**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**quản lí lớp học tín chỉ**

***Nhóm 07 - Sinh viên thực hiện***

1. Võ Minh Toàn– 20084581
2. Trần Thế Thành-20083661
3. Lê Cao Mỹ-20088891

# MỤC LỤC

[MỤC LỤC 1](#_Toc164786500)

[DANH MỤC CÁC HÌNH VẼ 3](#_Toc164786501)

[DANH MỤC CÁC BẢNG BIỂU 4](#_Toc164786502)

[CHƯƠNG 1 : GIỚI THIỆU 5](#_Toc164786503)

[1.1 Tổng quan 5](#_Toc164786504)

[1.2 Mục tiêu đề tài 5](#_Toc164786505)

[1.3 Phạm vi đề tài 6](#_Toc164786506)

[1.3.1.1 Chức năng dành cho người dùng: 6](#_Toc164786507)

[1.3.1.2 *Chức năng dành cho người quản trị*: 6](#_Toc164786508)

[1.4 Mô tả yêu cầu chức năng 6](#_Toc164786509)

[1.4.1.1 Phân tích yêu cầu của hệ thống 6](#_Toc164786510)

[1.4.1.2 *Mô tả hoạt động của hệ thống* 7](#_Toc164786511)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 8](#_Toc164786512)

[2.1 Node.js 8](#_Toc164786513)

[2.2 React.js 9](#_Toc164786514)

[2.3 MongoDB 10](#_Toc164786515)

[2.4 Kiến trúc 11](#_Toc164786516)

[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ 20](#_Toc164786517)

[3.1 Phân tích yêu cầu bằng UML 20](#_Toc164786518)

[3.1.1 Usecase tổng quát 20](#_Toc164786519)

[3.1.2 Danh sách tác nhân và mô tả 23](#_Toc164786520)

[3.1.3 Danh sách các tình huống hoạt động (Use cases) 24](#_Toc164786521)

[3.1.4 Tình huống hoạt động 26](#_Toc164786522)

[3.1.4.1 UC01\_Đăng nhập 26](#_Toc164786523)

[3.2 Class diagram 41](#_Toc164786524)

[3.3 Kiến trúc layer 42](#_Toc164786525)

[CHƯƠNG 4 : HIỆN THỰC 45](#_Toc164786526)

[4.1 Giao diện của hệ thống 45](#_Toc164786527)

[4.1.1 Giao diện web 45](#_Toc164786528)

[CHƯƠNG 5 : KẾT LUẬN 46](#_Toc164786529)

[5.1 Kết quả đạt được 46](#_Toc164786530)

[5.2 Hạn chế của đồ án 46](#_Toc164786531)

[5.3 Hướng phát triển 46](#_Toc164786532)

[TÀI LIỆU THAM KHẢO 47](#_Toc164786533)

# DANH MỤC CÁC HÌNH VẼ

[Hình 2‑1 Node.js development 1](#_Toc14693413)

# DANH MỤC CÁC BẢNG BIỂU

[Bảng 3‑1 Danh sách tác nhân và mô tả 1](#_Toc14682192)

[Bảng 3‑2 Danh sách các tình huống trong hệ thống 1](#_Toc14682193)

# : GIỚI THIỆU

## Tổng quan

Mô tả về hệ thống quản lý lớp học tín chỉ của một trường đại học. Hệ thống này được thiết kế để cung cấp cho sinh viên một cách tiếp cận dễ dàng và linh hoạt đối với các khóa học, đồng thời hỗ trợ họ trong việc quản lý lịch trình học tập và theo dõi tiến trình học tập của mình. Ngoài ra, hệ thống còn cung cấp các tính năng hỗ trợ cho việc chuẩn bị và xin cấp bằng tốt nghiệp sau khi hoàn thành chương trình học. Điều này giúp tạo ra một môi trường học tập hiệu quả và đảm bảo sự linh hoạt và thuận tiện cho sinh viên trong quá trình học tập và phát triển sự nghiệp sau này.

## Mục tiêu đề tài

Mục tiêu của hệ thống quản lý lớp học tín chỉ là cung cấp cho sinh viên một trải nghiệm học tập linh hoạt và hiệu quả. Hệ thống nhằm giúp sinh viên dễ dàng đăng ký các khóa học phù hợp với ngành học của mình, đồng thời giúp họ theo dõi và quản lý lịch trình học tập một cách thuận tiện. Mục tiêu của hệ thống không chỉ là hỗ trợ quá trình đăng ký học phần mà còn là cung cấp các tiện ích như xem thông tin học tập cá nhân, thời khóa biểu, và các dịch vụ khác để tối ưu hóa trải nghiệm học tập của sinh viên. Đồng thời, hệ thống cũng nhằm tạo điều kiện thuận lợi cho sinh viên sau khi hoàn thành chương trình học để xin cấp bằng tốt nghiệp và tiếp tục phát triển sự nghiệp. Phần mở rộng của hệ thống có thể bao gồm việc phân tích dữ liệu học tập để cung cấp cá nhân hóa hơn cho sinh viên, cũng như kết nối sinh viên với cơ hội thực tập và việc làm sau này.Với nền tảng website người dùng có thể :

* Xem và chỉnh sửa thông tin cá nhân.
* Xem lịch học
* Xem kết quả học tập
* Tra cứu công nợ
* Lịch học theo tiến độ
* Tiến trình học tập
* Xem Chương trình khung
* Đăng kí học phần

## Phạm vi đề tài

#### Chức năng dành cho người dùng:

* SinhViên xem thông tin cá nhân, đăng ký học phần, xem lịch học, xem kết quả học tập, tra cứu công nợ, xem lịch học theo tiến độ, tiến trình học tập, xem chương trình khung.
* Có giao diện trực quan và thân thiện, dễ tương tác và sử dụng thuận tiện.
* Có thể thay đổi : Thông tin tài khoản,và thông tin cá nhân,…

#### **Chức năng dành cho người quản trị**:

Ngoài các yêu cầu giống như của khách hàng, thì hệ thống phải đảm bảo những yêu cầu sau:

* Thêm, xóa, chỉnh sửa học phần.

## Mô tả yêu cầu chức năng

#### Phân tích yêu cầu của hệ thống

- Người dùng có thể thay đổi thông tin cá nhân như,tên, giới tính…

- Người dùng thực hiện đăng ký học phần trên web, hệ thống sẽ tiếp nhận và phản hồi cho đối tượng đăng kí học phần.

- Người dùng có thể hủy đăng ký học phần( nếu hệ thống cho phép).

- Từ mô tả trên, có thể đưa ra yêu cầu của hệ thống với hai đối tượng chính tương tác với hệ thống như sau:

• Đối với hệ thống:

- Thêm, sửa, xóa các học phần

- Tìm kiếm thông tin của học phần

- Quản lý thông tin của người dùng

• Đối với người dùng:

- Đăng kí học phần,xem lịch học,xem điểm,tra cứu công nợ, xem lịch học theo tiến độ, tiến trình học tập, xem chương trình khung, thanh toán trực tuyến, xem và chỉnh sửa thông tin cá nhân.

- Hủy đăng ký học phần

#### Mô tả hoạt động của hệ thống

1. **Biên chế sinh viên theo ngành học**: Hệ thống phải tự động phân loại sinh viên vào các khoa và ngành học tương ứng dựa trên thông tin đăng ký ban đầu của sinh viên.
2. **Chọn môn học cho học kỳ**: Nhà trường sẽ chọn một danh sách các môn học cần mở cho mỗi ngành học và đưa vào hệ thống.
3. **Đăng ký học phần**: Sinh viên sẽ đăng nhập vào hệ thống và xem danh sách các môn học có sẵn để đăng ký trong học kỳ đó. Họ chọn các môn học phù hợp và tiến hành đăng ký.
4. **Xác nhận đăng ký**: Để tránh trường hợp đăng ký nhầm lẫn hoặc đăng ký xong rồi hủy bỏ, sinh viên phải xác nhận đăng ký trước khi hệ thống chấp nhận.
5. **Xử lý đăng ký**: Hệ thống sẽ xử lý đăng ký của sinh viên, kiểm tra số tín chỉ đã đăng ký và tính toán số lượng sinh viên trong mỗi lớp học.
6. **Quản lý danh sách dự bị**: Nếu các lớp học đã đầy, sinh viên sẽ được đưa vào danh sách dự bị. Hệ thống sẽ cân nhắc mở thêm lớp nếu cần.
7. **Gửi thông báo đăng ký**: Sau khi đăng ký thành công, hệ thống sẽ gửi email thông báo xác nhận đăng ký và yêu cầu đóng học phí cho sinh viên.
8. **Xem thông tin học tập**: Sinh viên có thể truy cập hệ thống để xem thông tin học tập cá nhân như số tín chỉ đã đạt, số môn đã học, điểm số, và điểm trung bình tích lũy.
9. **Xem lịch học:** Sinh viên có thể truy cập vào hệ thống để xem lịch học của mình cho từng học kỳ, bao gồm thời gian, địa điểm, và thông tin liên quan đến các lớp học.
10. **Đăng ký xét tốt nghiệp**: Khi đạt đủ số tín chỉ theo yêu cầu, sinh viên có thể đăng ký xét tốt nghiệp. Hệ thống sẽ kiểm tra và cấp bằng tốt nghiệp cho sinh viên nếu đủ điều kiện. Thông tin về bằng cấp sẽ được công khai trên trang web của trường.

Top of Form

# : CƠ SỞ LÝ THUYẾTTop of Form

## Node.js



Hình 2‑1 Node.js development

**Tổng quan về Node.js:**

Node.js là một nền tảng phát triển độc lập, được xây dựng dựa trên môi trường thực thi JavaScript của Google Chrome. Nền tảng này cho phép phát triển các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

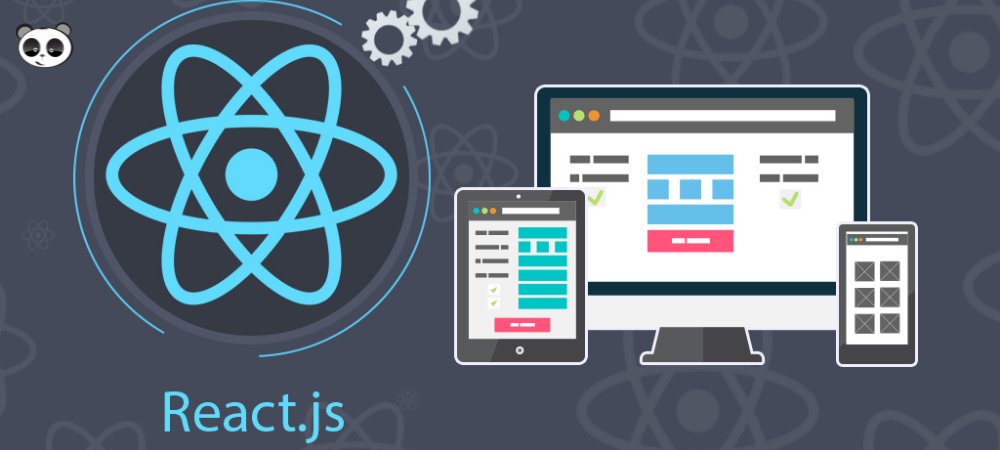
**Ưu điểm của Node.js:**

1. **Đơn ngôn ngữ:** Node.js sử dụng JavaScript, một trong những ngôn ngữ lập trình phổ biến nhất, cho phép phát triển ứng dụng đồng nhất từ client đến server.
2. **Kiến trúc hiệu quả:** Với kiến trúc hướng sự kiện và cơ chế non-blocking I/O, Node.js mang lại hiệu suất cao và khả năng xử lý đồng thời lớn, làm cho nó trở thành lựa chọn lý tưởng cho các ứng dụng cần xử lý nhiều kết nối cùng một lúc.
3. **Đa nền tảng:** Node.js có thể chạy trên nhiều hệ điều hành khác nhau như macOS, Windows và Linux, đảm bảo tính linh hoạt và dễ dàng triển khai trong môi trường phát triển đa dạng.
4. **Cộng đồng mạnh mẽ:** Có một cộng đồng lớn và tích cực hỗ trợ Node.js, với hàng ngàn mô-đun miễn phí có sẵn trên npm (Node Package Manager), giúp tăng tốc độ phát triển và cung cấp các giải pháp sẵn có cho nhiều vấn đề phức tạp.
5. **Miễn phí:** Node.js là một nền tảng mã nguồn mở, cung cấp miễn phí cho các nhà phát triển, giúp giảm chi phí phát triển phần mềm.

Nhờ vào những ưu điểm này, Node.js đã trở thành một trong những lựa chọn hàng đầu cho các nhà phát triển web và ứng dụng di động trên toàn thế giới.

Top of Form

## React.js



**Tổng quan về React:**

React là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng tương tác. Nó khuyến khích việc tạo ra các thành phần giao diện có thể tái sử dụng, và cho phép hiển thị dữ liệu thay đổi theo thời gian. Rất nhiều người sử dụng React làm phần giao diện người dùng (View) trong mô hình MVC (Model-View-Controller).

**Ưu điểm của React:**

1. **Sử dụng DOM ảo:** React sử dụng một cấu trúc dữ liệu gọi là DOM ảo (Virtual DOM) là một đối tượng JavaScript. Điều này cải thiện hiệu suất ứng dụng, vì DOM ảo JavaScript thường nhanh hơn DOM thực sự và giúp tránh trạng thái đồng bộ hóa tốn kém.
2. **Có thể sử dụng trên máy khách và máy chủ:** React có thể được sử dụng không chỉ trên phía máy khách (client-side) mà còn trên phía máy chủ (server-side), cung cấp khả năng rendering đa nền tảng và tối ưu hóa SEO.
3. **Các mẫu thành phần và dữ liệu cải thiện khả năng đọc:** React thúc đẩy việc sử dụng các mẫu thành phần (component-based) và quản lý dữ liệu một cách hiệu quả, giúp duy trì và phát triển các ứng dụng lớn một cách dễ dàng và hợp lý hơn.

Nhờ vào những ưu điểm này, React đã trở thành một trong những công cụ phát triển giao diện người dùng phổ biến nhất hiện nay, được sử dụng rộng rãi trong cả ứng dụng web và di động.

Top of Form

## MongoDB



**Tổng quan về MongoDB:**

MongoDB là một hệ thống quản lý cơ sở dữ liệu (DBMS) mã nguồn mở, được thiết kế với mô hình cơ sở dữ liệu hướng đối tượng. Đặc điểm nổi bật của MongoDB là khả năng lưu trữ dữ liệu một cách linh hoạt mà không yêu cầu phải tuân theo một cấu trúc nhất định.

**Ưu điểm của MongoDB:**

1. **Hiệu suất cao:** MongoDB được tối ưu hóa để đạt hiệu suất cao trong việc truy xuất và xử lý dữ liệu.
2. **Tính sẵn sàng cao - Nhân rộng:** MongoDB hỗ trợ tính năng nhân rộng tự động, giúp cân bằng tải và tăng khả năng chịu tải của hệ thống một cách linh hoạt.
3. **Khả năng mở rộng cao - Sharding:** MongoDB cho phép phân chia dữ liệu thành các phân đoạn nhỏ hơn (shards), giúp phân tải dữ liệu và cải thiện hiệu suất của hệ thống.
4. **Năng động:** MongoDB không yêu cầu một lược đồ cứng nhắc, cho phép thêm, xóa trường dữ liệu một cách linh hoạt mà không ảnh hưởng đến ứng dụng.
5. **Linh hoạt:** MongoDB cho phép dữ liệu được biểu diễn dưới dạng JSON hoặc BSON, đồng thời hỗ trợ các truy vấn phong phú và phức tạp.
6. **Dữ liệu không đồng nhất:** MongoDB cho phép lưu trữ dữ liệu không đồng nhất, tức là các bản ghi trong cùng một bảng không cần phải có cấu trúc giống nhau.
7. **Không joins:** MongoDB hạn chế việc sử dụng joins trong truy vấn, thay vào đó, nó sử dụng mô hình nhúng dữ liệu để tối ưu hóa hiệu suất.
8. **Phân phối được:** MongoDB có thể được triển khai trên nhiều nền tảng và môi trường, từ máy tính cá nhân đến các môi trường điện toán đám mây.

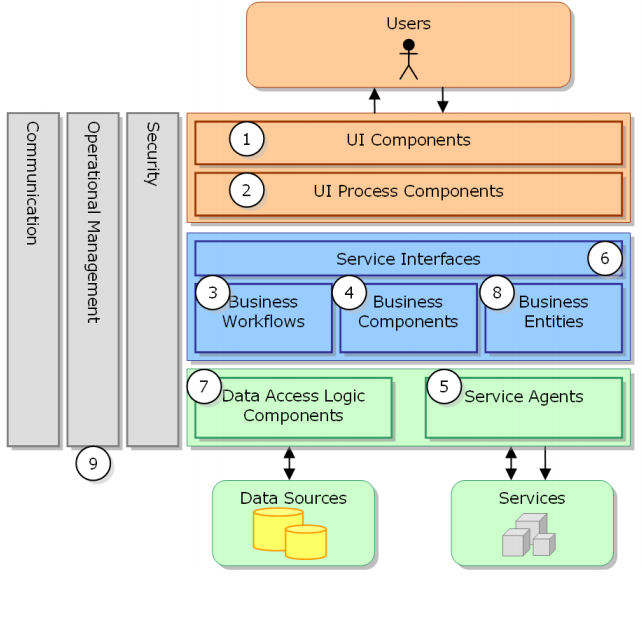
Nhờ vào những ưu điểm này, MongoDB trở thành một lựa chọn phổ biến cho các ứng dụng cần xử lý dữ liệu lớn và có tính linh hoạt cao.

## Kiến trúc 3 *layered architecture* ( Áp dụng vào project Nodejs)

1. **Kiến trúc 3 *layered architecture* là gì?**

**Mô hình 3 layered architecture**  hay còn được gọi là mô hình Three Layer(3-Layer), mô hình này ra đời nhằm phân chia các thành phần trong hệ thống, các thành phần cùng chức năng sẽ được nhóm lại với nhau và phân chia công việc cho từng nhóm để dữ liệu không bị chồng chéo và chạy lộn xộn.

* Mô hình này phát huy hiệu quả nhất khi bạn xây dựng một hệ thống lớn, việc quản lý code và xử lý dữ liệu lỗi dễ dàng hơn.
* Ưu điểm:
* Phân loại rõ ràng các lớp có các nhiệm vụ khác nhau.
* Từ đó ta có thể quản lý và maintain project tốt hơn.
* Dễ dàng phân loại các hành động tại Business.
* Dễ dàng phân loại các hàm truy xuất tại Database, phân loại hàm theo table,…
* Ứng dụng được cho các project lớn ở bên ngoài.



Mô hình 3-layer gồm có 3 phần chính:

Presentation Layer (GUI):

* Lớp này có nhiệm vụ chính là giao tiếp với người dùng. Nó gồm các thành phần giao diện ( winform, webform, …) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).

Business Logic Layer (BLL) :

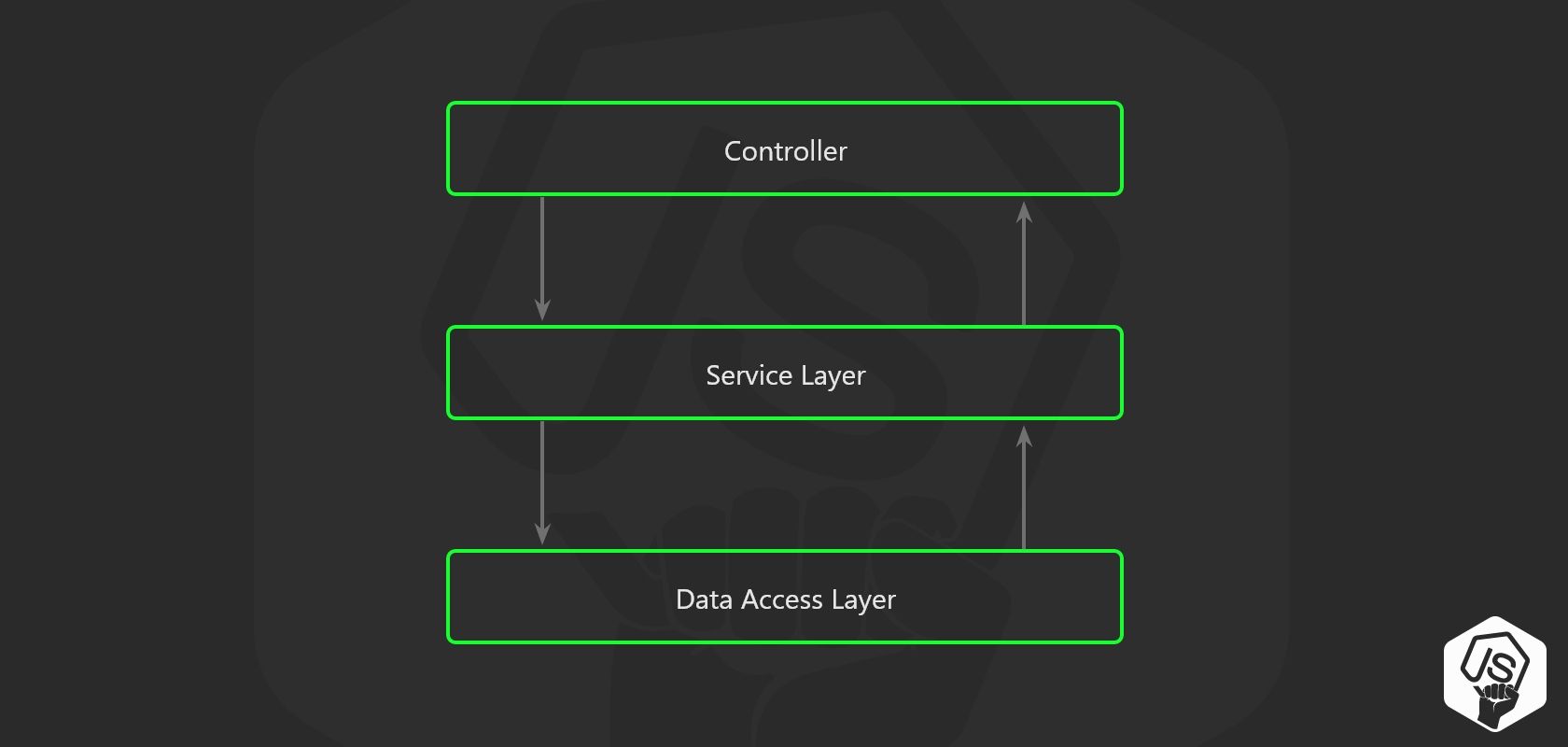
* Layer này phân ra 2 thành nhiệm vụ: Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.
* Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

Data Access Layer (DAL) :

* Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu ( tìm kiếm, thêm, xóa, sửa,…).

1. **Áp dụng kiến trúc 3 Layer Architecture vào project NodeJS**
   1. **Các vấn đề gặp phải khi sử dụng nền tảng NodeJS**

* Các framework nodejs phổ biết như Express cho phép chúng ta dễ dàng tạo ra Resful API xử lí các request từ phía client một cách nhanh chóng và linh hoạt. Tuy nhiên, Express lại không có một kiến trúc hay một quy tắc tổ chức codde một cách cụ thể. Điều này sẽ là một vấn đề lớn khi dự án của bạn được scale với nhiều chức năng cũng như logic nghiệp vụ ngày các phức tạp. Tình trạng này sẽ gây ra một số vấn đề:
* Rối code và trùng lặp code
* Khó quản lý và maintain
* Khó khăn trong việc mở rộng cũng như tính ổn định
* Cấu trúc tệ khiến khó khăn hoặc không thể viết unit-test
* Việc Không tổ chức tốt cấu trúc dự án sau này sẽ dẫn đến phát sinh chi phí cũng như rủi do trong quá trình update cũng như maintain.
  1. **Áp dụng kiến trúc 3 Layer**

****

* Ý tưởng kiến trúc này là tách biệt hoàn toàn logic nghiệp vụ cũng như truy vấn database ra khỏi API route và chuyển chúng đến các layer tương ứng như bên trên.
* Dưới đây là mô hình sau khi apply vào Express trong NodeJS:

A screenshot of a computer

Description automatically generated

* Controller layer (API routes and endpoints): Đây là module code cho phép định nghĩa các API route trong ứng dụng. Nhiệm vụ module này là xác thức data và chuyển đến Service Layer để xử lý.
* Service layer (for business logic): Module này cho phép tạo ra lớp tách biệt ra khỏi các API Router. Mỗi một service tương ứng với một nhiệm vụ riêng biệt (Nên tuân theo nguyên tắc SOLID). Việc chuyển logic nghiệp vụ vào module này cho phép các thể tái sử dụng ở các API khác nhau một cách hiệu quả.
* Data access layer (for working with a database): Module này đảm nhận trách nhiệm giao tiếp với database với các công việc truy vấn như get, update, create,.... Ngoài ra các công việc như định nghĩa model, ORM, connect databse đều được thực hiện tại đây.
* Một số packege nodejs nên cài đặt trước khi bắt đầu một dự án NodeJS:

A screenshot of a computer

Description automatically generated

* Cấu trúc thư mục (Folder structure)

A screenshot of a computer

Description automatically generated

* Controller: Đây là nơi xác định các api router cũng như các lớp service layer chưa logic xử lý nghiệp vụ.
* Models: Module này cho phép định nghĩa các model, schema cho database. Các truy vấn đến database sẽ thực hiện trong module
* Router: Ý tưởng của module này là thay vì khởi tạo và cấu hình các service trong một nơi duy nhất là index.js thì ta nên tách ra thành các module nhỏ khác nhau tương ứng với các service khác nhau.
  1. **Ưu điểm:**
  2. **Tính Tổ Chức (Organization)**: Phân tách ứng dụng thành 3 lớp giúp tổ chức code một cách rõ ràng và dễ dàng hiểu. Mỗi lớp chịu trách nhiệm riêng biệt, giúp quản lý và bảo trì dễ dàng hơn.
  3. **Tính Linh Hoạt và Mở Rộng (Flexibility and Extensibility)**: Mỗi lớp ẩn đi chi tiết triển khai của các lớp dưới cùng, chỉ cung cấp một giao diện cho các lớp trên cùng. Điều này giúp tạo ra một cách nhìn trừu tượng và dễ hiểu về cách hoạt động của hệ thống.
  4. **Dễ Kiểm Thử**: Tính phân tách rõ ràng giữa các lớp giúp cho việc kiểm thử dễ dàng hơn, từ việc kiểm tra giao diện người dùng đến kiểm tra logic kinh doanh và truy cập dữ liệu.
  5. **Nhược điểm:**

1. **Tăng Overhead (Tăng Chi phí)**: Mỗi lớp thêm vào đều tăng overhead cho hệ thống, từ việc tạo và duy trì các giao diện đến việc truyền thông qua các lớp. Điều này có thể ảnh hưởng đến hiệu suất của hệ thống, đặc biệt là trong các hệ thống có số lượng lớn các lớp.
2. **Khó Khăn Trong Việc Đồng Bộ Hóa**: Trong một số trường hợp, có thể khó khăn để đồng bộ hóa logic giữa các lớp, đặc biệt là khi cần thực hiện các thao tác phức tạp như giao tiếp với các dịch vụ ngoại vi.
   1. **Một số Khía cạnh Khác:**
   2. **Tính Bảo mật (Security)**: Kiến trúc 3 layer có thể được sử dụng để cải thiện bảo mật bằng cách tách biệt các lớp và áp dụng các biện pháp bảo mật tại mỗi lớp.
   3. **Tính Tái sử dụng (Reusability)**: Các thành phần logic và dữ liệu được phân chia rõ ràng, dễ dàng tái sử dụng trong các phần của ứng dụng khác nhau
   4. **Trường hợp nên dùng kiến trúc 3 lớp**

Khi bạn cần một cách tổ chức rõ ràng cho mã nguồn của mình, để dễ dàng bảo trì và mở rộng, kiến trúc 3 lớp là một lựa chọn tốt vì:

* Kiến trúc 3 lớp phân chia ứng dụng thành 3 lớp chính: lớp trình bày, lớp logic kinh doanh và lớp truy cập dữ liệu. Mỗi lớp có một trách nhiệm cụ thể, giúp giảm thiểu sự phức tạp và làm cho mã nguồn trở nên dễ hiểu hơn.
* Với việc mã nguồn được phân chia thành các phần nhỏ, mỗi phần chịu trách nhiệm cho một nhiệm vụ cụ thể, việc bảo trì trở nên dễ dàng hơn. Khi cần sửa đổi hoặc thêm mới tính năng, bạn chỉ cần tìm và chỉnh sửa trong phần tương ứng mà không cần lo lắng về ảnh hưởng đến các phần khác của ứng dụng.
* Kiến trúc 3 lớp tạo ra một môi trường linh hoạt cho việc mở rộng ứng dụng. Bạn có thể mở rộng chức năng của ứng dụng bằng cách thêm mới logic kinh doanh hoặc cải thiện lớp truy cập dữ liệu mà không cần phải thay đổi hoặc can thiệp vào các phần khác của ứng dụng.
  1. **Trường hợp không nên dùng kiến trúc 3 lớp**
* Nếu ứng dụng có độ phức tạp cực đại kiến trúc 3 lớp có thể trở nên không đủ linh hoạt và quá giản đơn để đáp ứng các yêu cầu phức tạp của dự án. Trong kiến trúc 3 lớp, các lớp thường giao tiếp một cách trực tiếp và có mối quan hệ chặt chẽ với nhau. Điều này có thể tạo ra ràng buộc quá mức giữa các lớp và làm giảm tính linh hoạt của hệ thống.
* Nếu ứng dụng cần tính Mở rộng cao. Trong các ứng dụng cần tính mở rộng cao, kiến trúc 3 lớp có thể gặp phải những vấn đề liên quan đến mối quan hệ giữa các lớp, gây ra các ràng buộc không cần thiết và làm giảm khả năng mở rộng của hệ thống.
* Nếu Ứng dụng Yêu cầu Bảo mật Cực cao. Trong một số trường hợp, đặc biệt là trong các ứng dụng yêu cầu bảo mật cực cao, kiến trúc 3 lớp có thể không cung cấp đủ cơ chế bảo mật để đáp ứng được các yêu cầu bảo mật nghiêm ngặt.
  1. **Khả năng kiểm thử**

Kiến trúc 3 lớp cung cấp một cơ sở lý tưởng để thực hiện kiểm thử cho các ứng dụng phần mềm. Bằng cách phân chia ứng dụng thành ba lớp chính - lớp trình bày, lớp logic kinh doanh và lớp truy cập dữ liệu - kiến trúc này giúp tạo điều kiện thuận lợi cho việc kiểm thử mọi khía cạnh của hệ thống.

* 1. **Hiệu năng**

Mặc dù kiến trúc 3 lớp mang lại nhiều lợi ích trong việc tổ chức và bảo trì mã nguồn, nhưng hiệu năng của nó có thể trở thành một vấn đề đối với các ứng dụng có yêu cầu cao về tốc độ và hiệu suất.

* 1. **Khả năng mở rộng**

Do xu hướng triển khai nguyên khối và liên kết chặt chẽ của kiến trúc này, các ứng dụng được xây dựng bằng cách sử dụng mẫu kiến trúc này thường khó mở rộng quy mô. Chúng ta có thể mở rộng quy mô kiến trúc phân 3 lớp bằng cách tách 3 lo thành các triển khai vật lý riêng biệt hoặc sao chép toàn bộ ứng dụng thành nhiều nút, nhưng nhìn chung mức độ chi tiết quá rộng, khiến việc mở rộng quy mô trở nên tốn kém.

# : PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

### Usecase tổng quát

A diagram of a network

Description automatically generated

Hình 3.1 Usecase tổng quát

A screen shot of a diagram

Description automatically generated

Hình 3.2 Usecase Đăng ký học phần

A diagram of a system

Description automatically generated

Hình 3.3 Usecase Quản lí thông tin

A screen shot of a computer

Description automatically generated

Hình 3.4 Usecase Thông Báo

A screen shot of a diagram

Description automatically generated

Hình 3.5 Usecase Xem Thông tin

A diagram of a system

Description automatically generated

Hình 3.6 Usecase Quản lí điểm

### 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 | * Đăng nhập * Đăng kí * Đăng ký học phần: * Cập nhật sửa đăng ký học phần * Tìm kiếm * Xem Thông tin * Xem thời khóa biểu * Xem danh sách lớp * Quản lý Điểm:   + Nhập điểm   + Sửa điểm * Thông Báo * TKB * Phòng học * Sửa password * Quản lí thông tin * Quản lí khoa * Quản lí môn học * Quản lí học kì * Quản lí thông tin sinh viên * Thoát |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Bảng 3‑1 Danh sách tác nhân và mô tả

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

|  |  |
| --- | --- |
| **ID use case** | **Tên Use case** |
| **U1** | Đăng nhập |
| U2 | Đăng xuất |
| U3 | Thay Đổi mật khẩu |
| U4 | Đăng ký môn học |
| U5 | Sửa Đăng ký môn học |
| U6 | Xem kết quả đăng ký học |
| U7 | Đăng ký môn dạy học |
| U8 | Sửa Đăng ký môn dạy học |
| U9 | Xem kết quả đăng ký dạy học |
| U10 | Xem Điểm |
| U11 | Đăng Thông Báo |
| U12 | Xem Thông Báo |
| U13 | Tìm kiếm |
| U14 | Đăng kí |

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

#### Tình huống hoạt động

#### UC01\_Đăng nhập

|  |  |
| --- | --- |
| * **Tên use case**: UC01\_Đăng nhập | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã có tài khoản đăng nhập | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 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 email và mật khẩu |  |
| 1. Chọn nút đăng nhập | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Hệ thống chuyển hướng người dùng vào giao diện trang chủ. |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 6.1. Hệ thống thông báo thông tin tài khoản đăng nhập không đúng |
|  | 6.2. Hệ thống quay lại bước 3 |

Activity diagram:

A diagram of a diagram

Description automatically generated with medium confidence

Sequence diagram:

A blueprint of a diagram

Description automatically generated

#### 3.1.4.2 UC02\_Đăng xuất

|  |  |
| --- | --- |
| * **Tên use case**: UC02\_Đăng xuất | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng xuất | 2. Hệ thống hiển thị xác nhận muốn thoát |
| 1. Người dùng ấn xác nhận | 1. Hệ thống trả về giao diện đăng nhập |
| * **Luồng sự kiện thay thế (alternate flow):** | |
| **Không có** | |

A diagram of a system

Description automatically generated

#### 3.1.4.3 UC03\_Sửa password

|  |  |
| --- | --- |
| * **Tên use case**: UC03\_ Sửa password | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| * 1. Người dùng chọn chức năng sửa password | * 1. Hệ thống yêu cầu nhập pass cũ và pass mới |
| * 1. Người dùng nhập vào pass cũ và mới | * 1. Hệ thống thực hiện kiểm tra pass cũ, xác nhận đã sửa passwork |
|  | * 1. Hệ thống thông báo thành công và quay vè màn hình đăng nhập |
|  |  |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.1 Hệ thống thông báo passwork cũ không đúng |
|  | 4.2 Trở về giao diện sửa pass ban đầu |

A diagram of a system

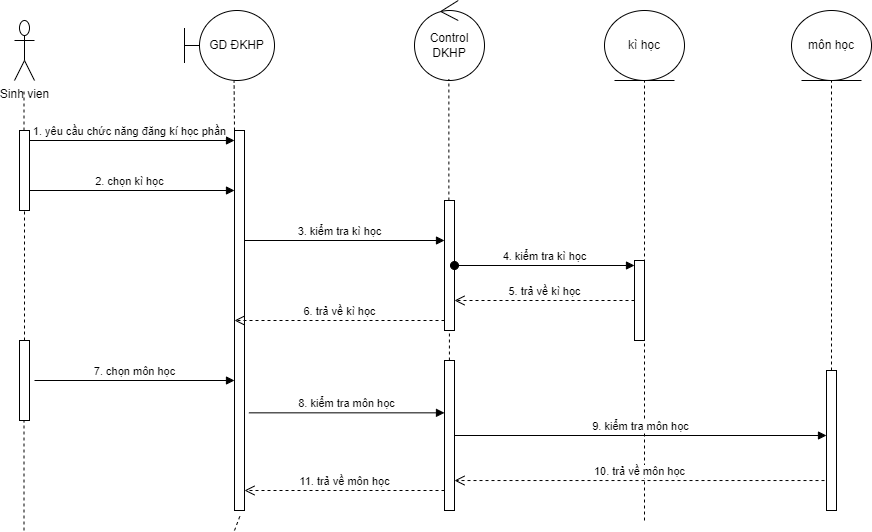
Description automatically generated

#### 3.1.4.4 UC04\_Đăng kí học phần

|  |  |
| --- | --- |
| * **Tên use case**: UC04\_ Đăng kí học phần | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| * 1. Sinh viên chọn chức năng đăng kí học phần | * 1. Hệ thống hiển thị form đăng kí |
| * 1. Sinh viên chọn môn học muốn đăng ký và click OK | * 1. Hệ thống kiểm tra thông tin đăng kí và các điều kiện đăng kí trong quy định |
|  | * 1. Hệ thống thông báo đăng ký thành công, lưu thông tin đăng ký vào cơ sở dữ liệu |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.a.1. sinh viên không đủ điều kiện để học môn này  4.a.2. Trả về form đăng kí |
|  | 4.b.1. hệ thông thông báo không còn lớp môn học này  4.b.2. Trả về form đăng kí |

A diagram of a system

Description automatically generated



#### 3.1.4.5 UC05\_Sửa đăng kí học phần

|  |  |
| --- | --- |
| * **Tên use case**: UC05\_Sửa đăng kí học phần | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Sinh viên | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Sinh viên phải đăng kí ít nhất 1 môn học | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| * 1. Sinh viên chọn chức năng Sửa đăng kí học phần | * 1. Hệ thống hiển thị danh sách các môm sinh viên đã đăng kí |
| * 1. Sinh viên chọn các môn muốn sửa và thay đổi | 4.Hệ thống yêu cầu xác nhạn thay đổi |
| 1. Sinh viên xác nhận | 1. Hệ thống thông báo sửa thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 2.1 Hệ thống thông báo không cho phép sửa vì đã quá hạn thời gian  2.2 Hệ thống quay lại giao diện của sinh viên |

A diagram of a system

Description automatically generated

#### 3.1.4.6 UC06\_Xem điểm

|  |  |
| --- | --- |
| * **Tên use case**: UC06\_Xem điểm | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Sinh viên | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| * 1. Sinh viên chọn chức năng xem điểm | * 1. Hệ thống tìm kiếm trong CSDL |
|  | * 1. Hệ thống hiển thị bảng điểm các môn của sinh viên |
|  |  |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 2.1 Hệ thống thông báo chưa cập nhật điểm  2.2 Hệ thống quay lại giao diện chức năng sinh viên |

A diagram of a system

Description automatically generated

#### 3.1.4.7 UC07\_Đăng kí dạy môn học

|  |  |
| --- | --- |
| * **Tên use case**: UC07\_Đăng kí dạy môn học | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Giảng viên | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Giảng viên đã đăng nhập vào hệ thống | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| * + 1. Giảng viên chọn chức năng đăng kí dạy môn học | * + 1. Hệ thống hiển thị form danh sách các môn học giảng viên có thể dạy |
| 3.Giảng viên chọn môn học mong muốn và click OK | 4. Hệ thống kiểm tra thông tin đăng kí và các điều kiện đăng kí trong quy định |
|  | 5. Hệ thống thông báo đăng ký thành công, lưu thông tin đăng ký vào cơ sở dữ liệu |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.1. Hệ thống thông báo giảng viên đăng ký quá số môn cho phép  4.2. Trả về form đăng kí |

A diagram of a system

Description automatically generated

#### 3.1.4.8 UC08 \_Thông báo

|  |  |
| --- | --- |
| * **Tên use case**: UC08 \_Thông báo | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Nhân viên phòng đào tạo | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn chức năng đăng thông báo | 1. Hệ thống hiển thị form yêu cầu nhập các thông tin cần thông báo |
| 1. Nhân viên nhập thông tin cần thông báo | 1. Hệ thống lưu thông báo vào csdl và hiển thị trên các mục thông báo của hệ thống |
|  |  |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | * 1. Hệ thông báo lỗi do file thông tin quá lớn   2. Trả về giao diện thông báo |

A diagram of a system

Description automatically generated

#### 3.1.4.9 UC09 \_Nhập điểm

|  |  |
| --- | --- |
| * **Tên use case**: UC09 \_Nhập điểm | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Nhân viên phòng đào tạo | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đã đăng nhập vào tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên nhập mã lớp muốn nhập điểm vào form xác nhận và click OK | 1. Hệ thống kiểm tra và hiển thị form nhập điểm |
| 1. Nhân viên nhập điểm cho các sinh viên | 1. Hệ thống kiểm tra và thông báo nhập thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 2.a.1. Hệ thống thông báo nhập sai mã  2.a.2.Quay lại form xác nhận để người dùng nhập lại thông tin khác |
|  | 4.a.1.Hệ thống thông báo nhập sai điểm(>10<)  4.a.2.Quay lại form nhập điểm |

A diagram of a diagram

Description automatically generated

#### 3.1.4.10 UC10 \_Đăng kí tài khoản

|  |  |
| --- | --- |
| * **Tên use case**: UC10 \_Đăng kí tài khoản | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng vào website trường | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng kí tài khoản | 1. Hệ thống hiển thị form đăng kí tài khoản |
| 1. Người dùng nhập vào form đăng kí | 1. Hệ thống kiểm tra thông tin người dung điền vào |
|  | 1. Hệ thống thông báo đăng kí tài khoản thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 4.a.1Hệ thống thông báo nhập sai tên đăng nhập( phải bắt đầu bằng chữ và kêt thúc là @gmail.com)  4.a.2.Quay lại form đăng kí |
|  | 4.a.1.Hệ thống thông báo mật khẩu không được nhập kí tự đặc biệt  4.a.2. Quay lại form đăng kí |
|  |  |

A diagram of a system

Description automatically generated

A diagram of a diagram

Description automatically generated

### Class diagram

A computer screen shot of a diagram

Description automatically generated

## Kiến trúc *layered architecture*

### Quản lý Sinh viên và Ngành học:

Sinh viên được phân chia vào các khoa dựa trên ngành học của mình để đăng ký học phần.

Hệ thống lưu trữ thông tin sinh viên bao gồm: mã sinh viên, tên, ngày sinh, khoa, ngành học, v.v.

### Quản lý Học kỳ và Môn học:

Trong mỗi học kỳ, nhà trường chọn một số môn học cho từng ngành học để sinh viên đăng ký.

Mỗi môn học có thông tin như: mã môn học, tên môn học, số tín chỉ, số lớp học, v.v.

Môn học có thể có môn tiên quyết, và hệ thống sẽ kiểm tra xem sinh viên đã học môn tiên quyết đó chưa trước khi cho phép đăng ký môn học.

### Quản lý Đăng ký học phần:

Sinh viên đăng ký học phần thông qua hệ thống sau khi đăng nhập.

Hệ thống hiển thị danh sách các môn học mà sinh viên có khả năng đăng ký dựa trên các yêu cầu và môn học đã đăng ký trước đó.

Sinh viên chọn các môn học và tiến hành đăng ký. Nếu lớp đã đầy, sinh viên sẽ được đưa vào danh sách dự bị.

### Quản lý Xác nhận và Thông báo:

Trước khi đăng ký, sinh viên phải xác nhận để tránh trường hợp đăng ký rồi hủy bỏ.

Sau khi đăng ký thành công, hệ thống gửi email thông báo xác nhận đăng ký và quyết định đóng học phí.

### Quản lý Thông tin học tập:

Hệ thống cung cấp cho sinh viên xem thông tin học tập cá nhân như số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy, v.v.

### Quản lý Xét tốt nghiệp và Cựu sinh viên:

Sau khi đạt đủ số tín chỉ theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp.

Thông tin về bằng cấp sẽ được công khai trên trang web của nhà trường và thông tin của cựu sinh viên sẽ được lưu giữ cho các mục đích theo dõi và liên lạc.

### Quản lý Lớp học:

Mỗi môn học có thể có nhiều lớp học khác nhau, được mở vào các kỳ học khác nhau.

Thông tin về lớp học bao gồm: số lượng sinh viên tối đa, số lượng sinh viên hiện tại, giáo viên phụ trách, thời gian học, v.v.

Hệ thống cung cấp khả năng quản lý lớp học, bao gồm tạo mới, chỉnh sửa và xóa lớp học.

### Quản lý Danh sách dự bị:

Sinh viên được đưa vào danh sách dự bị khi lớp học đã đầy.

Hệ thống tự động di chuyển sinh viên từ danh sách dự bị vào lớp học khi có chỗ trống.

### Quản lý Thời khóa biểu:

Hệ thống cung cấp thời khóa biểu cho sinh viên, hiển thị các lớp học mà sinh viên đã đăng ký theo từng tuần học.

Sinh viên có thể xem và in thời khóa biểu của mình từ hệ thống.

### Quản lý Tính năng Tìm kiếm và Lọc:

Hệ thống cung cấp tính năng tìm kiếm và lọc để sinh viên có thể dễ dàng tìm kiếm và lựa chọn các môn học phù hợp với nhu cầu học tập của mình.

Sinh viên có thể tìm kiếm môn học theo tên, ngành học, giáo viên, số tín chỉ, v.v.

### Quản lý Thông tin Tốt nghiệp:

Hệ thống lưu trữ thông tin về việc xét tốt nghiệp của sinh viên sau khi đạt đủ số tín chỉ yêu cầu.

Sinh viên có thể theo dõi trạng thái xét tốt nghiệp và yêu cầu cấp bằng từ hệ thống.

# : HIỆN THỰC

## Giao diện của hệ thống

### Giao diện web

A screenshot of a phone

Description automatically generated

Hình 4.1 giao diện Đăng Nhập

A screenshot of a computer

Description automatically generated

Hình 4.2 giao diện Đăng kí học phần sinh viên

A screenshot of a computer

Description automatically generated

Hình 4.3 giao diện Quản lý môn học của Admin

A screenshot of a computer

Description automatically generated

Hình 4.4 giao diện Quản lý học kỳ của Admin

A screenshot of a computer

Description automatically generated

Hình 4.5 giao diện Quản lý Khoa của Admin

### : KẾT LUẬN

## Kết quả đạt được

* Quản lý thông tin sinh viên: Hệ thống giúp quản lý thông tin cá nhân và học tập của sinh viên một cách chi tiết và có hệ thống.
* Lưu trữ và truy xuất dữ liệu dễ dàng: Dữ liệu về sinh viên và các khóa học được lưu trữ một cách khoa học, dễ dàng truy xuất và cập nhật khi cần.
* Tự động hóa quy trình đăng ký: Sinh viên có thể tự đăng ký học phần trực tuyến mà không cần phải tới văn phòng hoặc điền các mẫu giấy tờ thủ công.
* Giảm bớt khối lượng công việc cho nhân viên hành chính: Nhân viên chỉ cần giám sát và hỗ trợ khi cần, không phải xử lý thủ công từng trường hợp đăng ký.
* Giao diện thân thiện và dễ sử dụng: Sinh viên có thể dễ dàng tìm kiếm, đăng ký và theo dõi các khóa học của mình.
* Giảm thiểu sai sót: Hệ thống tự động hóa giúp giảm thiểu các lỗi phát sinh từ quy trình thủ công như nhập sai thông tin, thiếu dữ liệu, hoặc mất mát giấy tờ.
* Tăng cường minh bạch: Mọi thông tin về quá trình đăng ký và lựa chọn học phần đều được ghi nhận và có thể kiểm tra lại khi cần.
* Thu được gồm các chức năng sau:
* Đăng kí, Đăng nhập tài khoản
* Đăng kí học phần dành cho sinh viên
* Quản lý môn học
* Quản lý Học kỳ
* Quản lý Khoa

## Hạn chế của đồ án

* Ứng dụng chưa có chức năng giúp giảng viên có thể quản lý môn học và đăng ký dạy học
* Khả năng bảo mật: Do ứng dụng mới nên khả năng bảo mật của ứng dụng khá kém
* Hạn chế về mặt chi phí

## Hướng phát triển

* Mở rộng phần mềm giúp nó có thể đồng bộ dữ liệu, đảm bảo dữ liệu về lịch học, bài tập, và điểm số được đồng bộ tự động giữa các hệ thống.
* Phát Triển Ứng dụng di động cho sinh viên và giảng viên: Cung cấp ứng dụng di động để sinh viên và giảng viên có thể truy cập và quản lý thông tin học phần, đăng ký lớp học, và nhận thông báo ngay trên điện thoại.
* Phát triển thêm khả năng thông báo, nhắc nhở sinh viên về các thời hạn đăng ký, lịch thi, hoặc các thông tin quan trọng khác.
* Tăng khả năng bảo mật cho phần mềm bằng các dung biện pháp xác thực hai yếu tố và Mã hóa dữ liệu

## TÀI LIỆU THAM KHẢO

Các tài liệu Tiếng Anh

1. Fundamentals of Software Architecture - Mark Richards
2. Software Architecture Monday - Mark Richards

Các tài liệu từ Internet

1. <https://nodejs.org/en/docs/>
2. <https://expressjs.com/en/guide/routing.html>
3. <https://www.mongodb.com/docs/manual/reference/operator/aggregation/graphLookup/>
4. <https://topdev.vn/blog/kien-truc-phan-lop-layered-architecture/?gidzl=9GbQV5FuxoSW5bv-LkBj9o4HMdb8vBnnQqK4B4-oioSnHWHw5-2y8ZKP3Yr2whvoOn8FA3D7m81kKFZY90>
5. <https://200lab.io/blog/kien-truc-phan-lop/>
6. [Software Architecture Patterns](Software%20Architecture%20Patterns)