách cho phép các nhà phát triển sử dụng React cùng với môi trường ứng dụng gốc.

1. Ưu điểm:

* React Native có thể viết được cả App Android và IOS, điều này sẽ giúp:
* Tiết kiệm thời gian và giảm chi phí phát triển của một ứng dụng.
* Tận dụng nguồn nhân lực tốt hơn.
* Duy trì ít code hơn, ít bugs hơn.
* Các tính năng trong cả 2 platforms cũng tương tự nhau.
* React Native cho phép các developer có thể tái sử dụng code trong khi phát triển các ứng dụng đa nền tảng.
* React Native được đánh giá là một trong những Framework được yêu thích nhất, bạn có thể dễ dàng fix các lỗi thông qua các diễn đàn lớn nhỏ. Đồng thời có rất nhiều thư viện hỗ trợ.
* Tính ổn định và tối ưu: Được phát triển bởi Facebook, React Native có hiệu năng ổn định khá cao.
* Mã React Native giúp đơn giản hóa quá trình xử lý dữ liệu.
* Đội ngũ phát triển ứng dụng không quá lớn.
* Xây dựng ứng dụng ít native code nhất cho nhiều hệ điều hành khác nhau.
* Trải nghiệm người dùng tốt hơn khi so sánh với ứng dụng Hybrids

## MongoDB



Hình 2- 5 MongoDB

1. Tổng quan:

* MongoDB: là một trong những cơ sở dữ liệu phi quan hệ (NoSQL) phổ biến nhất hiện nay. MongoDB là một hệ thống cơ sở dữ liệu phi quan hệ (NoSQL), mã nguồn mở, được phát hành lần đầu vào tháng 2 năm 2009 bởi MongoDB Inc và quản lý theo SSPL (Server Side Public License). MongoDB lưu trữ dữ liệu dưới dạng tài liệu (document) JSON, cho phép các ứng dụng lưu trữ và truy vấn dữ liệu một cách linh hoạt và hiệu quả.

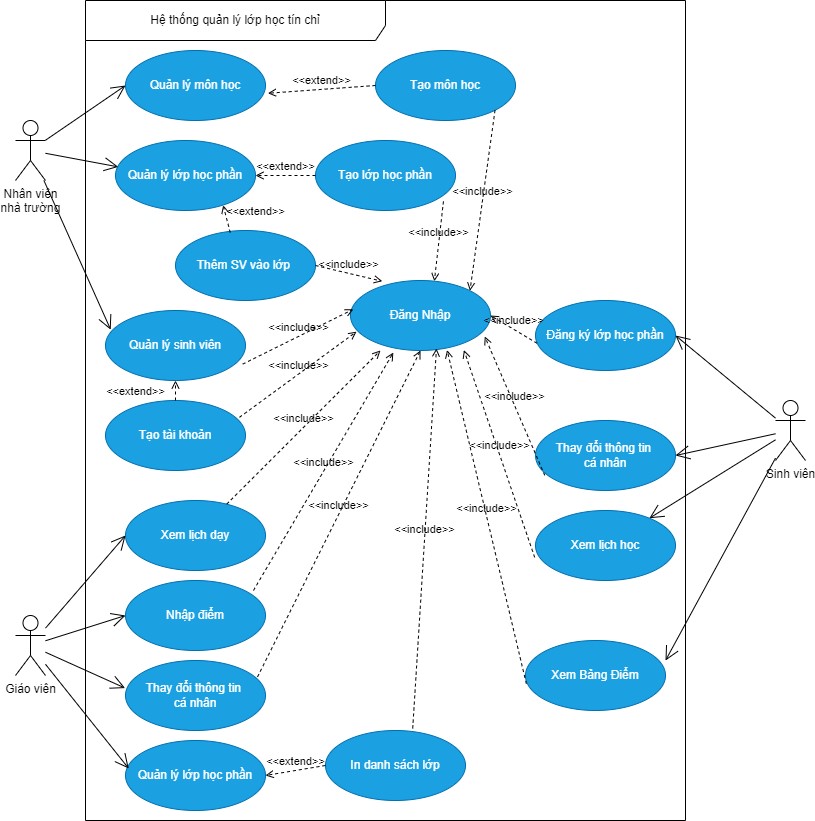
1. Ưu điểm của MongoDB:

* Khả năng lưu trữ dữ liệu phân tán, linh hoạt trong cấu trúc dữ liệu, có thể mở rộng, tốc độ truy vấn nhanh và hỗ trợ các tính năng như indexing, replication, sharding và map-reduce.
* MongoDB hỗ trợ tất cả các ngôn ngữ phổ biến như C, C++, C# và .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. Vì vậy, bạn có thể tạo một ứng dụng bằng bất kỳ ngôn ngữ nào trong số này.
* Hiện nay có rất nhiều công ty đang sử dụng MongoDB như Facebook, Nokia, eBay, Adobe, Google,... để lưu trữ lượng lớn dữ liệu của họ.

# : PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

### Use case tổng quát



Hình 2.3 Sơ đồ UseCase của hệ thống quản lý lớp học tín chỉ

### Danh sách tác nhân và mô tả

|  |  |  |
| --- | --- | --- |
| **Tác nhân** | **Mô tả tác nhân** | **Chức năng của tác nhân** |
| **Sinh viên** | Những người theo học tại trường | - Quản lý thông tin cá nhân  - Đăng ký học phần  - Xem lịch học, lịch thi  - Xem tiến độ học tập  - Đánh giá lớp học phần |
| **Giảng viên** | Những người có nhiệm vụ giảng dạy, quản lý các lớp học phần của mình | - Quản lý thông tin cá nhân  - Quản lý thông tin lớp học  - Quản lý điểm  - Xem lịch học, lịch thi |
| **Nhân viên**  **nhà trường** | Những người có nhiệm vụ quản lý một phần của hệ thống mà mình được phân quyền | - Quản lý sinh viên  - Quản lý giảng viên  - Quản lý môn học  - Quản lý lớp học phần |

Bảng 3.1.2 Danh sách tác nhân và mô tả

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

|  |  |
| --- | --- |
| **ID use case** | **Tên Use case** |
| UC01 | Đăng nhập |
| UC02 | Tạo tài khoản cho người dùng |
| UC03 | Cập nhật thông tin người dùng |
| UC04 | Tạo môn học |
| UC05 | Tạo lớp học phần |
| UC06 | Đăng ký lớp học phần |
| UC07 | Nhập điểm của lớp học phần |

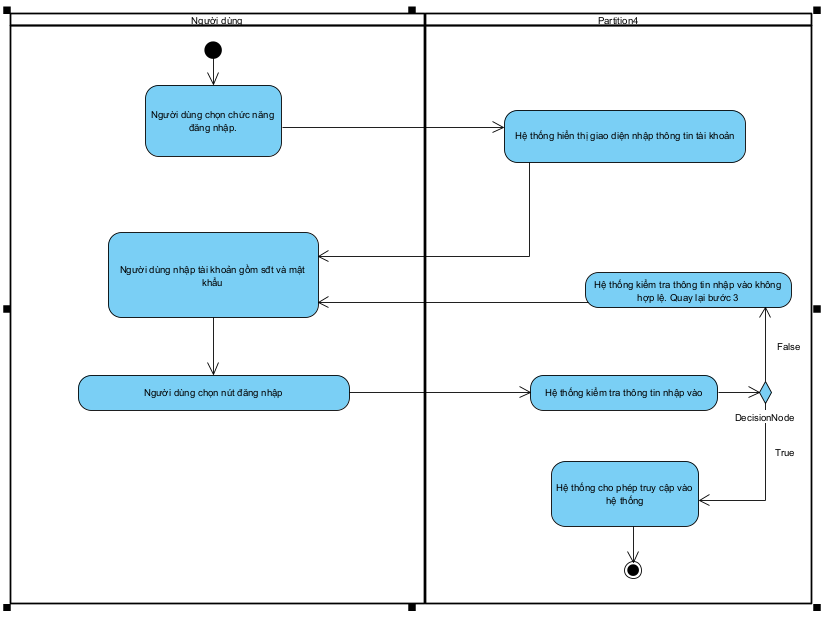
Bảng 3- 1.3 Danh sách các tình huống trong hệ thống

### Tình huống hoạt động

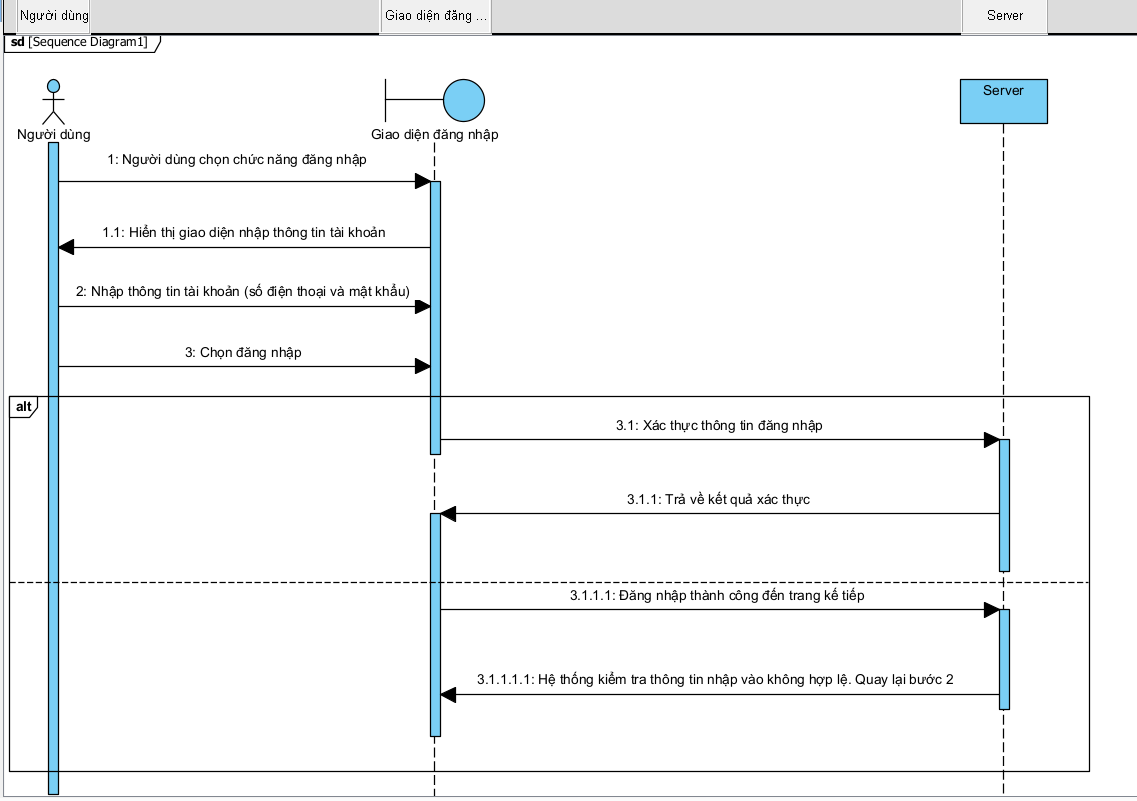
#### Use case đăng nhập

|  |  |
| --- | --- |
| * **Tên use case**: UC01\_Đăng nhập | |
| * **Mô tả sơ lược**: Người dùng đăng nhập vào hệ thống | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):**Người dùng phải có tài khoản trong hệ thống và phải truye cập vào trang web riêng dành cho loại người dùng đó | |
| * **Hậu điều kiện (Post-condition):** Người dùng truy cập được vào hệ thống | |
| * **Luồng sự kiện chính (main flow):** | |
| **Người Dùng** | **Hệ thống** |
| 1. Người dùng chọn chức năng đăng nhập. | 1. Hệ thống hiển thị giao diện nhập thông tin tài khoản |
| 1. Người dùng nhập tài khoản gồm mã dành riêng cho người dùng và mật khẩu |  |
| 1. Người dùng chọn nút đăng nhập | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Hệ thống cho phép truy cập vào hệ thống |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 5.1. Hệ thống kiểm tra thông tin nhập vào không hợp lệ. Quay lại bước 3 |

Bảng 3.1.4.1 Use case đăng nhập



Hình 3.1.4.1a. Activity diagram đăng nhập

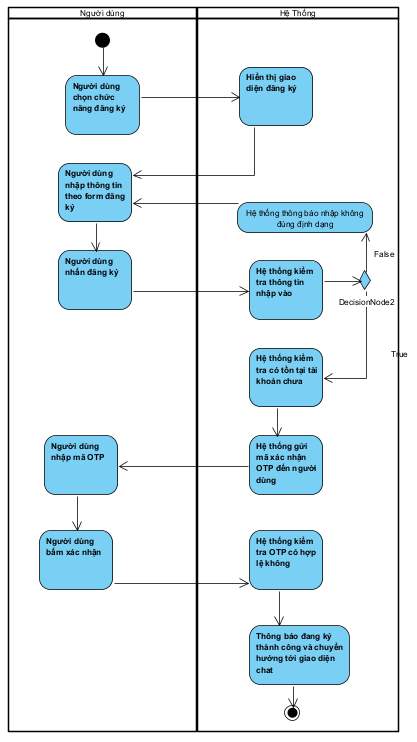


Hình 3.1.4.1b. Sequence diagram đăng nhập

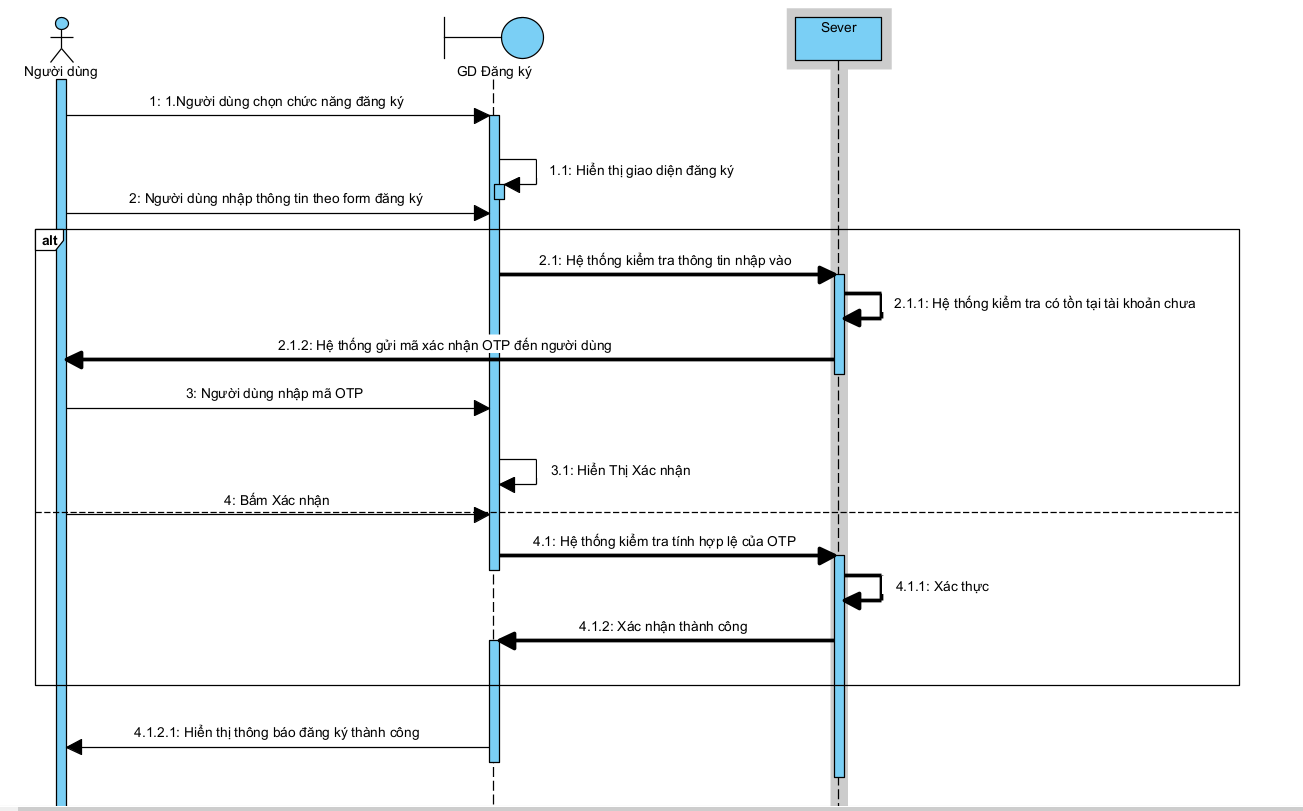
#### Use case tạo tài khoản

|  |  |
| --- | --- |
| **Tên UseCase:** UC02\_Tạo tài khoản | |
| **Mô tả sơ lược:** Cho phép nhân viên nhà trường tạo tài khoản cho người dùng | |
| **Tác nhân chính:** Nhân viên nhà trường | |
| **Tác nhân phụ:** | |
| **Tiền điều kiện:** Nhân viên được phân quyền chọn chức năng tạo tài khoản | |
| **Hậu điều kiện:** Tài khoản được tạo thành công | |
| **Luồng sự kiện chính** | |
| **Nhân viên** | **Hệ thống** |
| 1. Nhân viên nhà trường chọn chức năng tạo loại tài khoản cần tạo. | 2. Hệ thống hiển thị giao diện tạo tài khoản cần tạo. |
| 3. Nhân viên nhập thông tin của tài khoản cần tạo và nhấn “Lưu”. | 4. Hệ thống kiểm tra thông tin nhập vào. |
|  | 5. Hệ thống cập nhật thông tin vào cơ sở dữ liệu. |
|  | 6. Hệ thống thông báo tạo tài khoản thành công. |
| **Luồng sự kiện thay thế** | |
|  | 2.1. Hệ thống thông báo nhân viên chưa được cấp quyền tạo loại tài khoản này. |
|  | 4.1. Hệ thống thông báo thông tin không đúng định dạng. Quay lại bước 3. |
|  | 5.1. Hệ thống thông báo tạo tài khoản không thành công. |

Bảng 3.1.4.2. Use case tạo tài khoản



Hình 3- 3.1.4.2 Activity diagram tạo tài khoản

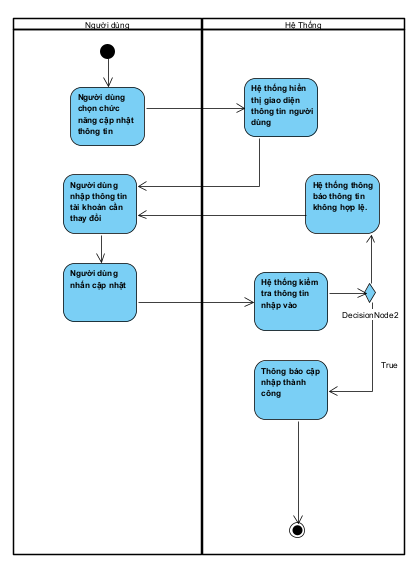


Hình 3- 3.1.4.2 sequence diagram tạo tài khoản

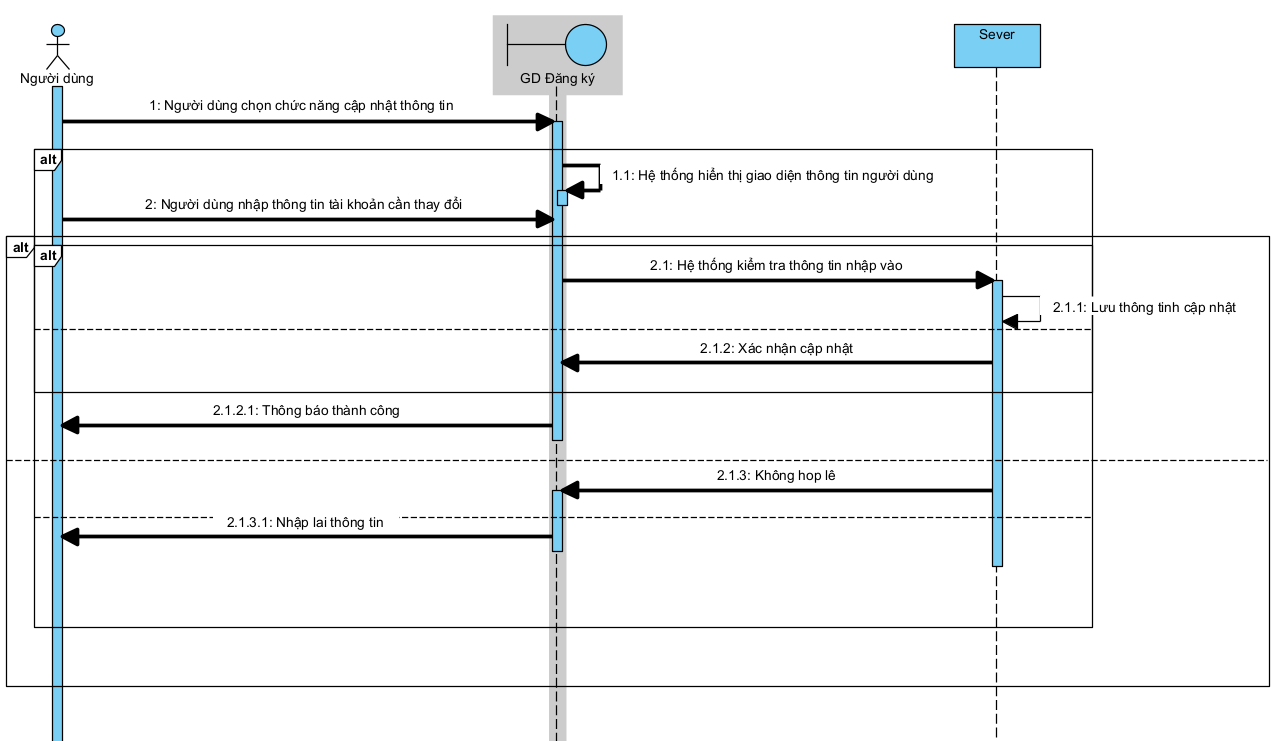
#### Use case cập nhập thông tin người dùng

|  |  |
| --- | --- |
| * **Tên use case**: UC03\_Cập nhập thông tin người dùng | |
| * **Mô tả sơ lược**: Giúp người dùng cập nhập thông tin của người dùng trong hệ thống | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đăng nhập thành công vào hệ thống | |
| * **Hậu điều kiện (Post-condition):** Thông tin người dùng được cập nhập | |
| * **Luồng sự kiện chính (main flow):** | |
| **Người dùng** | **Hệ Thống** |
| 1. Người dùng chọn chức năng cập nhật thông tin | 1. Hệ thống hiển thị giao diện thông tin người dùng |
| 1. Người dùng nhập thông tin tài khoản cần thay đổi |  |
| 1. Người dùng nhấn cập nhật | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Thông báo cập nhập thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 5.1. Hệ thống thông báo thông tin không hợp lệ. Quay lại bước 3 |

Bảng 3.1.4.3 Use case cập nhật thông tin người dùng



Hình 3.1.4.3a. Activity diagram cập nhật thông tin người dùng

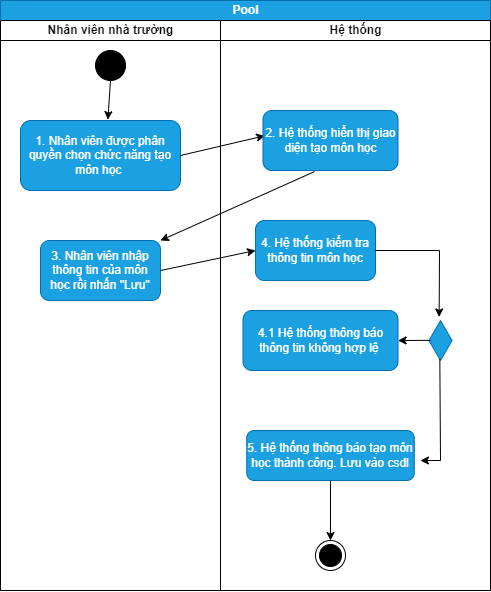
****

Hình 3.1.4.3b. sequence cập nhật thông tin người dùng

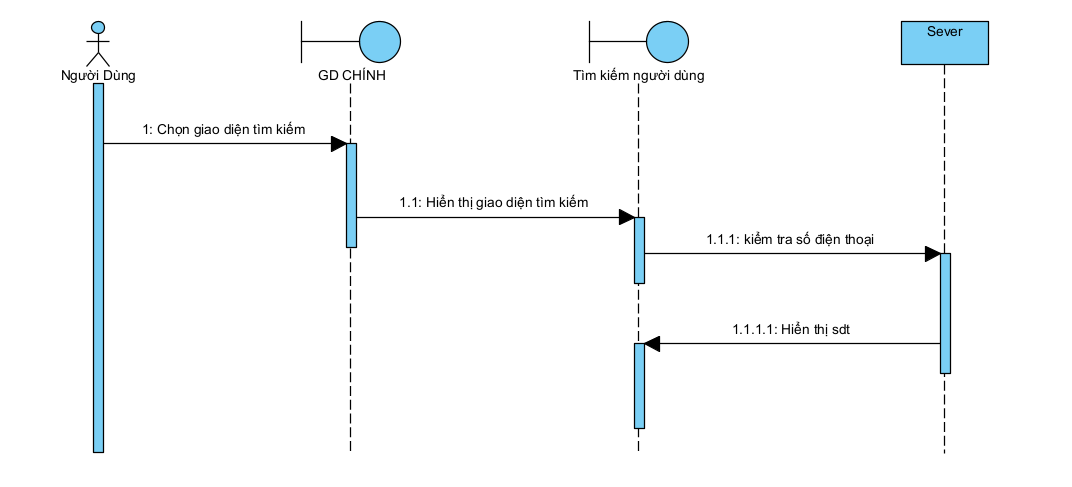
#### Use case tạo môn học

|  |  |
| --- | --- |
| **Use case:** UC04\_Tạo môn học | |
| **Mô tả sơ lược**: Giúp cho nhân viên nhà trường tạo môn học cho từng khoa | |
| **Actor chính**: Nhân viên nhà trường | |
| **Actor phụ**: | |
| **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào hệ thống. | |
| **Hậu điều kiện (Post-condition):** Môn học được lưu vào hệ thống | |
| * **Luồng sự kiện chính (main flow):** | |
| **Nhân viên** | **Hệ Thống** |
| 1. Nhân viên được phân quyền chọn vào biểu tượng “tạo môn học” | 1. Hệ thống hiển thị giao diện “tạo môn học” |
| 1. Nhân viên nhập thông tin của môn học |  |
| 1. Người dùng bấm nút tạo môn học | 1. Hệ thống kiểm tra tính hợp lệ thông tin |
|  | 1. Hệ thống lưu thông tin của môn học vào csdl. Thông báo môn học đã được tạo thành công. |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 5.1 Hệ thống thông báo thông tin hợp lệ. Quay lại bước 3 |
| * **Luồng sự kiện ngoại lệ (exception flow):** | |

Bảng 3.1.4.4 Use case tạo môn học



Hình 3- 3.1.4.4 Activity diagram tạo môn học

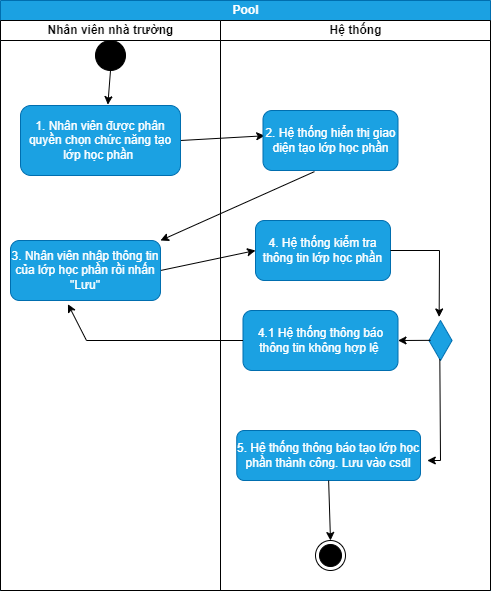
****

Hình 3- 3.1.4.4 sequence tạo môn học

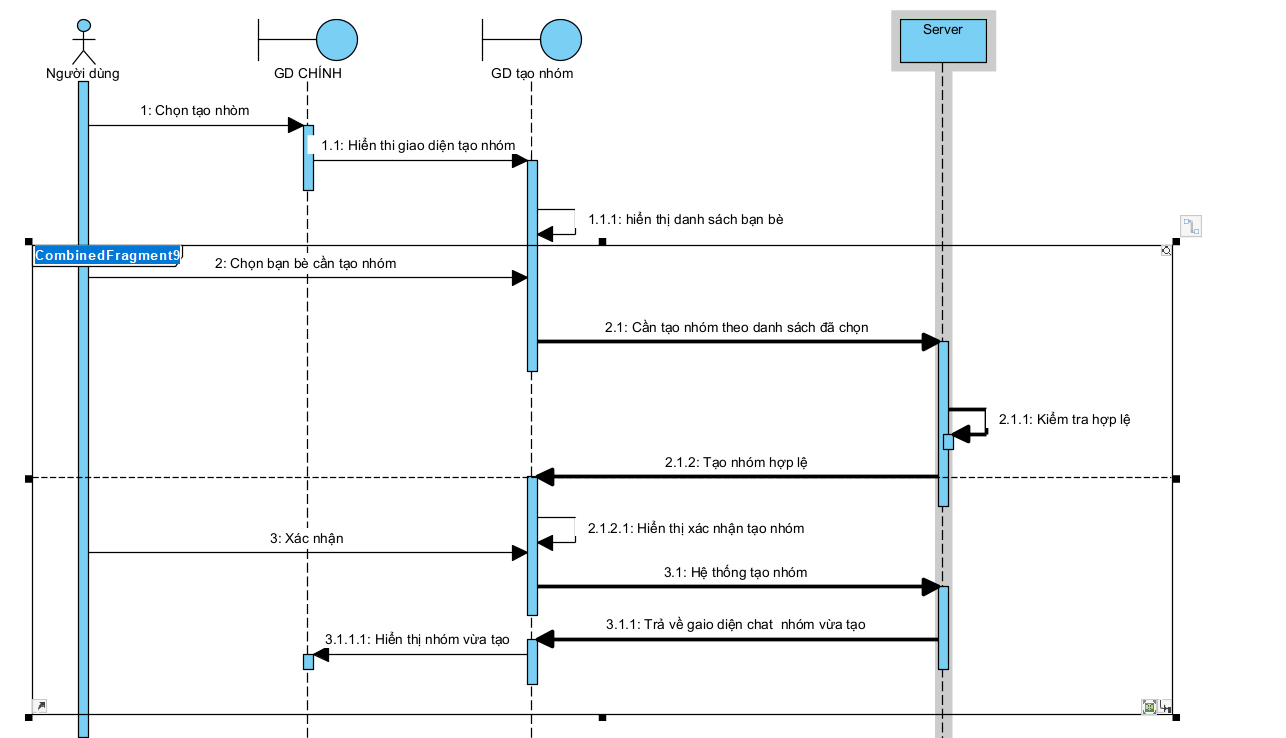
#### Use case tạo tạo lớp học phần

|  |  |
| --- | --- |
| **Use case:** UC05\_Tạo lớp học phần | |
| **Mô tả sơ lược**: Giúp nhân viên tạo ra các lớp học phần cho các môn học | |
| **Actor chính**: Nhân viên nhà trường. | |
| **Actor phụ**: | |
| **Tiền điều kiện (Pre-condition):** Nhân viên đã đăng nhập vào hệ thống | |
| **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Nhân viên** | **Hệ thống** |
| 1. Nhân viên được phân quyền chọn chức năng tạo lớp học phần | 1. Hệ thống hiển thị giao diện tạo lớp học phần |
| 1. Nhân viên nhập thông tin của lớp học phần rồi nhấn “Lưu” | 1. Hệ thống kiểm tra thông tin lớp học phần |
|  | 1. Hệ thống thông báo đã tạo được lớp học phần. Hệ thống lưu thông tin vào cơ sở dữ liệu |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.1 Hệ thống thông báo thông tin không hợp lệ. Quay lại 3 |
| * **Luồng sự kiện ngoại lệ (exception flow):** | |

Bảng 3.1.4.5 Use case tạo lớp học phần



Hình 3- 3.1.4.5 Activity diagram tạo lớp học phần

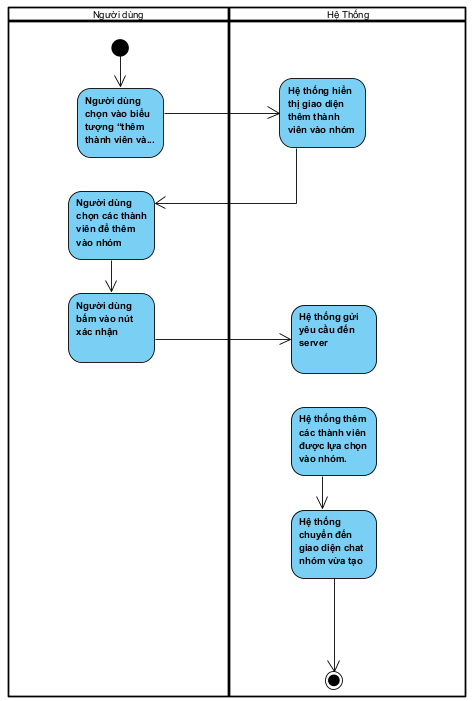


Hình 3- 3.1.4.5 sequence tạo lớp học phần

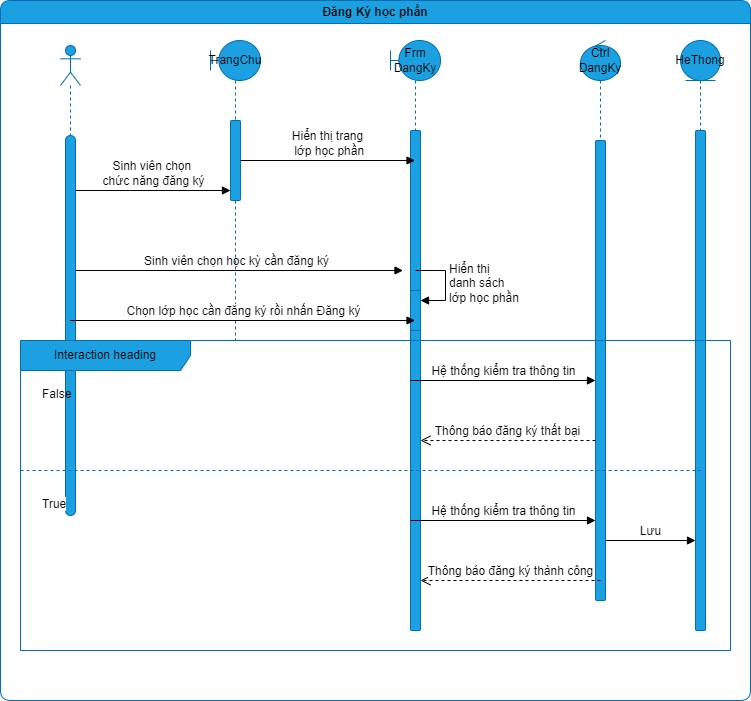
#### Use case đăng ký lớp học phần

|  |  |  |
| --- | --- | --- |
| **Use case:** UC06\_Đăng ký lớp học phần | | |
| **Mô tả sơ lược**: Cho phép sinh viên đăng ký lớp học phần | | |
| **Actor chính**: Sinh viên | | |
| **Actor phụ**: | | |
| **Tiền điều kiện (Pre-condition):** Sinh viên còn trong thời hạn học tập | | |
| **Hậu điều kiện (Post-condition):** Đăng ký lớp học phần thành công | | |
| * **Luồng sự kiện chính (main flow):** | | |
| **Sinh viên** | | **Hệ thống** |
| 1. Sinh viên nhà trường chọn chức năng đăng ký lớp học phần. | | 2. Hệ thống hiển thị giao diện đăng ký lớp học phần. |
| 3. Sinh viên chọn học kỳ cần đăng ký. | | 4. Hệ thống hiển thị thông tin của các môn học phần trong kỳ đó. |
| 5. Sinh viên chọn môn học cần đăng ký . | | 6. Hệ thống hiển thị thông tin các lớp học phần của môn học đó. |
| 7. Sinh viên chọn lớp học phần cần đăng ký rồi nhấn “Đăng ký”. | | 8. Hệ thống kiểm tra thông tin đăng ký. |
|  | | 9. Hệ thống cập nhập thông tin đăng ký vào cơ sở dữ liệu. Hệ thống thông báo đăng ký thành công. |
| * **Luồng sự kiện thay thế (alternate flow):** | | |
|  | 4.1. Hệ thống thông báo học kỳ đó chưa được mở đăng ký. | |
| * **Luồng sự kiện ngoại lệ (exception flow):** | | |

Bảng 3- 3.1.4.6 Use case đăng ký thành công



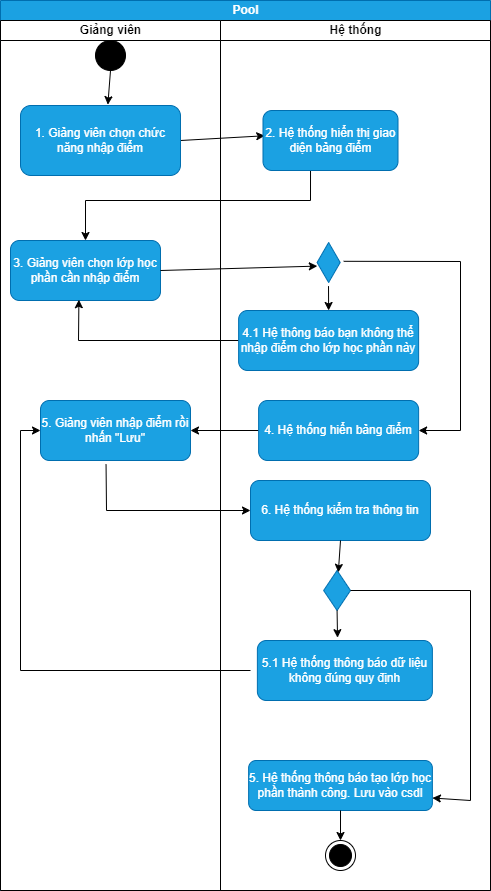
Hình 3- 3.1.4.6 Activity diagram đăng ký



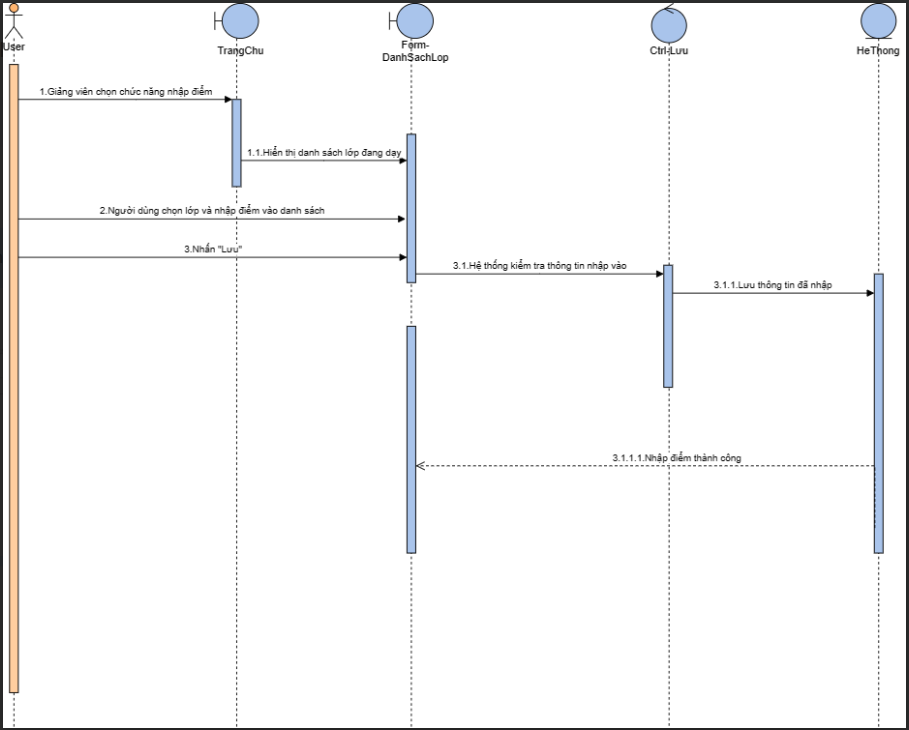
#### Use case nhập điểm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC07\_Nhập điểm | | |
| **Mô tả sơ lược**: Cho phép giáo viên nhập điểm cho lớp học phần mình đang giảng dạy | | |
| **Actor chính**: Giảng viên | | |
| **Actor phụ**: | | |
| **Tiền điều kiện (Pre-condition):** Giảng viên được phân quyền chọn chức năng nhập điểm | | |
| **Hậu điều kiện (Post-condition):** Điểm được nhập thành công | | |
| * **Luồng sự kiện chính (main flow):** | | |
| **Giảng viên** | **Hệ Thống** | |
| 1. Giảng viên chọn chức năng nhập điểm | 2. Hệ thống hiển thị giao diện nhập điểm | |
| 3. Giảng viên chọn lớp học phần cần nhập điểm | 4. Hệ thống hiển thị bảng điểm của lớp học phần | |
| 5. Giảng viên nhập điểm rồi nhấn “Lưu” | 6. Hệ thống kiểm tra thông tin được nhập vào | |
|  | 7. Hệ thống thông báo nhập điểm thành công. Lưu thông tin vào csdl | |
| * **Luồng sự kiện thay thế (alternate flow):** | | |
|  | | 4.1. Hệ thống thông báo bạn không có quyền để nhập điểm cho lớp học phần. Quay lại bước 3 |
|  | | 6.1 Hệ thống thông báo điểm nhập không đúng quy định. Quan lại bước 5. |
| * **Luồng sự kiện ngoại lệ (exception flow):** | | |

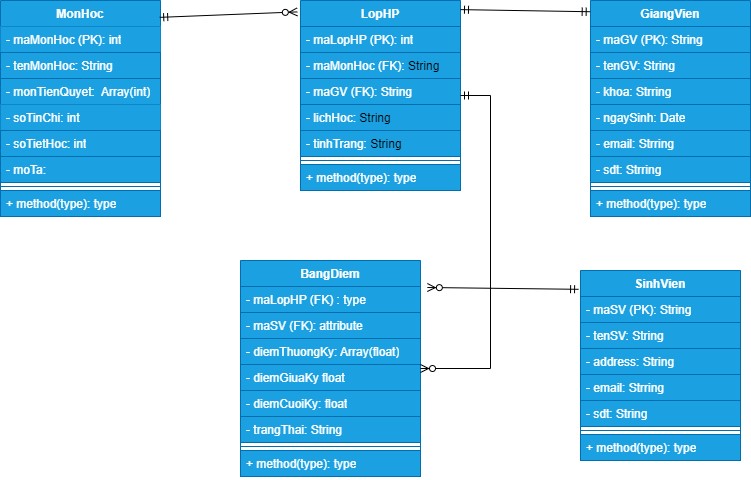
Bảng 3.1.4.7. usecase nhập điểm cho lớp học phần



Hình 3.1.4.7b. Activity nhập điểm cho lớp học phần



## Class diagram



Hình 2.5 Sơ đồ class dỉagram

# : HIỆN THỰC

## Cấu hình phần cứng, phần mềm

* Các dịch vụ Firebase:
  + - Firebase: lưu trữ hình ảnh
* Cấu hình Server: Visual code studio, MongoDB Compass
* Cấu hình Client: Visual code studio, MongoDB Compass

## Giao diện của hệ thống

### Giao diện web

### Giao diện mobile

# : KẾT LUẬN

## Kết quả đạt được

## Hạn chế của đồ án

## Hướng phát triển

Tiếp tục hoàn thành project, fix các lỗi và thêm vào 1 số chức năng như call video, đăng bài viết như 1 mạng xã hội thực sự

# 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
3. Richards,Mark;Ford, Neal (2020-01-28)-Fundamentals of Software Architecture

Các tài liệu từ Internet

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