**BỘ CÔNG THƯƠNG TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

**Logo, company name

Description automatically generated**

**BÁO CÁO CUỐI KÌ MÔN KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

**XÂY DỰNG WEBSITE QUẢN LÝ LỚP HỌC TÍN CHỈ CHO TRƯƠNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

**Khối ngành: Công nghệ thông tin**

**Chuyên ngành: Kỹ thuật phần mềm**

***Sinh viên thực hiện***

1. Ngô Hữu Nghị – 19495921

2. Trần Bảo Dự – 19509301

3. Võ Cường – 20002345

**Giảng viên: Thầy Võ Văn Hải**

**TP HỒ CHÍ MINH, THÁNG 05 NĂM 2024**

# **MỤC LỤC**

[**MỤC LỤC** 1](#_Toc167029349)

[**1. Tổng quan tình hình nghiên cứu thuộc lĩnh vực của đề tài** 2](#_Toc167029350)

[**2. Ý nghĩa khoa học thực tiễn của đề tài** 3](#_Toc167029351)

[**CHƯƠNG 1: GIỚI THIỆU** 3](#_Toc167029352)

[**1.1** **Tổng quan** 3](#_Toc167029353)

[**1.2** **Mục tiêu cơ bản của đề tài** 4](#_Toc167029354)

[**1.3 Phạm vi đề tài** 5](#_Toc167029355)

[**1.4 Mô tả yêu cầu chức năng** 5](#_Toc167029356)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 6](#_Toc167029357)

[**2.1 MongoDB** 6](#_Toc167029358)

[**2.2 ReactJS** 7](#_Toc167029359)

[**2.3 Typescript** 8](#_Toc167029360)

[**2.4 ViteJS** 9](#_Toc167029361)

[**2.5 GIT** 10](#_Toc167029362)

[**2.6 NodeJS** 11](#_Toc167029363)

[**2.7 Redux** 13](#_Toc167029364)

[**2.8 SASS** 14](#_Toc167029365)

[**2.9 SocketIO** 15](#_Toc167029366)

[**CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ** 17](#_Toc167029367)

[**3.1 Mô hình Use case tổng quát** 17](#_Toc167029368)

[**3.2 Danh sách các tác nhân và mô tả** 17](#_Toc167029369)

[**3.3 Danh sách Use Case và mô tả** 18](#_Toc167029370)

[**3.4 Đặc tả yêu cầu chức năng** 19](#_Toc167029371)

[**3.4.1 UC001 Đăng nhập** 19](#_Toc167029372)

[**3.4.2 UC002 Xem lịch học** 21](#_Toc167029373)

[**3.4.3 UC003 Xem thông báo** 24](#_Toc167029374)

[**3.4.4 UC004 Đăng kí học phần** 26](#_Toc167029375)

[**3.5 Class Diagram** 30](#_Toc167029376)

[**3.6 Kiến trúc Client-Server** 30](#_Toc167029377)

[**3.6.1. Hoạt động của kiến trúc:** 31](#_Toc167029378)

[**3.6.2. Ưu điểm:** 32](#_Toc167029379)

[**3.6.3. Nhược điểm:** 32](#_Toc167029380)

[**CHƯƠNG 4: HIỆN THỰC** 33](#_Toc167029381)

[**4.1 Một số giao diện** 33](#_Toc167029382)

[**4.1.1 Giao diện đăng nhập** 33](#_Toc167029383)

[**4.1.2 Giao diện trang chủ cho sinh viên** 34](#_Toc167029384)

[**4.1.3 Giao diện xem lịch học của sinh viên** 35](#_Toc167029385)

[**4.1.4 Giao diện đăng kí học phần** 35](#_Toc167029386)

[**4.2 Kiểm thử một số chức năng** 36](#_Toc167029387)

[**4.2.1 Chức năng Đăng nhập** 36](#_Toc167029388)

[**4.2.2 Chức năng tạo thông báo** 37](#_Toc167029389)

[**CHƯƠNG 5: KẾT LUẬN** 38](#_Toc167029390)

[**5.1 Kết quả đạt được** 38](#_Toc167029391)

[**5.2 Hướng phát triển** 38](#_Toc167029392)

[**TÀI LIỆU THAM KHẢO** 40](#_Toc167029393)

**LỜI MỞ ĐẦU**

**1. Tổng quan tình hình nghiên cứu thuộc lĩnh vực của đề tài**

Hiện nay, với sự tiến bộ chóng mặt của khoa học kĩ thuật và công nghệ đã thúc đẩy việc phát triển các ứng dụng dành cho nghành giáo dục, hầu hết các trường đại học và các tổ chức giáo dục đã đẩy mạnh phát triển các hệ thống website và app quản lý tín chỉ, thời khóa biểu trực tuyến nó giúp quản lý thông tin, lịch học của sinh viên và giảng viên. Các hệ thống này thường cung cấp tính năng giúp quản lí sinh viên, học phần cũng như cập nhật thời khóa biểu linh hoạt, thông báo tự động và tích hợp với các ứng dụng di động.

**2. Ý nghĩa khoa học thực tiễn của đề tài**

Trong những năm gần đây, trường Đại học Công nghiệp TP.HCM đã trở thành một trong những trường có lượng tuyển sinh đầu vào rất đông đảo. Với số lượng sinh viên lớn, việc quản lý thông tin sinh viên, học phần và phân bổ phòng học một cách hiệu quả là vô cùng quan trọng. Phần mềm hỗ trợ sẽ giúp tối ưu hóa những vấn đề này

**CHƯƠNG 1: GIỚI THIỆU**

## **Tổng quan**

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

Mỗi học kỳ sinh viên được đăng ký tối đa 30 tín chỉ. Nếu quá số này, hệ thống sẽ không cho đăng ký. Để phòng ngừa trường hợp sinh viên đăng ký xong rồi hủy bỏ, nhà trường yêu cầu sinh viên phải xác nhận trước khi đăng ký. Vào ngày mở đăng ký, sinh viên sẽ đăng nhập vào hệ thống và sẽ nhìn thấy danh sách các môn học mà mình có khả năng đăng ký. Sinh viên chọn các môn học và tiến hành đăng ký. Trường hợp các lớp đã đầy, sinh viên sẽ được đưa vào một danh sách dự bị để nhà trường cân nhắc có mở thêm lớp hay không. Nếu không mở thêm lớp, sinh viên sẽ bị hủy đăng ký môn đó.Sau khi đăng ký thành công, một email thông báo sẽ được gửi cho sinh viên xác nhận việc đăng ký và nhận quyết định đóng học phí.

Ngoài ra, hệ thống còn có khả năng cho phép sinh viên xem thông tin học tập của mình (số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy…), thời khóa biểu theo tuần và các tiện ích khác.

Sau khi số tín chỉ đã đạt theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp. Nếu mọi tiêu chuẩn đều thỏa mãn, sinh viên sẽ được cấp bằng tốt nghiệp và sẽ được đưa vào danh sách các cựu sinh viên. Thông tin về bằng cấp sẽ được công khai trên trang web của nhà trường. Thông tin của cựu sinh viên sẽ được lưu giữ để theo dõi quá trình làm việc (nếu sinh viên đồng ý), làm các cuộc survey, cũng như nhiều hoạt động khác.

* 1. **Mục tiêu cơ bản của đề tài**

Quản lý học phần: Xây dựng hệ thống giúp quản lý thông tin sinh viên và các học phần của sinh viên, giúp sinh viên có thể xem, đăng ký và hủy (nếu có) các học phần của các học kì khi sinh viên còn tham gia khóa học. Việc đăng ký học phần hệ thống phải kiểm tra các điều kiện tiên quyết và yêu cầu.

Quản lý phòng học: Xây dựng một hệ thống quản lý tài nguyên phòng học giúp nhân viên của khoa có thể dễ dàng quản lý thông tin về các phòng học, số lượng phòng cũng như các thiết bị và các lịch học trong phòng tạo khả năng tìm kiếm và lọc thông tin phòng dựa theo các thông tin khác nhau, cùng những nhu cầu đặc biệt khác. Giúp đảm bảo rằng việc phân chia phòng học được tiến hành một cách hiệu quả và hạn chế việc dư thừa hoặc thiếu phòng học.

**1.3 Phạm vi đề tài**

Hệ thống được xây dựng với 3 tác nhân chính:

- Chuyên viên văn phòng khoa

- Sinh viên

**1.4 Mô tả yêu cầu chức năng**

Hệ thống trên nền tảng Website có các chức năng:

* **Chuyên viên văn phòng khoa** có thể sử dụng các chức năng bao gồm:
* Đăng nhập/đăng xuất.
* Quản lí học phần (cập nhật lịch học, mở đăng kí học phần cho sinh viên, thêm lớp học phần)
* Quản lý tài khoản người dùng.
* Tạo thông báo
* Quản lí phòng học (bảo trì phòng, chuyển lịch từ phòng này sang phòng khác)
* **Sinh viên** có thể sử dụng các chức năng bao gồm:
* Đăng nhập/đăng xuất.
* Đăn ký học phần
* Xem lịch học.
* Xem thông báo.
* Đổi mật khẩu
* Xem thông tin cá nhân

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1 MongoDB**

A diagram of a company's logo

Description automatically generated

**Hình 2.1.1 Kiến trúc NoSQL**

**MongoDB** là hệ thống cơ sở dữ liệu phi quan hệ (NoSQL) được sử dụng lưu trữ dữ liệu phi hoặc bán cấu trúc. Điều này cho phép MongoDB không bị ràng buộc chặt chẽ như SQL và dễ dàng thích ứng với các ngôn ngữ như NodeJS, PHP. Nó có tốc độ truy vấn nhanh đồng thời có khả năng mở rộng mạnh [6]

**Ưu điểm:**

* Hỗ trợ lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc, cho phép lưu trữ dễ dàng mà không cứng nhắc như SQL.
* Tính năng replica set cho phép backup và tạo bản sao dữ liệu trên nhiều máy chủ để khi có sự cố thì hệ thống vẫn hoạt động.
* Tương thích với nhiều ngôn ngữ và có thể được sử dụng rộng rãi

**Nhược điểm:**

* Yêu cầu phải có hiểu biết thiết kế cơ sở dữ liệu để tạn dụng tối đa rính năng của MongoDB.
* Không phù hợp với các ứng dụng yêu cầu tính ràng buộc và nhất quán dữ liệu cao

## **2.2 ReactJS**

A diagram of a atom

Description automatically generated

**Hình 2.2.1: Kiến trúc của ReactJS**

ReactJS là một framework được phát triển bởi Facebook, được sử dụng để xây dựng Website động một cách hiệu quả. Thư viện này cung cấp một cách tiếp cận mới trong việc xây dựng giao diện người dùng, giúp cho việc phát triển ứng dụng web trở nên dễ dàng hơn.

Virtual DOM là một phần quan trọng trong ReactJS khi có sự thay đổi trong dữ liệu ứng dụng, ReactJS sẽ tạo một Virtual DOM mới và so sánh nó với Virtual DOM cũ để tìm ra sự thay đổi. Chỉ những phần thay đổi mới được cập nhật vào DOM. [2]

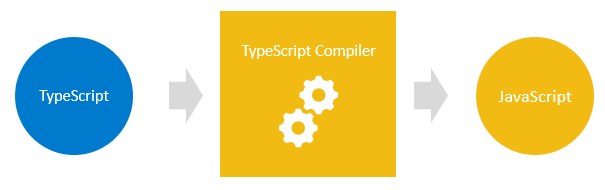
**Ưu điểm:**

* Sử dụng Virtual DOM (DOM ảo) để cập nhật giao diện người dùng nên hiệu suất cao hơn so với sử dụng DOM thật.
* ReactJS sử dụng thành phần (Component) giúp cho việc bảo trì và tái sử dụng thành phần giao dễ dàng hơn.

**Nhược điểm:**

* Học hỏi khó khăn đối với người mới bắt đầu và không có tính năng routing.

## **2.3 Typescript**



**Hình 2.3.1: Cách hoạt động Typescript**

TypeScript là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Microsoft, có thể biên dịch thành JavaScript. TypeScript là một phiên bản mở rộng của JavaScript, cung cấp các tính năng mới như kiểu dữ liệu tĩnh và các tính năng lập trình hướng đối tượng. Trong các file TypeScript (.ts) bạn có thể định nghĩa kiểu dữ liệu cho biến, tham số và giá trị trả về của hàm. Điều này giúp kiểm tra kiễu tĩnh trong quá tình phát triển, giảm thiểu số lượng lỗi. [4]

**Ưu điểm:**

* TypeScript hỗ trợ kiểu dữ liệu tĩnh, giúp cho việc phát hiện lỗi trước khi thực thi chương trình.
* TypeScript có thể biên dịch thành JavaScript và tương thích với các thư viện, framework và công cụ khác của JavaScript.
* TypeScript hỗ trợ các tính năng mới của JavaScript như async/await, map/filter/reduce, v.v., giúp cho việc phát triển ứng dụng trở nên hiệu quả hơn.

**Nhược điểm:**

* TypeScript cần phải được biên dịch thành JavaScript trước khi chạy, điều này có thể tốn thời gian trong quá trình phát triển.
* TypeScript có thể khó khăn đối với các nhà phát triển mới bắt đầu, đặc biệt là khi họ chưa có kiến thức về JavaScript.

## **2.4 ViteJS**



**Hình 2.4.1: Tổng quan về Vite**

Vite.js là một công cụ phát triển ứng dụng web với tốc độ nhanh và hiệu suất cao. Nó chạy nhanh nhờ sử dụng mô hình đóng gói tĩnh và sử dụng khả năng của trình duyệt để tải các module chỉ khi cần. Vite cung cấp hot-reload tức thì, cho phép bạn chỉnh sửa mã nguồn mà không cần xây dựng lại toàn bộ ứng dụng, giúp tiết kiệm thời gian phát triển. Nó hỗ trợ TypeScript mạnh mẽ. [7]

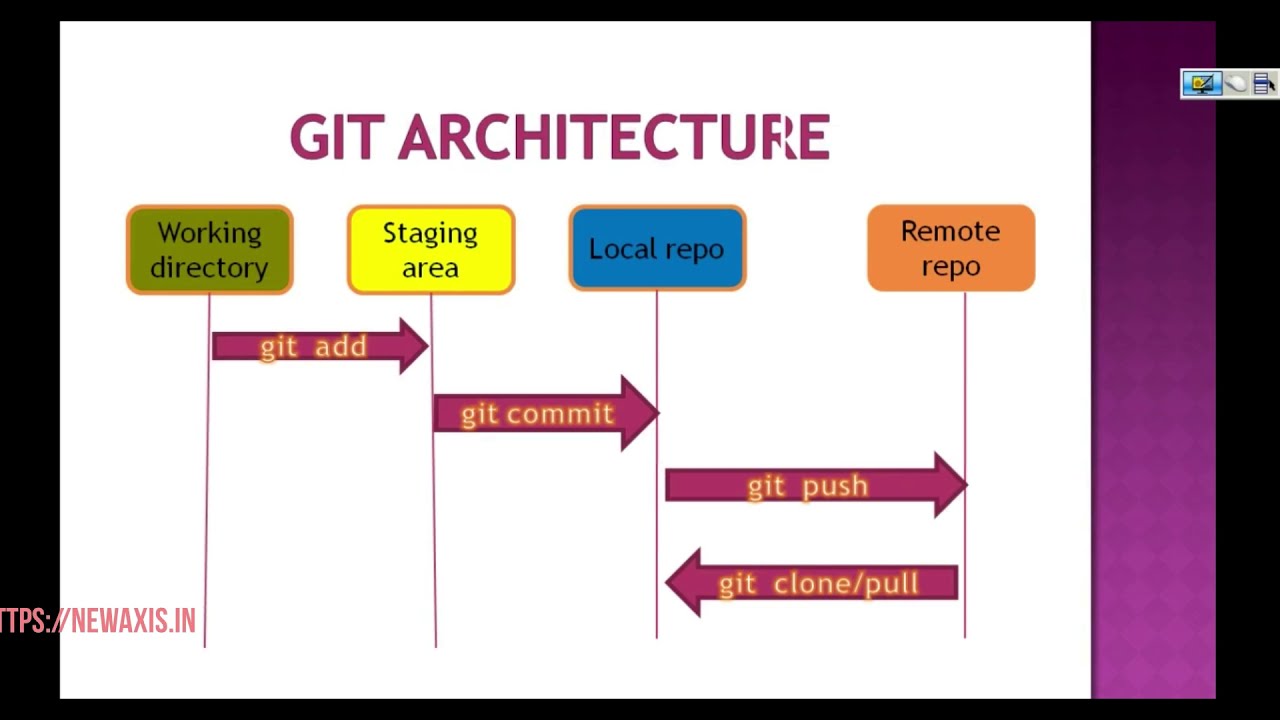
**Ưu điểm:**

* Vite.js sử dụng mô hình đóng gói tĩnh và khai thác trình duyệt để tải module khi cần. Điều này giúp giảm thời gian build và tăng tốc độ khởi động ứng dụng.
* Vite.js cung cấp hot-reloading ngay lập tức khi bạn chỉnh sửa mã nguồn điều này giúp tăng hiệu suất phát triển và tiết kiệm thời gian
* Vite.js tích hợp TypeScript một cách mạnh mẽ, cho phép bạn sử dụng kiểm tra kiểu tĩnh và tính năng của TypeScript.
* Vite.js có cấu hình đơn giản và dễ hiểu. Bạn không cần phải mất thời gian cấu hình phức tạp để bắt đầu phát triển ứng dụng

**Nhược điểm:**

* Hỗ trợ cộng đồng còn hạn chế.

## **2.5 GIT**



**Hình 2.5.1: Kiến trúc của GIT**

Git là một hệ thống quản lý phiên bản phân tán, được phát triển bởi Linus Torvalds. Git được sử dụng rộng rãi trong việc quản lý mã nguồn của các dự án phần mềm, cho phép các nhà phát triển làm việc cùng nhau trên cùng một mã nguồn, theo dõi các thay đổi và hợp nhất các phiên bản khác nhau của mã nguồn. [3]

**Ưu điểm:**

* GIT có thể tích hợp dễ dàng với các dịch vụ lưu trữ mã nguồn.
* GIT cho phép các nhà phát triển làm việc trên cùng một mã nguồn mà không cần phải kết nối với một máy chủ
* GIT cung cấp các tính năng bảo mật để bảo vệ mã nguồn của bạn, bao gồm quản lý quyền truy cập, mã hóa và chứng thực.

**Nhược điểm:**

* Trong trường hợp có nhiều người cùng làm việc trên cùng một mã nguồn và thực hiện các thay đổi trên cùng một tệp, có thể gây ra xung đột confict và khó khăn trong việc giải quyết các xung đột này.

## **2.6 NodeJS**

A diagram of a company

Description automatically generated

**Hình 2.6.1: NodeJS**

Nodejs là một môi trường chạy mã JavaScript bên ngoài trình duyệt web. Điều này cho phép chúng ta sử dụng JavaScript để viết các ứng dụng và máy chủ phía sau web, không chỉ trong trình duyệt. Node.js giúp xây dựng các ứng dụng máy chủ, ứng dụng dòng lệnh, và nhiều loại ứng dụng khác sử dụng JavaScript. Điều này làm cho việc chia sẻ mã nguồn giữa máy chủ và trình duyệt dễ dàng hơn và giúp tạo ra các ứng dụng web hiệu quả và nhanh chóng hơn. [1]

**Ưu Điểm**

* Các ứng dụng Nodejs được viết bằng javascript, ngôn ngữ này là một ngôn ngữ khá thông dụng.
* Hiệu suất cao: Node.js được xây dựng dựa trên kiến trúc non-blocking và sử dụng sự kiện (event-driven), giúp xử lý nhiều yêu cầu đồng thời mà không tạo ra các luồng mới, làm tối ưu hiệu năng ứng dụng.
* Thích hợp cho ứng dụng thời gian thực: Với khả năng xử lý đồng thời và sự kiện, Node.js thích hợp cho việc xây dựng các ứng dụng thời gian thực như ứng dụng trò chơi trực tuyến hoặc chat realtime.

**Nhược Điểm**

* Xử lí đơn luồng: NodeJS sử dụng mô dình đơn luồng nên chỉ có 1 luồng chính xử lí yêu cầu. Việc này có thể gây khó khăn khi xử lí các tác vụ đòi hỏi xử lí đồng thời nặng.
* Khó khăn quản lí bộ nhớ: NodeJS sử dụng mô hình quản lí bộ nhớ không đồng bộ trong đó bộ nhớ được quản lí bởi garbage collector. Nó có thể gây hao hụt tài nguyên đòi hỏi lập trinh viên phải có cách quản lí tốt

## **2.7 Redux**



**Hình 2.7.1: Kiến trúc của Redux**

Redux là một thư viện quản lý trạng thái cho ứng dụng JavaScript. Redux giúp cho việc quản lý trạng thái ứng dụng trở nên dễ dàng và có thể được sử dụng trong các ứng dụng web, mobile và desktop. [12]

Redux giúp cho việc quản lý trạng thái ứng dụng trở nên dễ dàng bằng cách giữ cho tất cả trạng thái của ứng dụng được lưu trữ tại một nơi duy nhất, được gọi là "store". Trạng thái này được quản lý bởi một "reducer", là một hàm JavaScript đơn giản nhận vào trạng thái hiện tại và một hành động và trả về trạng thái mới của ứng dụng.. [2]

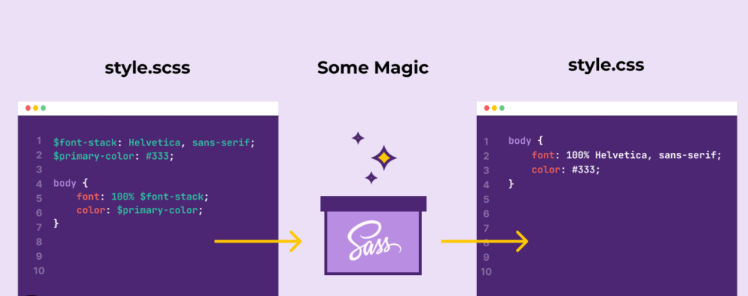
**Ưu điểm:**

* + Redux giúp cho việc quản lý trạng thái của ứng dụng trở nên dễ dàng hơn.
  + Redux giúp cho việc kiểm tra và hiểu rõ các thay đổi trong trạng thái của ứng dụng trở nên dễ dàng hơn.
  + Redux có thể tích hợp dễ dàng với các thư viện khác như React, Angular, Vue.

**Nhược điểm:**

* Khi ứng dụng không quá phức tạp, việc sử dụng Redux có thể làm tăng độ phức tạp của ứng dụng một cách không cần thiết.
* Redux có một số khái niệm mới và phải học một cách cẩn thận để sử dụng hiệu quả.

## **2.8 SASS**



**Hình 2.8.1: cách SASS hoạt động**

Sass (Syntactically Awesome Style Sheets) là một ngôn ngữ mở rộng của CSS, được tạo ra để giúp bạn viết mã CSS dễ dàng hơn và hiệu quả hơn. Sass cung cấp nhiều tính năng mạnh mẽ như biến, lồng ghép, và hàm, giúp bạn tạo mã CSS. Bạn có thể viết mã Sass sử dụng cú pháp giống CSS hoặc cú pháp Sass.

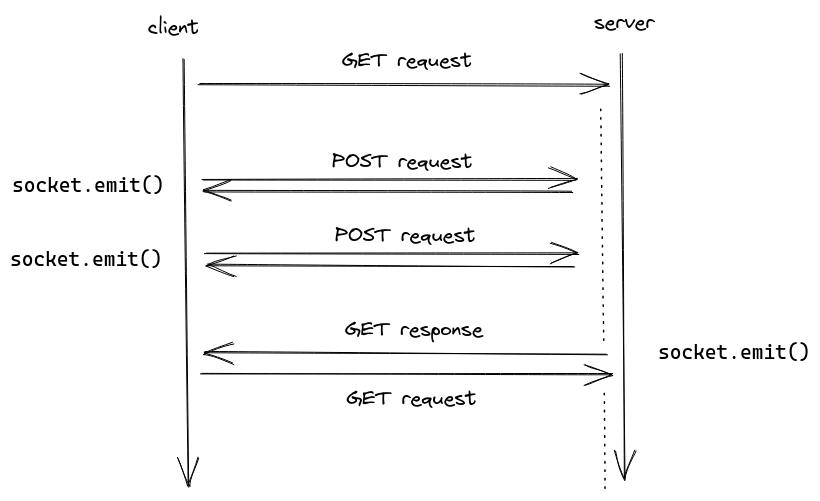
**Ưu điểm:**

* Sass cho phép bạn lồng ghép các quy tắc CSS bên trong nhau, dễ đọc và quản lý hơn.
* Bạn có thể định nghĩa và sử dụng biến trong mã Sass, giúp bạn tái sử dụng giá trị dễ dàng, và thay đổi một lần để thay đổi toàn bộ kiểu.

**Nhược điểm:**

* Sass cần sử dụng một trình biên dịch để chuyển đổi mã nguồn thành CSS.
* Việc biên dịch từ Sass sang CSS có thể tạo ra thời gian chờ đợi, đặc biệt đối với dự án lớn.

## **2.9 SocketIO**



**Hình 2.9.1: Socket IO**

Socket.IO là 1 thư viện javaScript mã nguồn mở được sử dụng để phát triển ứng dụng web thời gian thực. Với khả năng tương tác và truyền dữ liệu nhanh giữa client và server nó giúp xây dựng các website đa người dùng nó giúp truyền và tải dữ liệu mà không cần phải làm mới trang

Socket.IO sử dụng WebSocket để tạo kết nối thời gian thực, mở khả năng truyền và nhận dữ liệu ngay lập tức nó giúp tăng cường hiệu suất và trải nghiệm người dùng khi không phải tải lại trang khi có thay đổi.

**Ưu điểm:**

* Cho phép truyền tải dữ liệu real-time giữa client và server mà không cần làm mới trang
* Linh hoạt có thể tự động tái kết nối trong điều kiện phù hợp khi có sự cố mât kết nối, giảm thiểu gián đoạn truyền tải dữ liệu
* Dễ sử dụng đơn giản có thể tích hợp các framework web

**Nhược điểm:**

* Do yêu cầu môi trường bảo mật nên việc cấu hình và cài đặc gặp nhiều khó khăn cho người mới.
* Mặc dù có khả năng mở rộng nhưng việc quản lý môi trường với quá nhiều kết nối gặp khó khăn

# **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ**

## **3.1 Mô hình Use case tổng quát**

A diagram of a diagram

Description automatically generated

**Hình 3.1: Mô hình use case tổng quát quản lí phòng học**

## **3.2 Danh sách các tác nhân và mô tả**

|  |  |  |
| --- | --- | --- |
| STT | Tác nhân | Mô tả chức năng |
| 1 | **Giáo Vụ Khoa** | - Đăng nhập vào website.  - Quản lí học phần học phần (mở đăng kí học phần cho sinh viên, cập nhật hoặc tạm ngưng lịch, them lớp học phần mới).  - Quản lí phòng (bảo trì phòng và chuyển lịch từ phòng này sang phòng khác theo ngày)  - Quản lí tài khoản  - Tạo các thông báo cho sinh viên. |
| 2 | **Sinh Viên** | - Đăng nhâp vào hệ thống và xem lịch học cá nhân có thể chọn ngày để xem lịch (xem lịch theo ngày, theo tháng)  - Đăng ký học phần  - Xem các thông báo từ khoa.  - Xem thông tin cá nhân  - Đổi mật khẩu |

## **3.3 Danh sách Use Case và mô tả**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tên use case** | **Mô tả ngắn gọn Use Case** | **Chức năng** | **Ghi chú** |
| UC001 | Đăng nhập | Giúp người dùng đăng nhập vào hệ thống thực hiện các chức năng khác | Đăng nhập |  |
| UC002 | Xem lịch học | Giúp sinh viên xem lịch học của mình trên web | Xem lịch học |  |
| UC003 | Xem thông báo | Giúp sinh viên có thể xem thông báo của khoa (sự kiên, lịch thi, thời gian đăng kí học phần,…) | Xem thông báo |  |
| UC004 | Đăng kí học phần | Giúp sinh viên đăng kí học phần | Giúp sinh viên đăng ký các môn học cần thiết |  |

**Bảng 3.3 Danh sách Use Case và mô tả**

## **3.4 Đặc tả yêu cầu chức năng**

### **3.4.1 UC001 Đăng nhập**

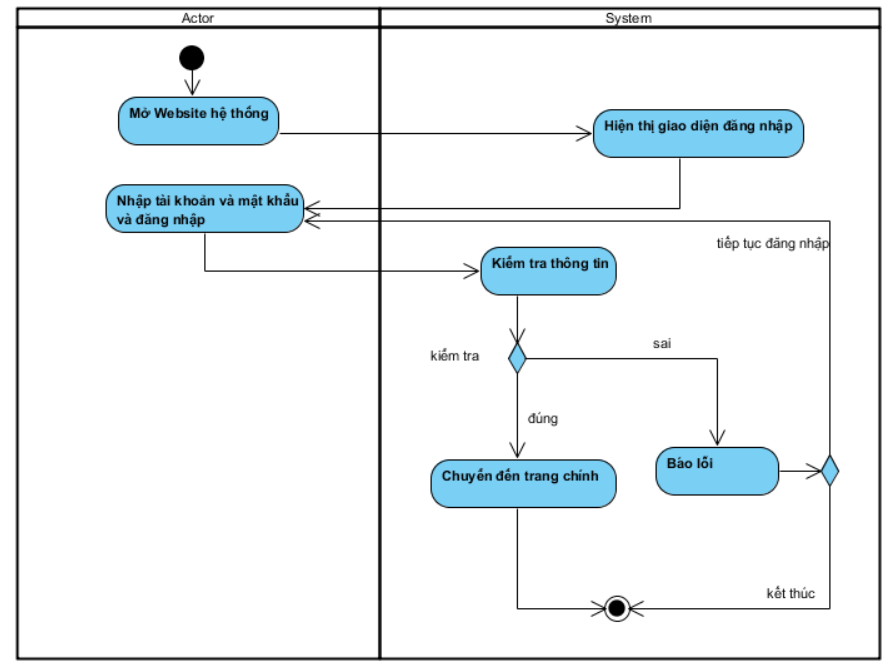
**3.4.1.1 Đặc tả**

|  |  |
| --- | --- |
| **Use case: UC001** | |
| Tên use case | Đăng nhập |
| Actor | Sinh viên, giảng viên, Khoa |
| Mô tả | Giúp đăng nhập vào hệ thống và thực hiện các chức năng dựa trên quyền của tài khoản đã đăng nhập vào |
| Precondition: | Điền đầy đủ thông tin đăng nhập tài khoản có tồn tại trên hệ thống |
| Poscondition: | Nếu thực hiện chức năng thành công tiến hành đăng nhập vào trang chủ của hệ thống. Nếu thất bại hiện thông báo lỗi trên giao diện của hệ thống. |
| **Luồng sự kiện chính (Basic flows)**   |  |  | | --- | --- | | Actor | System | | 1. Mở website  3. thông tin đăng nhập vào các ô input hoặc bấm vào nút đăng nhập | 2. Hệ thống hiển thị trang đăng nhập gồm các ô input tài khoản, mật khẩu và nút đăng nhập  4. Hệ thống kiểm tra thông tin đăng nhập  5. Hệ thống chuyển đến trang chủ | | |
| **Luồng sự kiện phụ (Alternative Flows):** | |
| |  |  | | --- | --- | | Actor | System | |  | 5.1 Hệ thống hiện thông báo lỗi lên website | | |

**Bảng 3.4.1.1 Mô tả UC001 Đăng nhập**

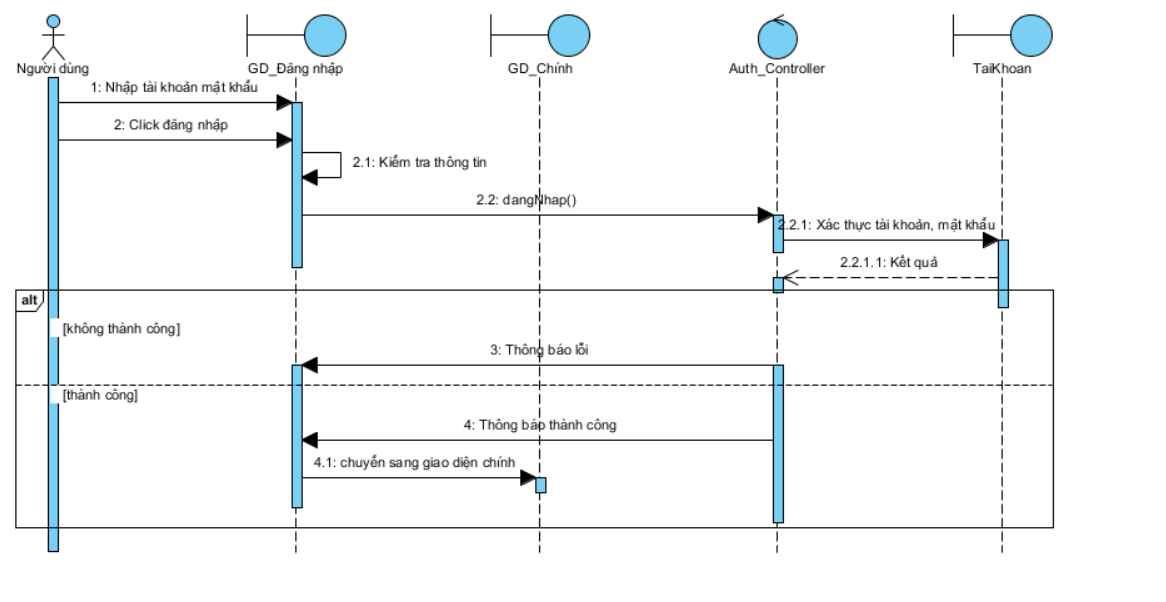
**3.4.1.2 Các sơ đồ**

**a. Sơ đồ Activity Diagram**

****

**Hình 3.4.1.2.1 Sơ đồ Activity**

**b.Sơ đồ Sequence Diagram**

****

**Hình 3.4.1.1.2 Sơ đồ Sequence Diagram**

### 

### **3.4.2 UC002 Xem lịch học**

**3.4.2.1 Đặc tả**

|  |  |
| --- | --- |
| **Use case: UC004** | |
| Tên use case | Xem lịch học |
| Actor | Sinh viên |
| Mô tả | Giúp sinh viên có thể kiểm tra lịch học cá nhân |
| Precondition: | Đăng nhập thành công vào hệ thống và có data danh sách lịch học |
| Poscondition: | Nếu thực hiện chức năng thành công hiển thị danh sách lịch học theo tuần cho sinh viên. Nếu thực hiện thất bại hiện danh sách trống không có lịch |
| **Luồng sự kiện chính (Basic flows)**   |  |  | | --- | --- | | Actor | System | | 1. Đăng nhập thành công và click chọn lịch học trên menu  5. Xem lịch | 2. Kiểm tra và lấy danh sách theo mã  3. Chuyển sang giao diện Lịch  4. Show danh sách lịch học | | |
| **Luồng sự kiện phụ (Alternative Flows):** | |
| |  |  | | --- | --- | | Actor | System | |  |  | | |

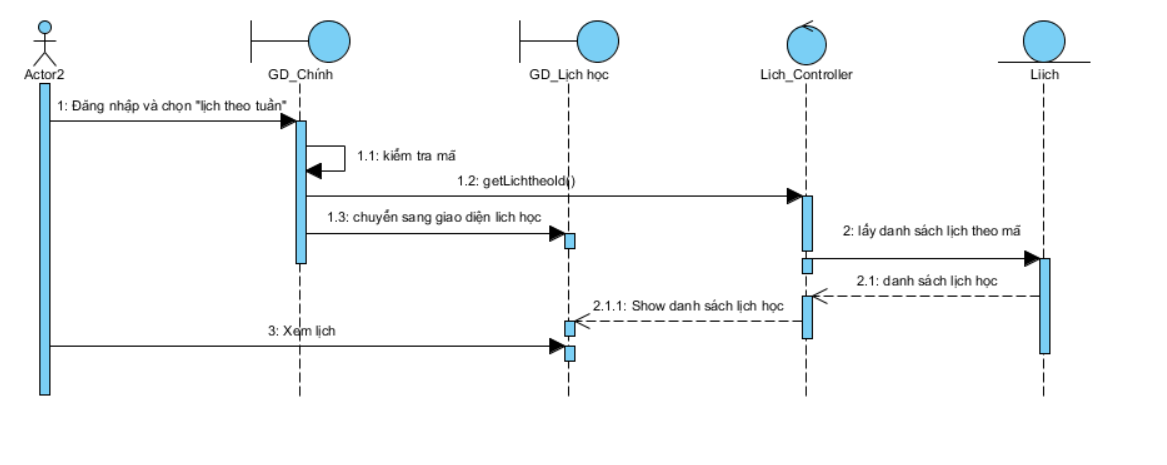
**3.4.2.2 Các sơ đồ**

**a. Sơ đồ Activity Diagram**

****

**Hình 3.4.2.2.1 Sơ đồ Activity**

**b.Sơ đồ Sequence Diagram**

****

**Hình 3.4.2.2. 2 Sơ đồ Sequence Diagram**

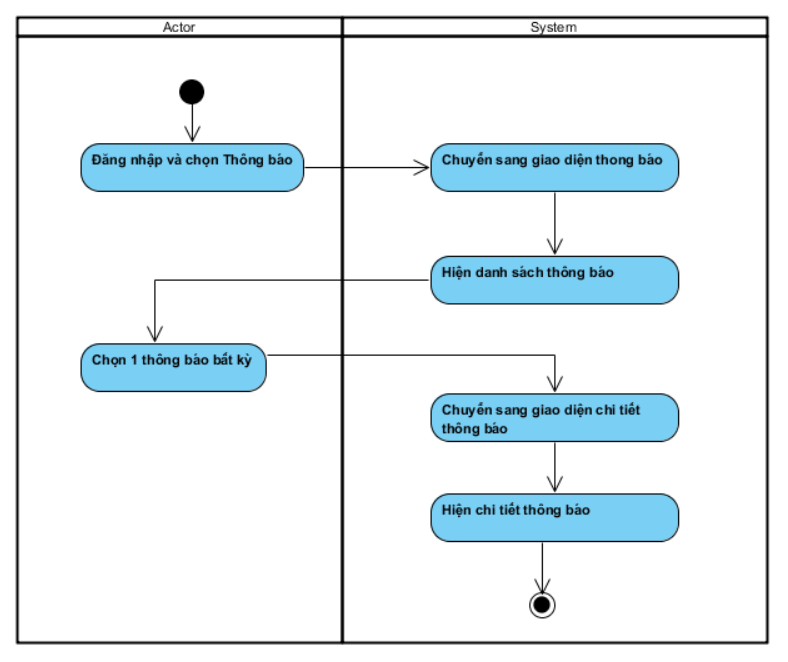
### **3.4.3 UC003 Xem thông báo**

**3.4.3.1 Đặc tả**

|  |  |
| --- | --- |
| **Use case: UC005** | |
| Tên use case | Xem lịch thông báo |
| Actor | Sinh viên |
| Mô tả | Giúp sinh viên có xem các thông báo từ khoa |
| Precondition: | Đăng nhập thành công vào hệ thống |
| Poscondition: | Nếu thực hiện chức năng thành công hiển thị danh sách các thông báo cho sinh viên. Nếu thực hiện thất bại hiện danh sách trống |
| **Luồng sự kiện chính (Basic flows)**   |  |  | | --- | --- | | Actor | System | | 1. Đăng nhập thành công và click chọn thông báo trên thanh menu  4. Click chọn 1 thông báo bất kì  7. Kiểm tra thông báo | 2. Chuyển sang giao diện thông báo  3. Show danh sách các thống báo  5. Chuyển sang giao diện chi tiết thông báo  6. Show chi tiết thông báo | | |
| **Luồng sự kiện phụ (Alternative Flows):** | |
| |  |  | | --- | --- | | Actor | System | |  |  | | |

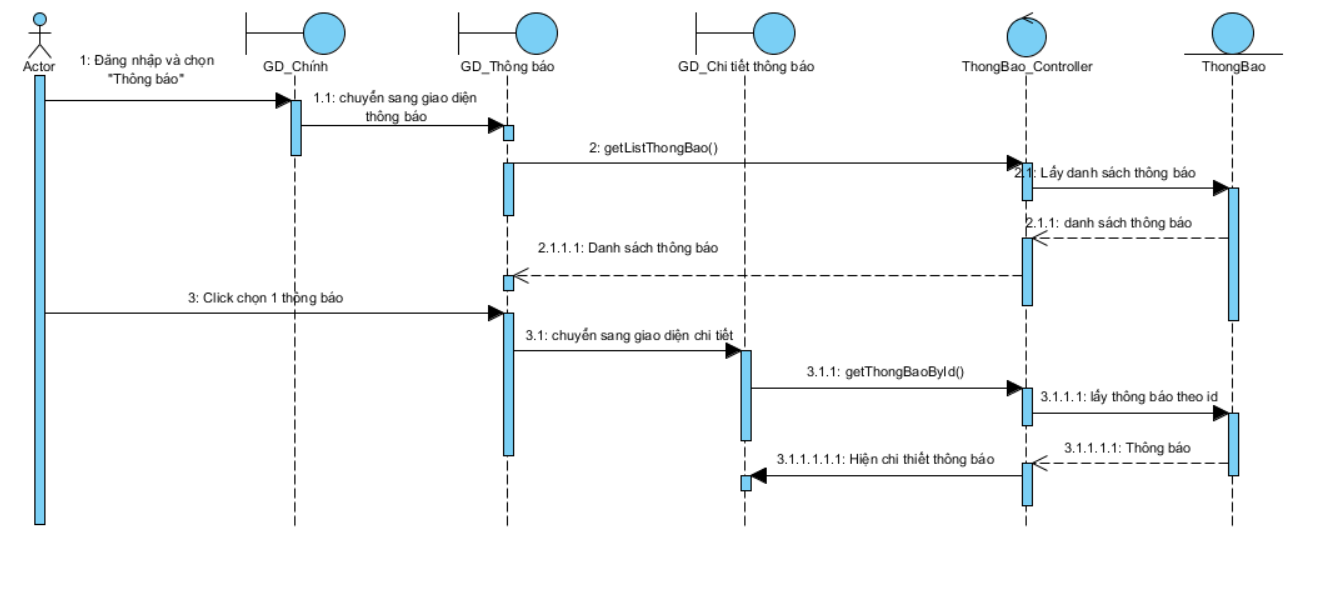
**3.4.3.2 Các sơ đồ**

**a. Sơ đồ Activity Diagram**

****

**Hình 3.4.3.2.1 Sơ đồ Activity**

**b.Sơ đồ Sequence Diagram**

****

**Hình 3.4.3.2.2 Sơ đồ Sequence Diagram**

### 

### **3.4.4 UC004 Đăng kí học phần**

**3.4.4.1 Đặc tả**

|  |  |
| --- | --- |
| **Use case: UC004** | |
| Tên use case | Đăng kí học phần |
| Actor | Sinh viên |
| Mô tả | Giúp sinh viên đăng kí môn học cần thiết theo niên giám |
| Precondition: | Đăng nhập thành công vào hệ thống, có học phần cần đăng kí và đã hoàn thành các học phần tiên quyết nếu có |
| Poscondition: | Nếu thực hiện chức năng thành công thì thông báo thành công, và them học phần đã đăng kí vào danh sách học phần của học kì đó |
| **Luồng sự kiện chính (Basic flows)**   |  |  | | --- | --- | | Actor | System | | 1. Đăng nhập thành công và chọn đăng kí học phần  4. Chọn học kì  5. Chọn 1 môn học phần cần đăng kí  7. Chọn lớp phù hợp và click đăng ký | 2. Chuyển sang giao diện đăng kí học phần  3.1 Hiện danh sách học phần theo học kì  6. Hiển thị các lớp học phần của môn đó  8. Kiểm tra xem đã hoàn thành các học phần tiên quyết và có trùng lịch không  9.1 Thông báo thành công | | |
| **Luồng sự kiện phụ (Alternative Flows):** | |
| |  |  | | --- | --- | | Actor | System | |  | 3.2 Thông báo học kì không được đăng kí  9.2 Thông báo đăng kí thất bại | | |

**3.4.4.2 Các sơ đồ**

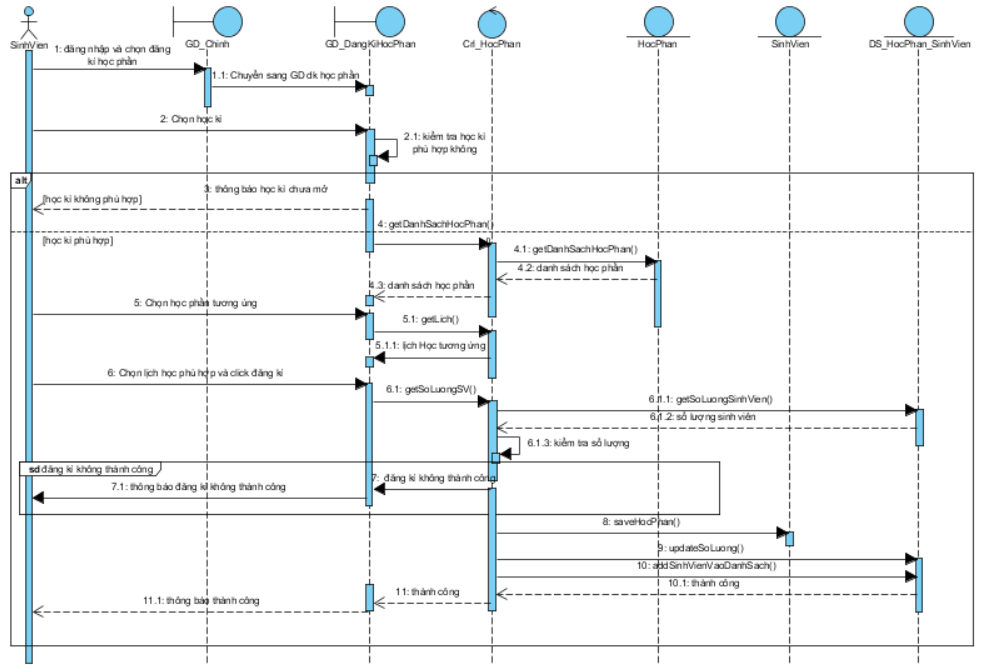
**a. Sơ đồ Activity Diagram**

**A diagram of a system

Description automatically generated with medium confidence**

**Hình 3.4.4.2.1 Sơ đồ Activity**

**b.Sơ đồ Sequence Diagram**

****

**Hình 3.4.4.2.2 Sơ đồ Sequence Diagram**

### 

## **3.5 Class Diagram**

**A computer screen shot of a diagram

Description automatically generated**

**Hình 3.5.1: Sơ đồ Class Diagram**

## **3.6 Kiến trúc Client-Server**

Chọn kiến trúc client-server vì có thể xây dựng nhiều server hoạt động cùng lúc tránh trường hợp nghẽn tăng tốc độ xử lí. Có thể xây dựng 2 hoặc nhiều server chạy đồng thời, khi có nhiều request từ nhiều client khác nhau thì các request sẻ được “load balancer” điều hướng. Ở đây “load balancer” có thể kiểm tra server nào còn hoạt động và có các thuật toán cân bằng tải cho server để server không bị quá tải (Trong trường hợp có server không hoạt động hoặc bảo trì thì website vẫn hoạt động bình thường vì “load balancer” có thể loại trừ server đã chết ra điều này giúp web hoạt động tốt dễ bảo trì, cập nhật và mở rộng). Sau khi request được điều hướng xuống Server xử lí và sau khi xử lí các yêu cầu Server connect với Database. Ở đây Database chọn MongoDB vì có chức năng Replica set tương tự như server khi 1 data mất thì sẻ có 1 data khác thay thế kịp thời

**A diagram of a server

Description automatically generated**

**Hình 3.6: Mô hình kiến trúc**

### **3.6.1. Hoạt động của kiến trúc:**

- **Client:** là giao diện người dùng có thể là Web được xây dựng bằng ngôn ngữ ReactJS + Vite hoặc có thể là Mobile được xây dựng bằng React Navite. Khi người dùng tương tác với giao diện các request sẻ được gửi đi tới Load balancer (nếu có nhiều SERVER hoặc nếu là 1 server thì có thể gửi trực tiếp đến SERVER) bằng http/https (gồm các phương thức GET, POST, PUT, DELETE). Ở đây có thể có nhiều client đồng thời sử dụng như client 1, client2, …

**- Load Balancer:** là môi trường broker trung gian tiếp nhận và điều hướng các request từ client đến với các server phù hợp:

+ Listener: nơi nhận các request từ phía client

+ Health Checked: đây là phần giúp luôn kiểm tra server còn hoạt động và có sẳn sàn nhận request không

+ Schedule: Chứa các thuật toán cân bằng tải giúp chỉ định request sẻ được đưa đến server nào.

Đây là thành phần quan trọng giúp đảm bảo website hoạt động tốt không bị nghẽn và các server hoạt động đồng đều. Thường được đặt trên đám may hoặc các máy ảo mạnh. Tùy theo cấu hình các request sẻ được điều hướng đến các server tương ứng

**- Server:** được viết bằng NodeJS nơinhận các request được điều hướng từ phía load balancer. Nơi xử lý dữ liệu và trả về dữ liệu tương ứng cho người dùng gồm:

+ Controller: nơi nhận request, kiểm tra route và các tham số từ request sau đó gọi gọi các phương thức từ business logic để xử lí.

+ Business Logic: thực hiện các quy tắc nghiệp vụ, kiểm tra tính hợp lệ dữ liệu và xử lí các logic sau đó gọi các phương thức từ Data Access để truy xuất và cập nhật dữ liệu.

+ Data Access: noi tương tác với Database

+ Model: Định nghĩa cấu trúc dữ liệu và quản lý schema

**- Database:** Sử dụng MongoDB để lưu trữ dữ liệu vì MongoDB có chức năng Replica Set. Sau khi nhận yêu cầu từ Data Access sẻ được chuyển tới primary node, ở đây dữ liệu được xử lí theo yêu cầu và save đồng thời lưu vào “OPLOG”. Các secondary node sẻ sao chép “OPLOG” từ primary và tự cập nhật. Nếu Primary node gặp sự cố FAILOVER sẻ chạy để bầu 1 secondary node thành 1 primary node và tiếp tục hoạt động.

### **3.6.2. Ưu điểm:**

- Có thể xây dựng nhiều server giúp giảm số lượng request của server nâng cao hiệu suất và tốc độ.

- Dễ dàng mở rộng khi cần thiết.

- Việc bảo trì hoặc sửa lỗi dễ dàng do có nhiều server có thể chạy đồng thời

### **3.6.3. Nhược điểm:**

- Tạo nhiều server có thể làm kinh phí rất lớn.

- Việc quản lý nhiều server đòi hỏi nhiều kinh nghiệm

- Khó khan khi xây dụng các thuật toán về việc điều tải hợp lí

# **CHƯƠNG 4: HIỆN THỰC**

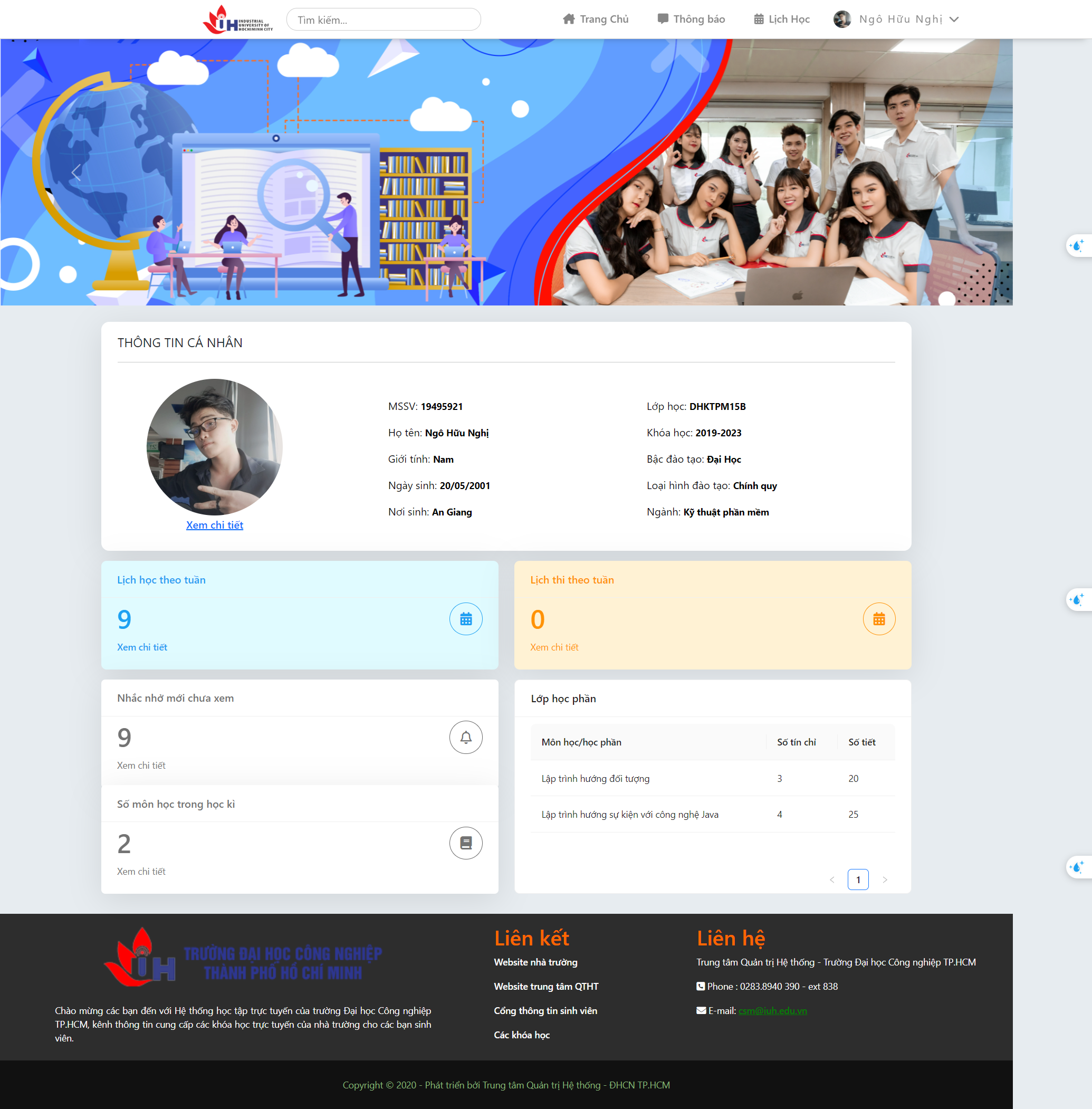
## **4.1 Một số giao diện**

### **4.1.1 Giao diện đăng nhập**



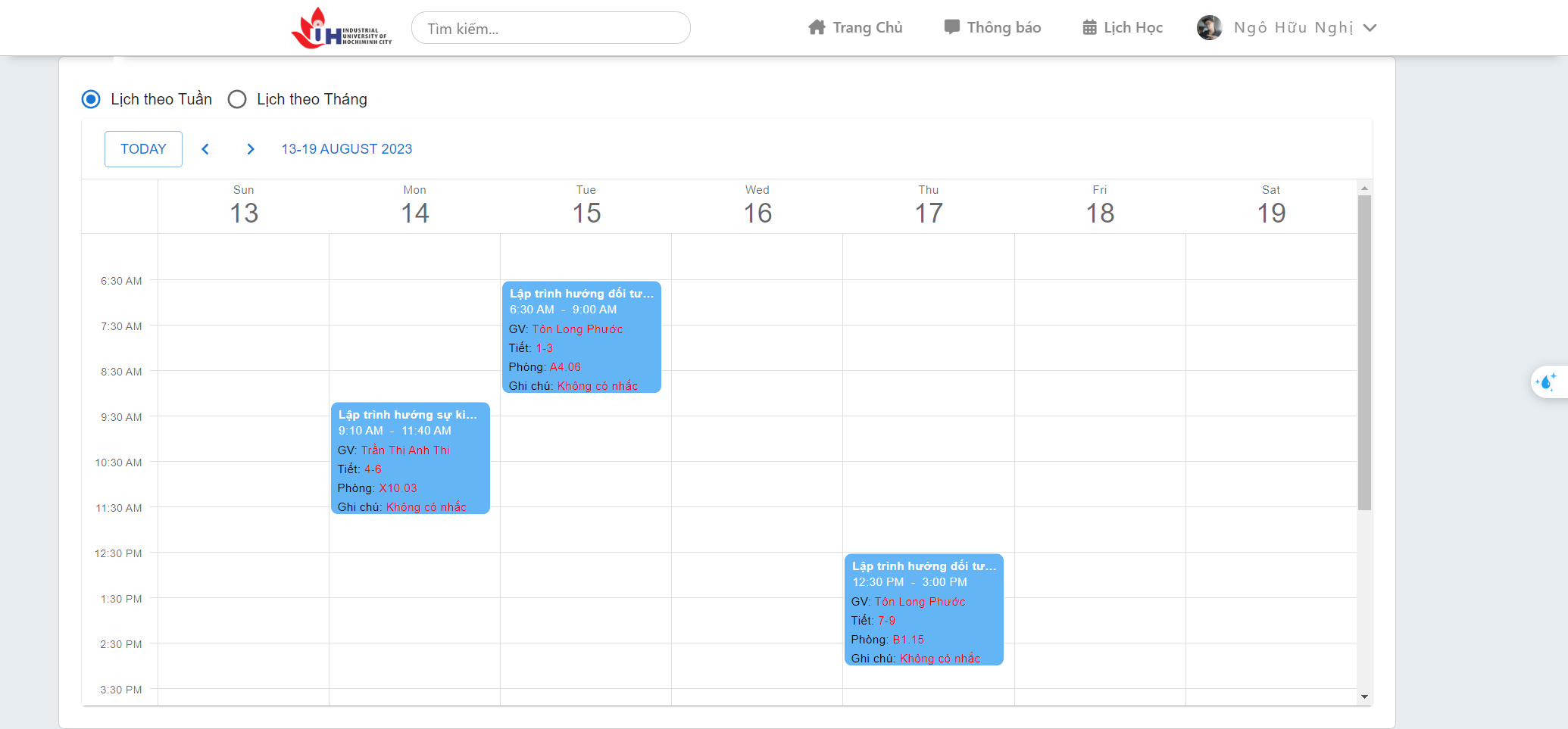
**Hình 4.1.1: Giao diện đăng nhập**

### **4.1.2 Giao diện trang chủ cho sinh viên**



**Hình 4.1.2: Giao diện trang chủ cho sinh viên**

### **4.1.3 Giao diện xem lịch học của sinh viên**



**Hình 4.1.3: Giao diện xem lịch của sinh viên**

### **4.1.4 Giao diện đăng kí học phần**

A screenshot of a computer

Description automatically generated

**Hình 4.1.5: Giao diện đăng kí học phần**

## **4.2 Kiểm thử một số chức năng**

### **4.2.1 Chức năng Đăng nhập**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên testcase | Dữ liệu nhập | | Kết quả mong đợi | Kết quả đạt được |
| Tên tài khoản | Mật khẩu |
| 1 | Vùng hợp lệ | admin | admin | Chuyển sang giao diện chính với quyền Khoa | Đúng |
| 2 | Vùng hợp lệ | 19495921 | 19495921 | Chuyển sang giao diện chính với quyền sinh viên | Đúng |
| 3 | Vùng tên tài khoản sai 1 | “rỗng” | 19509301 | Vui lòng nhập tên tài khoản của bạn | Đúng |
| 4 | Vùng tên tài khoản sai 2 | 1111111 | 19509301 | Thông báo tài khoản không hợp lệ | Đúng |
| 5 | Vùng mật khẩu sai 1 | 19509301 | “rỗng” | Vui lòng nhập mật khẩu của bạn | Đúng |
| 6 | Vùng mật khẩu sai 2 | 01120050 | 01120050 | Chuyển sang giao diện quyền giảng viên | Đúng |

**Bảng 4.2.1 Bảng kiểm thử chức năng đăng nhập**

### **4.2.2 Chức năng tạo thông báo**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên testcase | Dữ liệu nhập | | | | Kết quả mong đợi | Kết quả đạt được |
| Tên | Chi tiết | Ngày tạo | Loại |
| 1 | Vùng hợp lệ 1 | “Thông báo 1” | “Chi tiết thông báo 1” | 25/11/2023 | “sinhvien” | Hợp lệ và thông báo thành công | Đúng |
| 2 | Vùng không hợp lệ 1 | “Rỗng” | “Chi tiết thông báo ” | 25/11/2023 | “sinhvien” | Thông báo nhập tên Thông báo | Đúng |
| 3 | Vùng không hợp lệ 2 | “Thông báo ” | “Rỗng” | 25/11/2023 | “sinhvien” | Báo nhập chi tiết thông báo | Đúng |
| 4 | Vùng không hợp lệ 3 | “Rỗng” | “Rỗng” | 25/11/2023 | “sinhvien” | Thông báo nhập tên Thông báo | Đúng |

**Bảng 4.2.2 Bảng kiểm thử chức năng tạo thông báo**

# **CHƯƠNG 5: KẾT LUẬN**

Báo cáo đã phân tích những chức năng cơ bản mà hệ thống cần thực hiện. Từ bước phân tích thiết kế đến xây dựng hệ thống ứng dụng.Dự án này đã được phát triển với mục đích giúp quản lí học phần và phòng học trong trường trở nên dễ dàng, khách quan và hiệu quả hơn.

Với sự phát triển không ngừng của công nghệ và trí tuệ nhân tạo, dự án này có thể sẽ tiếp tục được cải tiến và phát triển để trở nên tối ưu hơn và có thể áp dụng rộng rãi hơn.

## **5.1 Kết quả đạt được**

**Chức năng chính mà hệ thống đã phát triển bao gồm:**

* Sinh viên: thực hiện được các chức năng xem lịch, xem thông báo và chi tiết thông báo cá nhân , xem thông tin cá nhân, đổi mật khẩu, xem tất cả môn học hiện tại.
* Khoa: thực hiện các chức năng như xem, tạo và xóa thông báo, tạo tài khoản cho từng loại người dùng, cập nhật lịch theo học phần (xem lịch, cập nhật thông tin lịch, tạm ngưng lịch), quản lí phòng theo loại (thêm, cập nhật thông tin phòng , xem lịch của phòng, tạo cuộc họp mới, bảo trì phòng và chuyển lịch của phòng cũ sang phòng mới, xem thông tin chi tiết phòng và thiết bị của phòng)

**Ưu điểm:**

* Giao diện dễ dàng sử dụng.
* Tổng quan hệ thống đạt mức cơ sở đã có được những chức năng hệ thông cần.
* Áp dụng những công nghệ mới, hiện đại
* Khả năng mở rộng cao

**Nhược điểm:**

* Chưa phát triển trên các nền tảng khác nhau và code chưa tối ưu hiệu suất.
* Chưa thể triển khai thành nhiều server khác nhau
* Thành viên nhóm không có nhiều thời gian để khảo sát và hiện thực đề tài.
* Một số chức năng chưa được hoàn thiện theo ý định ban đầu.

## **5.2 Hướng phát triển**

**-** Kết hợp các chức năng quản lý học phần và phòng học 1 cách logic hơn và phù hợp hơn

- Tối ưu hóa và điều chỉnh nghiệp vụ một cách chặt chẽ hơn sao cho phù hợp với thực tế hiện nay.

- Phát triển hệ thống trên ứng dụng mobile với các giao diện đồng bộ. Các chức năng của hệ thống sẽ được tích hợp để người dùng dễ dàng sử dụng mọi nơi.

# **TÀI LIỆU THAM KHẢO**

**Các tài liệu Tiếng Việt**

[1]. E. Brown. (2020). JavaScript: the Complete Refernce. McGraw-Hill Education

**Các tài liệu Tiếng Anh**

[2]. R. E. Johnson, R. Helms, and C. D. Johnson. (2019). Learning React: Functional Web Development with React and Redux. Packt Publishing

**Các tài liệu từ Internet**

[3]. Git scm –fast-version-control. Ngày 26/08/2023. <https://git-scm.com/>

[4]. Typescript lang. Ngày 28/08/2023. <https://www.typescriptlang.org/>

[5]. Topdev MongoDB là gì? Ngày 29/08/2023. <https://topdev.vn/blog/mongodb-la-gi/>

[6]. MongoDB Cloud. Ngày 29/08/2023. <https://www.mongodb.com/cloud>

[7]. Vite cho người mới. Ngày 05/09/2023 https://vitejs.dev/guide/