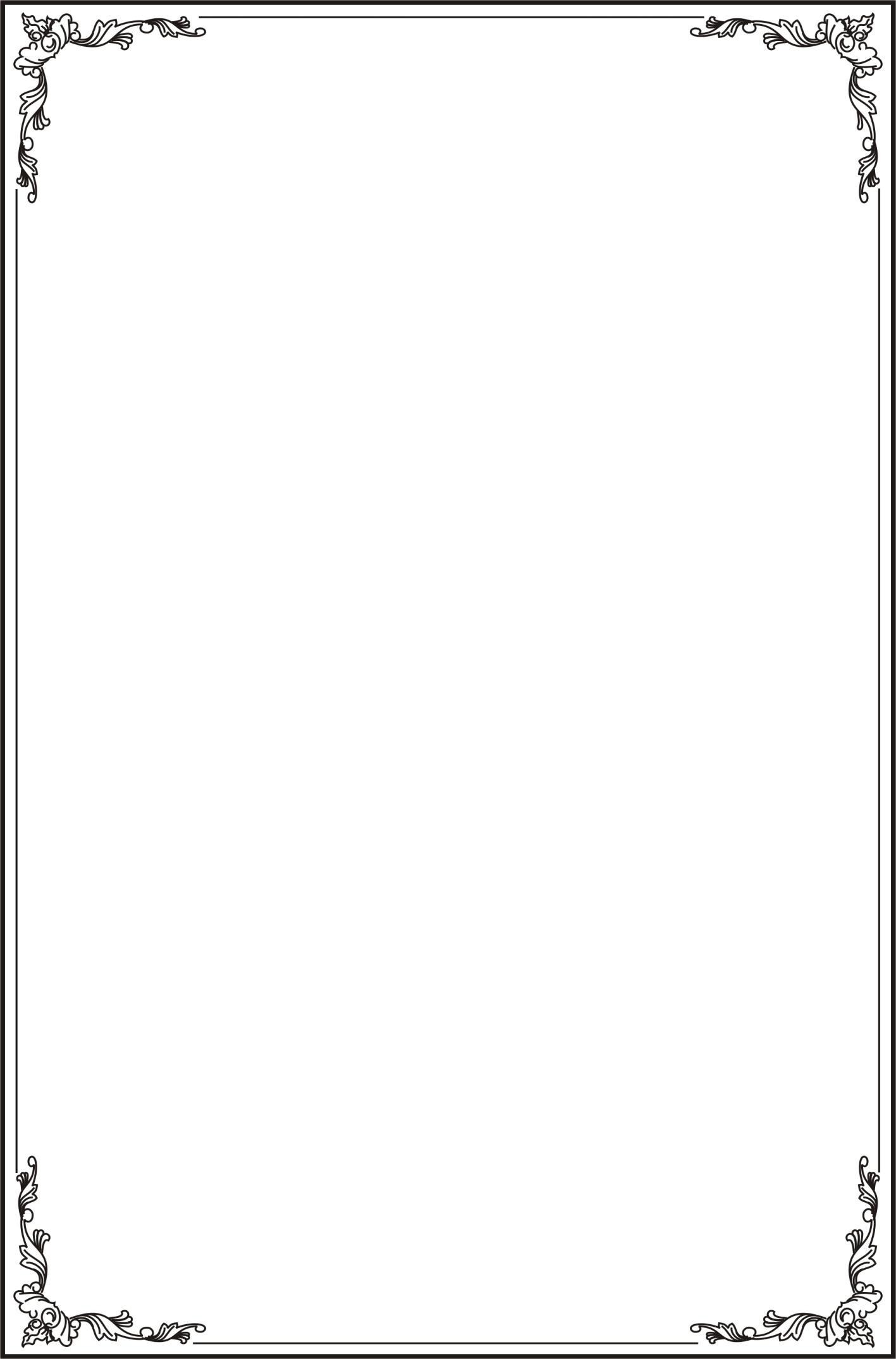
**A picture containing text

Description automatically generated**

**BÁO CÁO TIỂU LUẬN**

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

**ĐỀ TÀI: ỨNG DỤNG ĐĂNG KÝ HỌC PHẦN**

**Giảng viên hướng dẫn: TS. Võ Văn Hải**

**Các thành viên thực hiện:**

1. Nguyễn Huy Hoàng - 21084321

2. Đỗ Chí Tường - 21129041

3. Nguyễn Hồ Đăng Quang - 21054971

**Hồ Chí Minh, 13 tháng 5 năm 2024**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**MỤC LỤC**

[**BẢNG PHÂN CÔNG NHIỆM VỤ** 3](#_Toc166512669)

[**CHƯƠNG 1: GIỚI THIỆU CHUNG** 4](#_Toc166512670)

[**1.1** **Mục đích** 4](#_Toc166512671)

[**1.2** **Đặc tả** 4](#_Toc166512672)

[**1.3** **Phân tích và thu thập yêu cầu** 5](#_Toc166512673)

[**1.3.1 Yêu cầu chức năng** 5](#_Toc166512674)

[**1.3.2 Yêu cầu phi chức năng:** 6](#_Toc166512675)

[**1.4** **Kiến trúc hệ thống** 6](#_Toc166512676)

[**CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ** 10](#_Toc166512677)

[**2.1** **Sơ đồ UseCase** 10](#_Toc166512678)

[**2.2** **Sơ đồ Class** 10](#_Toc166512679)

[**2.3** **Đặc tả Đăng ký học phần** 10](#_Toc166512680)

[**2.4** **Sơ đồ Activity** 10](#_Toc166512681)

[**2.5** **Sơ đồ Sequence** 10](#_Toc166512682)

[**2.6** **Sơ đồ Statechart** 10](#_Toc166512683)

[**2.7** **Sơ đồ Deployment** 10](#_Toc166512684)

[**CHƯƠNG 3: HIỆN THỰC** 11](#_Toc166512685)

[**3.1 Cấu hình phần cứng, phần mềm** 11](#_Toc166512686)

[**3.2 Giao diện của hệ thống** 11](#_Toc166512687)

[**CHƯƠNG 4: KẾT LUẬN** 12](#_Toc166512688)

[**4.1 Kết quả đạt được** 12](#_Toc166512689)

[**4.2 Hạn chế của đồ án** 12](#_Toc166512690)

[**4.3 Hướng phát triển** 12](#_Toc166512691)

[**TÀI LIỆU THAM KHẢO** 13](#_Toc166512692)

# **BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ tên** | **Nhiệm vụ** |
| 21084321 | Nguyễn Huy Hoàng | Code backend, làm file báo cáo |
| 21129041 | Đỗ Chí Tường | Code frontend |
| 21054971 | Nguyễn Hồ Đăng Quang | Code frontend |

# **CHƯƠNG 1: GIỚI THIỆU CHUNG**

* 1. **Mục đích**

Mục đích chính của ứng dụng phần mềm đăng ký học phần của trường đại học là tạo ra một hệ thống linh hoạt và tiện lợi để sinh viên có thể đăng ký các môn học một cách dễ dàng và nhanh chóng. Điều này giúp tối ưu hóa quy trình đăng ký học phần, giảm thiểu sự phức tạp và thời gian mất mát, cũng như tăng cường tính minh bạch và công bằng trong việc phân chia cơ hội học tập cho sinh viên. Ngoài ra, ứng dụng cũng có thể cung cấp thông tin chi tiết về các môn học, giúp sinh viên lựa chọn môn học phù hợp với lịch trình và mục tiêu học tập của mình. Qua đó, ứng dụng đóng vai trò quan trọng trong việc tối ưu hóa trải nghiệm học tập của sinh viên và nâng cao chất lượng giáo dục tại trường đại học.

* 1. **Đặc tả**

Một trường đại học cần quản lý hệ thống đăng ký học phần của sinh viên 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 gồm có thông tin cá nhân(mã số sinh viên, họ tên, giới tính, ngày sinh, email, số điện thoại, khoa). Mỗi 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.

Mỗi lớp học gồm có tên lớp danh nghĩa, loại lớp(lý thuyết hoặc thực hành. Nếu là lớp thực hành thì phải thêm mã lớp lý thuyết), trạng thái(Đang mở hoặc đã đóng. Nếu đã đóng thì đồng nghĩa với việc nhà trường chấp nhận mở lớp), sĩ số tối đa, sĩ số hiện tại, ngày bắt đầu và ngày kết thúc. Mỗi lớp có một giảng viên và nhiều sinh viên.

Mỗi giảng viên gồm có thông tin cá nhân(mã số giảng viên, họ tên, giới tính, ngày sinh, email, số điện thoại, khoa). Mỗi giảng viên có thể dạy một hoặc nhiều lớp học.

Mỗi môn học gồm có tên môn học, mô tả, số tín chỉ, môn tiên quyết. 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 và nhà trường chấp nhận mở lớp học phần, 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. **Phân tích và thu thập yêu cầu**

### **1.3.1 Yêu cầu chức năng**

- Đối với sinh viên:

+ Sinh viên có thể xem lịch học

+ Sinh viên có thể xem, cập nhật thông tin cá nhân

+ Sinh viên có thể thực hiện đăng ký học phần

- Đối với giảng viên:

+ Giảng viên có thể xem lịch dạy học

+ Giảng viên có thể xem, cập nhật thông tin cá nhân

- Đối với người quản trị hệ thống:

+ Người quản trị hệ thống có thể thêm, xoá, sửa sinh viên, giảng viên, môn học, lớp học phần, lịch học.

### **1.3.2 Yêu cầu phi chức năng:**

* **Giao diện: thân thiện, dễ nhìn, dễ sử dụng.**
* **Bảo mật:** Đảm bảo tính bảo mật cho dữ liệu người dùng thông qua các biện pháp bảo mật như mã hóa dữ liệu, xác thực người dùng, quản lý phiên đăng nhập và kiểm soát truy cập.
* **Hiệu suất:** Đảm bảo ứng dụng hoạt động một cách mượt mà và hiệu quả, đặc biệt là trong việc xử lý lượng truy cập lớn và truy cập đồng thời từ nhiều người dùng.
* **Tính sẵn sàng và tin cậy:** Đảm bảo ứng dụng luôn sẵn sàng hoạt động và không gặp sự cố đáng kể, bao gồm cả việc sửa lỗi và bảo trì hệ thống định kỳ.
  1. **Kiến trúc hệ thống**

Hệ thống sử dụng Client-Server Architecture hoặc Two-tier Architecture

A diagram of a computer network

Description automatically generated

* **Client:** là một thiết bị hoặc ứng dụng phần mềm, bắt đầu liên lạc bằng cách yêu cầu các dịch vụ hoặc tài nguyên từ máy chủ. Hãy nghĩ đến việc trình duyệt web của bạn yêu cầu một trang web hoặc điện thoại thông minh của bạn tìm nạp các bản cập nhật từ server.
* **Server:** là một hệ thống hoặc ứng dụng mạnh mẽ được thiết kế để đáp ứng các yêu cầu của khách hàng. Nó xử lý dữ liệu, thực hiện tính toán và đóng vai trò là trung tâm để quản lý nhiều tương tác của khách hàng.
* **Về công nghệ sử dụng:**

+ Client-side(Front-end): ReactJS

+ Server-side(Back-end): Spring boot, Spring Data JPA, MariaDB

* **Ưu điểm:**

*+ Deployability*: Kiến trúc tạo điều kiện triển khai dễ dàng các ứng dụng trên nhiều client vì client chỉ cần cài đặt phần mềm nhẹ. Kiến trúc này cho phép quản lý tập trung các bản cập nhật trên server, giảm bớt sự phức tạp khi triển khai.

+ *Elasticity*: Kiến trúc có thể được thiết kế để mở rộng quy mô tài nguyên một cách linh hoạt dựa trên nhu cầu. Server có thể được cấu hình để xử lý các tải khác nhau bằng cách thêm hoặc xóa tài nguyên khi cần, đảm bảo hiệu suất và sử dụng tài nguyên tối ưu.

+ *Evolutionary*: Nó cho phép dễ dàng tiến hóa và nâng cấp các thành phần hệ thống mà không làm gián đoạn toàn bộ hệ thống. Khi công nghệ phát triển, các bản cập nhật và cải tiến có thể được thực hiện một cách độc lập đối với các thành phần server và client, cho phép áp dụng dần dần các tính năng mới.

+ *Fault tolerance*: Kiến trúc cung cấp các cơ chế chịu lỗi tích hợp. Dự phòng có thể được triển khai ở cấp máy chủ để đảm bảo tính sẵn sàng và tính toàn vẹn dữ liệu cao, giảm thiểu tác động của lỗi phần cứng hoặc gián đoạn mạng.

+ *Modularity*: Kiến trúc thúc đẩy tính mô đun và phát triển dựa trên thành phần. Bằng cách tách biệt các thành phần client và server, nhà phát triển có thể tập trung vào việc xây dựng mã mô-đun và có thể tái sử dụng, nâng cao khả năng bảo trì và khả năng mở rộng.

+ *Overall cost*: Kiến trúc có thể giảm chi phí tổng thể bằng cách tập trung tài nguyên và quản lý. Server tập trung cho phép sử dụng tài nguyên hiệu quả, bảo trì đơn giản và giảm yêu cầu phần cứng so với kiến ​​trúc phân tán.

+ *Performance*: Nó cho phép sử dụng hiệu quả các nguồn tài nguyên và hiệu suất được tối ưu hóa. Kiến trúc này cho phép các tài nguyên máy chủ chuyên dụng xử lý các tác vụ xử lý chuyên sâu, giúp cải thiện thời gian phản hồi và hiệu suất tổng thể của hệ thống.

+ *Reliability*: Kiến trúc nâng cao độ tin cậy thông qua quản lý tập trung và dự phòng. Bằng cách tập trung các chức năng quan trọng trên máy chủ và thực hiện các biện pháp dự phòng, hệ thống có thể duy trì mức độ tin cậy cao ngay cả khi gặp lỗi.

+ *Scalability*: Kiến trúc hỗ trợ khả năng mở rộng theo chiều ngang và chiều dọc. Server có thể được mở rộng quy mô (tỉ lệ theo chiều dọc) bằng cách nâng cấp phần cứng hoặc thu nhỏ quy mô (tỉ lệ theo chiều ngang) bằng cách thêm nhiều server hơn để phân phối khối lượng công việc, đảm bảo khả năng mở rộng nhằm đáp ứng nhu cầu ngày càng tăng.

+ *Simplicity*: Nó mang lại sự đơn giản trong thiết kế và quản lý hệ thống. Việc phân tách rõ ràng trách nhiệm của client và server giúp đơn giản hóa kiến ​​trúc, phát triển và bảo trì hệ thống, dẫn đến việc khắc phục sự cố và nâng cấp dễ dàng hơn.

+ *Testability*: Kiến trúc tạo điều kiện cho việc kiểm tra và gỡ lỗi hiệu quả. Với các giao diện được xác định rõ ràng giữa client và server, việc tách biệt và kiểm tra các thành phần riêng lẻ trở nên dễ dàng hơn, đảm bảo kiểm soát chất lượng tốt hơn và chu kỳ phát triển nhanh hơn.

* **Nhược điểm:**

+ *Deployability*: Việc triển khai có thể trở nên phức tạp đối với các ứng dụng, đặc biệt nếu phát sinh vấn đề tương thích. Việc quản lý các phiên bản khác nhau của phần mềm trên nhiều nền tảng và thiết bị có thể gây ra những thách thức khi triển khai và các vấn đề về khả năng tương thích.

+ *Elasticity*: Việc mở rộng quy mô tài nguyên server một cách linh hoạt có thể yêu cầu các cơ chế quản lý tài nguyên và cân bằng tải phức tạp. Để đạt được tính linh hoạt liền mạch trong kiến ​​trúc client-server đòi hỏi phải lập kế hoạch và triển khai cẩn thận các chiến lược phân bổ tài nguyên động, điều này có thể làm tăng thêm độ phức tạp cho hệ thống.

+ *Evolutionary*: Sự phát triển của kiến ​​trúc có thể yêu cầu cập nhật cho cả thành phần client và server, dẫn đến những thách thức về đồng bộ hóa. Việc giới thiệu các tính năng hoặc công nghệ mới có thể cần phải cập nhật cho cả phần mềm client và server, có khả năng gây ra sự cố tương thích và yêu cầu các bản phát hành được đồng bộ hóa.

+ *Fault tolerance*: Việc triển khai các cơ chế chịu lỗi sẽ làm tăng thêm độ phức tạp và chi phí chung cho hệ thống. Cơ chế dự phòng và chuyển đổi dự phòng yêu cầu nguồn lực bổ sung và chi phí quản lý, điều này có thể làm tăng độ phức tạp và chi phí của hệ thống.

+ *Modularity*: Quá phụ thuộc vào giao tiếp giữa client và server có thể dẫn đến sự liên kết chặt chẽ giữa các thành phần. Việc ghép nối quá mức giữa các thành phần client và server có thể cản trở tính mô đun hóa và khiến hệ thống khó bảo trì và phát triển hơn.

+ *Overall cost*: Chi phí thiết lập và bảo trì ban đầu cho server có thể rất lớn. Việc thiết lập và duy trì cơ sở hạ tầng server tập trung, bao gồm phần cứng, phần mềm và kết nối mạng, có thể phải chịu chi phí trả trước cao so với kiến ​​trúc ngang hàng hoặc kiến ​​trúc phân tán.

+ *Performance*: Hiệu suất có thể bị ảnh hưởng bởi độ trễ mạng và giới hạn băng thông. Giao tiếp giữa client và server qua mạng gây ra các hạn chế về độ trễ và băng thông, điều này có thể ảnh hưởng đến khả năng phản hồi và thông lượng theo thời gian thực, đặc biệt là trong môi trường phân tán.

+ *Reliability*: Sự phụ thuộc vào các server tập trung làm tăng nguy cơ xảy ra lỗi ở một điểm. Nếu server bị lỗi hoặc ngừng hoạt động, nó có thể làm gián đoạn quyền truy cập vào dịch vụ của tất cả khách hàng, có khả năng dẫn đến thời gian ngừng hoạt động và mất dữ liệu đáng kể.

+ *Scalability*: Việc mở rộng quy mô tài nguyên server có thể đạt đến giới hạn do hạn chế về phần cứng hoặc tắc nghẽn kiến ​​​​trúc. Mặc dù kiến ​​trúc client-server hỗ trợ triển khai có thể mở rộng nhưng có thể có những hạn chế thực tế đối với việc mở rộng quy mô tài nguyên server, chẳng hạn như giới hạn phần cứng hoặc tắc nghẽn kiến ​​trúc.

+ *Simplicity*: Độ phức tạp có thể tăng lên khi hệ thống phát triển về quy mô và độ phức tạp. Việc quản lý một hệ thống client-server lớn và phức tạp với nhiều client, server và các thành phần được kết nối với nhau có thể ngày càng trở nên khó khăn hơn, dẫn đến độ phức tạp và chi phí bảo trì cao hơn.

+ *Testability*: Việc kiểm thử các hệ thống phân tán có thể phức tạp hơn và yêu cầu môi trường kiểm thử chuyên biệt. Việc kiểm tra sự tương tác giữa các thành phần client và server, đặc biệt là trong môi trường phân tán, có thể khó khăn hơn so với việc kiểm tra các hệ thống độc lập, đòi hỏi các công cụ và môi trường kiểm tra chuyên dụng.

# **CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ**

* 1. **Usecase diagram**

A diagram of a company

Description automatically generated

* 1. **Class diagram**

A diagram of a computer program

Description automatically generated with medium confidence

* 1. **Đặc tả**

|  |  |
| --- | --- |
| * **Tên use case**: Đăng ký học phần | |
| * **Mô tả sơ lược**: Sinh viên đăng ký lớp học phần mà mình muốn đăng ký | |
| * **Actor chính**: Sinh viên | |
| * **Tiền điều kiện (Pre-condition):** Sinh viên đăng nhập vào thành công | |
| * **Hậu điều kiện (Post-condition):** Sinh viên đăng ký thành công và dữ liệu được lưu vào cơ sở dữ liệu | |
| * **Luồng sự kiện chính (main flow):** | |
| **Sinh viên** | **Hệ thống** |
| 1. Sinh viên chọn học kỳ |  |
| 1. Sinh viên chọn môn học |  |
| 1. Sinh viên chọn lớp học phần |  |
| 1. Sinh viên nhấn Đăng ký học phần |  |
|  | 1. Hệ thống kiểm tra môn học tiên quyết |
|  | 1. Hệ thống kiểm tra trạng thái lớp học phần đang mở đăng ký |
|  | 1. Hệ thống kiểm tra sĩ số đăng ký |
|  | 1. Hệ thống lưu vào cơ sở dữ liệu, cập nhập sĩ số lớp học phần và hiển thị thông báo đăng ký thành công. |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 5.1 Hệ thống hiển thị thông báo Môn học tiên quyết chưa học. |
|  | 6.1 Hệ thống hiển thị thông báo Lớp học phần đã đóng đăng ký. |
|  | 7.1 Hệ thống hiển thị thông báo Sĩ số đã đầy. |

* 1. **Activity diagram**

A diagram of a flowchart

Description automatically generated

* 1. **Sequence diagram**

A diagram of a project

Description automatically generated

* 1. **Statechart diagram**

A diagram of a course

Description automatically generated

* 1. **Component diagram**

A diagram of a course register system

Description automatically generated

* 1. **Deployment diagram**

A diagram of a computer application

Description automatically generated

**CHƯƠNG 3: HIỆN THỰC**

## **Cấu hình phần cứng, phần mềm**

* **Phần cứng**
* Máy tính ít nhất 4 GB RAM.
* Hệ điều hành Win 10 trở lên.
* Độ phân giải tối thiểu 1366 x 768 pixel trở lên.
* **Phần mềm**
* Kết nối mạng: Cần có kết nối Internet ổn định và đủ băng thông.

## **3.2** **Giao diện của hệ thống**

### **3.2.1 Giao diện đăng nhập**

A screenshot of a computer

Description automatically generated

### **3.2.2 Giao diện sinh viên**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*Giao diện đăng ký học phần*

A white background with black text

Description automatically generated

*Giao diện lịch học*

A close-up of a person

Description automatically generated

*Giao diện thông tin cá nhân*

### **3.2.3 Giao diện giảng viên**

A screenshot of a computer

Description automatically generated

*Giao diện lịch dạy học*

A close-up of a person

Description automatically generated

*Giao diện thông tin cá nhân*

### **3.2.4 Giao diện quản trị viên**

A screenshot of a computer

Description automatically generated

*Giao diện quản lý sinh viên*

A screenshot of a computer

Description automatically generated

*Giao diện quản lý giảng viên*

A screenshot of a computer

Description automatically generated

*Giao diện quản lý môn học*

A screenshot of a computer

Description automatically generated

*Giao diện quản lý lớp học phần*

A screenshot of a computer

Description automatically generated

*Giao diện quản lý lịch học*

**CHƯƠNG 4: KẾT LUẬN**

**4.1 Kết quả đạt được**

Nhóm đã hoàn thành các chức năng đã đặt ra như:

* Đối với sinh viên:

- Có thể thực hiện đăng nhập tài khoản, cập nhật thông tin cá nhân.

- Có thể đăng ký lớp học phần với môn học mà mình muốn.

- Có thể xem lịch học mỗi tuần

* Đối với giảng viên:

- Có thể thực hiện đăng nhập tài khoản, cập nhật thông tin cá nhân.

- Có thể xem lịch dạy học mỗi tuần

* Đối với quản trị viên:

- Có thể quản lý sinh viên, giảng viên, môn học, lớp học phần và lịch học

**4.2 Hạn chế của đồ án**

Đồ án chưa thể hoàn thành 1 số mục tiêu đã đề ra, một số chứ năng còn thiếu như:

**-** Mỗi môn học có nhiều môn tiên quyết

**-** Chưa có điểm số đánh giá sau khi hoàn thành môn học.

**-** Chưa có chức năng xem điểm và số tín chỉ đã học.

**4.3 Hướng phát triển**

Tiếp tục hoàn thiện các chức năng còn thiếu và một số lỗi ràng buộc.

# **TÀI LIỆU THAM KHẢO**

1. <https://react.dev/learn> (05/2024)

[2]. <https://spring.io/projects/spring-boot> (05/2024)

[3] <https://spring.io/projects/spring-data> (05/2024)

[4] <https://mariadb.org/> (05/2024)