



**TÊN ĐỀ TÀI: hệ thống quản lý lớp học tín chỉ**

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

**Lớp HP: DHKTPM16A**

***Nhóm 02 - Sinh viên thực hiện***

1. Lê Quốc Phòng - 20070031
2. Nguyễn Đức Tài - 20077681

# 

# MỤC LỤC

[MỤC LỤC 1](#_heading=h.1fob9te)

[DANH MỤC CÁC HÌNH VẼ 1](#_heading=h.3znysh7)

[DANH MỤC CÁC BẢNG BIỂU 1](#_heading=h.2et92p0)

[CHƯƠNG 1 : GIỚI THIỆU 1](#_heading=h.tyjcwt)

[1.1 Tổng quan 1](#_heading=h.3dy6vkm)

[1.2 Mục tiêu đề tài 1](#_heading=h.1t3h5sf)

[1.3 Phạm vi đề tài 1](#_heading=h.4d34og8)

[1.4 Mô tả yêu cầu chức năng 1](#_heading=h.2s8eyo1)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT](#_heading=h.17dp8vu) 1

[2.1 Node.js 1](#_heading=h.26in1rg)

[2.2 MongoDB / DynamoDB 1](#_heading=h.35nkun2)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 1](#_heading=h.1ksv4uv)

[3.1 Phân tích yêu cầu bằng UML 1](#_heading=h.44sinio)

[3.1.1 Usecase tổng quát 1](#_heading=h.2jxsxqh)

[3.1.2 Danh sách tác nhân và mô tả 1](#_heading=h.z337ya)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 1](#_heading=h.1y810tw)

[3.1.4 Tình huống hoạt động 1](#_heading=h.2xcytpi)

[3.2 Class diagram 1](#_heading=h.1ci93xb)

[3.3 Deployment diagram 1](#_heading=h.3whwml4)

[CHƯƠNG 4 : HIỆN THỰC 1](#_heading=h.2bn6wsx)

[4.1 Cấu hình phần cứng, phần mềm 1](#_heading=h.qsh70q)

[4.2 Giao diện của hệ thống 1](#_heading=h.3as4poj)

[CHƯƠNG 5 : KẾT LUẬN 1](#_heading=h.49x2ik5)

[5.1 Kết quả đạt được 1](#_heading=h.2p2csry)

[5.2 Hạn chế của đồ án 1](#_heading=h.147n2zr)

[5.3 Hướng phát triển 1](#_heading=h.3o7alnk)

[TÀI LIỆU THAM KHẢO 1](#_heading=h.ihv636)

# 

# DANH MỤC CÁC HÌNH VẼ

[Hình 2-1 Node.js development 1](#_heading=h.lnxbz9)

# DANH MỤC CÁC BẢNG BIỂU

[Bảng 3-1 Danh sách tác nhân và mô tả 1](#_heading=h.3j2qqm3)

[Bảng 3-2 Danh sách các tình huống trong hệ thống 1](#_heading=h.4i7ojhp)

# : GIỚI THIỆU

## Tổng quan

Hệ thống quản lý lớp học tín chỉ là một giải pháp quan trọng giúp trường đại học tự động hóa quy trình đăng ký học phần của sinh viên. Hệ thống này được thiết kế để đáp ứng nhu cầu cụ thể của mô hình giáo dục tín chỉ, trong đó sinh viên có thể tự do lựa chọn các môn học mà họ muốn học trong mỗi học kỳ, tuân thủ các quy định về môn học tiên quyết và giới hạn tín chỉ tối đa.

Hệ thống cho phép sinh viên đăng ký vào các lớp học tín chỉ cụ thể, với số lượng sinh viên tối đa cho mỗi lớp được xác định trước. Trong trường hợp lớp học đã đầy, sinh viên có thể được đưa vào danh sách chờ, và nhà trường có thể cân nhắc việc mở thêm lớp học.

Một tính năng quan trọng khác của hệ thống là khả năng xác nhận đăng ký của sinh viên. Điều này giúp ngăn chặn trường hợp sinh viên đăng ký rồi sau đó hủy bỏ, gây rối loạn cho quy trình quản lý.

Ngoài ra, hệ thống cung cấp cho sinh viên thông tin chi tiết về quá trình học tập của họ, bao gồm số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy, và thời khóa biểu theo tuần.

Cuối cùng, sau khi đạt đủ số tín chỉ theo yêu cầu của ngành học, sinh viên có thể đăng ký xét tốt nghiệp. Thông tin về bằng cấp và thông tin của cựu sinh viên sẽ được lưu giữ và công khai trên trang web của nhà trường.

## Mục tiêu đề tài

Hệ thống quản lý lớp học tín chỉ nhằm mục đích tạo ra một giải pháp hiệu quả và tự động để quản lý quy trình đăng ký học phần của sinh viên tại trường đại học. Mục tiêu cụ thể của đề tài bao gồm:

* Tự động hóa quy trình đăng ký học phần: Hệ thống sẽ cho phép sinh viên tự đăng ký các môn học mà họ muốn học trong mỗi học kỳ, tuân thủ các quy định về môn học tiên quyết và giới hạn tín chỉ tối đa.
* Quản lý lớp học tín chỉ: Hệ thống sẽ quản lý số lượng sinh viên tối đa cho mỗi lớp học tín chỉ và cung cấp danh sách chờ cho những sinh viên không thể đăng ký do lớp đã đầy.
* Xác nhận đăng ký: Hệ thống sẽ yêu cầu sinh viên xác nhận đăng ký của họ để ngăn chặn trường hợp đăng ký rồi hủy bỏ.
* Cung cấp thông tin học tập: Hệ thống sẽ cung cấp cho sinh viên thông tin chi tiết về quá trình học tập của họ, bao gồm số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy, và thời khóa biểu theo tuần.
* Quản lý xét tốt nghiệp: Hệ thống sẽ cho phép sinh viên đăng ký xét tốt nghiệp sau khi họ đã đạt đủ số tín chỉ theo yêu cầu của ngành học. Thông tin về bằng cấp và thông tin của cựu sinh viên sẽ được lưu giữ và công khai trên trang web của nhà trường.

## Phạm vi đề tài

Triển khai trên nền tảng Web

## Mô tả yêu cầu chức năng

Người dùng có thể thực hiện một loạt các hoạt động trên ứng dụng như sau:

**Đăng nhập:**

* Đăng nhập thông qua user code và mật khẩu đã đăng ký trước đó.
* Cập nhật thông tin cá nhân và đặt lại mật khẩu nếu cần.

**Quản lý đăng ký học phần:**

* Đăng ký các môn học mà họ muốn học trong mỗi học kỳ,
* Xác nhận đăng ký của họ để ngăn chặn trường hợp đăng ký rồi hủy bỏ.
* Xem danh sách các lớp học tín chỉ mà họ đã đăng ký.

**Quản lý thông tin học tập:**

* Xem thông tin chi tiết về quá trình học tập của họ, bao gồm số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy, và thời khóa biểu theo tuần.

**Quản lý xét tốt nghiệp:**

* Đăng ký xét tốt nghiệp sau khi họ đã đạt đủ số tín chỉ theo yêu cầu của ngành học.
* Xem thông tin về bằng cấp và thông tin của cựu sinh viên.

**Quản lý lớp học :**

* Tạo mới lớp học: Admin có thể tạo mới lớp học, bao gồm việc nhập thông tin chi tiết về lớp học như tên lớp, môn học, giáo viên, thời gian học, và số lượng sinh viên tối đa.
* Xóa lớp học: Admin có thể xóa lớp học nếu họ cảm thấy lớp học đó không còn cần thiết hoặc đã hoàn thành.
* Cập nhật thông tin lớp học: Admin có thể cập nhật thông tin về lớp học, bao gồm việc thay đổi tên lớp, môn học, giáo viên, thời gian học, và số lượng sinh viên tối đa.

**Ngoài ra, ứng dụng cũng phải đáp ứng các yêu cầu phi chức năng như:**

* Giao diện thân thiện, màu sắc phù hợp và dễ sử dụng.
* Đảm bảo tính bảo mật và xác thực.
* Cung cấp dung lượng lưu trữ đủ lớn.
* Hệ thống hoạt động ổn định và nhanh chóng.

# : CƠ SỞ LÝ THUYẾT

## Kiến trúc thiết kế (Architectural design)

## Mô hình Client-Server MVC

**Kiến trúc MVC truyền thống:**

Mô hình MVC truyền thống thường được sử dụng trong các ứng dụng web, nơi trình duyệt web đóng vai trò là client và máy chủ web đóng vai trò là server. Dưới đây là mô tả chi tiết cách thức hoạt động của mô hình MVC trong kiến trúc client-server:

**1. Client:**

* Người dùng gửi yêu cầu thông qua trình duyệt web đến máy chủ web.
* Yêu cầu này thường bao gồm thông tin về hành động cần thực hiện (ví dụ: nhấp vào nút "Gửi") và dữ liệu đầu vào (ví dụ: thông tin trong biểu mẫu).

**2. Server:**

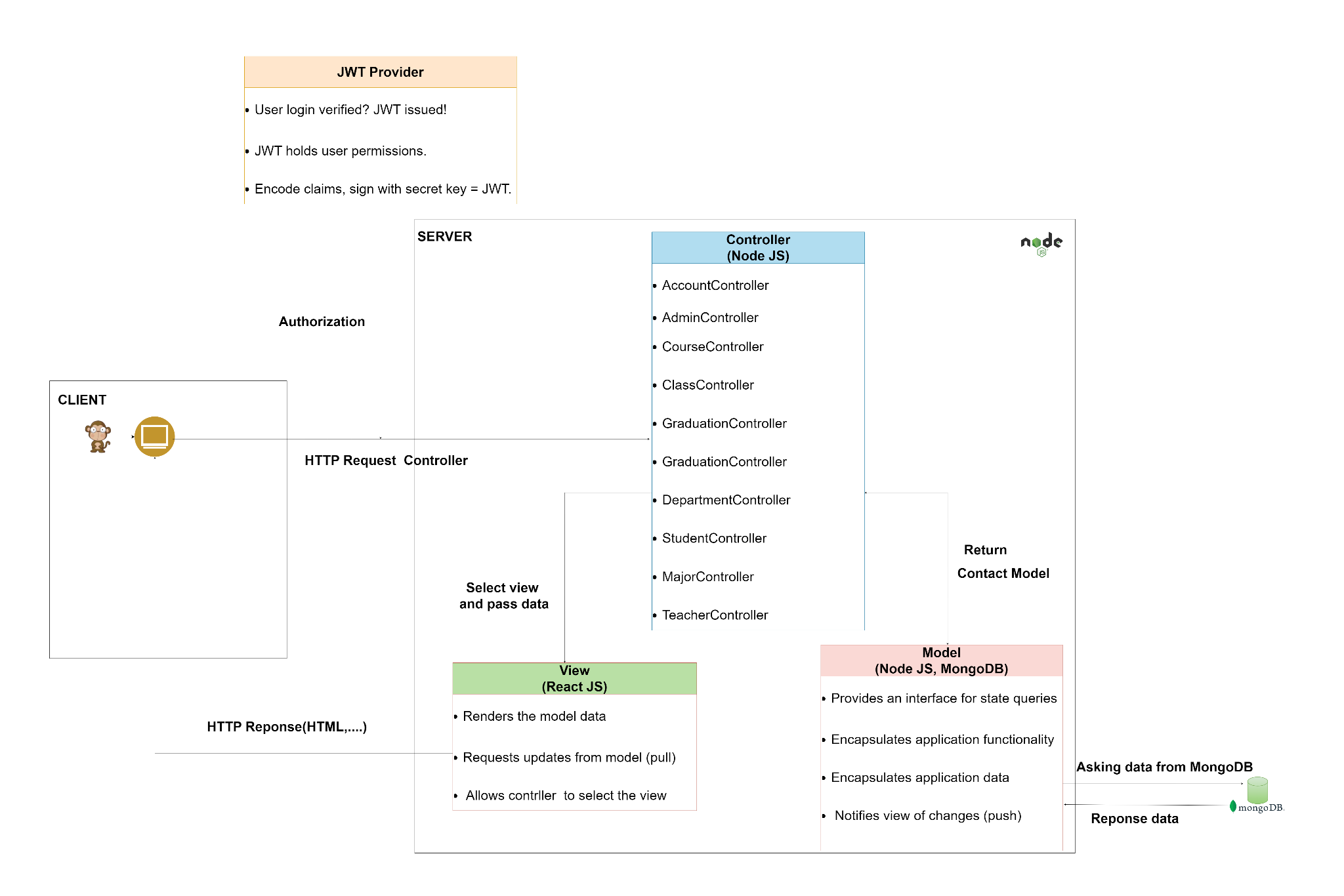
* Máy chủ web nhận được yêu cầu từ client.
* Controller của ứng dụng web MVC sẽ được kích hoạt để xử lý yêu cầu.
* Controller tương tác với Model để truy xuất hoặc cập nhật dữ liệu cần thiết.
* Sau khi xử lý dữ liệu, Controller trả về dữ liệu cho View dưới dạng phù hợp.

**3. View:**

* View xử lý dữ liệu nhận được từ Controller và hiển thị nó cho người dùng dưới dạng giao diện người dùng.
* Giao diện người dùng có thể được tạo bằng HTML, CSS, JavaScript hoặc các ngôn ngữ lập trình web khác.
* View sau đó gửi giao diện người dùng đến trình duyệt web.

**4. Phản hồi:**

* Trình duyệt web hiển thị giao diện người dùng cho người dùng.
* Người dùng có thể tương tác với giao diện người dùng để gửi các yêu cầu mới đến máy chủ web.

**Sơ đồ mô hình Client-Server MVC:**

Trong đó:

* **Model (Kiểu mẫu)**: Thể hiện các cấu trúc dữ liệu. Các lớp của thành phần Model thường thực hiện các tác vụ như truy vấn, thêm, xóa và cập nhật dữ liệu. Khi dữ liệu trong Model thay đổi, thành phần View sẽ được cập nhật.
* **View (Xem)**: Là một thành phần thể hiện dữ liệu trong Model thành các giao diện tương tác với người dùng. Một Model có thể có nhiều View tùy thuộc vào mục đích khác nhau.
* **Controller (Bộ điều khiển)**: Hoạt động như một trung gian giữa Model và View. Thông tin người dùng từ View được gửi đến Controller để xử lý, sau đó Controller tương tác với Model để truy xuất dữ liệu được yêu cầu và cuối cùng Controller trả lại dữ liệu này cho View.

**Lợi ích của mô hình Client-Server MVC:**

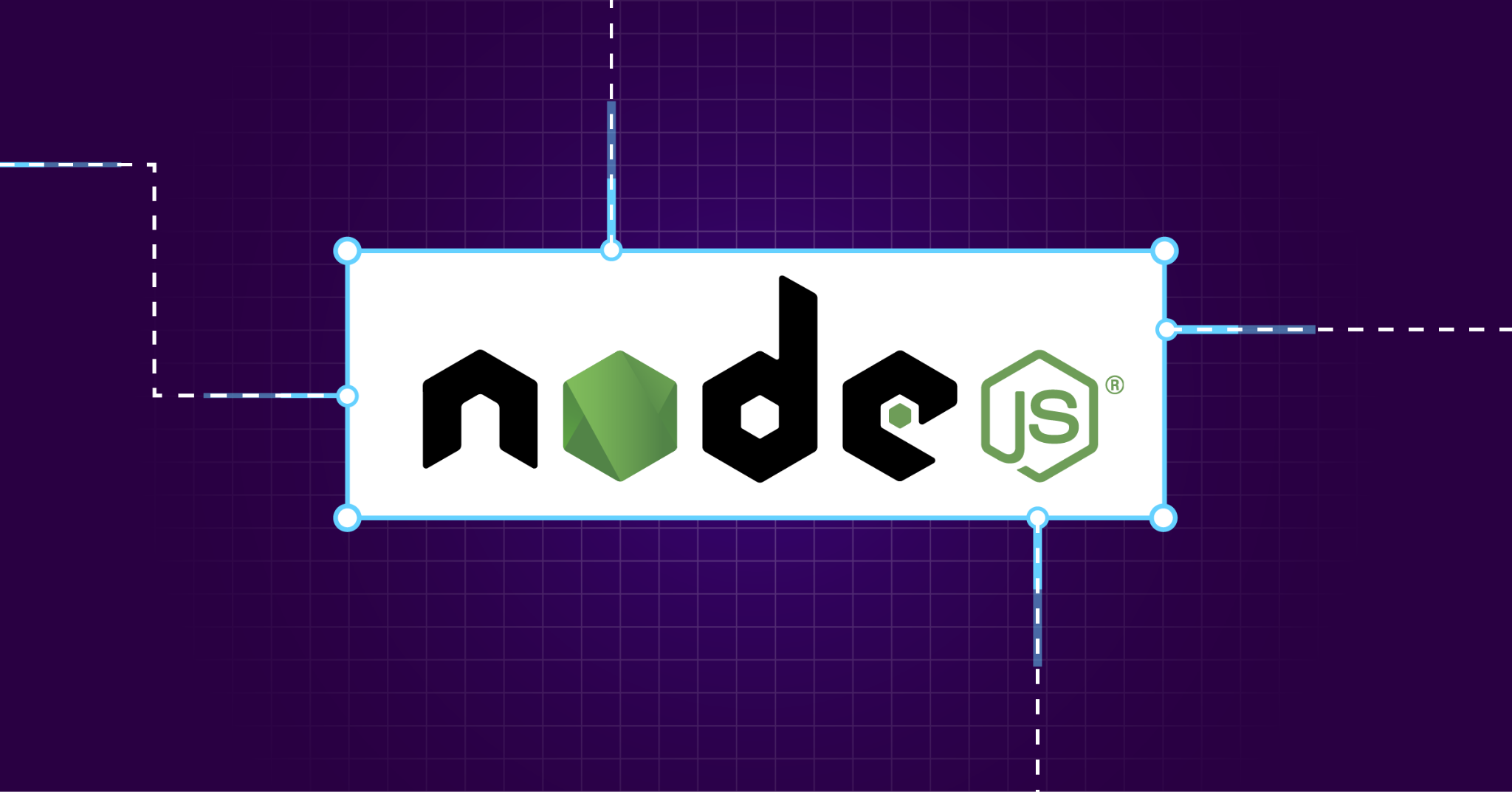
* **Tách biệt mối quan tâm:** Mô hình MVC giúp tách biệt logic nghiệp vụ (Model), giao diện người dùng (View) và logic điều khiển (Controller). Điều này giúp cho việc phát triển, kiểm thử và bảo trì ứng dụng dễ dàng hơn.
* **Tái sử dụng:** Các thành phần Model, View và Controller có thể được tái sử dụng trong các ứng dụng khác nhau.
* **Bảo trì:** Mô hình MVC giúp dễ dàng bảo trì ứng dụng bằng cách tách biệt các thành phần khác nhau.
* **Khả năng mở rộng:** Mô hình MVC có thể dễ dàng mở rộng bằng cách thêm các thành phần Model, View và Controller mới.

**Khó khăn**

* Phụ thuộc vào kết nối mạng:
* Hiệu suất của hệ thống phụ thuộc vào chất lượng kết nối mạng giữa client và server.
* Chi phí cơ sở hạ tầng:
* Cần đầu tư vào phần cứng và phần mềm cho server.
* Chi phí bảo trì và vận hành server có thể cao.
* Độ trễ và hiệu suất:
* Có thể xảy ra độ trễ khi truyền dữ liệu giữa client và server.
* Hiệu suất của hệ thống có thể bị ảnh hưởng nếu server bị quá tải.

Mô hình MVC thường được sử dụng trong các ứng dụng web, vì thành phần View (mã HTML / XHTML) được tạo từ các ngôn ngữ lập trình web. Thành phần Controller sẽ nhận dữ liệu GET / POST, xử lý các dữ liệu này, sau đó chuyển sang Model để xử lý. Model sẽ trả lại dữ liệu cho Controller, sau đó Controller tạo ra mã HTML / XHTML để hiển thị trong View.

## Node.js



Hình 2.1 Node.js development

Node.js là một môi trường chạy mã JavaScript phía máy chủ, được xây dựng trên nền tảng V8 JavaScript engine của Google Chrome. Node.js không chỉ cho phép chúng ta sử dụng JavaScript ở phía máy chủ, mà còn cung cấp khả năng xây dựng các ứng dụng mạng có khả năng mở rộng lớn.

Một trong những đặc điểm nổi bật của Node.js là khả năng xử lý sự kiện không chặn (non-blocking event-driven). Điều này có nghĩa là Node.js không chờ đợi một hoạt động nào đó hoàn thành trước khi chuyển sang hoạt động khác. Thay vào đó, nó sẽ tiếp tục xử lý các hoạt động khác và khi hoạt động ban đầu hoàn thành, một sự kiện sẽ được phát ra để thông báo cho Node.js biết. Điều này giúp Node.js có thể xử lý hàng nghìn kết nối cùng một lúc mà không cần tạo ra các tiến trình mới cho mỗi kết nối.

Node.js được sử dụng rộng rãi trong việc phát triển các ứng dụng web thời gian thực (real-time web applications), ứng dụng đơn trang (single-page applications), ứng dụng di động, và nhiều loại ứng dụng khác.

## MongoDB



Hình 2.2 MongoDB

MongoDB là một hệ thống quản lý cơ sở dữ liệu mã nguồn mở, không sử dụng mô hình quan hệ (NoSQL), được thiết kế để lưu trữ dữ liệu theo cấu trúc giống JSON (BSON), nghĩa là dữ liệu được tổ chức theo hình thức các tài liệu có khả năng lồng vào nhau. MongoDB được ứng dụng rộng rãi trong các ứng dụng web và di động, cũng như các ứng dụng đòi hỏi khả năng mở rộng và linh hoạt trong việc lưu trữ dữ liệu. Nó cung cấp các chức năng như bộ sao chép (replica sets), phân mảnh (sharding), chỉ mục (indexation), và tìm kiếm toàn văn (full-text search). MongoDB là một lựa chọn được ưa chuộng cho việc phát triển các ứng dụng yêu cầu lưu trữ dữ liệu phức tạp và linh hoạt.

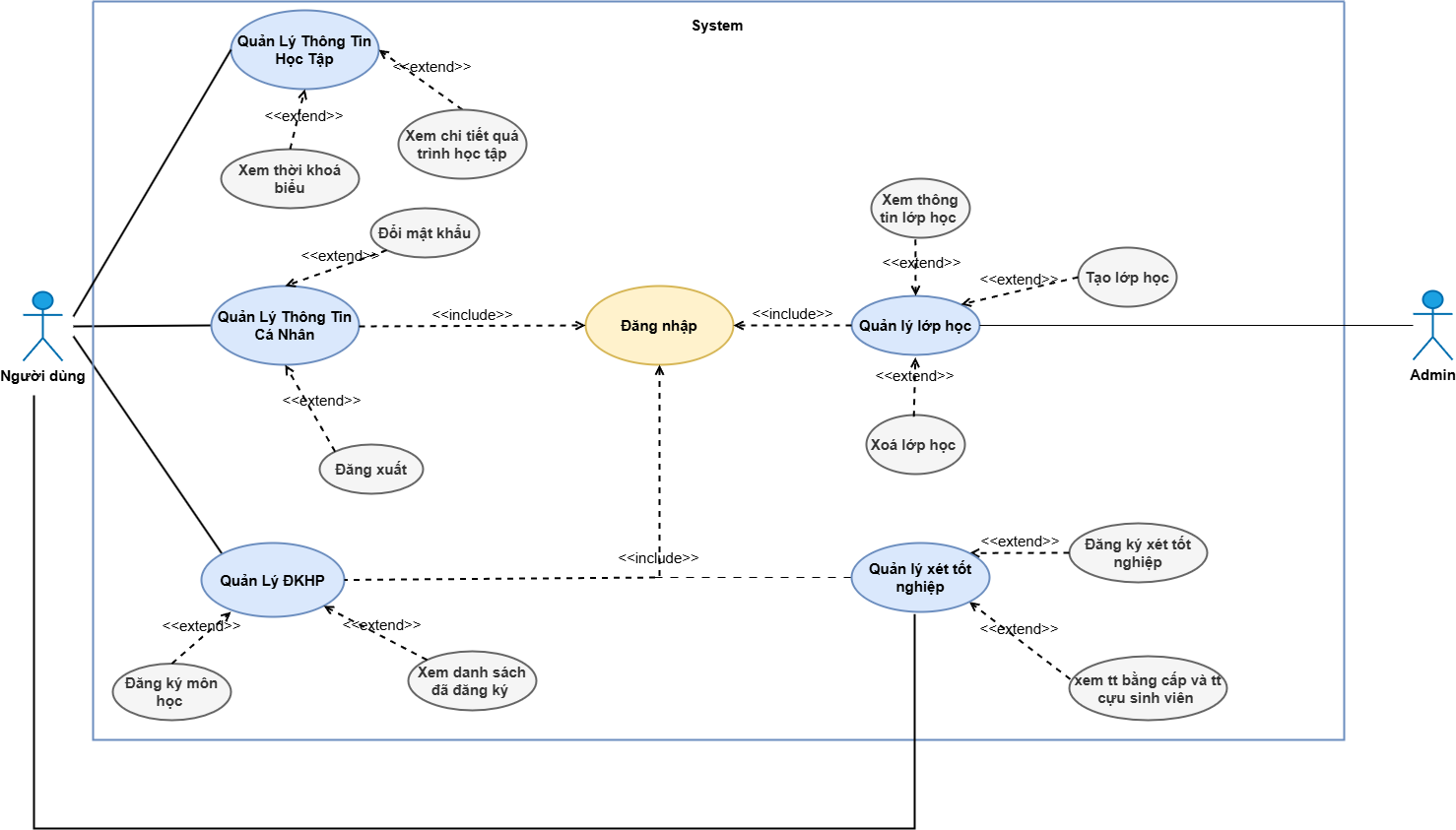
## ReactJS

Hình 2.5 ReactJS

ReactJS, một thư viện JavaScript mã nguồn mở do Facebook tạo ra, được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web hiện đại. ReactJS cho phép bạn tạo ra các thành phần UI có thể tái sử dụng và linh hoạt, giúp quản lý trạng thái ứng dụng một cách dễ dàng và hiệu quả.

Điểm nổi bật của ReactJS bao gồm:

* Kiến trúc dựa trên thành phần: ReactJS được xây dựng dựa trên mô hình thành phần. Mỗi phần của giao diện người dùng được xem như một "thành phần", cho phép tái sử dụng dễ dàng và phân chia giao diện thành các phần nhỏ hơn, dễ quản lý.
* Virtual DOM: ReactJS sử dụng một cơ chế gọi là "Virtual DOM" để tối ưu hóa hiệu suất của ứng dụng. Thay vì cập nhật trực tiếp DOM (Document Object Model) khi có thay đổi trong dữ liệu, ReactJS sẽ tạo ra một bản sao của DOM gọi là Virtual DOM, và chỉ cập nhật lại DOM thực sự khi cần thiết. Điều này giúp giảm thiểu số lượng thao tác với DOM và cải thiện hiệu suất ứng dụng. thiểu số lượng thao tác với DOM và cải thiện hiệu suất ứng dụng.
* Cú pháp khai báo: ReactJS sử dụng cú pháp khai báo để xây dựng giao diện người dùng, có nghĩa là bạn chỉ cần chỉ định "cách mà giao diện của bạn nên trông như thế nào" và React sẽ tự động xử lý các thay đổi khi trạng thái của ứng dụng thay đổi.



Hình 3.1.1 Usecase tổng quan

### 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 | Người trực tiếp sử dụng, thao tác với các chức năng của hệ thống đăng ký tín chỉ |
| Admin | Người quản trị hệ thống , thao tác với chắc năng quản lý lớp học và tạo lớp học phần |

Bảng 3-1 Danh sách tác nhân và mô tả

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

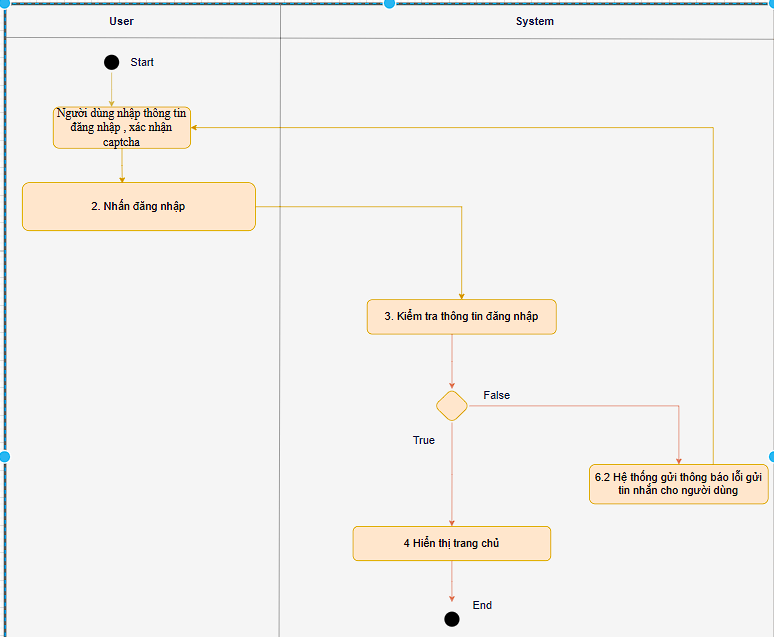
| **ID** | **Tên Use case** |
| --- | --- |
| UC01 | Đăng nhập |
| UC02 | Đăng xuất |
| UC03 | Đăng ký môn học |
| UC04 | Xem danh sách đã đăng ký |
| UC05 | Xem kết quả học tập |
| UC06 | Xem học phần đã đăng ký |
| UC07 | Xem thời khóa biểu |
| UC08 | Tạo lớp học |
| UC09 | Xem Thông tin lớp học |
| UC10 | Xoá lớp học |
| UC11 | Đổi mật khẩu |
| UC12 | Đăng ký xét tốt nghiệp |
| UC13 | Xem thông tin bằng cấp và thông tin cựu sinh viên |
| UC14 | Xem chi tiết quá trình học tập |

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

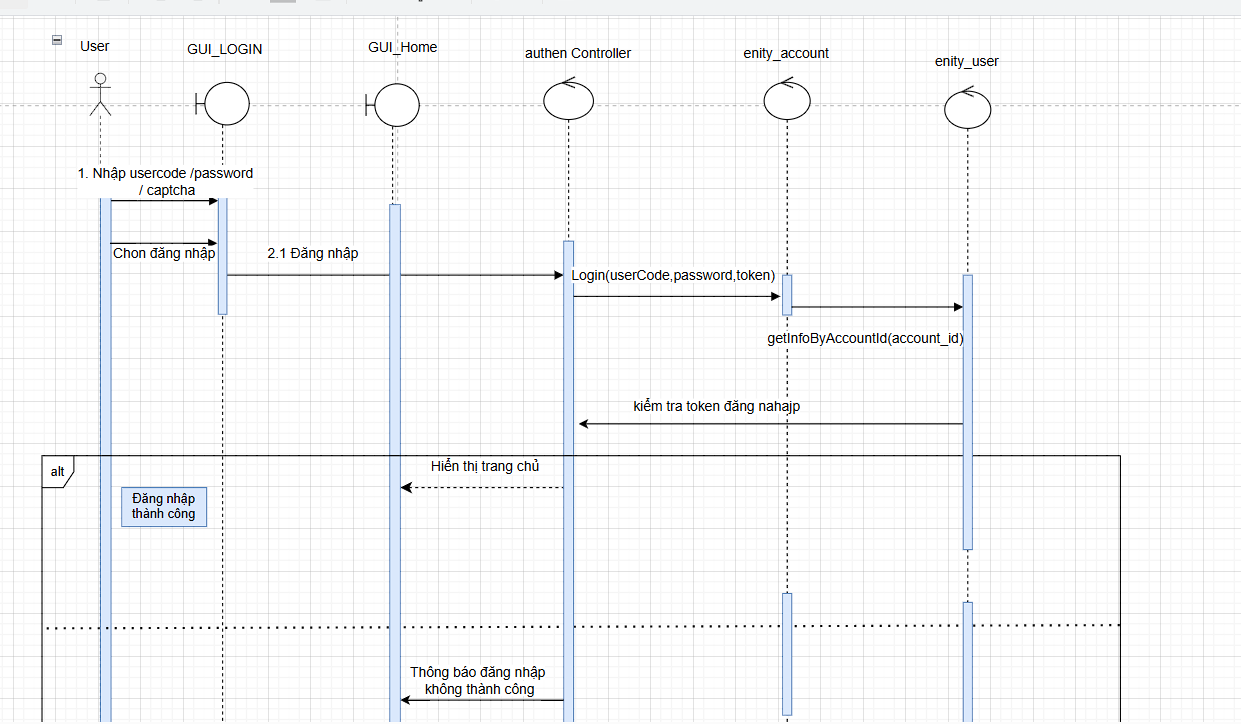
### Tình huống hoạt động

* + - 1. **Đặc tả use case “Đăng nhập”**

| − **Tên use case**: Đăng nhập | |
| --- | --- |
| − **Mô tả sơ lược:** Đăng nhập giúp sinh viên vào tài khoản sinh viên của mình | |
| − **Actor chính**: User , Admin | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống | |
| − **Hậu điều kiện (Post-condition): Đăng nhập vào được hệ thống** | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. 1. Người dùng nhập thông tin đăng nhập , xác nhận captcha  (userCode, password) |  |
| 2. Nhấp vào đăng nhập | 4. Hệ thống kiểm tra thông tin đăng nhập |
|  | 1. Hệ thống hiển thị trang chủ |
| − **Luồng sự kiện thay thế (alternate flow):** | |
|  | 3.1 Hệ thống kiểm tra đăng nhập sai |
|  | 3.2 Hiển thị thông báo sai mật khẩu hoặc số điện thoại hoặc chưa xác nhận captcha |
| 3.3 Quay lại bước 1 |  |

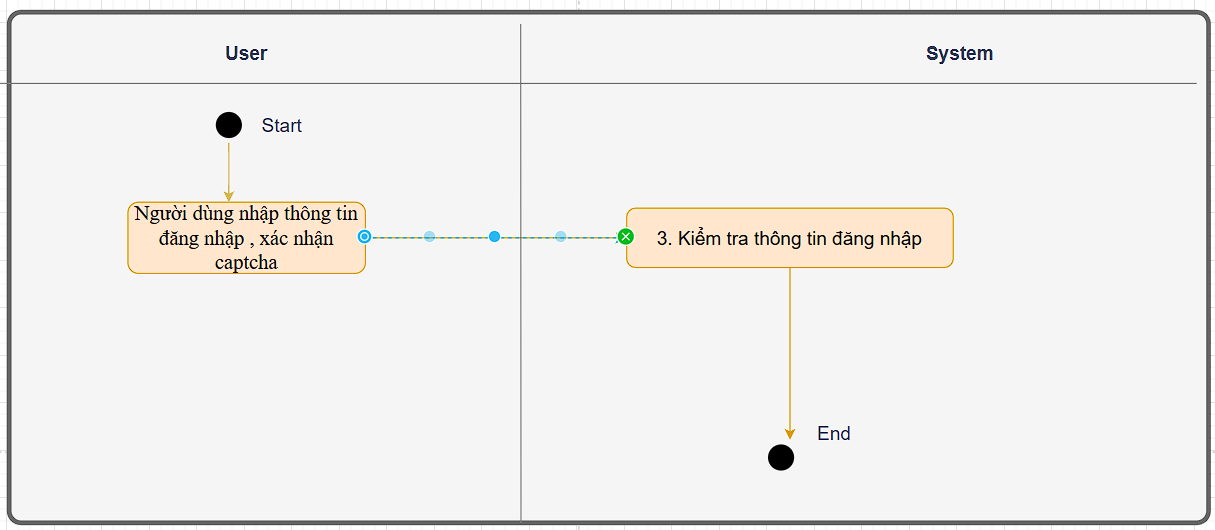
****

Hình Sequence diagram của đăng nhập

****

* + - 1. **Đặc tả use case “Đăng xuất”**

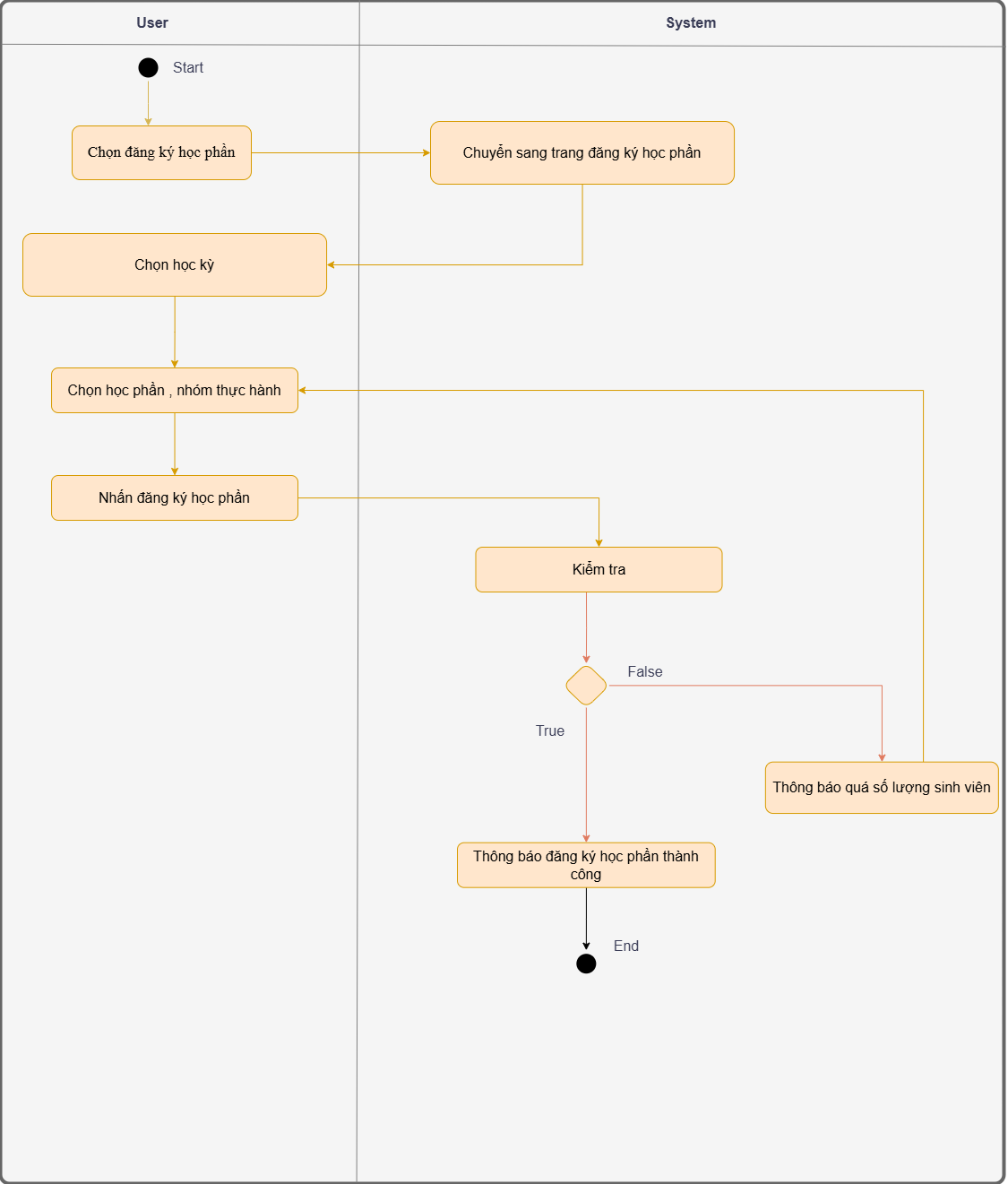
| − **Tên use case**: Đăng xuất | |
| --- | --- |
| − **Mô tả sơ lược:** Sinh viên or Admin đăng xuất khỏi tài khoản sinh viên | |
| − **Actor chính**:User, Admin | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):**Tài khoản tồn tại trong hệ thống | |
| − **Hậu điều kiện (Post-condition):**  Đã đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn đăng xuất | 2. Hệ thống chuyển về trang đăng nhập |
| − **Luồng sự kiện thay thế (alternate flow):** | |



Hình Activity diagram của nhắn tin

* + - 1. **Đặc tả use case “Đăng ký môn học ”**

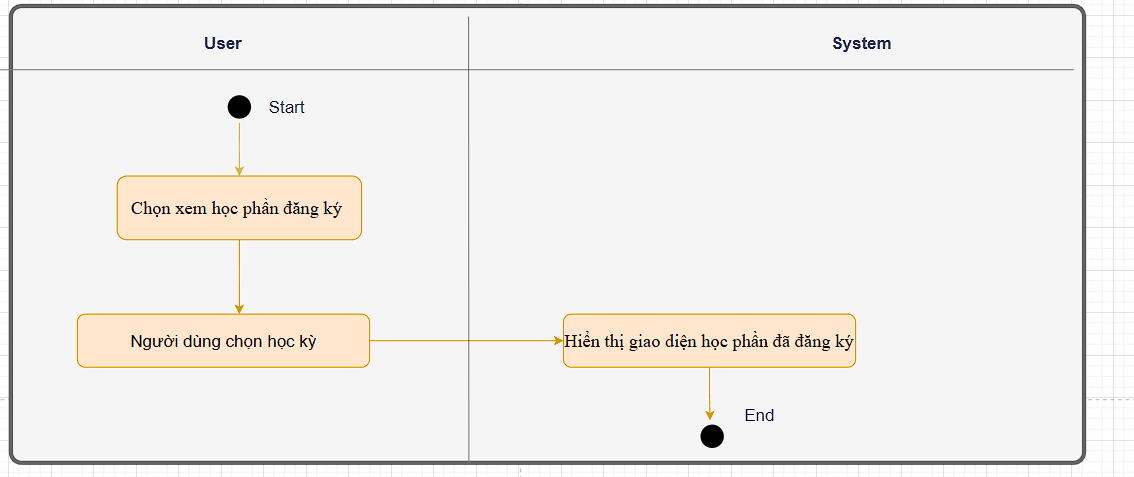
| − **Tên use case**: Đăng ký môn học | |
| --- | --- |
| − **Mô tả sơ lược:** Sinh viên đăng ký học phần | |
| − **Actor chính**: User | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , đã đăng nhập | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn đăng ký học phần | 2. Hệ thống chuyển đến trang đăng ký học phần |
| 3. Người dùng chọn học kỳ đăng ký |  |
| 4. Người dùng chọn học phần đăng ký, chọn lớp thực hành |  |
| 5. Nhấn đăng ký môn học | 6.Hệ thống kiểm tra |
|  | 7.Thông báo đăng ký học phần thành công |
| − **Luồng sự kiện thay thế (alternate flow):** | |
|  | 7.1 Hệ thống báo quá số lượng đăng ký |
| 7.2 Quay lại bước 4 |  |
|  | 7.1 Hệ thống thông báo quá số tín chỉ học kỳ |

****

Hình activity ram của đăng ký môn học

* + - 1. **Đặc tả use case “Xem danh sách đã đăng ký ”**

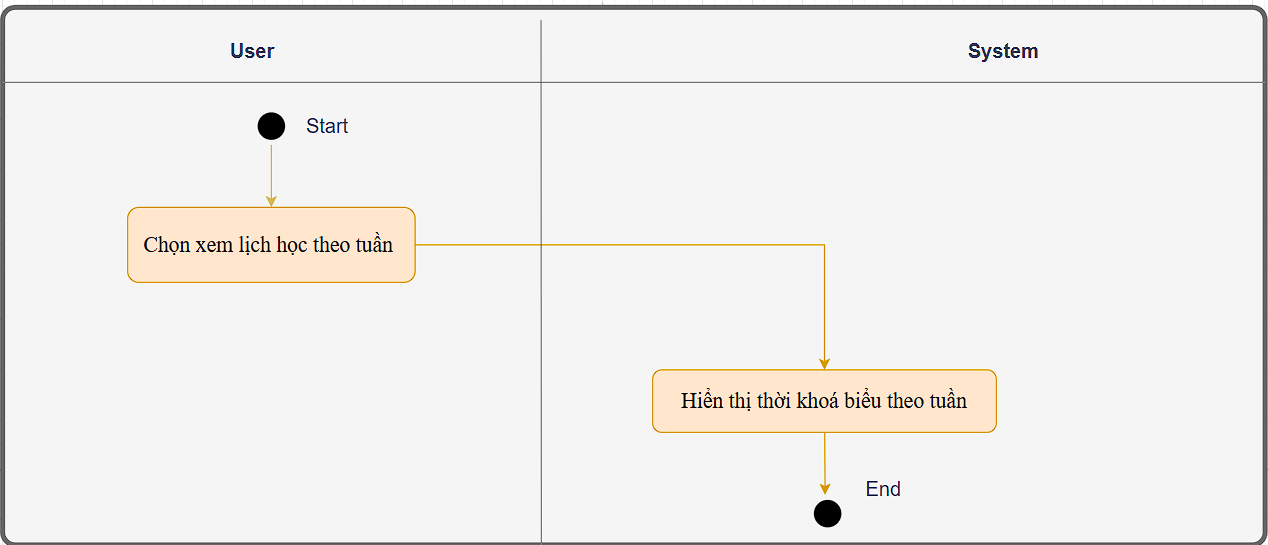
| − **Tên use case**: Xem danh sách đã đăng ký | |
| --- | --- |
| − **Mô tả sơ lược:** Sinh viên xem danh sách đã đăng ký | |
| − **Actor chính**: User | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , đã đăng nhập , | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn xem học phần đã đăng ký | 3. Hiển thị giao diện học phần đã đăng ký |
| 2. Người dùng chọn học kỳ |  |
| − **Luồng sự kiện thay thế (alternate flow):** | |

****

Hình activity ram xem học phần đã đăng ký

* + - 1. **Đặc tả use case “Xem thời khóa biểu ”**

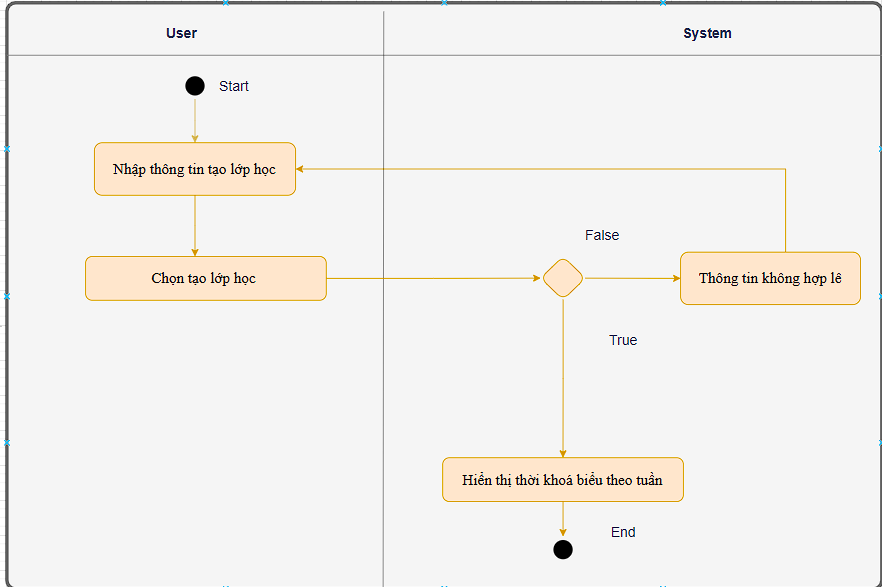
| − **Tên use case**: Xem thời khóa biểu | |
| --- | --- |
| − **Mô tả sơ lược:** Sinh viên xem thời khóa biểu | |
| − **Actor chính**: User | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , đã đăng nhập , | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn xem lịch học theo tuần | 2.Hệ thống hiển thị thời khoá biểu theo tuần |
| − **Luồng sự kiện thay thế (alternate flow):** | |

****

Hình activity ram xem thời khóa biểu

* + - 1. **Đặc tả use case “Tạo lớp học ”**

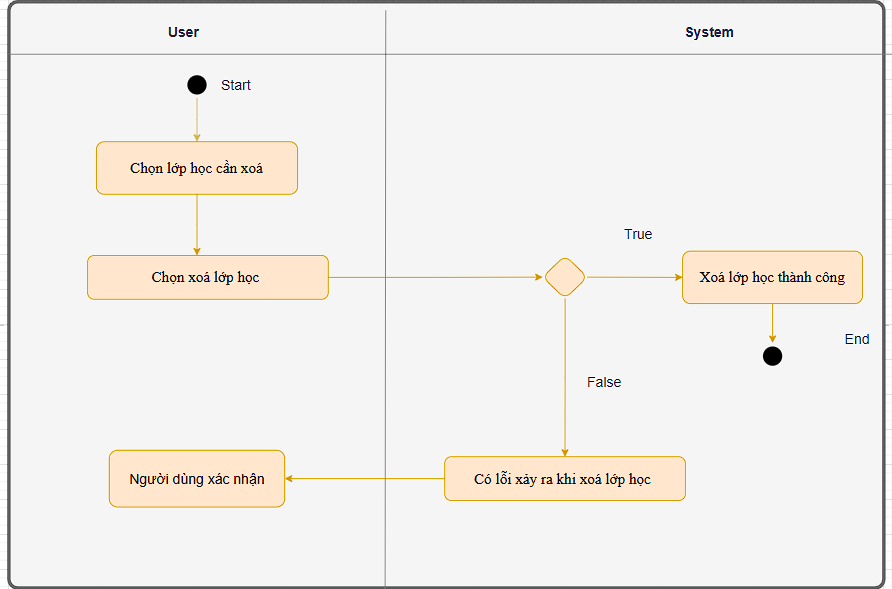
| − **Tên use case**: Tạo lớp học | |
| --- | --- |
| − **Mô tả sơ lược:** Admin tạo lớp học phần để sinh viên đăng ký | |
| − **Actor chính**: Admin | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , có quyền admin | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Admin nhập thông tin lớp học |  |
| 2.Admin bấm submit tạo lớp học | 3. Tạo lớp học thành công |
| − **Luồng sự kiện thay thế (alternate flow):** | |
|  | 3.1 Thông tin không hợp lệ |
| 2.1 Admin nhập lại thông tin lớp học |  |

****

Hình activity ram tạo lớp học

* + - 1. **Đặc tả use case “Xóa lớp học ”**

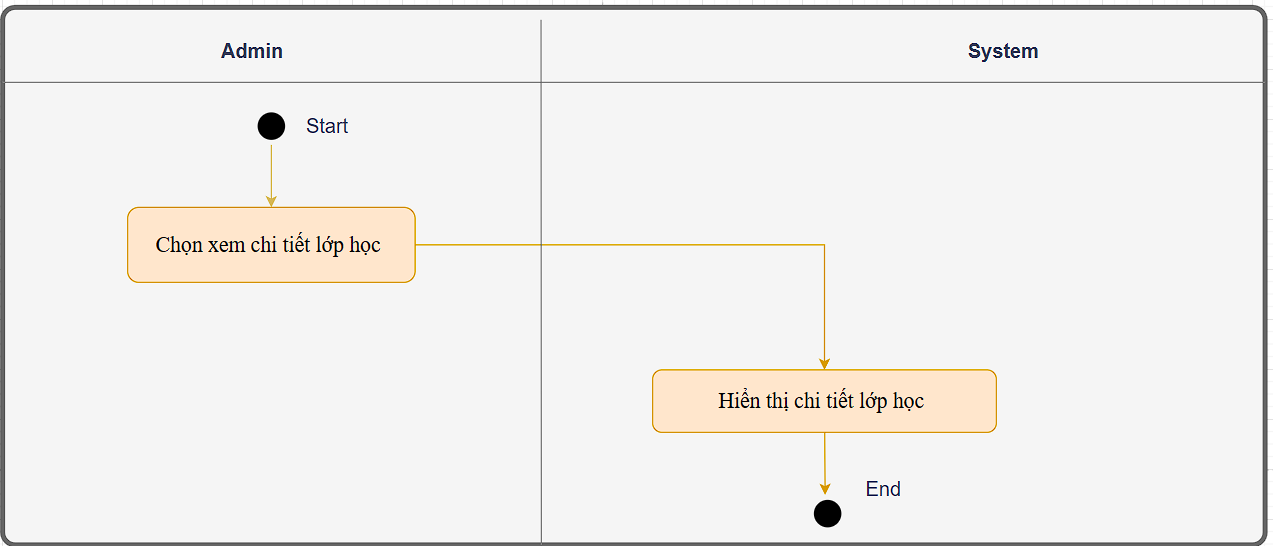
| − **Tên use case**: Xóa lớp học | |
| --- | --- |
| − **Mô tả sơ lược:** Admin chọn xóa lớp học | |
| − **Actor chính**: Admin | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , có quyền admin | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Admin chọn lớp học cần xoá |  |
| 2.Admin bấm submit xóa lớp học | 3. Xóa lớp học thành công |
| − **Luồng sự kiện thay thế (alternate flow):** | |
|  | 3.1 Xoá không thành công |
| 2.1 Admin xác nhận thông tin |  |

****

Hình activity ram xoá lớp học

* + - 1. **Đặc tả use case “Xem thông tin lớp học ”**

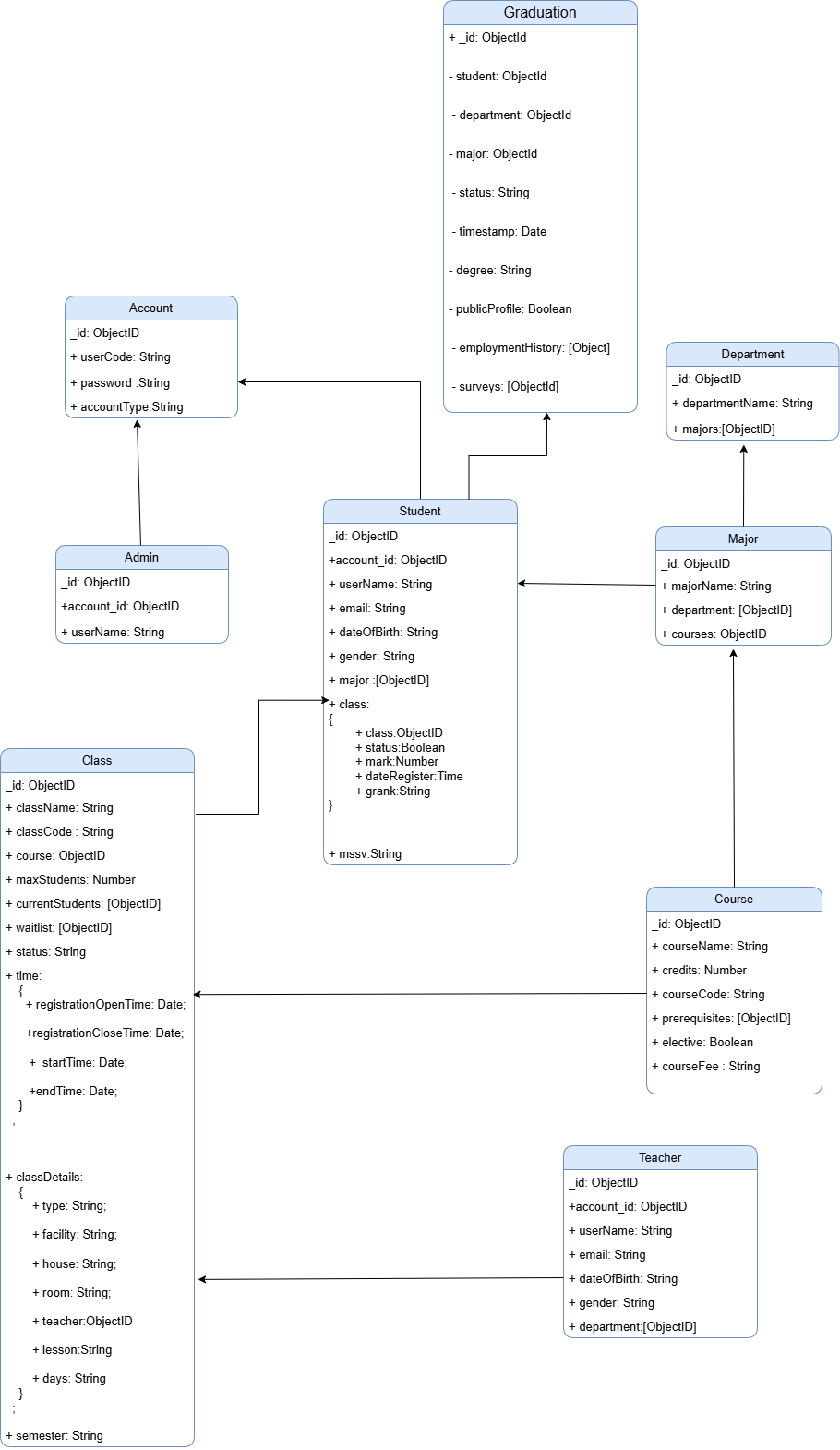
| − **Tên use case**: Xem thông tin lớp học | |
| --- | --- |
| − **Mô tả sơ lược:** Admin xem thông tin chi tiết lớp học | |
| − **Actor chính**: Admin | |
| − **Actor phụ**:Không | |
| − **Tiền điều kiện (Pre-condition):** Tài khoản tồn tại trong hệ thống , có quyền admin | |
| − **Hậu điều kiện (Post-condition):** Đăng nhập vào được hệ thống | |
| − **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Admin chọn xem chi tiết lớp học | 2.Hệ thống hiển thị thông tin chi tiết lớp h c |
| − **Luồng sự kiện thay thế (alternate flow):** | |

****

Hình activity xem chi tiết lớp học

## Class diagram

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



# : HIỆN THỰC

## Cấu hình phần cứng, phần mềm

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

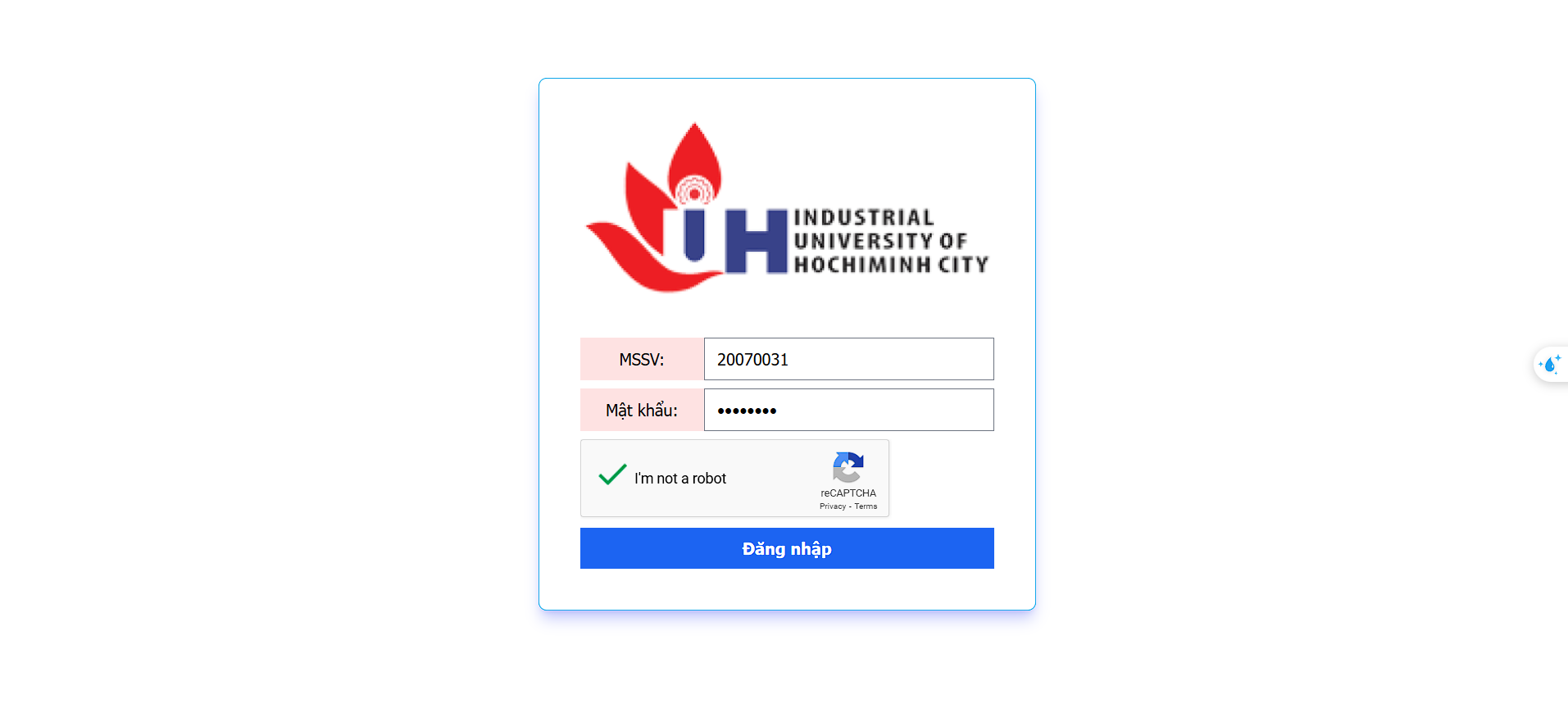
* Phần mềm cơ sở dữ liệu: Sử dụng hệ quản trị cơ sở dữ liệu (DBMS) MongoDB để lưu trữ và quản lý dữ liệu người dùng và thông tin ứng dụng.
* Phần mềm ứng dụng: Triển khai và cấu hình các ứng dụng server-side Node.js xử lý logic nghiệp vụ của ứng dụng chat và mạng xã hội.

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

## Giao diện của hệ thống

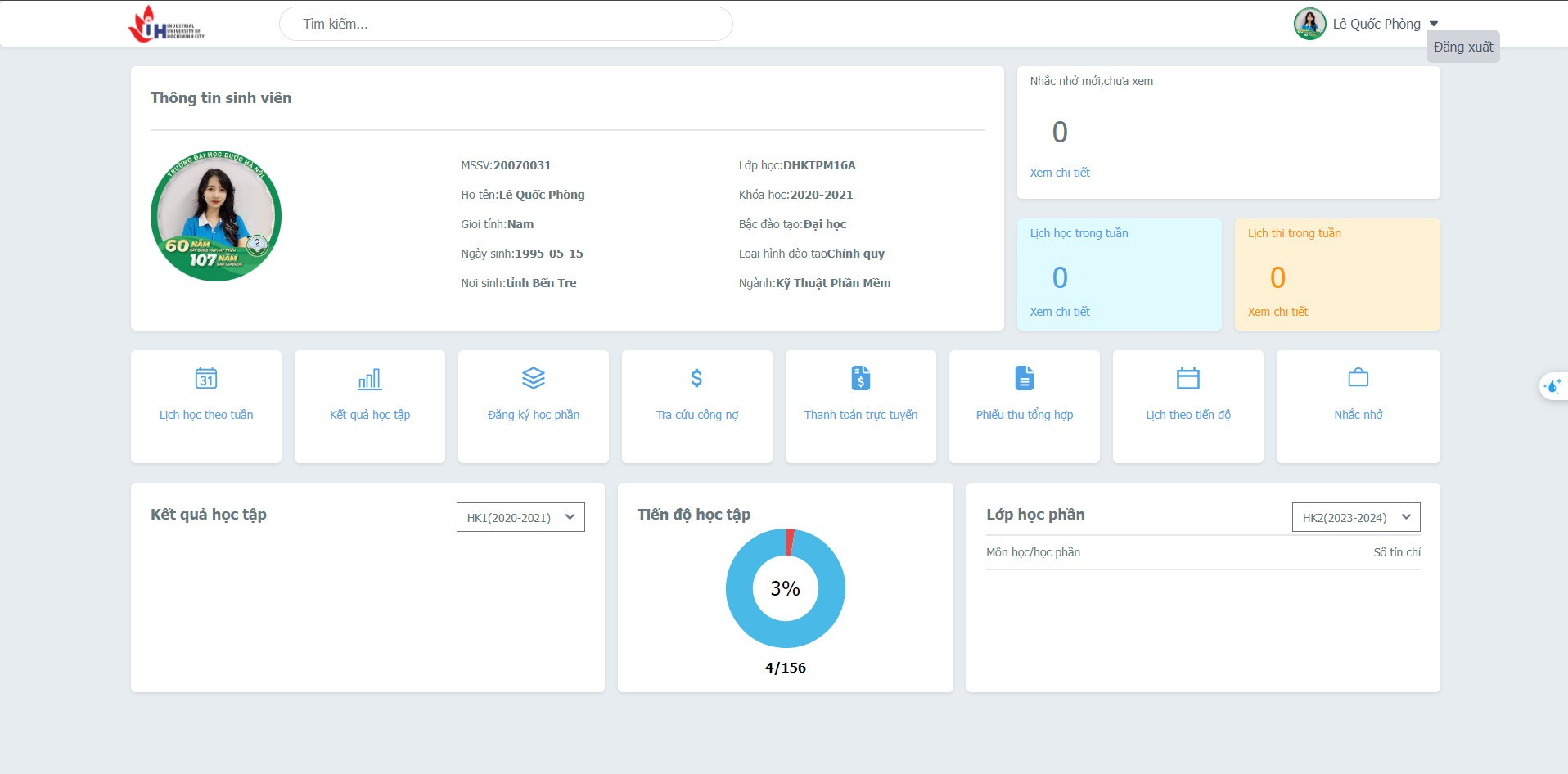
### Giao diện trên website

#### Màn hình đăng nhập



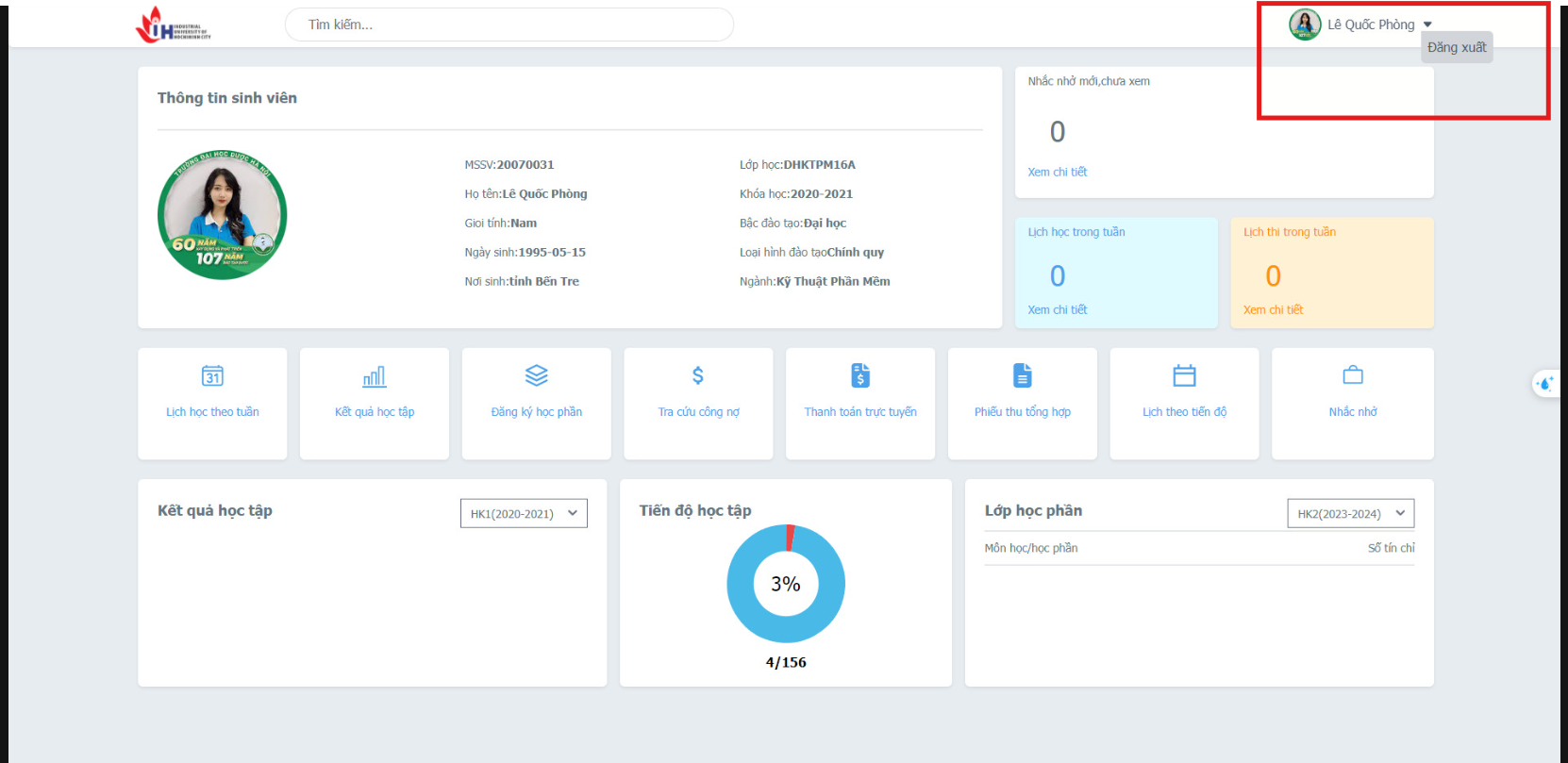
Nhập thông tin user Code , password và xác nhận captcha

#### Màn hình trang chủ

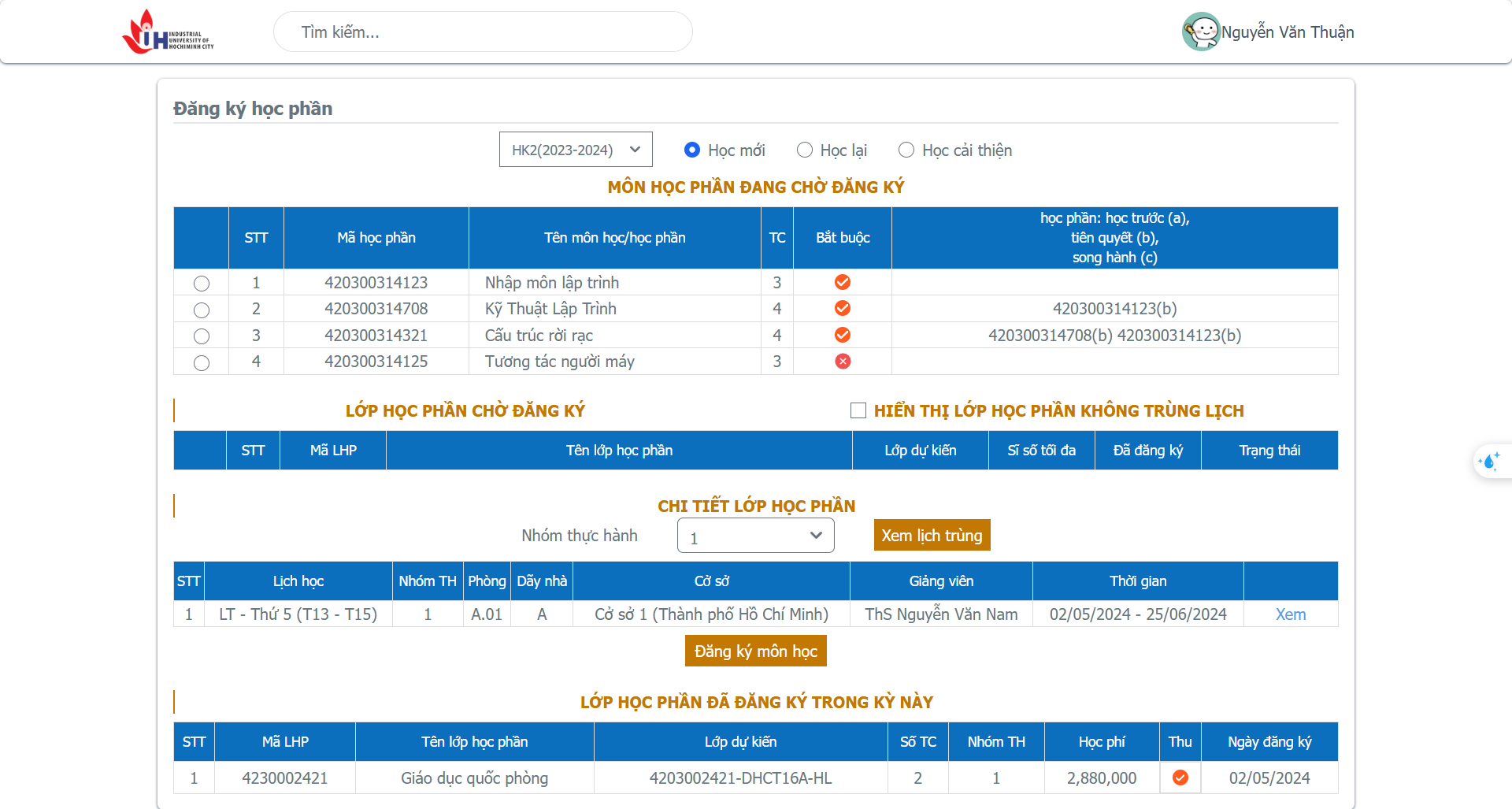


Khi đã đăng nhập vào ứng dụng thành công, ta sẽ thấy được màn hình chờ giao diện của hệ thống

#### Màn hình đăng xuất

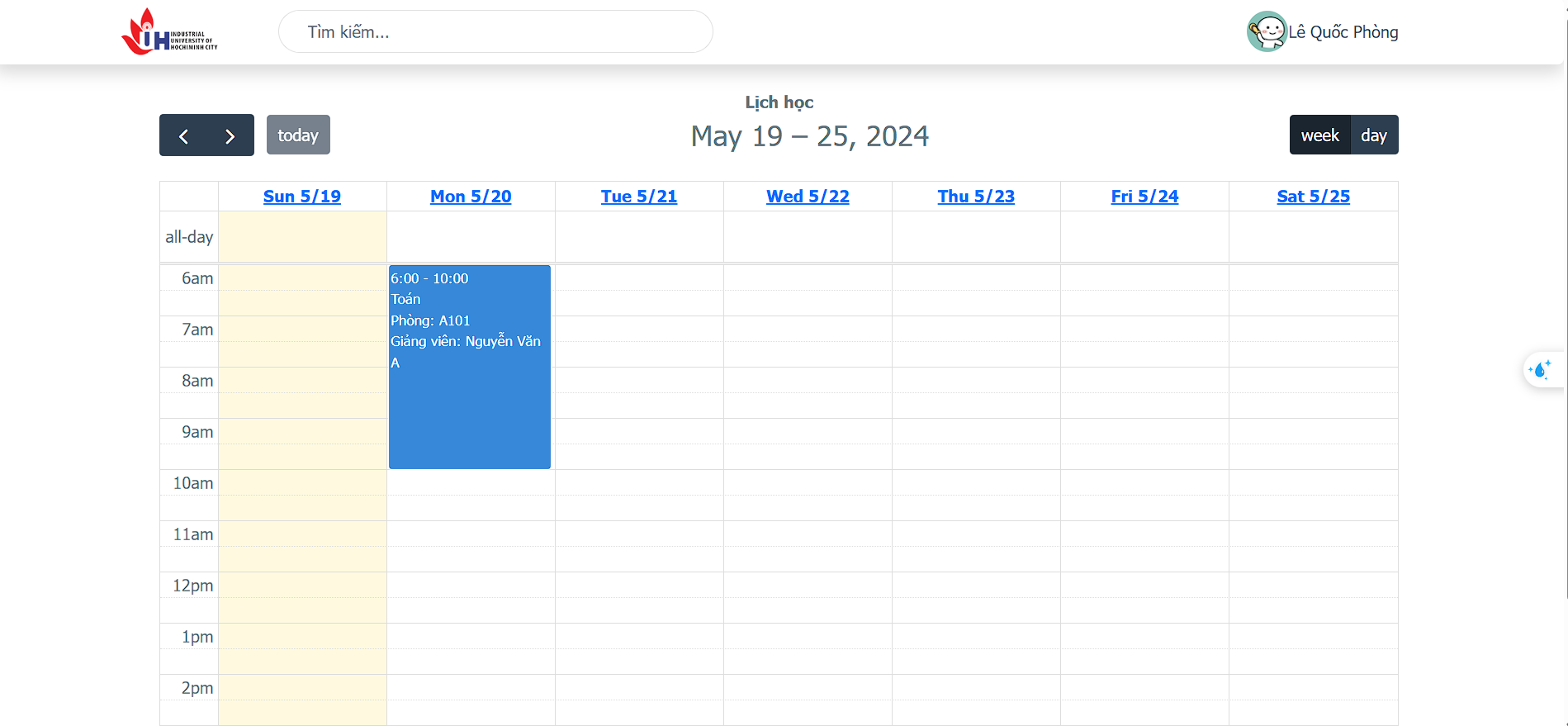


#### Màn hình đăng ký học phần

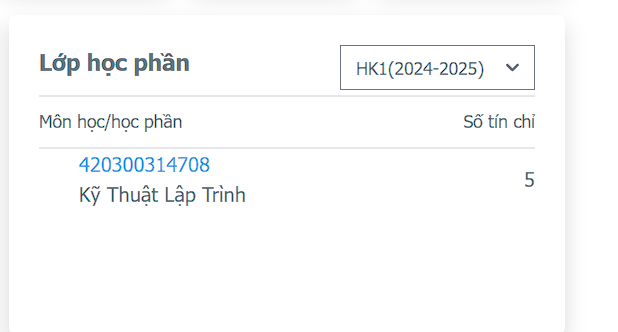


Màn hình chat khi ta muốn nhắn tin, liên lạc với ai đó thì bấm vào tên người đó để hiện ra khung chat để trò chuyện

#### Màn hình xem thời khoá biểu

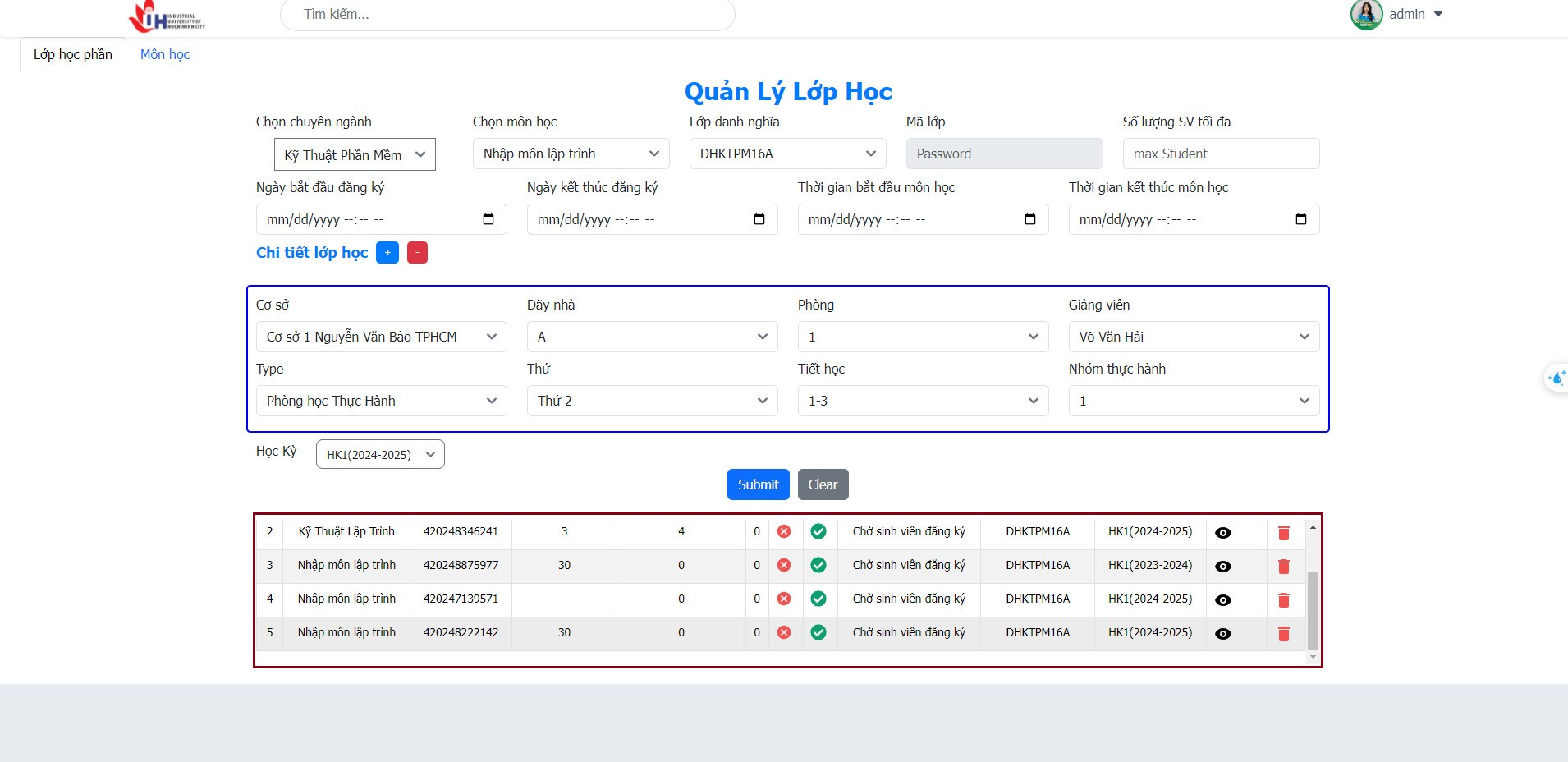


#### Màn hình xem danh sách đã đăng ký



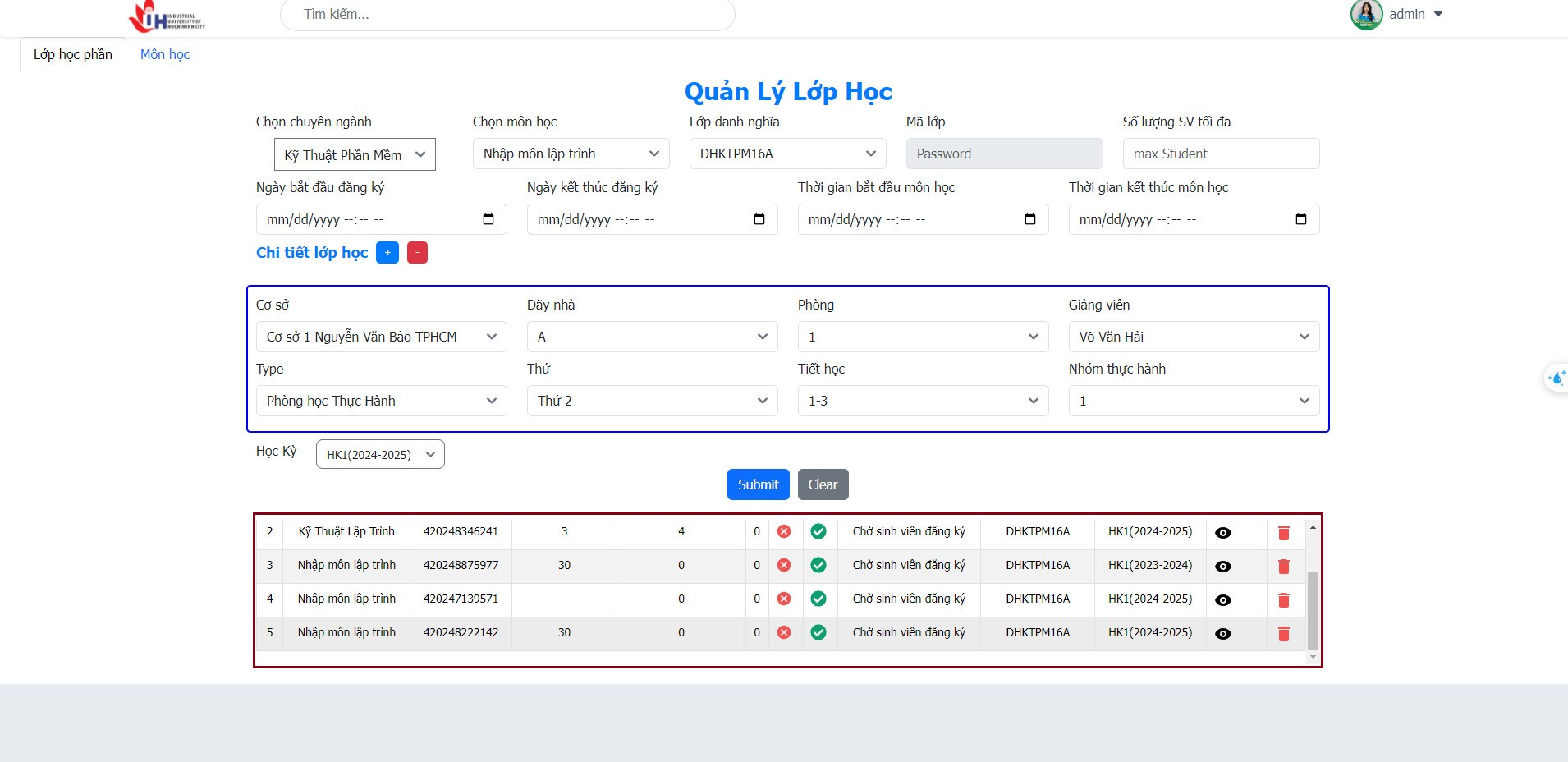
Màn hình xem danh sách đã đăng ký

#### Màn hình tạo lớp học



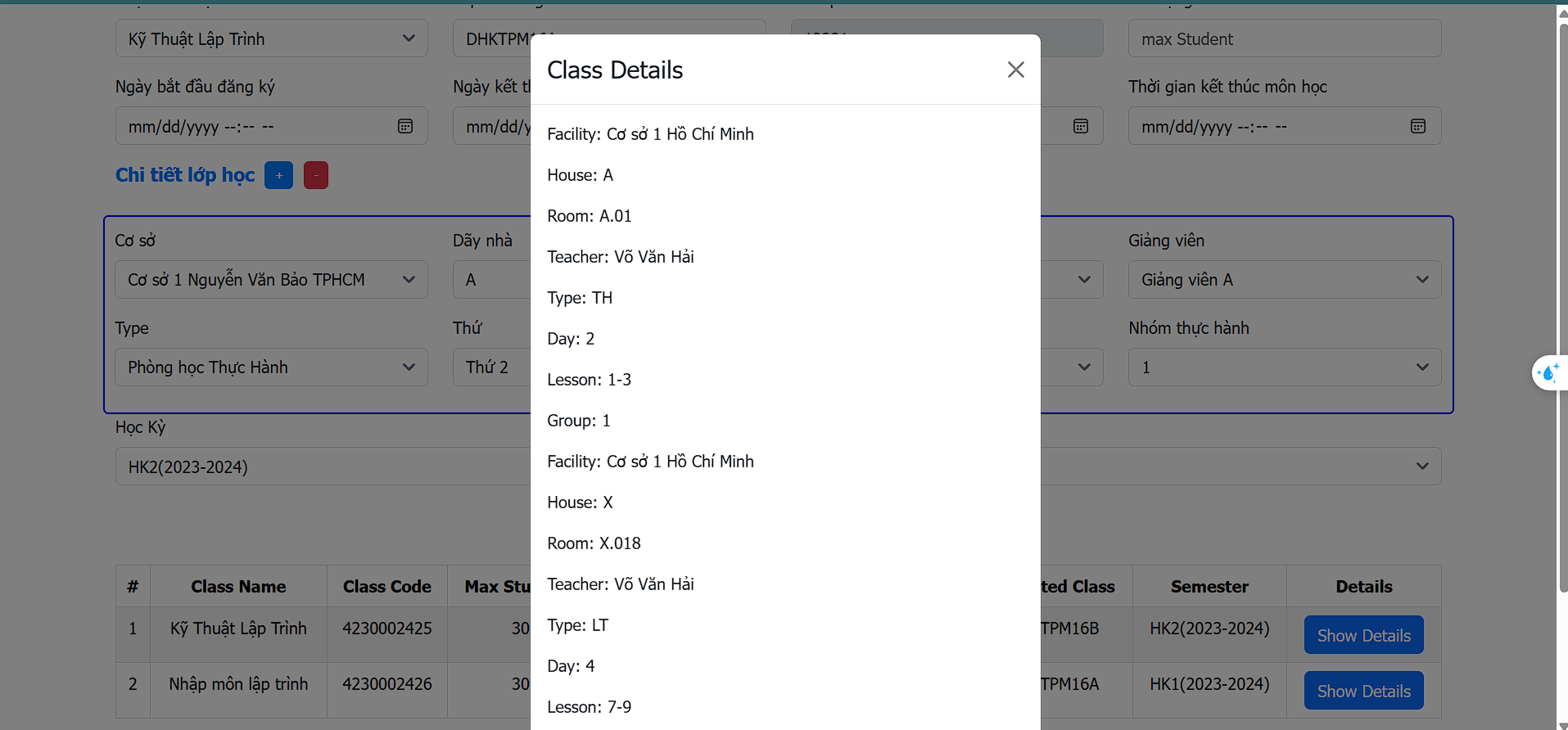
Màn hình xuất hiện tạo lớp học

#### Màn hình xoá lớp học



Màn hình xuất hiện xóa lớp học

#### Màn hình xem chi tiết lớp học



## Kế hoạch và hiện thực kiểm thử hệ thống

### Kế hoạch kiểm thử

### Kiểm thử hệ thống

# : KẾT LUẬN

## Kết quả đạt được

* Đăng nhập , đăng xuất ra
* Quản Lý Đăng ký học phần (đăng ký môn học , xem danh sách đã đăng ký )
* Xem thời khóa biểu
* Quản lý lớp học ( Xem thông tin lớp học , Tạo lớp học , xóa lớp học )

## Hạn chế của đồ án

## Tốc độ 1 số chức năng chưa được tối ưu.

* Chưa kết nối đồng bộ được mobile và web

## Hướng phát triển

* Phát triển tính năng cho GUI giảng viên đăng ký
* Bổ sung các chức năng: đổi ảnh đại diện ,...

# 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>