

**TRƯỜNG ĐẠI HỌC SAO ĐỎ**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO**

**ĐỒ ÁN CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI: NGHIÊN CỨU HỆ THỐNG HỖ TRỢ MỘT CỬA CHO SINH VIÊN ĐẠI HỌC SAO ĐỎ**

|  |  |
| --- | --- |
| **Nhóm tác giả:** | 1. Tăng Quang Nghĩa |
|  | 2. Nguyễn Chí Thọ |
| **Lớp:** | DK12\_CNTT2 |
| **Giảng viên giảng dạy:** | Phạm Thị Hường |
| **Đơn vị:** | Khoa Công nghệ thông tin |

**MỤC LỤC**

[1. Lý do chọn đề tài 2](#_Toc195391991)

[2. Mục đích đề tài 2](#_Toc195391992)

[3. Đối tượng nghiên cứu 2](#_Toc195391993)

[4. Phạm vi nghiên cứu 2](#_Toc195391994)

[5. Phương pháp nghiên cứu 3](#_Toc195391995)

[6. Ý nghĩa thực tiễn của đề tài 3](#_Toc195391996)

[7. Bố cục của đề tài 3](#_Toc195391997)

[CHƯƠNG 1. CƠ SỞ LÍ THUYẾT 4](#_Toc195391998)

[1.1. Tổng quan Javascript 4](#_Toc195391999)

[*1.1.1. Khái niệm* 4](#_Toc195392000)

[*1.1.2. Lịch sử của Javascript* 5](#_Toc195392001)

[*1.1.3. Ưu điểm của Javascript* 6](#_Toc195392002)

[*1.1.4. Nhược điểm của Javascript* 6](#_Toc195392003)

[1.2. Giới thiệu về ReactJs 7](#_Toc195392004)

[*1.2.1. Khái niệm* 7](#_Toc195392005)

[*1.2.2. Những lợi ích của ReactJS* 8](#_Toc195392006)

[*1.2.3. Các tính năng nổi bật của ReactJS* 8](#_Toc195392007)

[*1.2.4. Cách sử dụng ReactJS trong phát triển web* 9](#_Toc195392008)

[*1.2.5. Các dự án sử dụng ReactJS trong thực tế* 10](#_Toc195392009)

[1.3. Giới thiệu về TailwindCss 11](#_Toc195392010)

[*1.3.1. Đặc điểm nổi bật của TailwindCss* 11](#_Toc195392011)

[*1.3.2 Ưu điểm của TailwindCss* 11](#_Toc195392012)

[*1.3.3. Nhược điểm của TailwindCss* 12](#_Toc195392013)

[1.4. Giới thiệu về Ant Design 12](#_Toc195392014)

[*1.4.1. Đặc điểm nổi bật của Ant Design* 12](#_Toc195392015)

[*1.5.2. Lợi ích khi sử dụng cho dự án thực tế* 13](#_Toc195392016)

[*1.5.3. Ứng dụng cụ thể trong hệ thống* 13](#_Toc195392017)

[1.5. SQL Server 14](#_Toc195392018)

[*1.5.1. Các thành phần của SQL Server* 14](#_Toc195392019)

[*1.5.2. Các tính năng nâng cấp của SQL Server* 15](#_Toc195392020)

[1.6. NodeJs 15](#_Toc195392021)

[*1.6.1. Cách hoạt động của NodeJS* 16](#_Toc195392022)

[*1.6.2. Ứng dụng của NodeJs* 17](#_Toc195392023)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 19](#_Toc195392024)

[2.1.Mô tả đề tài 19](#_Toc195392025)

[*2.1.1. Giới thiệu* 19](#_Toc195392026)

[*2.1.2. Bối cảnh và thực trạng hiện tại* 19](#_Toc195392027)

[*2.1.3. Tính cấp thiết của đề tài* 20](#_Toc195392028)

[*2.1.4. Yêu cầu đặt ra đối với hệ thống mới* 20](#_Toc195392029)

[*2.1.5. Phương pháp triển khai* 21](#_Toc195392030)

[*2.1.6. Đối tượng sử dụng và lợi ích mang lại* 23](#_Toc195392031)

[2.2. Thiết kế cơ sở dữ liệu 23](#_Toc195392032)

[*2.2.1. Mô tả chi tiết các bảng* 23](#_Toc195392033)

[*2.2.2. Sơ đồ quan hệ* 27](#_Toc195392034)

[2.3. Thiết kế Back-end 28](#_Toc195392035)

[*2.3.1. Tổng quan kiến trúc* 28](#_Toc195392036)

[*2.3.2. Chi tiết các thành phần trong kiến trúc Backend* 29](#_Toc195392037)

[2.4. Thiết kế Front-end 29](#_Toc195392038)

[*2.2.1. Tổng quan kiến trúc* 29](#_Toc195392039)

[*2.2.2. Công nghệ sử dụng* 30](#_Toc195392040)

[*2.2.3. Quy trình thiết kế giao diện* 30](#_Toc195392041)

[*2.2.4. Chi tiết các thành phần trong Front-end* 30](#_Toc195392042)

[*2.2.5. Tối ưu hóa giao diện* 31](#_Toc195392043)

[CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH 32](#_Toc195392044)

[3.1. Giao diện quản trị 32](#_Toc195392045)

[3.2. Giao diện người dùng 35](#_Toc195392046)

[KẾT LUẬN 43](#_Toc195392047)

[1. Kết quả đạt được 43](#_Toc195392049)

[2. Hạn chế 43](#_Toc195392050)

[3. Hướng phát triển 43](#_Toc195392051)

[TÀI LIỆU THAM KHẢO 44](#_Toc195392052)

**DANH MỤC HÌNH ẢNH**

[Bảng 2.2 Bảng Students 24](#_Toc195391953)

[Bảng 2.3 Bảng documents 25](#_Toc195391954)

[Bảng 2.4 Bảng loại yêu cầu 26](#_Toc195391955)

[Bảng 2.5 Bảng yêu cầu 26](#_Toc195391956)

[Hình 2.1: Sơ đồ quan hệ 27](#_Toc195391957)

[Hình 3.1: Giao diện quản lý tài liệu 32](#_Toc195391958)

[Hình 3.2: Giao diện thêm tài liệu 32](#_Toc195391959)

[Hình 3.3: Giao diện quản lý sinh viên 33](#_Toc195391960)

[Hình 3.4: Giao diện thêm sinh viên 33](#_Toc195391961)

[Hình 3.5: Giao diện loại yêu cầu 34](#_Toc195391962)

[Hình 3.6: Giao diện thêm loại yêu cầu 34](#_Toc195391963)

[Hình 3.7: Giao diện yêu cầu 35](#_Toc195391964)

[Hình 3.8: Giao diện đăng nhập 35](#_Toc195391965)

[Hình 3.9: Giao diện header và slider 36](#_Toc195391966)

[Hình 3.10: Giao diện phần giới thiệu sơ qua về trường 36](#_Toc195391967)

[Hình 3.11: Giao diện các câu slogan 37](#_Toc195391968)

[Hình 3.12: Giao diện footer 37](#_Toc195391969)

[Hình 3.13: Giao diện giới thiệu 38](#_Toc195391970)

[Hình 3.14: Giao diện giới thiệu thành tựu 38](#_Toc195391971)

[Hình 3.15: Giao diện danh sách các thủ tục 39](#_Toc195391972)

[Hình 3.16: Giao diện hướng dẫn làm thủ tục 40](#_Toc195391973)

[Hình 3.17: Giao diện tiến trình xử lý yêu cầu 40](#_Toc195391974)

[Hình 3.18: Giao diện xử lý yêu cầu nhưng chưa có yêu cầu 41](#_Toc195391975)

[Hình 3.19: Giao diện tạo hồ sơ 41](#_Toc195391976)

[Hình 3.20: Giao diện thông tin sinh viên 42](#_Toc195391977)

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Hiện nay, sinh viên phải di chuyển nhiều nơi (phòng đào tạo, phòng công tác sinh viên, phòng kế toán...) để giải quyết các thủ tục hành chính, dẫn đến mất thời gian, phức tạp và dễ gây nhầm lẫn. Hệ thống một cửa sẽ tập trung hóa dịch vụ, giúp sinh viên tiếp cận thông tin và xử lý công việc nhanh chóng, thuận tiện. Giúp việc nhà trường kết nối với sinh viên một cách hiệu quả. Phù hợp hơn với xu thế hiện nay

## 2. Mục đích đề tài

Việc xử lý các thủ tục hành chính gây ra rất nhiều phiền toái và tốn nhiều thời gian, công sức cho sinh viên. Đề tài này giúp giảm thiểu thời gian và công sức sinh viên phải bỏ ra khi giải quyết thủ tục bằng cách tích hợp các dịch vụ thành một điểm tiếp nhận duy nhất. Loại bỏ đi sự phiền phức của việc di chuyển nhiều phòng ban.

Cung cấp một giải pháp toàn diện, trải nghiệm thuận tiện hơn cho sinh viên. Thông qua hệ thống trực tuyến, giúp sinh viên dễ dàng truy cập thông tin, gửi yêu cầu và theo dõi tiến độ xử lý.

Mục tiêu là xây dựng giao diện web/mobile thân thiện, dễ sử dụng. Tự động hóa quy trình xử lý yêu cầu. Xây dựng cơ sở dữ liệu tập trung, đồng bộ hóa thông tin giữa các phòng ban. Cung cấp công cụ theo dõi trạng thái yêu cầu và thống kê dữ liệu

## 3. Đối tượng nghiên cứu

* Đối tượng nghiên cứu: Sinh viên trường Đại học Sao Đỏ, cán bộ hành chính các phòng ban liên quan, quy trình nghiệp vụ hiện hành và hệ thống công nghệ thông tin của trường.
* Phạm vi nghiên cứu: các phòng ban, đào tạo, kế toán, công tác sinh viên, thời gian chia các giai đoạn thiết kế, phát triển và thử nghiệm hệ thống.

## 4. Phạm vi nghiên cứu

* Phương pháp khảo sát bao gồm thu thập ý kiến sinh viên và cán bộ về khó khăn khi sử dụng phương pháp truyền thống, nhu cầu về hệ thống cũng như phân tích yêu cầu chức năng từ các phòng ban.
* Phương pháp triển khai bao gồm thiết kế cơ sở dữ liệu với ERD (Entity-Relationship Diagram), phát triển hệ thống với công nghệ MERN Stack (SQL Server, Express.js, ReactJS, Node.js), và kiểm thử chức năng và hiệu năng bằng Jest và Postman.

## 5. Phương pháp nghiên cứu

Phương pháp nghiên cứu bao gồm:

* Nghiên cứu lý thuyết: Tìm hiểu về các công nghệ, công cụ phát triển ứng dụng.
* Thiết kế ứng dụng: Lập kế hoạch và thiết kế giao diện người dùng.
* Phát triển ứng dụng: Lập trình và triển khai ứng dụng.
* Kiểm thử: Thực hiện các bài kiểm tra để đảm bảo ứng dụng hoạt động ổn định.

## 6. Ý nghĩa thực tiễn của đề tài

* Tiết kiệm thời gian và công sức: Sinh viên không cần di chuyển qua nhiều phòng ban để giải quyết thủ tục. Truy cập dịch vụ 24/7 qua website mọi lúc, mọi nơi
* Có thể theo dõi trạng thái yêu cầu trực tuyến, giảm thiểu thắc mắc và lo lắng.
* Tránh tiêu cực hoặc hiểu lần trong xử lý hồ sơ.
* Giảm tải công việc thủ công cho các cán bộ hành chính. Giúp nâng cao hiệu suất làm việc.

## 7. Bố cục của đề tài

Chương 1 : Cơ sở lý thuyết

Chuong 2 : Phân tích và thiết kế hệ thống

Chương 3: Xây dựng chương trình

# CHƯƠNG 1. CƠ SỞ LÍ THUYẾT

1.1. Tổng quan Javascript

JavaScript đã trở thành một trong những ngôn ngữ lập trình quan trọng và phổ biến nhất trong thế giới phát triển web hiện đại. Đây là ngôn ngữ kịch bản (scripting language) bậc cao, được tích hợp và nhúng trực tiếp vào HTML, giúp các trang web tĩnh trở nên sống động và tương tác hơn rất nhiều so với khi chỉ sử dụng mỗi HTML thuần túy.

Khả năng độc đáo của JavaScript nằm ở chỗ nó có thể kiểm soát gần như mọi hành vi và tương tác trên trang web. Từ những thao tác đơn giản như hiển thị popup thông báo, kiểm tra tính hợp lệ của form nhập liệu, cho đến những ứng dụng phức tạp như xử lý dữ liệu thời gian thực, tạo hiệu ứng đồ họa ấn tượng - tất cả đều có thể thực hiện được nhờ JavaScript.

Theo thống kê mới nhất, JavaScript hiện được sử dụng trong hơn 92% các website trên toàn thế giới, một con số ấn tượng chứng minh cho vị thế không thể thay thế của nó trong hệ sinh thái web. Điều này xuất phát từ nhiều ưu điểm vượt trội của JavaScript: cú pháp tương đối dễ học, thực thi nhanh, nhẹ và mượt mà trên hầu hết các trình duyệt hiện đại, cùng với khả năng tích hợp linh hoạt.

Trong thực tế phát triển, JavaScript có thể được sử dụng theo nhiều cách khác nhau. Cách đơn giản nhất là nhúng trực tiếp mã JavaScript vào file HTML thông qua thẻ <script>. Đối với các dự án phức tạp hơn, các đoạn mã JavaScript thường được tách riêng thành các file độc lập với phần mở rộng .js, sau đó được liên kết với file HTML chính. Cách làm này giúp mã nguồn trở nên có tổ chức hơn, dễ bảo trì và tái sử dụng

### *1.1.1. Khái niệm*

JavaScript là ngôn ngữ lập trình website phổ biến hiện nay, nó được tích hợp và nhúng vào HTML giúp website trở nên sống động hơn. JavaScript đóng vai trò như là một phần của trang web, thực thi cho phép Client-side script từ phía người dùng cũng như phía máy chủ (Nodejs) tạo ra các trang web động.

JavaScript là một ngôn ngữ lập trình thông dịch với khả năng hướng đến đối tượng. Là một trong 3 ngôn ngữ chính trong lập trình web và có mối liên hệ lẫn nhau để xây dựng một website sống động, chuyên nghiệp:

* HTML: Hỗ trợ trong việc xây dựng layout, thêm nội dung dễ dàng trên website.
* JavaScript: Tạo nên những nội dung “động” trên website. Cùng tìm hiểu rõ hơn ở phần dưới đây.
* JS là viết tắt của JavaScript, khi có JS bạn sẽ hiểu đó đang nói đến JavaScript.

Nhiệm vụ của Javascript là xử lý những đối tượng HTML trên trình duyệt. Nó có thể can thiệp với các hành động như thêm / xóa / sửa các thuộc tính CSS và các thẻ HTML một cách dễ dàng. Hay nói cách khác, Javascript là một ngôn ngữ lập trình trên trình duyệt ở phía client. Tuy nhiên, hiện nay với sự xuất hiện của NodeJS đã giúp cho Javascript có thể làm việc ở backend.

### *1.1.2. Lịch sử của Javascript*

* Brendan Eich chính là người đã phát triển JS tại Netscape với tiền thân là Mocha. Sau đó, Mocha được đổi thành LiveScript và cuối cùng mới đổi thành JavaScript.
* Năm 1998, JavaScript với phiên bản mới nhất là ECMAScript 2 phát hành và đến năm 1999 thì ECMAScript 3 được ra mắt.
* Năm 2016, ứng dụng JavaScript đã đạt kỷ lục lên tới 92% website sử dụng, đồng thời cũng được đánh giá là một công cụ cực kỳ quan trọng đối với lập trình viên.

Bên cạnh việc tìm hiểu javascript là ngôn ngữ gì thì chúng được sử dụng để làm gì cũng rất quan trọng. Việc nắm bắt được mục đích của ngôn ngữ đặc biệt này sẽ giúp bạn dễ dàng sử dụng chúng hơn trong công việc. Cụ thể như sau:

* Thay đổi nội dung HTML: Một trong số nhiều phương thức HTML JavaScript chính là getElementById (). Chúng được sử dụng để tìm một phần tử của HTML với id =”demo” và dùng để thay đổi nội dung của phần từ (Internal HTML) sang thành “Hello JavaScript”
* Thay đổi giá trị thuộc tính HTML: Tổng quan về javascript còn có thể sử dụng để thay đổi các giá trị của thuộc tính. Ví dụ: thay đổi thuộc tính src (source) của tag<img>.
* Thay đổi kiểu HTML: Đây chính là một hoạt động biến thể của việc thay đổi thuộc tính HTML ở trên. Ví dụ: document.getElementById(‘demo’).style.fontSize = ’35px;
* Ẩn các phần tử HTML: Một hoạt động tiếp theo là Javascript có thể ẩn được các phần tử HTML. Chúng có thể được thực hiện thông qua hoạt động thay đổi kiểu hiển thị các phần tử HTML.
* Hiển thị các phần tử HTML: Một điểm đặc biệt là JavaScript có thể hiển thị được các yếu tố HTML ẩn. Đồng thời, cũng có thể thực hiện được thông qua cách thay đổi kiểu hiển thị phần tử.

### *1.1.3. Ưu điểm của Javascript*

Một số ưu điểm nổi bật của ngôn ngữ lập trình JS như sau:

* Chương trình rất dễ học.
* Những lỗi Javascript rất dễ để phát hiện, từ đó giúp bạn sửa lỗi một cách nhanh chóng hơn.
* Những trình duyệt web có thể dịch thông qua HTML mà không cần sử dụng đến một compiler.
* JS có thể hoạt động ở trên nhiều nền tảng và các trình duyệt web khác nhau.
* Được các chuyên gia đánh giá là một loại ngôn ngữ lập trình nhẹ và nhanh hơn nhiều so với các ngôn ngữ lập trình khác.
* JS còn có thể được gắn trên một số các element hoặc những events của các trang web.
* Những website có sử dụng JS thì chúng sẽ giúp cho trang web đó có sự tương tác cũng như tăng thêm nhiều trải nghiệm mới cho người dùng.
* Người dùng cũng có thể tận dụng JS với mục đích là để kiểm tra những input thay vì cách kiểm tra thủ công thông qua hoạt động truy xuất database.
* Giao diện của ứng dụng phong phú với nhiều thành phần như Drag and Drop, Slider để cung cấp đến cho người dùng một Rich Interface (giao diện giàu tính năng).
* Giúp thao tác với người dùng phía Client và tách biệt giữa các Client với nhau.

### *1.1.4. Nhược điểm của Javascript*

Bên cạnh những ưu điểm kể trên thì JS vẫn có những nhược điểm riêng tương tự như các ngôn ngữ lập trình khác hiện nay:

* JS Code Snippet khá lớn.
* JS dễ bị các hacker và scammer khai thác hơn.
* JS cũng không có khả năng đa luồng hoặc đa dạng xử lý.
* Có thể được dùng để thực thi những mã độc ở trên máy tính của người sử dụng.
* Những thiết bị khác nhau có thể sẽ thực hiện JS khác nhau, từ đó dẫn đến sự không đồng nhất.
* Vì tính bảo mật và an toàn nên các Client-Side Javascript sẽ không cho phép đọc hoặc ghi các file.
* JS không được hỗ trợ khi bạn sử dụng ở trong tình trạng thiết bị được kết nối mạng.

1.2. Giới thiệu về ReactJs

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javasscript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS.Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn.

### *1.2.1. Khái niệm*

ReactJS được phát triển bởi Facebook và được giới thiệu lần đầu tiên vào năm 2011. Ban đầu, ReactJS được phát triển để xây dựng giao diện người dùng trên trang web Facebook, nhằm cải thiện tốc độ và hiệu suất của ứng dụng web.

Tuy nhiên, ReactJS không được công bố cho cộng đồng phát triển cho đến năm 2013, khi Facebook công bố mã nguồn mở của nó và giới thiệu cho cộng đồng lập trình viên. Từ đó, ReactJS nhanh chóng trở thành một trong những thư viện phổ biến nhất để phát triển các ứng dụng web động.

Sau đó, vào năm 2015, Facebook giới thiệu phiên bản React Native, một framework phát triển ứng dụng di động sử dụng ReactJS. React Native cho phép các nhà phát triển xây dựng các ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn, tương tự như ReactJS trên web.

Hiện nay, ReactJS đã trở thành một trong những thư viện phát triển web phổ biến nhất, được sử dụng rộng rãi bởi các công ty lớn và nhỏ trên toàn thế giới. Facebook cũng tiếp tục đầu tư phát triển và nâng cấp ReactJS để đáp ứng nhu cầu của cộng đồng phát triển.

### *1.2.2. Những lợi ích của ReactJS*

ReactJS mang đến nhiều lợi ích cho việc phát triển ứng dụng web, bao gồm:

* Hiệu suất cao: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM cho phép ReactJS cập nhật các thay đổi trên trang web một cách nhanh chóng và hiệu quả hơn so với cách truyền thống, giúp tăng tốc độ và hiệu suất của ứng dụng.
* Tái sử dụng: ReactJS cho phép tái sử dụng các thành phần UI, giúp giảm thiểu thời gian và chi phí phát triển. Các thành phần UI có thể được sử dụng lại trong nhiều phần khác nhau của ứng dụng, giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.
* Dễ dàng quản lý trạng thái: ReactJS giúp quản lý trạng thái của ứng dụng một cách dễ dàng. Sử dụng State và Props, ReactJS cho phép các nhà phát triển quản lý trạng thái của các thành phần UI một cách chính xác và dễ dàng.
* Hỗ trợ tốt cho SEO: ReactJS cho phép các nhà phát triển xây dựng ứng dụng web với khả năng tương thích tốt với SEO. Với sự hỗ trợ của các thư viện như React Helmet, ReactJS cho phép các nhà phát triển tùy chỉnh và quản lý các phần tử meta và title cho từng trang web.
* Hỗ trợ đa nền tảng: ReactJS không chỉ được sử dụng để phát triển các ứng dụng web, mà còn được sử dụng để phát triển các ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.

### *1.2.3. Các tính năng nổi bật của ReactJS*

ReactJS có nhiều tính năng hữu ích cho việc phát triển ứng dụng web, bao gồm:

* Components: ReactJS cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.
* Virtual DOM: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.
* JSX: JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.
* State và Props: ReactJS cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
* Hỗ trợ tốt cho SEO: ReactJS hỗ trợ tốt cho việc tối ưu hóa SEO. Với các thư viện như React Helmet, các nhà phát triển có thể quản lý các phần tử meta và title cho từng trang web, giúp tăng khả năng tìm kiếm và tăng cường trải nghiệm người dùng.
* Hỗ trợ đa nền tảng: ReactJS không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.
* Redux: Redux là một thư viện quản lý trạng thái cho các ứng dụng ReactJS. Nó giúp quản lý trạng thái của ứng dụng một cách chính xác và dễ dàng, đồng thời giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.

### *1.2.4. Cách sử dụng ReactJS trong phát triển web*

Để sử dụng ReactJS trong phát triển web, bạn có thể làm theo các bước sau:

* Bước 1 Cài đặt Node.js và npm: ReactJS được xây dựng trên nền tảng Node.js, do đó bạn cần cài đặt Node.js và npm để phát triển ứng dụng ReactJS.
* Bước 2 Tạo một ứng dụng React: Bạn có thể tạo một ứng dụng React bằng cách sử dụng lệnh "create-react-app" trong Command Prompt hoặc Terminal.
* Bước 3 Tạo các component: Tạo các component để xây dựng giao diện người dùng cho ứng dụng của bạn. Bạn có thể tạo component bằng cách sử dụng class hoặc hàm.
* Bước 4 Xây dựng giao diện người dùng: Sử dụng JSX để xây dựng giao diện người dùng cho ứng dụng của bạn. JSX là một ngôn ngữ phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI.
* Bước 5 Quản lý trạng thái: Sử dụng State và Props để quản lý trạng thái của các thành phần UI. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
* Bước 6 Kết nối với API: Sử dụng thư viện như Axios để kết nối với API và lấy dữ liệu từ server.
* Bước 7 Build và triển khai ứng dụng: Sử dụng lệnh "npm run build" để build ứng dụng của bạn và triển khai nó trên môi trường sản phẩm.

Tóm lại, để sử dụng ReactJS trong phát triển web, bạn cần cài đặt Node.js và npm, tạo các component, xây dựng giao diện người dùng, quản lý trạng thái, kết nối với API, build và triển khai ứng dụng.

### *1.2.5. Các dự án sử dụng ReactJS trong thực tế*

ReactJS là một trong những công nghệ phát triển web phổ biến nhất hiện nay, được sử dụng trong nhiều dự án thành công của các công ty lớn trên thế giới. Dưới đây là một số dự án thành công sử dụng ReactJS:

* Facebook: ReactJS là một trong những công nghệ chính của Facebook, được sử dụng để xây dựng nhiều sản phẩm của họ như Facebook, Instagram, Messenger...
* Netflix: Netflix sử dụng ReactJS để xây dựng giao diện người dùng cho trang web của họ và cho các ứng dụng di động.
* Airbnb: Airbnb sử dụng ReactJS để xây dựng giao diện người dùng cho trang web của họ và cho các ứng dụng di động.
* Uber: Uber sử dụng ReactJS để xây dựng giao diện người dùng cho ứng dụng di động của họ.
* Dropbox: Dropbox sử dụng ReactJS để xây dựng giao diện người dùng cho trang web và cho các ứng dụng di động của họ.
* Atlassian: Atlassian sử dụng ReactJS để xây dựng giao diện người dùng cho nhiều sản phẩm của họ như Jira, Confluence...

Tất cả những dự án trên đều cho thấy rằng ReactJS là một công nghệ rất phổ biến và thành công trong việc xây dựng các ứng dụng web và di động.

1.3. Giới thiệu về TailwindCss

Tailwind CSS là một framework CSS tiện ích, công cụ này cung cấp các lớp tiện ích sẵn có. Điều này giúp người dùng dễ dàng tạo giao diện người dùng mà không cần viết CSS tùy chỉnh. Khác với các framework như Bootstrap, nơi sử dụng các thành phần đã được định nghĩa trước, Tailwind CSS cho phép bạn tạo kiểu linh hoạt và tùy chỉnh hoàn toàn theo ý muốn.

### *1.3.1. Đặc điểm nổi bật của TailwindCss*

Nói đến Tailwind phải nói đến những đặc điểm nổi bật dưới đây:

* Xây dựng giao diện nhanh chóng: Chỉ cần khai báo các lớp [CSS](https://aptech.fpt.edu.vn/css-la-gi.html) trực tiếp trong file [HTML](https://aptech.fpt.edu.vn/html-la-gi.html).
* Tối ưu hóa file CSS: Chỉ xuất ra các lớp CSS đã sử dụng trong giao diện, giúp giảm dung lượng file.
* Thiết lập Responsive dễ dàng: Có thể thực hiện ngay trên file HTML mà không cần thêm mã CSS riêng biệt.
* Hỗ trợ tương tác: Cung cấp các lớp cho hiệu ứng hover, focus và nhiều hiệu ứng tương tác khác ngay trong HTML.
* Tối ưu hóa cấu trúc lặp lại: Giúp giảm thiểu sự lặp lại trong HTML và CSS.
* Tạo lớp mới dễ dàng: Cho phép định nghĩa các lớp mới kết hợp từ các lớp có sẵn trong Tailwind. Hỗ trợ chế độ tối: Tích hợp Dark Mode để giao diện phù hợp với chế độ tối.
* Dễ dàng mở rộng và chỉnh sửa: Tailwind CSS cho phép bạn dễ dàng thay đổi và điều chỉnh cấu hình.
* Tăng tốc độ lập trình: Cung cấp các Extension thông minh trên VS Code để nâng cao hiệu quả công việc.

### *1.3.2 Ưu điểm của TailwindCss*

* Thiết kế nhanh chóng: Tailwind CSS cho phép bạn xây dựng giao diện nhanh chóng bằng cách khai báo các lớp CSS trực tiếp trên phần tử HTML, giảm thời gian viết CSS tùy chỉnh.
* Khả năng tùy chỉnh cao: Bạn có thể dễ dàng tùy chỉnh giao diện qua tệp cấu hình và thêm các lớp mới phối hợp với lớp có sẵn.
* Tối ưu hóa kích thước file CSS: Chỉ xuất ra các lớp CSS đã sử dụng, giúp giảm kích thước file và tăng hiệu suất tải trang.
* Hỗ trợ responsive design: Các lớp CSS cho responsive design giúp dễ dàng điều chỉnh giao diện cho các kích thước màn hình khác nhau.
* Tương thích với nhiều framework: Tailwind CSS có thể tích hợp dễ dàng với các framework front-end như React, Vue.js, Angular…
* Hỗ trợ hiệu ứng tương tác: Các lớp CSS cho hover, focus, active giúp tạo hiệu ứng tương tác dễ dàng.
* Tài liệu chi tiết: Tài liệu của Tailwind CSS rất chi tiết và dễ hiểu, hỗ trợ người dùng mới nhanh chóng làm quen với framework.

### *1.3.3. Nhược điểm của TailwindCss*

* HTML phức tạp: Mã HTML có thể trở nên dài và phức tạp do phải khai báo nhiều lớp CSS trực tiếp trên các phần tử.
* Khó nhớ các lớp: Việc ghi nhớ và sử dụng nhiều lớp CSS có thể đòi hỏi thời gian và công sức, đặc biệt với người mới.
* Phụ thuộc vào kiến thức CSS: Để sử dụng hiệu quả, bạn cần có kiến thức cơ bản về CSS. Nếu không, việc tùy chỉnh giao diện có thể gặp khó khăn.
* Tốn thời gian làm quen: Cần thời gian để làm quen và nhớ tên các lớp trong Tailwind CSS.

1.4. Giới thiệu về Ant Design

Trong bối cảnh phát triển ứng dụng web hiện đại, Ant Design đã khẳng định vị thế là một trong những thư viện UI component hàng đầu dành cho React. Được Alibaba Group phát triển và duy trì từ năm 2015, Ant Design không đơn thuần là tập hợp các component giao diện mà đã phát triển thành một hệ thống thiết kế hoàn chỉnh (Design System) với triết lý rõ ràng. Thư viện này đặc biệt phù hợp cho các ứng dụng doanh nghiệp, hệ thống quản trị nội bộ và các giải pháp phần mềm yêu cầu tính chuyên nghiệp cao.

*1.4.1. Đặc điểm nổi bật của Ant Design*

Về mặt thiết kế, Ant Design tuân thủ chặt chẽ các nguyên tắc thiết kế giao diện hiện đại với hệ thống màu sắc được cân chỉnh kỹ lưỡng, tỷ lệ khoảng cách hợp lý và typography rõ ràng. Điều này giúp tạo ra các giao diện không chỉ đẹp mắt mà còn đảm bảo tính nhất quán cao độ. Hệ thống grid 24 cột cùng với responsive layout được tối ưu hóa giúp giao diện hiển thị hoàn hảo trên mọi thiết bị. Thư viện components được tối ưu hóa đa dạng từ Form, Table đến các components phức tạp như Transfer, TreeSelect.

Tích hợp các công nghệ hiện đại, giúp hỗ trợ TypeScript mạnh mẽ, tương thích với React Hooks, khả năng customize theme dễ dàng. Giúp đảm bảo giao diện thân thiện và dễ sử dụng cho cả sinh viên lẫn cán bộ quản trị. Các thành phần tương tác được thiết kế khoa học giúp giảm thiểu sai sót trong quá trình nhập liệu và thao tác. Đặc biệt, khả năng hỗ trợ đa ngôn ngữ tích hợp sẵn tạo điều kiện thuận lợi nếu nhà trường có nhu cầu mở rộng phục vụ sinh viên quốc tế.

*1.5.2. Lợi ích khi sử dụng cho dự án thực tế*

Trong thực tế triển khai, Ant Design có thể dễ dàng kết hợp với các thư viện và framework phổ biến như Redux để quản lý trạng thái ứng dụng, Axios cho việc giao tiếp với backend, hay AntV cho các yêu cầu hiển thị biểu đồ phức tạp. Kiến trúc module hóa của Ant Design cho phép chỉ import những component cần thiết, giúp tối ưu kích thước bundle và hiệu năng ứng dụng. Giúp tiết kiệm 40-60% thời gian xây dựng UI.

Dễ dàng bảo trì và mở rộng với cấu trúc code rõ ràng, tách biệt giữa logic và giao diện. Các component như Form kết hợp Table sẽ là công cụ mạnh mẽ để xây dựng chức năng quản lý hồ sơ sinh viên. Component Steps có thể áp dụng để hiển thị quy trình xử lý yêu cầu, trong khi hệ thống Notification và Modal giúp cung cấp phản hồi tức thì cho người dùng. Các tiện ích như DatePicker, Upload hay AutoComplete sẽ nâng cao trải nghiệm nhập liệu cho các thủ tục hành chính phức tạp.

*1.5.3. Ứng dụng cụ thể trong hệ thống*

Dưới đây là bảng so sánh giữa Ant Design và các framework UI:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **Ant Design** | **Material UI** | **Bootstrap React** |
| **Phong cách** | Chuyên nghiệp | Hiện đại | Linh hoạt |
| **Hiệu năng** | Tốt | Rất tốt | Nhẹ |
| **Tài liệu** | Đầy đủ + ví dụ | Chi tiết | Cơ bản |
| **Phù hợp** | Hệ thống quản trị | Ứng dụng di động | Website đơn giản |

Ant Design tỏ rõ ưu thế khi xây dựng các hệ thống backend, admin dashboard với yêu cầu nghiệp vụ phức tạp, trong khi Material UI phù hợp hơn cho các ứng dụng mobile-first.

1.5. SQL Server

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, cung cấp giải pháp lưu trữ, truy vấn và quản lý dữ liệu mạnh mẽ cho doanh nghiệp. SQL Server hỗ trợ ngôn ngữ T-SQL (Transact-SQL), một phiên bản mở rộng của SQL, giúp tối ưu hóa hiệu suất truy vấn và tích hợp tốt với các nền tảng công nghệ của Microsoft như .NET, Azure và Power BI.

SQL Server phù hợp với nhiều quy mô tổ chức, từ hệ thống nhỏ (dùng SQL Server Express) đến hệ thống lớn với lượng dữ liệu khổng lồ (dùng SQL Server Enterprise). Đối với hệ thống hỗ trợ một cửa tại Trường Đại học Sao Đỏ, SQL Server cung cấp nền tảng ổn định, bảo mật cao và khả năng mở rộng linh hoạt.

### *1.5.1. Các thành phần của SQL Server*

SQL Server bao gồm 7 thành phần khác nhau, mỗi cấu phần có chức năng riêng. Tất cả những điều này tạo nên một hệ thống SQL Server hoàn chỉnh cho các chức năng phân tích và lưu trữ dữ liệu.

* Database Engine: Thành phần này lưu trữ dữ liệu dưới dạng table và support, Database Engine có thể tự điều chỉnh được.
* Integration Services: Là tập hợp các công cụ đồ họa, các đối tượng lập trình, nhằm thực hiện các thao tác di chuyển, sao chép, chuyển đổi dữ liệu.
* Analysis Services: Dịch vụ phân tích dữ liệu bằng kỹ thuật khai thác hình khối đa chiều dựa trên dữ liệu có sẵn.
* Notification Services: Nền tảng phát triển và triển khai các ứng dụng soạn cũng như gửi thông báo. Ngoài ra, Notification Services cũng cung cấp tính năng gửi thông báo đến những người đăng ký sử dụng trên nhiều thiết bị khác nhau.
* Reporting Services: Có chức năng tạo, triển khai và quản lý báo cáo cho Server và Client. Reporting Services là nền tảng đẩy xây dựng ứng dụng báo cáo.
* Full Text Search Services: Là thành phần đặc biệt trong việc đánh giá và truy vấn các chỉ mục dữ liệu văn bản lưu trong database.
* Service Broker: Môi trường để lập trình ra các ứng dụng.

### *1.5.2. Các tính năng nâng cấp của SQL Server*

SQL Server cung cấp nhiều tính năng hiện đại hỗ trợ quản lý dữ liệu tiên tiến:

* In-Memory OLTP: Công nghệ xử lý giao dịch trực tiếp trong bộ nhớ giúp tăng tốc độ xử lý lên đến 30 lần, đặc biệt hữu ích cho các hệ thống cần xử lý giao dịch nhanh như hệ thống một cửa.
* PolyBase: Cho phép truy vấn dữ liệu phân tán từ nhiều nguồn khác nhau (Hadoop, Azure Blob Storage) mà không cần di chuyển dữ liệu.
* Intelligent Query Processing: Các cải tiến tự động về tối ưu hóa truy vấn giúp nâng cao hiệu suất mà không cần thay đổi code ứng dụng.

1.6. NodeJs

NodeJS là một nền tảng được xây dựng trên “[V8 Javascript engine](https://topdev.vn/blog/cach-thuc-hoat-dong-cua-javascript-v8-engine-va-5-meo-toi-uu-hoa/)” được viết bằng c++ và Javascript. Nền tảng này được phát triển bởi Ryan Lienhart Dahl vào năm 2009.

Node.js ra đời khi các developer đời đầu của JavaScript mở rộng nó từ một thứ bạn chỉ chạy được trên trình duyệt thành một thứ bạn có thể chạy trên máy của mình dưới dạng ứng dụng độc lập.

* Nguồn mở (Open-source): Mã nguồn của Node.js được công bố công khai, điều này có nghĩa là bất kỳ ai cũng có thể truy cập, sử dụng, và đóng góp vào mã nguồn. Node.js được duy trì bởi cộng đồng lập trình viên trên toàn thế giới, và hướng dẫn đóng góp của Node.js hướng dẫn bạn cách để bạn có thể góp phần phát triển nó.
* Đa nền tảng (Cross-platform): Node.js không phụ thuộc vào bất kỳ hệ điều hành nào cụ thể nào, nghĩa là nó có thể chạy trên Linux, macOS hoặc Windows. Điều này làm cho Node.js trở thành một lựa chọn linh hoạt cho các nhà phát triển muốn xây dựng các ứng dụng có thể hoạt động trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn.
* Môi trường thực thi JavaScript (JavaScript runtime environment): Để mã JavaScript có thể được thực thi, nó cần một môi trường chạy phù hợp. Trong khi trình duyệt như Chrome và Firefox cung cấp một môi trường thực thi cho JavaScript, Node.js mở rộng khả năng này ra ngoài trình duyệt. Node.js cho phép chạy JavaScript trên máy chủ, hoặc trong bất kỳ môi trường máy tính nào khác, không chỉ trong trình duyệt.
* Dựa trên V8 JavaScript Engine: Node.js được xây dựng dựa trên V8, động cơ JavaScript được phát triển bởi Google cho trình duyệt Chrome. Điều này giúp Node.js có khả năng thực thi JavaScript nhanh và hiệu quả, đồng thời hỗ trợ các tính năng mới nhất của ngôn ngữ JavaScript.

Node.js đã mở rộng khả năng của JavaScript từ việc chỉ phát triển front-end trong trình duyệt để bao gồm cả phát triển back-end. Điều này có nghĩa là các lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình, JavaScript, để phát triển toàn bộ ứng dụng, từ front-end đến back-end, qua đó tạo điều kiện cho việc học tập và phát triển ứng dụng nhanh chóng và hiệu quả hơn.

### *1.6.1. Cách hoạt động của NodeJS*

Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi. Dưới đây là một số thành phần chính giải thích cách thức hoạt động của Node.js:

* Kiến trúc Non-blocking I/O và Event-Driven:

Node.js sử dụng một mô hình non-blocking I/O (input/output) và event-driven, nghĩa là các hoạt động như đọc file, truy vấn cơ sở dữ liệu, hoặc giao tiếp mạng được thực hiện mà không chặn tiến trình chính. Điều này cho phép xử lý nhiều yêu cầu cùng lúc mà không cần tạo nhiều luồng (thread), giúp giảm bớt chi phí liên quan đến quản lý luồng và tối ưu hóa hiệu suất.

Khi một hoạt động I/O được khởi tạo, nó sẽ được gửi đến thực thi trong hệ thống hoặc cơ sở dữ liệu mà không làm chậm tiến trình chính. Sau khi hoạt động hoàn tất, một sự kiện sẽ được phát đi và xử lý bằng các hàm gọi lại (callback).

* V8 JavaScript Engine

Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, đây là một động cơ rất nhanh cho phép biên dịch mã JavaScript thành mã máy để thực thi trực tiếp trên phần cứng, làm tăng hiệu suất thực thi.

* Single-Threaded

Mặc dù Node.js hoạt động trên một luồng duy nhất cho logic ứng dụng của người dùng, nó vẫn sử dụng nhiều luồng ở tầng thấp hơn thông qua thư viện libuv để xử lý các hoạt động I/O. Tuy nhiên, những chi tiết này được ẩn giấu khỏi người dùng, giúp việc lập trình đơn giản hơn mà vẫn đảm bảo hiệu suất.

* Event Loop

Trái tim của Node.js là “event loop”. Đây là vòng lặp sự kiện mà ở đó Node.js tiếp tục lắng nghe sự kiện và thực hiện các hàm gọi lại khi một sự kiện được kích hoạt. Vòng lặp sự kiện cho phép Node.js xử lý hàng nghìn kết nối đồng thời mà không cần phải tạo ra chi phí quản lý luồng.

* Trigger Callback

Khi thao tác I/O hoàn tất, hệ điều hành thông báo cho Node.js, và Node.js sau đó thực thi hàm callback tương ứng để xử lý kết quả hoặc tiếp tục xử lý logic.

* NPM (Node Package Manager)

[NPM](https://topdev.vn/blog/npm-la-gi/) là hệ thống quản lý gói cho Node.js, cho phép các nhà phát triển dễ dàng chia sẻ và sử dụng mã nguồn từ nhau. NPM là một trong những kho lưu trữ mã nguồn mở lớn nhất thế giới và chứa hàng ngàn module có thể được tích hợp vào ứng dụng của bạn.

Tổng hợp lại, Node.js mang đến một mô hình hiệu quả và mạnh mẽ cho các ứng dụng web và máy chủ, nhờ khả năng xử lý đồng thời nhiều hoạt động I/O mà không bị chặn, và qua đó tối ưu hóa việc sử dụng tài nguyên và cải thiện hiệu suất.

### *1.6.2. Ứng dụng của NodeJs*

Node.js được sử dụng rộng rãi trong nhiều loại ứng dụng web và server do khả năng xử lý bất đồng bộ, hiệu suất cao, và hệ sinh thái phong phú của nó. Dưới đây là một số ứng dụng phổ biến của Node.js

* Ứng dụng Web thời gian thực (Real-time Web Applications): Node.js là lựa chọn lý tưởng cho các ứng dụng web thời gian thực như trò chuyện trực tuyến và trò chơi trực tuyến do khả năng xử lý các sự kiện I/O một cách nhanh chóng và hiệu quả.
* APIs Server-side: Node.js thường được sử dụng để xây dựng RESTful APIs do khả năng xử lý đồng thời lớn và tốc độ phản hồi nhanh, làm cơ sở cho các ứng dụng di động và web.
* Streaming Data: Node.js hỗ trợ xử lý dữ liệu dạng stream, cho phép ứng dụng xử lý các tệp video, âm thanh hoặc các dữ liệu khác trong khi chúng vẫn đang được truyền, thay vì phải chờ cho đến khi toàn bộ tệp được tải về.
* Ứng dụng Một Trang (Single Page Applications): Node.js phù hợp với việc phát triển các ứng dụng một trang (SPA) như Gmail, Google Maps, hay Facebook, nơi mà nhiều tương tác xảy ra trên một trang duy nhất mà không cần tải lại trang.
* Công cụ và Tự Động Hóa: Node.js cũng được sử dụng để phát triển các công cụ dòng lệnh và các script tự động hóa quy trình làm việc, nhờ vào các gói NPM hỗ trợ đa dạng và khả năng tích hợp dễ dàng với các công nghệ khác.
* Microservices Architecture: Node.js là một lựa chọn phổ biến cho kiến trúc microservices, nơi các ứng dụng lớn được chia thành các dịch vụ nhỏ, độc lập và dễ quản lý hơn.
* Ứng dụng IoT (Internet of Things): Node.js thường được sử dụng trong các ứng dụng IoT, nơi cần xử lý một lượng lớn các kết nối đồng thời và các sự kiện từ các thiết bị IoT.
* Dashboard và Monitoring: Node.js được sử dụng để xây dựng các dashboard hiển thị dữ liệu thời gian thực và các công cụ giám sát, giúp doanh nghiệp dễ dàng theo dõi hiệu suất và tình trạng của các hệ thống.

Node.js, với các ưu điểm như xử lý nhanh, mô hình bất đồng bộ và không chặn, đã trở thành công nghệ lựa chọn cho nhiều nhà phát triển và doanh nghiệp khi xây dựng các ứng dụng hiện đại và hiệu quả.

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1.Mô tả đề tài

### *2.1.1. Giới thiệu*

Hệ thống hỗ trợ một cửa là một nền tảng số tích hợp đa dịch vụ, được thiết kế để tập trung hóa các thủ tục hành chính dành cho sinh viên tại Trường Đại học Sao Đỏ vào một điểm truy cập duy nhất. Hệ thống này không chỉ đơn thuần là cổng thông tin điện tử mà là một giải pháp tổng thể áp dụng công nghệ hiện đại để đơn giản hóa quy trình. Giảm thiểu các bước thủ công, loại bỏ sự phiền hà khi sinh viên phải di chuyển qua nhiều phòng ban. Tăng tốc độ xử lý nhờ tự động hóa các công đoạn như tiếp nhận, phân loại và chuyển yêu cầu đến bộ phận chịu trách nhiệm. Nâng cao trải nghiệm người dùng do giao diện thân thiện, hỗ trợ đa nền tảng (web, mobile), và cung cấp tính năng theo dõi tiến độ minh bạch.

Đề tài xây dựng hệ thống hỗ trợ một cửa cho sinh viên Trường đại học Sao Đỏ ứng dụng công nghệ JavaScript (ReactJS, Node.js) để xây dựng hệ thống, phù hợp với xu hướng chuyển đổi số trong giáo dục đại học. Đây không chỉ là bước hiện đại hóa công tác quản lý mà còn là chiến lược dài hạn để nâng cao chất lượng dịch vụ, tạo sự khác biệt trong môi trường giáo dục cạnh tranh.

### *2.1.2. Bối cảnh và thực trạng hiện tại*

Trong bối cảnh chuyển đổi số mạnh mẽ của ngành giáo dục, Trường Đại học Sao Đỏ nhận thấy nhu cầu cấp thiết phải hiện đại hóa hệ thống quản lý và hỗ trợ sinh viên. Hiện tại, các quy trình hành chính như đăng ký học phần, xin xác nhận sinh viên, làm thẻ thư viện, giải quyết các chế độ chính sách... đang tồn tại nhiều bất cập:

Thứ nhất, sinh viên phải di chuyển qua nhiều phòng ban khác nhau để hoàn thành một thủ tục, gây tốn thời gian và phiền hà. Mỗi dịch vụ lại có quy trình tiếp nhận và xử lý riêng, thiếu sự thống nhất. Các giấy tờ, hồ sơ phần lớn được lưu trữ thủ công dưới dạng văn bản giấy, gây khó khăn cho việc tra cứu và quản lý.

Thứ hai, phía nhà trường gặp khó khăn trong việc kiểm soát tiến độ giải quyết các yêu cầu của sinh viên. Việc tổng hợp báo cáo về các hoạt động hỗ trợ sinh viên mất nhiều thời gian và độ chính xác không cao. Các số liệu thống kê về nhu cầu của sinh viên không được cập nhật kịp thời để phục vụ công tác hoạch định chính sách.

### *2.1.3. Tính cấp thiết của đề tài*

Hiện nay, hầu hết các thủ tục hành chính tại Trường Đại học Sao Đỏ vẫn được xử lý theo phương thức truyền thống, nghĩa là sinh viên phải trực tiếp đến các phòng ban để nộp đơn, xin xác nhận hoặc giải quyết các vấn đề phát sinh. Điều này dẫn đến nhiều bất cập như tốn thời gian, quá tải tại các phòng ban, thiếu minh bạch trong quy trình, và khó khăn trong việc theo dõi tiến độ giải quyết hồ sơ.

Trong khi đó, nhiều trường đại học lớn trong và ngoài nước đã triển khai thành công hệ thống một cửa điện tử, giúp cải thiện đáng kể hiệu suất làm việc và sự hài lòng của sinh viên. Do đó, việc xây dựng hệ thống hỗ trợ một cửa tại Trường Đại học Sao Đỏ là vô cùng cấp thiết, không chỉ để bắt kịp xu thế chung mà còn để nâng cao chất lượng dịch vụ, tạo lợi thế cạnh tranh trong tuyển sinh và đào tạo.

Việc triển khai được hệ thống một cửa mang lại nhiều lợi ích cho sinh viên và nhà trường. Với sinh viên, đề tài này giúp tiết kiệm thời gian, chỉ cần truy cập hệ thống online một lần để giải quyết mọi thủ tục. Nhận thông báo tự động về tiến độ xử lý qua email hoặc ứng dụng di động. Với nhà trường, đề tài này giúp chuẩn hóa quy trình nghiệp vụ, dễ dàng mở rộng khi cần thêm dịch vụ. Tích hợp với các hệ thống hiện có (quản lý đào tạo, kế toán) thông qua API. Có dữ liệu tập trung để phân tích, đưa ra chính sách cải tiến dịch vụ.

### *2.1.4. Yêu cầu đặt ra đối với hệ thống mới*

Về mặt chức năng, hệ thống cần tích hợp tất cả các dịch vụ hỗ trợ sinh viên vào một giao diện duy nhất. Sinh viên có thể dễ dàng gửi các yêu cầu, theo dõi tiến độ xử lý và nhận kết quả mà không phải đến nhiều phòng ban. Hệ thống phải hỗ trợ đầy đủ các nghiệp vụ từ đơn giản như xin xác nhận sinh viên đến phức tạp như giải quyết chế độ chính sách.

Về mặt quy trình, cần xây dựng các workflow chuẩn hóa cho từng loại dịch vụ, tự động hóa các bước xử lý có thể để tăng hiệu suất. Hệ thống phải có khả năng định tuyến yêu cầu đến đúng bộ phận chịu trách nhiệm và nhắc nhở khi quá thời hạn xử lý.

Về mặt quản lý, hệ thống cần cung cấp các công cụ báo cáo đa chiều, giúp ban giám hiệu nắm được tình hình sử dụng dịch vụ, hiệu suất làm việc của các bộ phận và các vấn đề phát sinh. Dữ liệu phải được lưu trữ tập trung, đảm bảo tính thống nhất và có thể khai thác phục vụ công tác phân tích, dự báo.

### *2.1.5.* *Phương pháp triển khai*

Hệ thống được thiết kế theo mô hình 3-tier architecture:

* Frontend Layer: ReactJS với Ant Design
* Backend Layer: Node.js + Express.js
* Data Layer: SQL Server 2019

1. Lớp giao diện – Frontend Layer:

Sử dụng ReactJS cùng với Ant Design. Với ReactJS giúp xây dựng UI components có thể tái sử dụng. Ant Design cung cấp hoàn hảo các components có sẵn, giúp tiết kiệm thời gian thiết kế. ReactJS cung cấp React Hooks cho quản lý các components một cách dễ dàng. ReactJS cũng là ứng dụng Signle-Page, ứng dụng này tải trang duy nhất một lần. Sau đó cập nhật nội dung động qua AJAX, chỉ sửa các nội dung cần sửa mà không cần tải lại toàn bộ trang.

ReactJS cũng là một framework có hiệu năng tốt, nó áp dụng các kỹ thuật tiên tiến như:

* Phân tách mã (Code Splitting) với React.lazy()

Kỹ thuật này cho phép chia nhỏ ứng dụng thành các gói mã (bundles) nhỏ hơn, chỉ tải khi cần thiết. Thay vì tải toàn bộ ứng dụng ngay từ đầu, các thành phần không quan trọng ngay lập tức sẽ được tải sau (lazy-loaded). Điều này giảm đáng kể thời gian tải ban đầu, đặc biệt quan trọng khi ứng dụng có nhiều chức năng phức tạp. React.lazy() giúp triển khai code splitting một cách dễ dàng bằng cách cho phép import động các component như các component React thông thường.

* Import động cho các component lớn (Dynamic Import)

Đối với các component có kích thước lớn hoặc ít khi được sử dụng, chúng ta áp dụng dynamic import. Kỹ thuật này tải component chỉ khi người dùng thực sự cần đến chúng, giúp giảm tải lượng mã cần tải ban đầu. Các component này sẽ được tách thành các chunk riêng biệt và tự động tải khi được gọi đến. Điều này đặc biệt hữu ích cho các tính năng như trình xem file PDF, trình soạn thảo văn bản nâng cao, hoặc các báo cáo phức tạp.

* Ảo hóa danh sách (Virtualization) với react-window

Khi xử lý các danh sách dài chứa hàng trăm hoặc hàng ngàn mục, việc render toàn bộ cùng lúc sẽ gây ảnh hưởng nghiêm trọng đến hiệu suất. Giải pháp virtualization chỉ render các phần tử đang hiển thị trên viewport của người dùng, giảm đáng kể số lượng DOM nodes cần xử lý. Thư viện react-window cung cấp các component như FixedSizeList và VariableSizeList giúp triển khai kỹ thuật này một cách hiệu quả, đảm bảo trải nghiệm mượt mà ngay cả với danh sách cực lớn.

1. Lớp Backend:

Kiến trúc Backend sử dụng Node.js làm nền tảng back-end kết hợp với framework Express.js để xây dựng các API RESTful, đáp ứng các yêu cầu nghiệp vụ phức tạp của đề tài.

Các nghiệp vụ chính bao gồm: tiếp nhận và phân loại yêu cầu từ sinh viên, phê duyệt hồ sơ tự động/theo quy trình làm việc. đồng bộ dữ liệu với các hệ thống hiện có. Một số ưu điểm của hệ thống như:

* Hiệu suất cao nhờ cơ chế Non-blocking I/O
* Dễ dàng mở rộng theo mô hình Microservices
* Hệ sinh thái phong phú với NPM packages

Bảo mật với JWT (JSON Web Token), đảm bảo tính bảo mật và quản lý phiên làm việc của người dùng một cách hiệu quả. Quy trình xác thực được thiết kế theo một luồng làm việc chặt chẽ như sau:

Khi sinh viên thực hiện đăng nhập vào hệ thống bằng tài khoản được cấp bởi nhà trường, hệ thống sẽ tiến hành xác minh thông tin đăng nhập. Sau khi xác thực thành công, hệ thống sẽ phát hành một cặp token quan trọng: Access Token có thời hạn sử dụng ngắn (thường là 1 giờ) và Refresh Token có thời hạn dài hơn. Access Token này sẽ được sử dụng để xác thực cho mọi yêu cầu tiếp theo mà người dùng gửi đến hệ thống.

Đối với mỗi request gửi đến server, client bắt buộc phải đính kèm Access Token trong phần Authorization header theo chuẩn Bearer Token. Hệ thống sẽ kiểm tra tính hợp lệ của token trước khi xử lý yêu cầu, bao gồm việc xác minh chữ ký, thời hạn sử dụng và các claim quan trọng khác.

1. Lớp dữ liệu – Data layout:

Data Layer (tầng dữ liệu) là thành phần quan trọng nhất trong kiến trúc hệ thống, chịu trách nhiệm lưu trữ, truy xuất và quản lý dữ liệu một cách hiệu quả. Trong dự án Hệ thống hỗ trợ một cửa cho Trường Đại học Sao Đỏ, Data Layer được xây dựng dựa trên SQL Server 2019, kết hợp với các công nghệ bổ trợ để đảm bảo hiệu năng cao, tính sẵn sàng (availability) và bảo mật.

### *2.1.6. Đối tượng sử dụng và lợi ích mang lại*

Hệ thống được thiết kế cho nhiều nhóm người dùng với các vai trò khác nhau:

Đối với sinh viên, hệ thống mang lại trải nghiệm dịch vụ tốt hơn với thời gian chờ đợi được rút ngắn, quy trình minh bạch và có thể theo dõi trực tuyến. Các dịch vụ có thể được truy cập mọi lúc, mọi nơi thông qua giao diện web hoặc ứng dụng di động thân thiện.

Đối với cán bộ phòng ban, hệ thống giúp giảm tải công việc hành chính nhờ tự động hóa các thao tác lặp lại. Quy trình làm việc được chuẩn hóa giúp giảm sai sót và nâng cao chất lượng dịch vụ. Các nhiệm vụ được sắp xếp khoa học theo mức độ ưu tiên và thời hạn.

Đối với lãnh đạo nhà trường, hệ thống cung cấp cái nhìn toàn cảnh về hoạt động hỗ trợ sinh viên thông qua các báo cáo trực quan. Các chỉ số hiệu suất (KPI) được đo lường chính xác giúp đánh giá chất lượng dịch vụ và đưa ra quyết định điều chỉnh kịp thời.

2.2. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu là nền tảng trọng yếu của hệ thống một cửa, đóng vai trò là trung tâm lưu trữ và xử lý toàn bộ thông tin nghiệp vụ. Thiết kế cơ sở dữ liệu cho hệ thống này cần đảm bảo 5 yếu tố cốt lõi: tính toàn vẹn dữ liệu, hiệu năng cao, bảo mật nghiêm ngặt, khả năng mở rộng và dễ dàng bảo trì.

Hệ thống hỗ trợ một cửa cho sinh viên Trường đại học Sao Đỏ yêu cầu xử lý đồng thời hàng nghìn giao dịch mỗi ngày, bao gồm: đăng ký yêu cầu, phê duyệt hồ sơ, quản lý tài liệu và thông báo tới sinh viên. Do đó, cơ sở dữ liệu được thiết kế theo mô hình quan hệ (RDBMS) với SQL Server để tận dụng ưu điểm về transaction ACID, khả năng kiểm soát ràng buộc dữ liệu và hiệu suất ổn định.

### *2.2.1. Mô tả chi tiết các bảng*

1. Bảng Admin (Quản trị viên)

Quản lý thông tin tài khoản của cán bộ quản trị hệ thống, phân quyền truy cập và theo dõi hoạt động quản lý.

Đặc tả chi tiết:

Bảng 2.1 Bảng admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| manager\_id | INT | PRIMARY KEY, IDENTITY(1,1) | ID tự tăng của quản trị viên | 1, 2, 3, ... |
| full\_name | NVARCHAR(255) | NOT NULL | Họ và tên đầy đủ | “Nguyễn Văn A” |
| username | NVARCHAR(255) | NOT NULL, UNIQUE | Tên đăng nhập hệ thống | “admin.saodo” |
| password | NVARCHAR(255) | NOT NULL | Mật khẩu | “$2a$10$N9q...” |
| created\_at | DATETIME | DEFAULT GETDATE() | Thời gian tạo tài khoản | 2025-02-15 08:30:00 |
| updated\_at | DATETIME | DEFAULT GETDATE() | Thời gian cập nhật gần nhất | 2025-02-16 08:30:00 |

Giải thích thiết kế:

* Mật khẩu: Sử dụng bcrypt với salt để chống rainbow table attacks.
* Audit trail: Tự động ghi nhận thời gian tạo/cập nhật.

1. Bảng Students (Sinh viên)

Lưu trữ hồ sơ toàn diện của sinh viên từ thông tin cá nhân đến tài khoản hệ thống.

Đặc tả chi tiết:

Bảng 2.2 Bảng Students

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| student\_id | INT | PRIMARY KEY | Mã số sinh viên (theo quy định nhà trường) | 2100568, ... |
| full\_name | NVARCHAR(100) | NOT NULL | Họ tên đầy đủ | “Nguyễn Văn B” |
| date\_of\_birth | DATE | NOT NULL | Ngày tháng năm sinh | 2003-04-09 |
| address | NVARCHAR(255) |  | Địa chỉ nơi sinh sống | Số 1, Đường X, Hải Dương |
| phone\_number | NVARCHAR(15) | check( len( phone\_number) >=10 | Số điện thoại | “012918429” |
| email | NVARCHAR(100) |  | Email liên lạc | “chitho493@gmail .com” |
| password | NVARCHAR(255) | NOT NULL | Mật khẩu | “$2°$12f2s...” |
| create\_at | DATETIME | DEFAULT GETDATE() | Thời gian tạo hồ sơ | 2025-02-16 08:30:00 |
| updated\_at | DATETIME | DEFAULT GETDATE() | Thời gian cập nhật | 2025-02-17 08:30:00 |

Giải thích thiết kế:

* Mã sinh viên: Theo quy tắc của nhà trường.
* Ràng buộc email: Đảm bảo định dạng hợp lệ.

1. Bảng Documents (Tài liệu)

Quản lý hệ thống tài liệu mẫu và file đính kèm trong các yêu cầu.

Bảng 2.3 Bảng documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| document\_id | INT | PRIMARY KEY, IDENTITY(1,1) | ID tự tăng | 1, 2, 3, ... |
| document\_type | NVARCHAR(100) | NOT NULL | Loại tài liệu | “Mẫu đơn xin bảo lưu” |
| file\_name | NVARCHAR(255) | NOT NULL | Tên file gốc | “XacNhan.docx” |
| file\_path | NVARCHAR(255) | NOT NULL | Đường dẫn lưu trữ | “uploads/1411524.docx” |
| uploaded\_at | DATETIME | DEFAULT GETDATE() | Thời gian tải lên | 2025-02-17 08:30:00 |

Giải thích thiết kế:

* Lưu trữ file: File được lưu trực tiếp ở hệ thống, chỉ lưu trữ đường dẫn file.
* Bảo mật: Quét virus trước khi lưu, giới hạn loại tài liệu.

1. Bảng Loai\_yeu\_cau (Loại yêu cầu)

Phân loại và quản lý danh mục các loại yêu cầu sinh viên có thể gửi.

Đặc tả chi tiết:

Bảng 2.4 Bảng loại yêu cầu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| loai\_yeu\_cau\_id | INT | PRIMARY KEY, IDENTITY(1,1) | ID tự tăng | 1, 2, 3, ... |
| ten\_loai | NVARCHAR(200) | NOT NULL, UNIQUE | Tên loại yêu cầu | “Xin xác nhận sinh viên” |
| document\_id | INT | FOREIGN KEY | Tài liệu mẫu liên quan | 2 |

1. Bảng Yeu\_cau (Yêu cầu của sinh viên)

Lưu trữ toàn bộ các yêu cầu do sinh viên gửi lên hệ thống.

Đặc tả chi tiết:

Bảng 2.5 Bảng yêu cầu

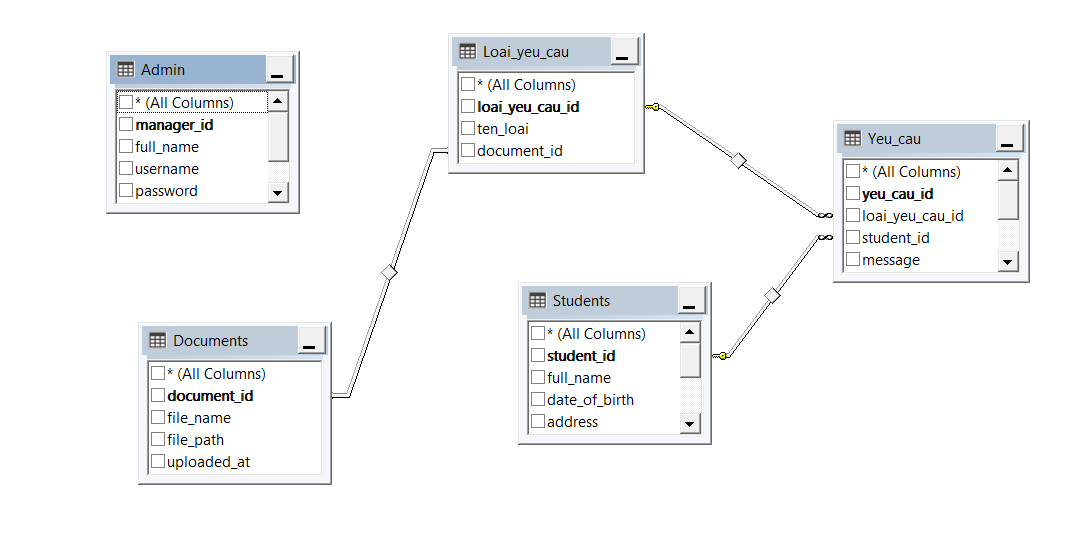
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| yeu\_cau\_id | INT | PRIMARY KEY, IDENTITY(1,1) | ID tự tăng | 1, 2, 3, ... |
| loai\_yeu\_cau\_id | INT | NOT NULL, FOREIGN KEY | Loại yêu cầu | 1 |
| student\_id | INT | FOREIGN KEY | ID sinh viên gửi yêu cầu | 2100568 |
| message | nvarchar(max) |  | Nội dung yêu cầu | “Em cần xác nhận...” |
| status | NVARCHAR(20) | CHECK:  pending/approved/rejected | Trạng thái xử lý | “pending” |
| created\_at | DATETIME | DEFAULT GETDATE() | Thời gian tạo | 2025-02-17 08:30:00 |
| updated\_at | DATETIME | DEFAULT GETDATE() | Thời gian cập nhật | 2025-02-18 08:30:00 |

Giải thích thiết kế:

* Status chỉ chấp nhận 3 trạng thái: pending (đang chờ) approved (chấp nhận) rejected(từ chối).

### *2.2.2. Sơ đồ quan hệ*

Dưới đây là sơ đồ quan hệ của cơ sở dữ liệu:



Hình 2.1: Sơ đồ quan hệ

Đặc điểm thiết kế nổi bật:

* Chuẩn hóa dữ liệu: Chuẩn dạng 3NF, giảm thiểu dư thừa dữ liệu.
* Ràng buộc toàn vẹn: Khóa ngoại đảm bảo quan hệ hợp lệ.
* Bảo mật: Audit trail qua created\_at/updated\_at.
* Hiệu năng: index trên các trường tìm kiếm thường xuyên.

2.3. Thiết kế Back-end

Hệ thống backend được thiết kế theo mô hình MVC (Model-View-Controller) sử dụng Express.js làm framework chính, cung cấp một RESTful API đạt chuẩn Level 2 theo Richardson Maturity Model. Kiến trúc này được xây dựng với các nguyên tắc cơ bản của kiến trúc này.

### *2.3.1. Tổng quan kiến trúc*

Mô hình MVC giúp phân tách rõ ràng các thành phần trong hệ thống:

* Model: Chịu trách nhiệm tương tác với cơ sở dữ liệu, thực hiện các thao tác CRUD (Create, Read, Update, Delete).
* View: Trong hệ thống backend thuần API, phần này được thay thế bằng các response JSON.
* Controller: Xử lý logic nghiệp vụ, điều hướng request từ client và trả về response tương ứng.

Hệ thống API được thiết kế theo chuẩn RESTful Level 2, bao gồm các đặc điểm như:

* Sử dụng HTTP methods đúng mục đích: GET để truy vấn dữ liệu, POST để tạo mới, PUT/PATCH để cập nhật thông tin, DELETE để xóa.
* HTTP Status Codes rõ ràng: 200 OK(Thành công), 404 Not Found(Không tìm thấy tài nguyên theo yêu cầu), 401 Unauthorized(Chưa xác thực).

Các API endpoint được tổ chức một cách rõ ràng và tách việt thông qua Express Router. Bằng việc mỗi nhóm chức năng có route riêng, các route được bảo vệ bởi middleware.

Hệ thống sử dụng JSON Web Token (JWT) để tăng sự bảo mật cho API, quy trình xác thực như sau:

* Người dùng đăng nhập, nếu hợp lệ thì server sẽ trả về access token.
* Token được gửi kèm trong header dạng là: Authorization: Bearer <token>
* Server kiểm tra token, nếu hợp lệ mới cho phép truy cập.

Ngoài việc kiểm tra token, JWT giới hạn token trong khoảng 1 giờ. Không lưu trữ thông tin nhạy cảm trong token.

### *2.3.2. Chi tiết các thành phần trong kiến trúc Backend*

1. Models - Tầng tương tác với Database

Tầng Model đóng vai trò là lớp trung gian giữa ứng dụng và cơ sở dữ liệu. Nó chịu trách nhiệm định nghĩa cấu trúc dữ liệu và thực hiện các truy vấn. Sử dụng parametezized queries để ngăn chặn SQL Injection. Tầng này xử lý các truy vấn findById(), create(), update(), delete().

Ví dụ:

const getStudentById = async (id) => {

const query = 'SELECT \* FROM Students WHERE student\_id = @id';

const result = await pool.request().input('id', id).query(query);

return result.recordset[0];

};

Giải thích chi tiết:

* pool.request(): Kết nối tới SQL Server thông qua connection pool.
* .input('id', id): Truyền tham số an toàn, tránh SQL injection.
* .query(query): Thực thi câu lệnh SQL.
* recordset[0]: Lấy bản ghi đầu tiên từ kết quả trả về.

Connection pool giúp hệ thống chịu tải tốt hơn. Ngăn chặn được 100% SQL Injection hạn chế rủi ro với database.

* Controllers - Tầng xử lý logic
* Routes (Định nghĩa API endpoints)

2.4. Thiết kế Front-end

Hệ thống Front-end được thiết kế với mục tiêu mang lại trải nghiệm người dùng thân thiện, dễ sử dụng và hiệu quả. Sử dụng React.js làm thư viện chính để xây dựng giao diện người dùng, kết hợp với các công nghệ hiện đại để tối ưu hóa hiệu suất và khả năng mở rộng.

### *2.2.1. Tổng quan kiến trúc*

* Kiến trúc Component-based: Giao diện được chia thành phần (component) nhỏ, độc lập, có thể tái sử dụng. Mỗi component đảm nhận một chức năng cụ thể, giúp việc phát triển và bảo trì dễ dàng hơn.
* Quản lý trạngthái: Sử dụng React Context API hoặc Redux Toolkit để quản lý trạng thái toàn cục, đảm bảo dữ liệu được đồng bộ giữa các thành phần trong ứng dụng.
* Routing: Sử dụng React Router để quản lý điều hướng giữa các trang, hỗ trợ các tính năng như routing động, bảo vệ route (protected route) và lazy loading để tối ưu hiệu suất.

### *2.2.2. Công nghệ sử dụng*

* React.js: Thư viện JavaScript mạnh mẽ để xây dựng giao diện người dùng.
* Tailwind CSS: Framework CSS tiện lợi, giúp xây dựng giao diện nhanh chóng và dễ dàng tùy chỉnh.
* Antd: Thư viện hỗ trợ các components có sẵn
* Framer Motion: Thư viện hỗ trợ animation, giúp giao diện trở nên sinh động và hiện đại hơn.

### *2.2.3. Quy trình thiết kế giao diện*

Phân tích yêu cầu giao diện: Dựa trên các yêu cầu từ người dùng và nghiệp vụ, xác định các chức năng chính cần hiển thị trên giao diện, bao gồm:

* Đăng nhập.
* Trang quản lý thông tin cá nhân.
* Trang gửi yêu cầu và theo dõi trạng thái xử lý.
* Trang lý hồ sơ và tải xuống tài liệu.
* Tạo hồ sơ

Triển khai giao diện:

* Xây dựng các component React.js tương ứng với từng phần của giao diện.
* Sử dụng Tailwind CSS để tùy chỉnh giao diện theo thiết kế.
* Tích hợp animation bằng Framer Motion để tăng tính tương tác.

### *2.2.4. Chi tiết các thành phần trong Front-end*

Authentication (Xác thực người dùng):

* Trang đăng nhập/đăng ký được thiết kế với giao diện đơn giản, hỗ trợ xác thực thông qua API backend.
* Sử dụng React Context API để lưu trữ thông tin người dùng sau khi đăng nhập.

Dashboard (Bảng điều khiển):

* Hiển thị thông tin cá nhân của sinh viên, bao gồm mã sinh viên, họ tên, ngày sinh, địa chỉ, và trạng thái các yêu cầu đã gửi.
* Sử dụng React Router để điều hướng giữa các trang như "Thông tin cá nhân", "Gửi yêu cầu", và "Theo dõi trạng thái".

Form gửi yêu cầu:

* Form được thiết kế với các trường nhập liệu như loại yêu cầu, ghi chú, và tải lên tài liệu.
* Sử dụng Formik hoặc React Hook Form để xử lý form, đảm bảo tính năng kiểm tra dữ liệu đầu vào (validation).

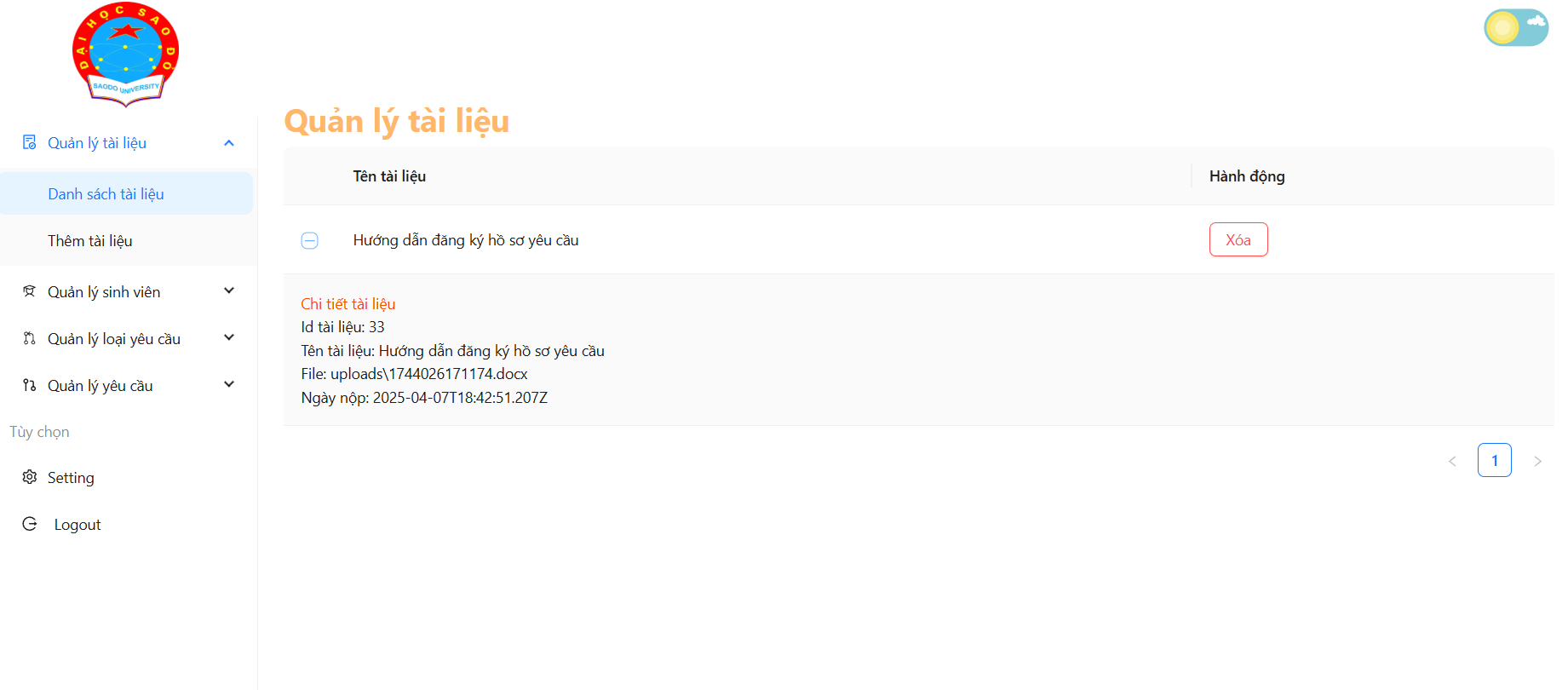
### *2.2.5. Tối ưu hóa giao diện*

* Responsive Design: Giao diện được thiết kế để tương thích với nhiều kích thước màn hình khác nhau (desktop, tablet, mobile).
* Lazy Loading: Sử dụng lazy loading để tải các component hoặc hình ảnh khi cần thiết, giúp cải thiện hiệu suất.
* Caching: Sử dụng React Query hoặc localStorage để lưu trữ tạm thời dữ liệu, giảm số lượng request đến backend.

# CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH

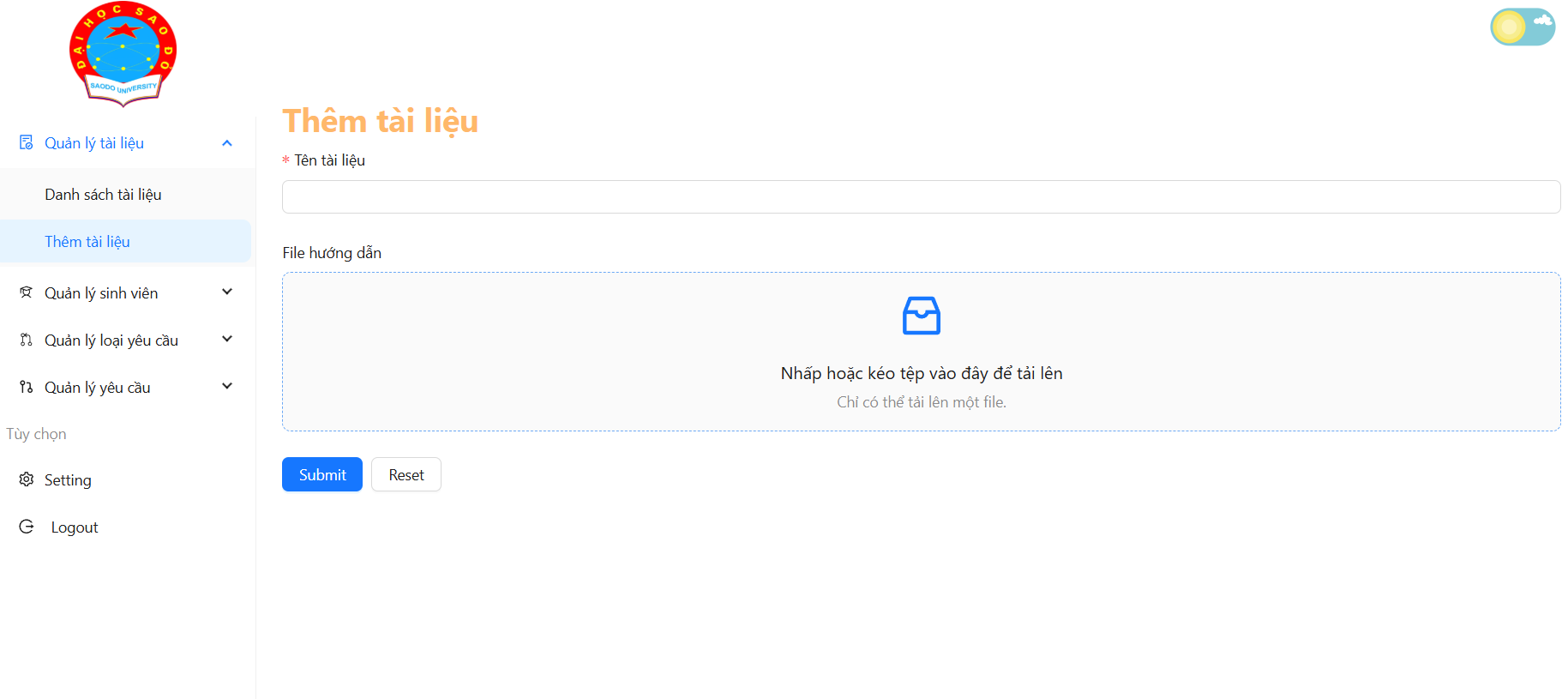
3.1. Giao diện quản trị

Đây là trang quản lý tài liệu. Trang web cho phép người dùng tải lên, xem và quản lý tài liệu một cách dễ dàng. Người dùng có thể xem danh sách tài liệu đã tải lên cùng với thông tin chi tiết như tên tài liệu, đường dẫn tải xuống và ngày tải lên. Ngoài ra, có các tùy chọn để điều chỉnh cài đặt tài khoản và đăng xuất khi không sử dụng.

****

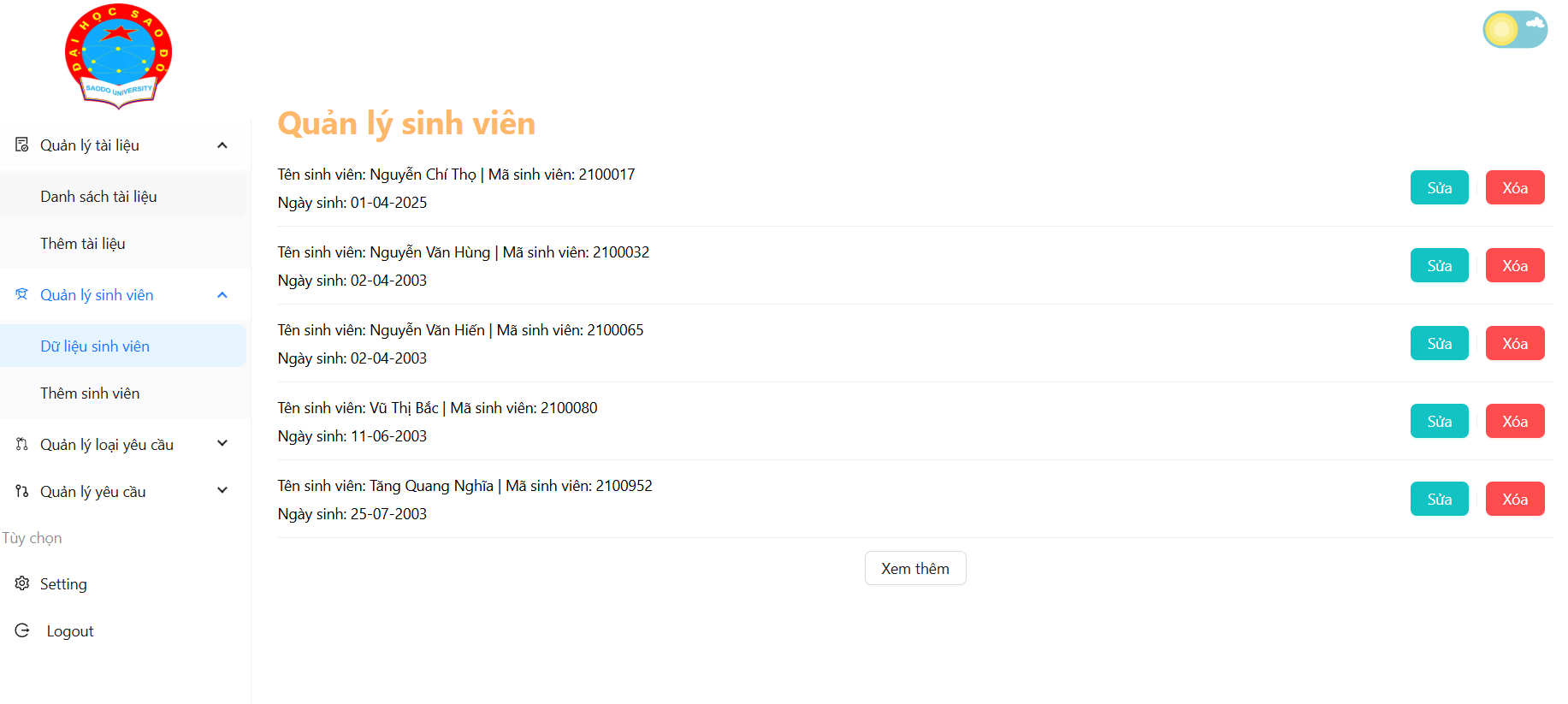
Hình 3.1: Giao diện quản lý tài liệu

Quản lý tài liệu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép người dùng xem danh sách tài liệu, chi tiết từng tài liệu (tên, file đính kèm, ngày nộp), và thực hiện các hành động như xóa tài liệu.

****

Hình 3.2: Giao diện thêm tài liệu

Giao diện thêm tài liệu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép người dùng nhập tên tài liệu và tải lên file hướng dẫn bằng cách kéo thả hoặc chọn file từ thiết bị. Sau khi hoàn tất, người dùng có thể nhấn nút Submit để lưu tài liệu hoặc Reset để xóa dữ liệu đã nhập.

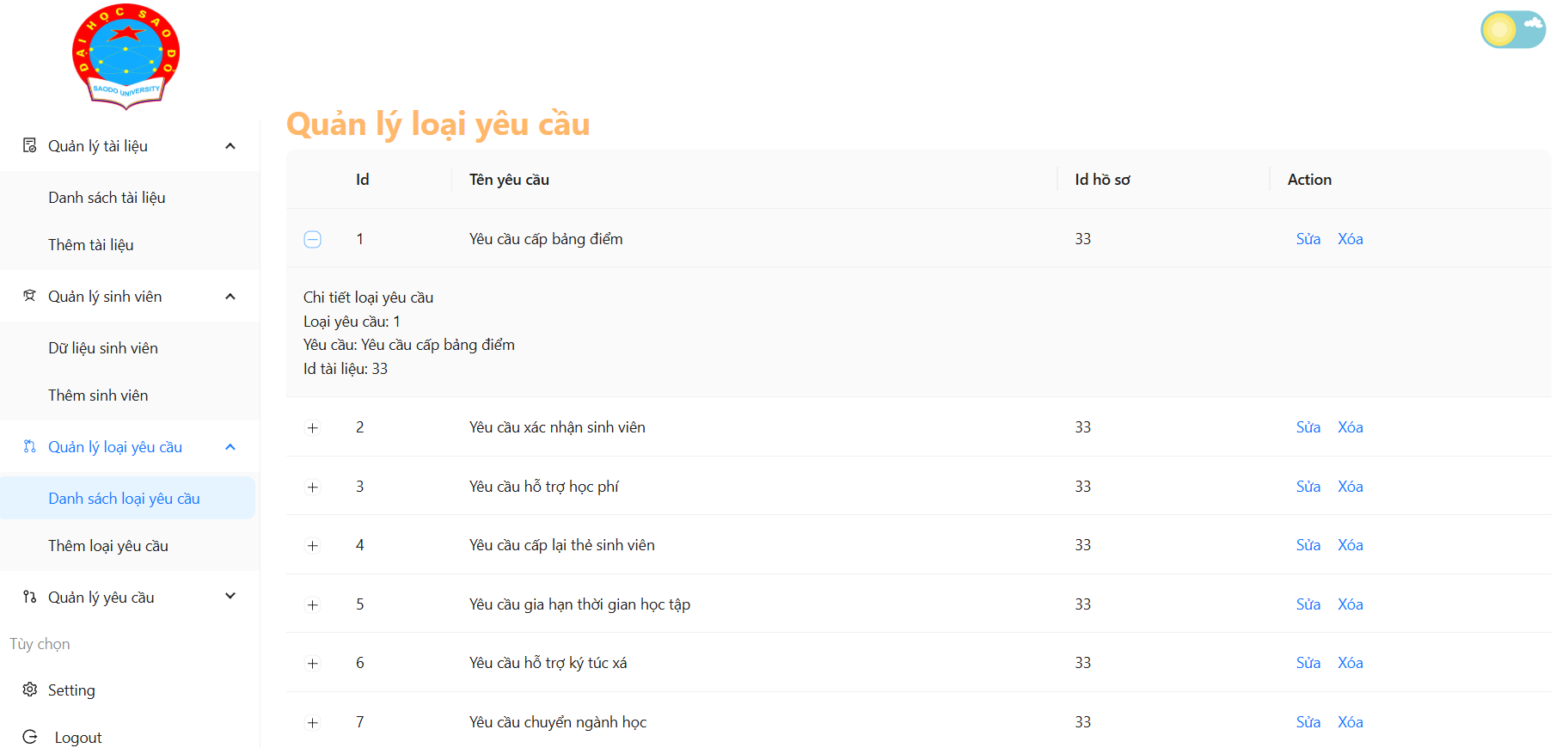
****

Hình 3.3: Giao diện quản lý sinh viên

Giao diện Quản lý sinh viên trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép hiển thị danh sách sinh viên với thông tin như tên, mã sinh viên, và ngày sinh. Người dùng có thể thực hiện thao tác như Sửa thông tin hoặc Xóa sinh viên khỏi danh sách. Nút Xem thêm được cung cấp để tải thêm dữ liệu nếu danh sách dài.****

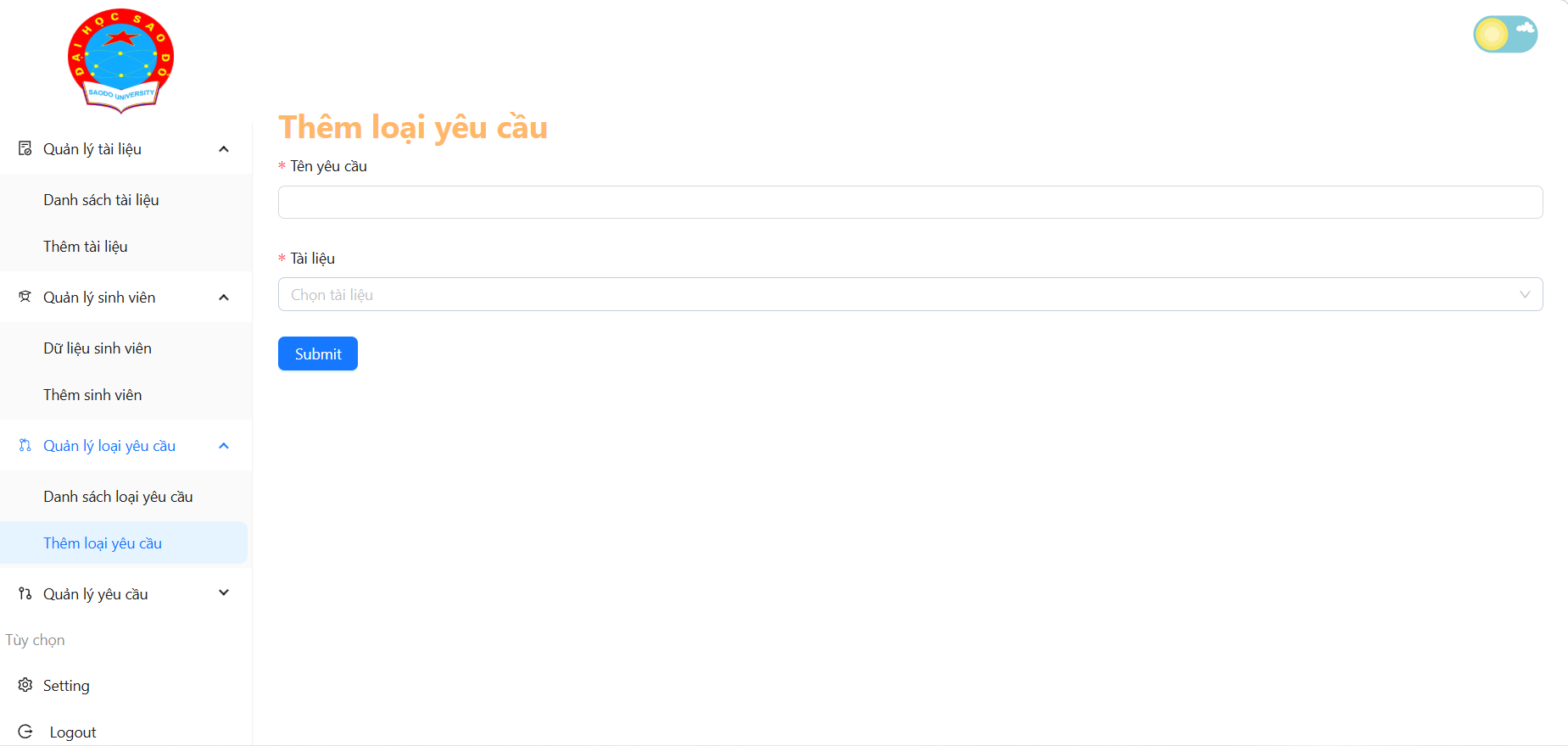
Hình 3.4: Giao diện thêm sinh viên

Giao diện thêm sinh viên trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép người dùng nhập thông tin chi tiết của sinh viên, bao gồm mã sinh viên, họ tên, địa chỉ, số điện thoại, ngày sinh, email, và mật khẩu. Sau khi điền đầy đủ thông tin, người dùng có thể lưu dữ liệu để thêm sinh viên mới vào hệ thống.

****

Hình 3.5: Giao diện loại yêu cầu

Giao diện Quản lý loại yêu cầu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép hiển thị danh sách các loại yêu cầu như yêu cầu cấp bảng điểm, xác nhận sinh viên, hỗ trợ học phí, v.v. Người dùng có thể xem chi tiết từng loại yêu cầu, chỉnh sửa thông tin bằng nút Sửa hoặc xóa loại yêu cầu bằng nút Xóa.

****

Hình 3.6: Giao diện thêm loại yêu cầu

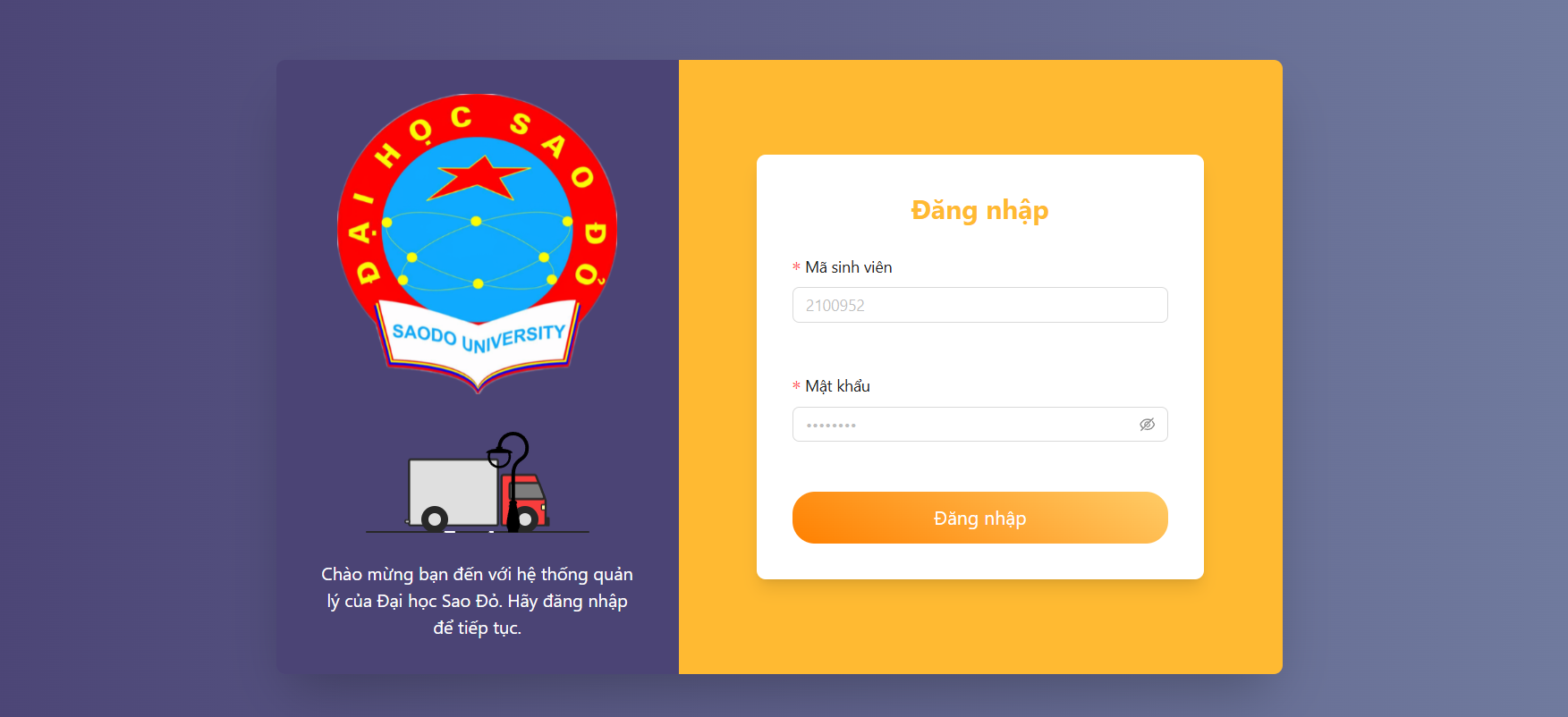
Giao diện thêm loại yêu cầu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép người dùng nhập tên loại yêu cầu và chọn tài liệu liên quan từ danh sách có sẵn. Sau khi hoàn tất, người dùng có thể nhấn nút Submit để lưu loại yêu cầu mới vào hệ thống.



Hình 3.7: Giao diện yêu cầu

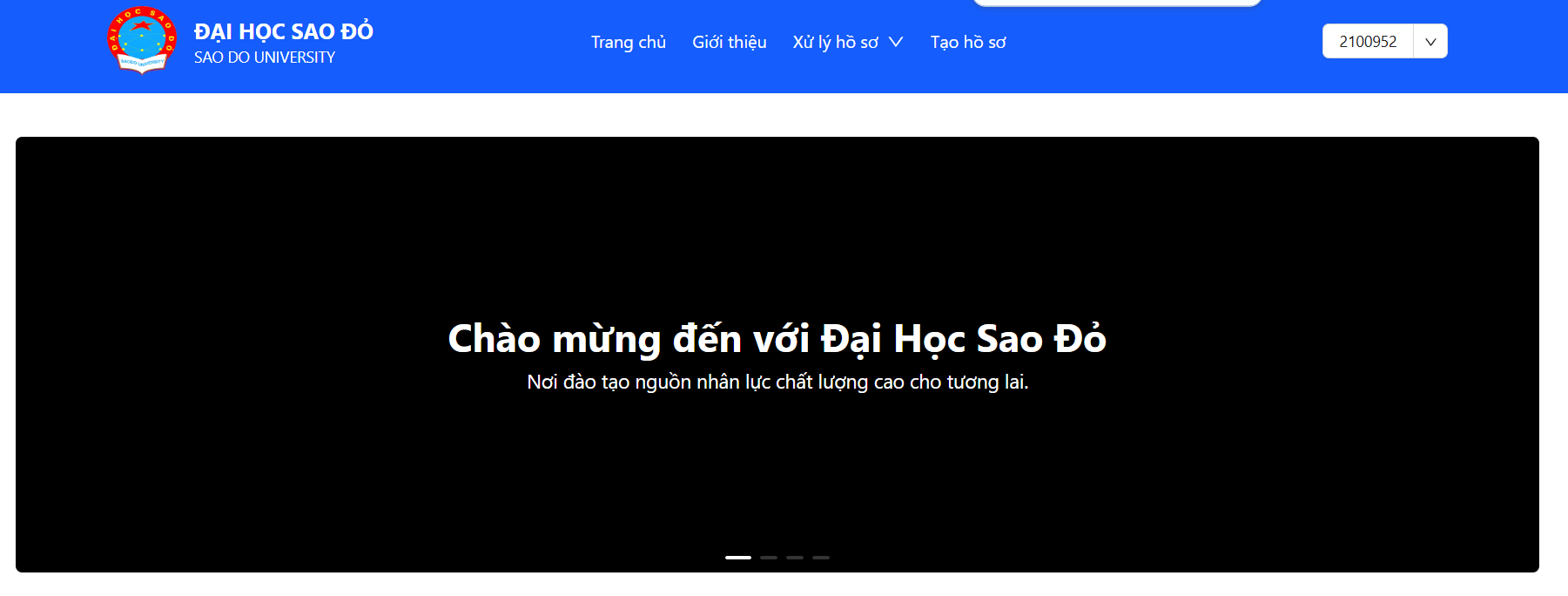
Giao diện Quản lý yêu cầu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép hiển thị danh sách các yêu cầu của sinh viên, bao gồm thông tin như ID, loại yêu cầu, mã sinh viên, mô tả, trạng thái, và các hành động (sửa, xóa). Hiện tại, danh sách chưa có dữ liệu để hiển thị.

3.2. Giao diện người dùng



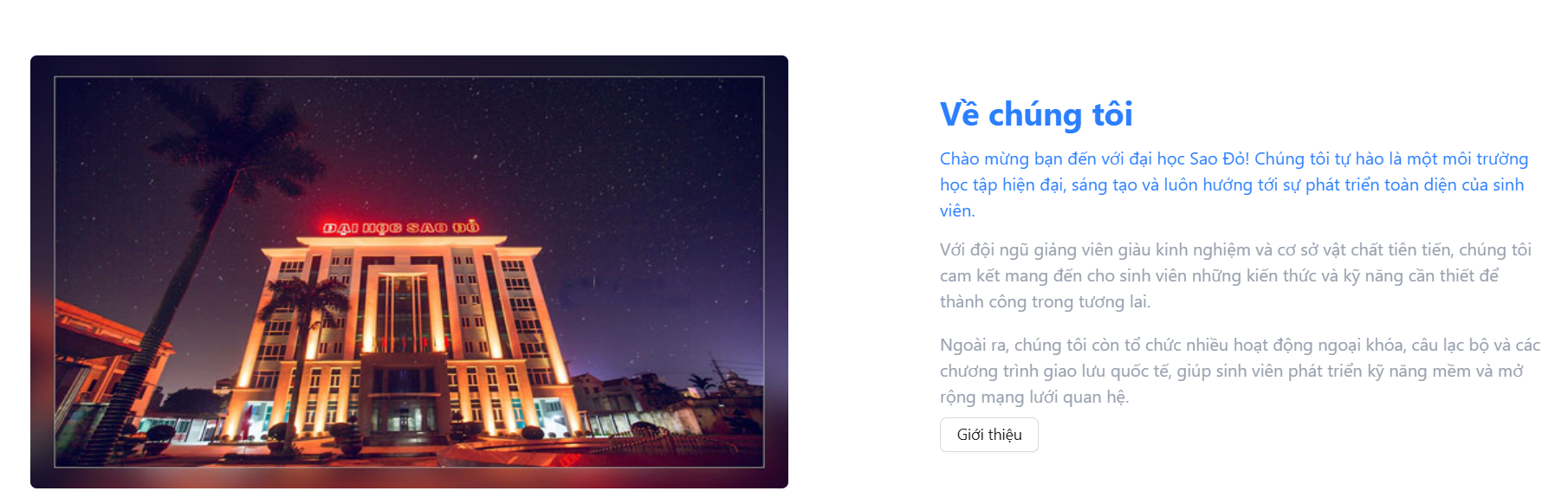
Hình 3.8: Giao diện đăng nhập

Giao diện đăng nhập trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép sinh viên nhập mã sinh viên và mật khẩu để truy cập vào hệ thống. Giao diện được thiết kế đơn giản, trực quan với nút ‘Đăng nhập’ nổi bật, giúp người dùng dễ dàng thao tác.



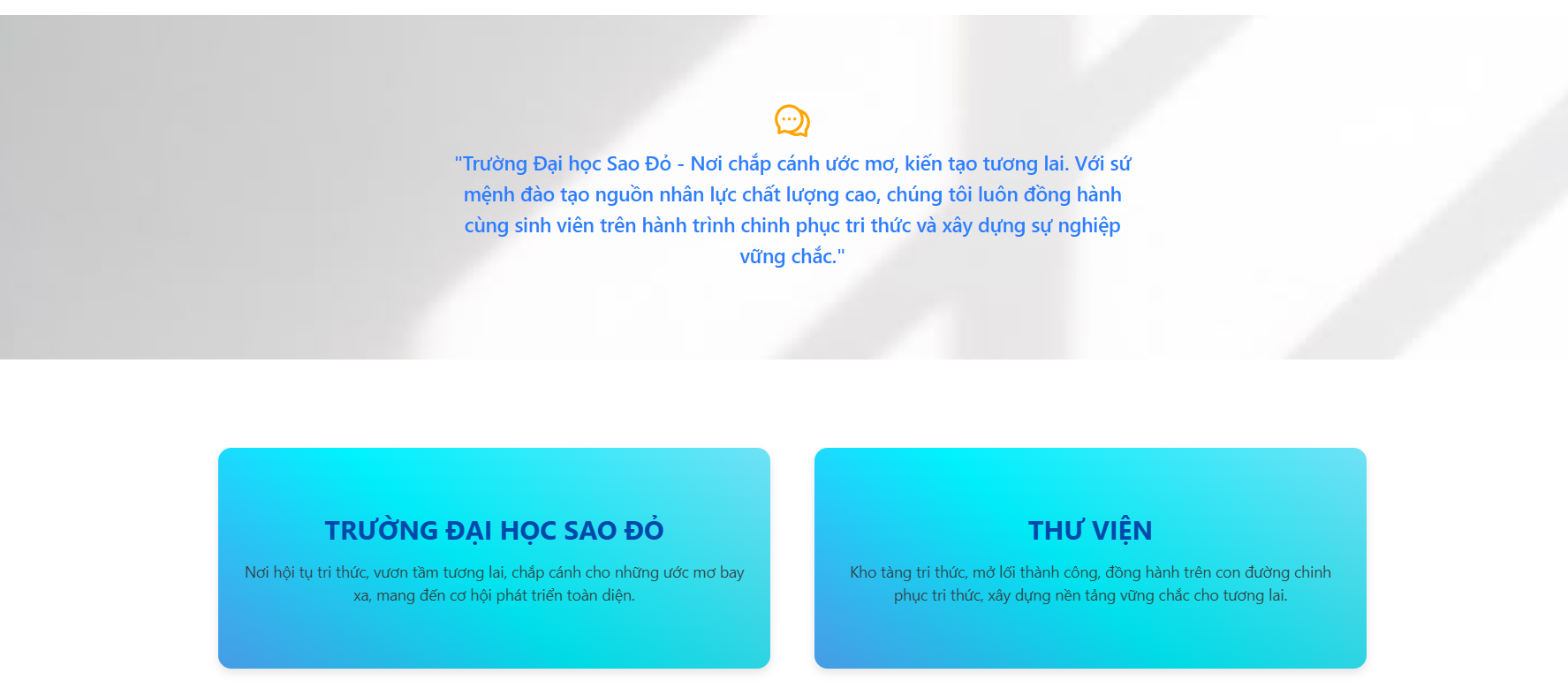
Hình 3.9: Giao diện header và slider

Giao diện Trang chủ với phần Header và Slider của hệ thống hỗ trợ một cửa cho sinh viên. Header bao gồm logo trường, thanh điều hướng với các mục như Trang chủ, Giới thiệu, Xử lý hồ sơ, Tạo hồ sơ, và menu chọn mã sinh viên. Slider hiển thị thông điệp chào mừng và slogan của trường, mang lại cái nhìn tổng quan và chuyên nghiệp cho người dùng khi truy cập.



Hình 3.10: Giao diện phần giới thiệu sơ qua về trường

Giao diện phần Giới thiệu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cung cấp thông tin tổng quan về trường Đại học Sao Đỏ, bao gồm sứ mệnh, tầm nhìn, và cam kết mang lại môi trường học tập hiện đại cho sinh viên. Phần này kết hợp hình ảnh minh họa và nội dung văn bản để tạo ấn tượng chuyên nghiệp và thu hút người dùng.



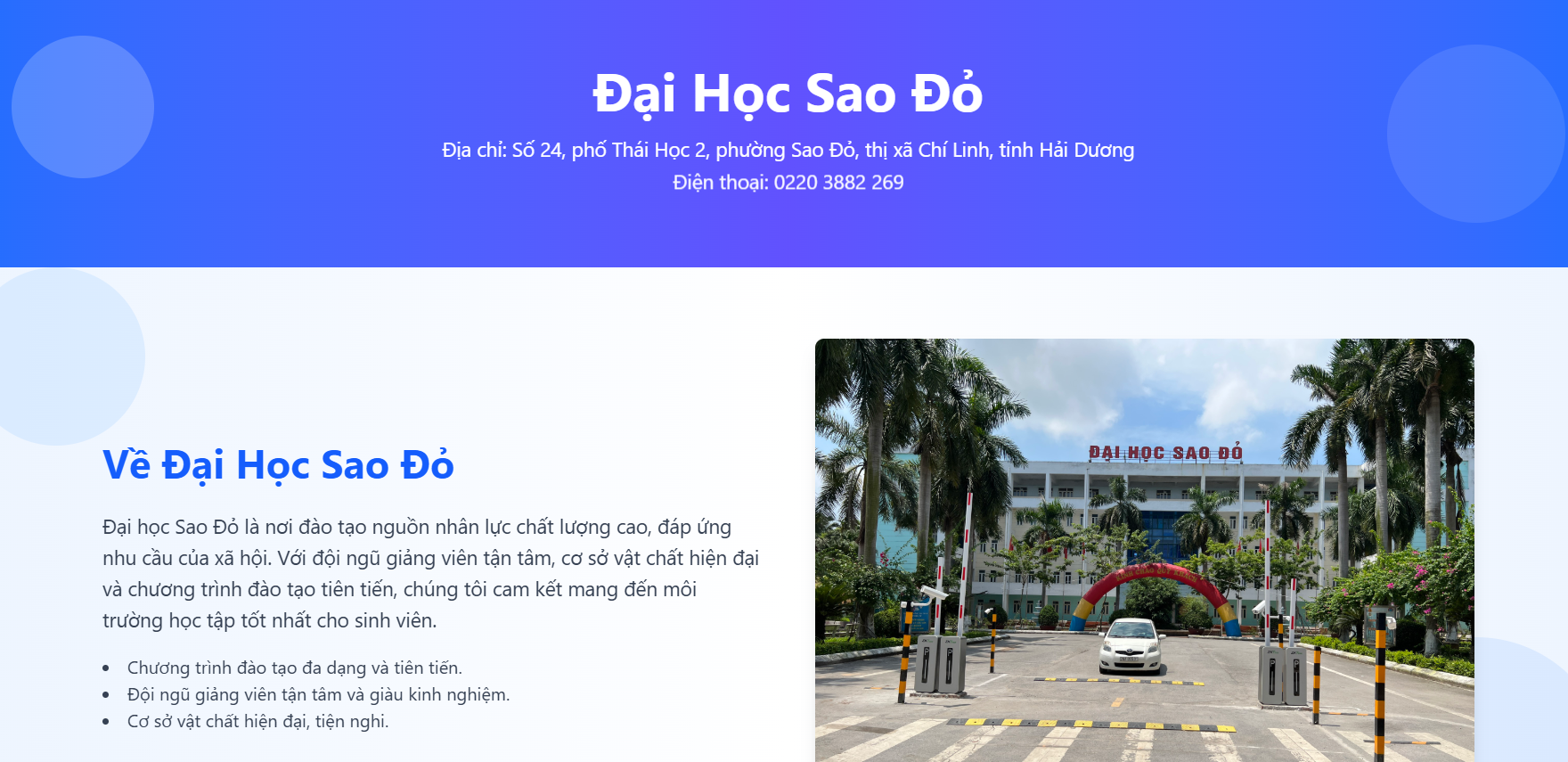
Hình 3.11: Giao diện các câu slogan

Giao diện phần Slogan trong hệ thống hỗ trợ một cửa cho sinh viên. Phần này hiển thị thông điệp truyền cảm hứng của trường Đại học Sao Đỏ, nhấn mạnh sứ mệnh và tầm nhìn trong việc đồng hành cùng sinh viên. Bên dưới là các mục thông tin nổi bật như Trường Đại học Sao Đỏ và Thư viện, giúp người dùng dễ dàng tiếp cận các nội dung quan trọng.



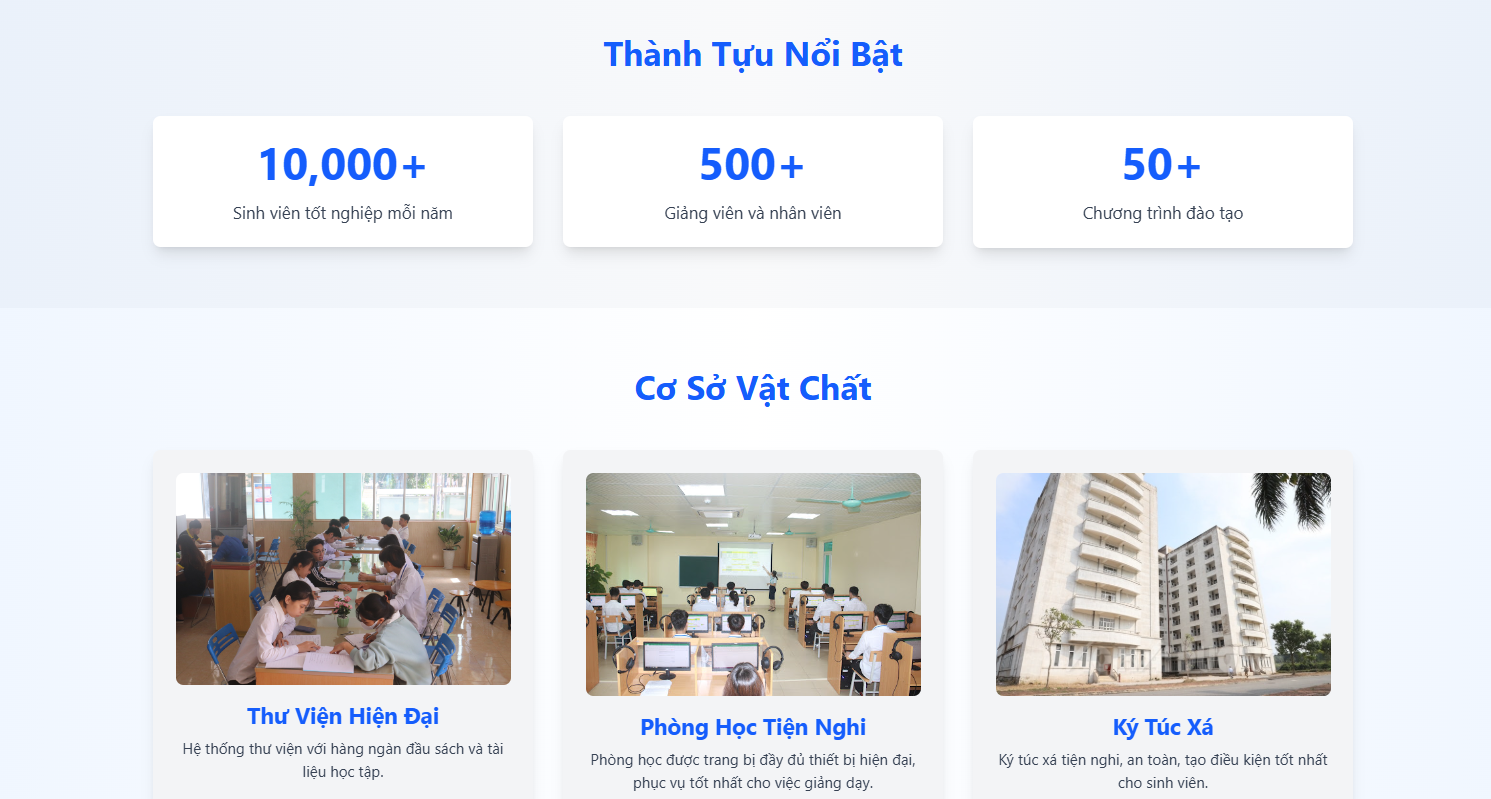
Hình 3.12: Giao diện footer

Giao diện Footer trong hệ thống hỗ trợ một cửa cho sinh viên. Phần này hiển thị thông tin liên hệ của trường Đại học Sao Đỏ, bao gồm địa chỉ, website, và số điện thoại. Footer được thiết kế đơn giản, rõ ràng, giúp người dùng dễ dàng tìm kiếm thông tin liên lạc khi cần thiết.



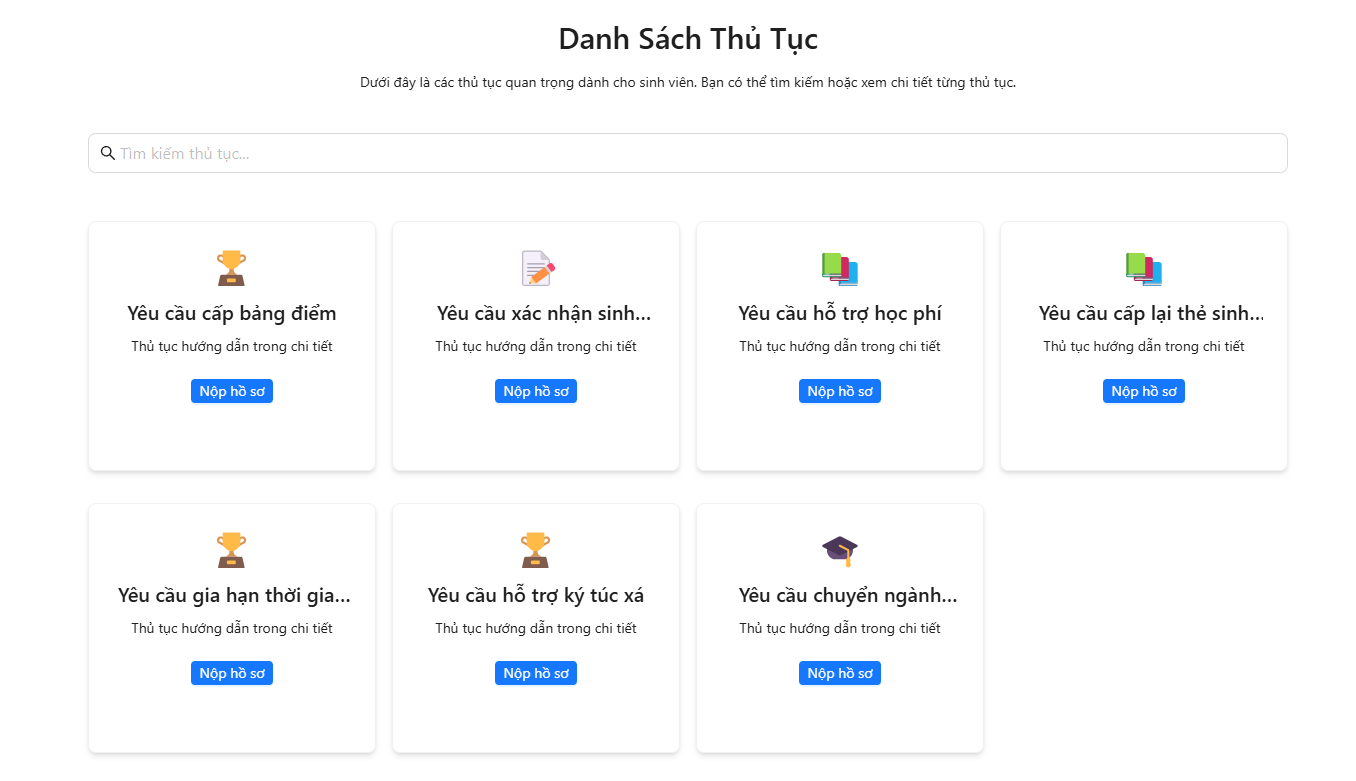
Hình 3.13: Giao diện giới thiệu

Giao diện phần giới thiệu trong hệ thống hỗ trợ một cửa cho sinh viên. Phần này cung cấp thông tin tổng quan về trường Đại học Sao Đỏ, bao gồm địa chỉ, số điện thoại liên hệ, và các điểm nổi bật như chương trình đào tạo, đội ngũ giảng viên, và cơ sở vật chất hiện đại. Kết hợp hình ảnh minh họa và nội dung chi tiết, phần này giúp người dùng hiểu rõ hơn về trường.



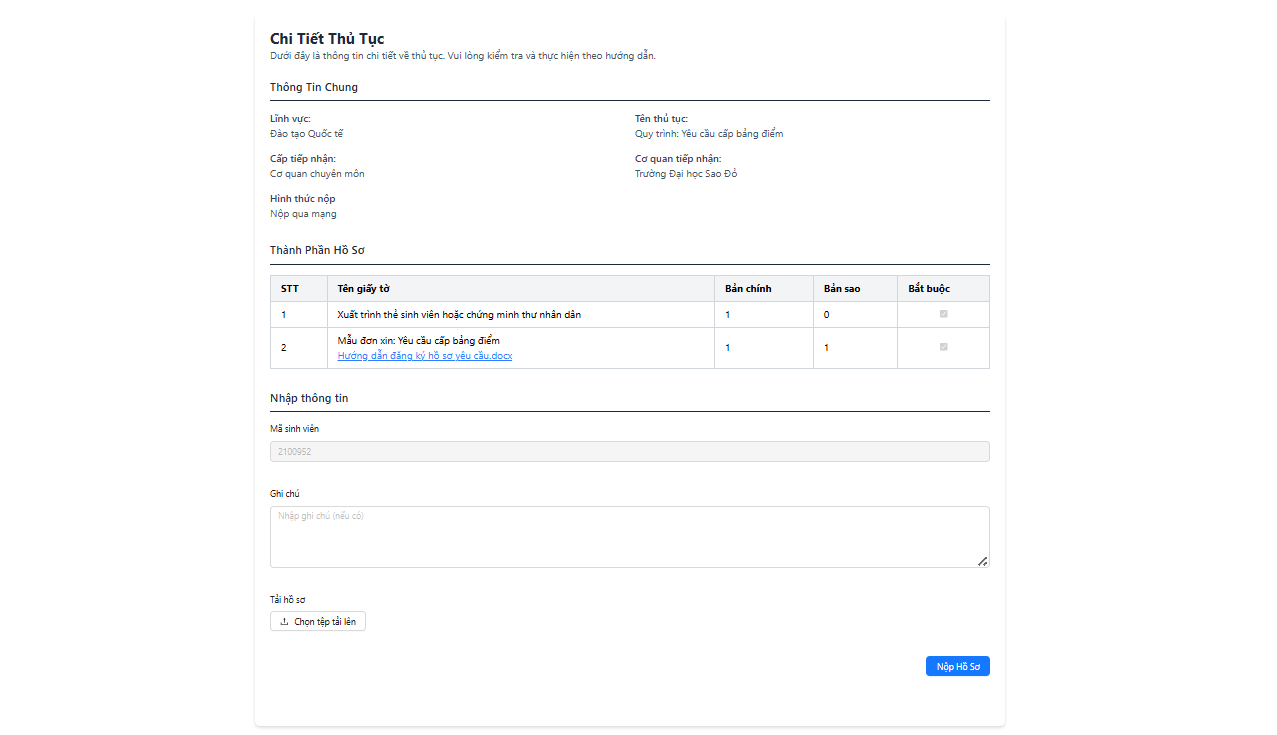
Hình 3.14: Giao diện giới thiệu thành tựu

Giao diện phần thành tựu nổi bật và cơ sở vật chất trong hệ thống hỗ trợ một cửa cho sinh viên. Phần thành tựu nổi bật hiển thị các số liệu ấn tượng như số lượng sinh viên tốt nghiệp, giảng viên, và chương trình đào tạo. Phần cơ sở vật chất giới thiệu các tiện ích hiện đại của trường như thư viện, phòng học, và ký túc xá, giúp người dùng có cái nhìn tổng quan về môi trường học tập và sinh hoạt.



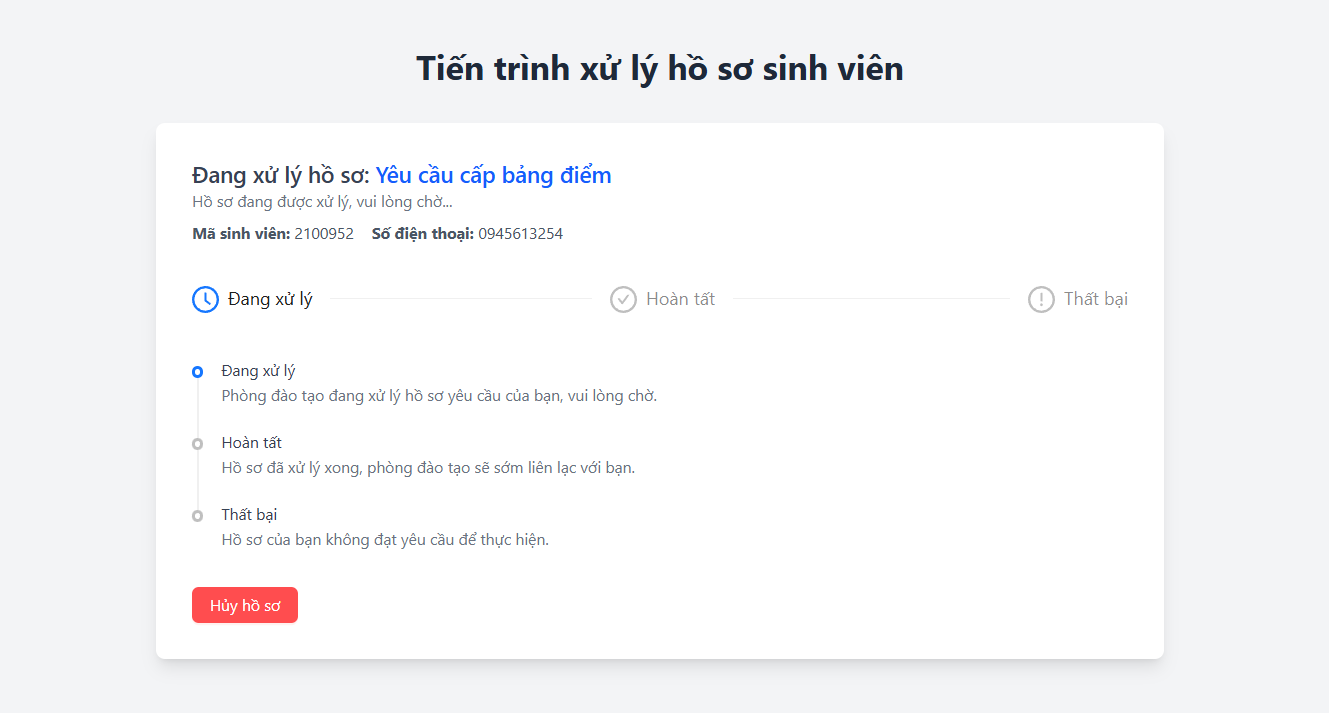
Hình 3.15: Giao diện danh sách các thủ tục

Giao diện danh sách thủ tục trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này hiển thị các thủ tục quan trọng như yêu cầu cấp bảng điểm, xác nhận sinh viên, hỗ trợ học phí, và chuyển ngành. Người dùng có thể tìm kiếm thủ tục bằng thanh tìm kiếm hoặc nhấn nút nộp hồ sơ để bắt đầu quy trình xử lý cho từng thủ tục cụ thể.



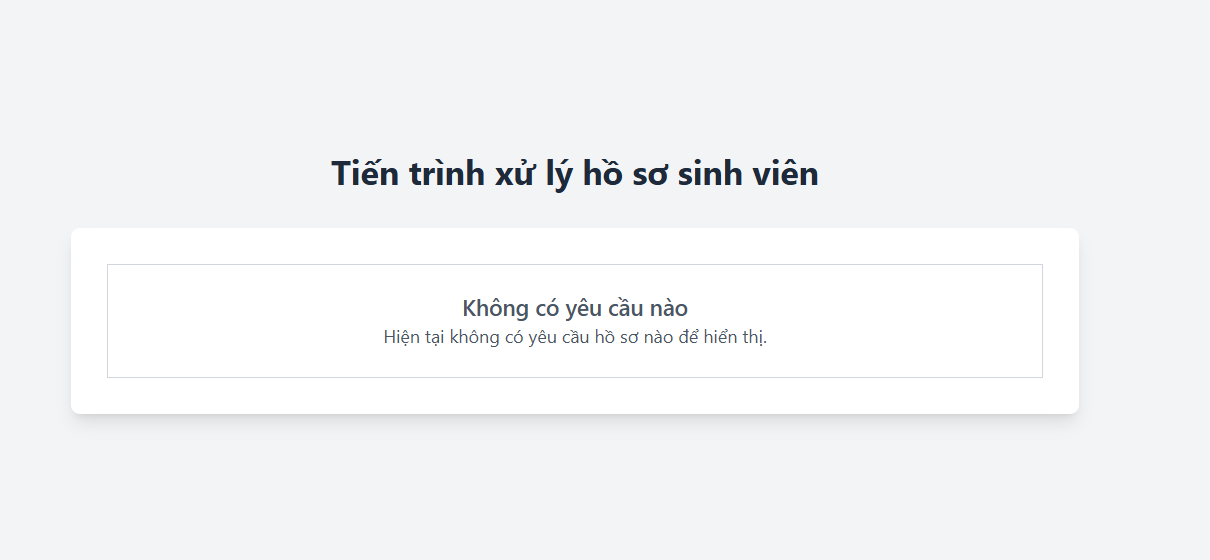
Hình 3.16: Giao diện hướng dẫn làm thủ tục

Giao diện chi tiết thủ tục trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cung cấp thông tin chi tiết về thủ tục, bao gồm lĩnh vực, tên thủ tục, cơ quan tiếp nhận, và hình thức nộp. Bên dưới là danh sách thành phần hồ sơ cần chuẩn bị, kèm theo số lượng bản chính, bản sao, và yêu cầu bắt buộc. Người dùng có thể nhập thông tin, ghi chú, tải tệp đính kèm, và nhấn nút Nộp hồ sơ để hoàn tất quy trình.



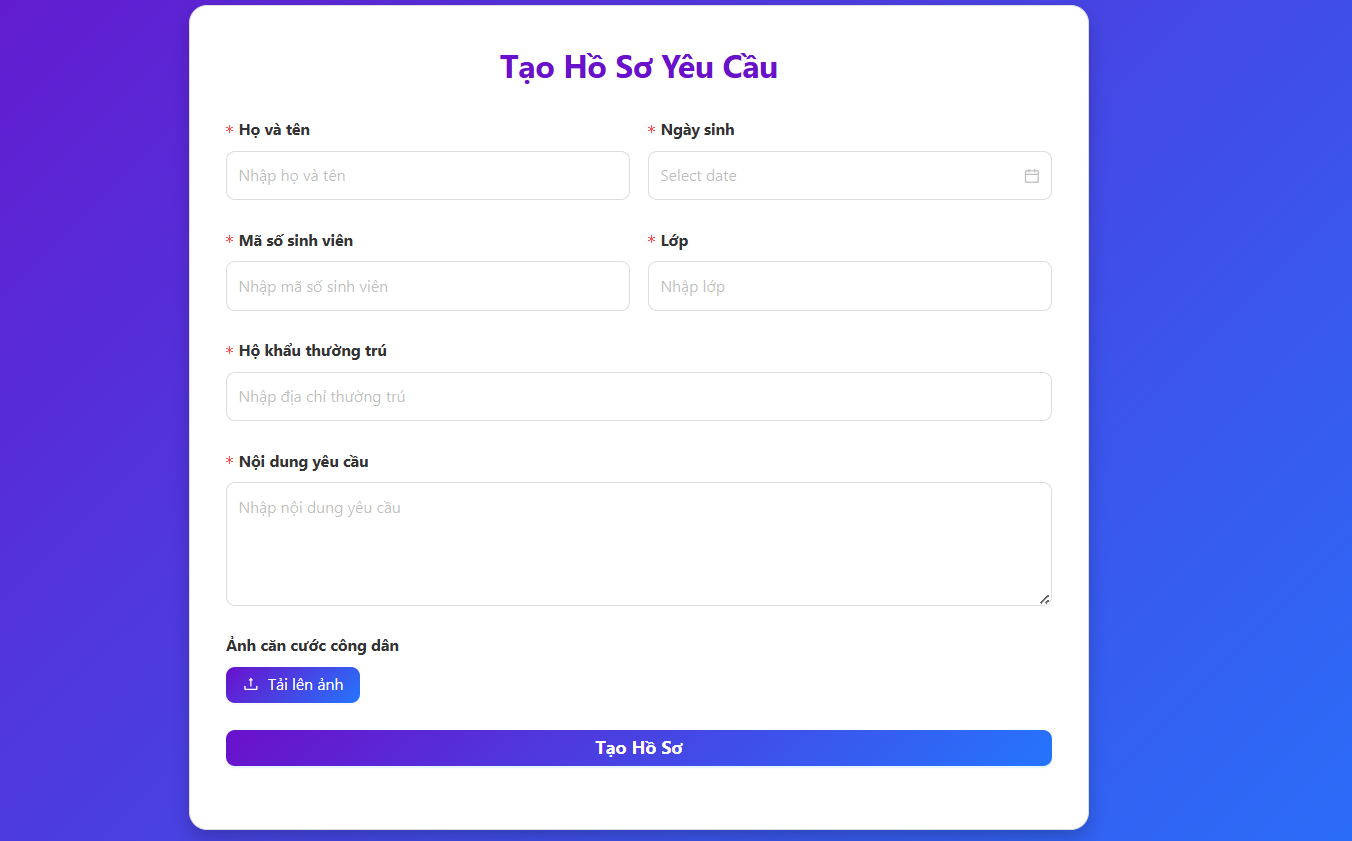
Hình 3.17: Giao diện tiến trình xử lý yêu cầu

Giao diện tiến trình xử lý hồ sơ sinh viên trong hệ thống hỗ trợ một cửa. Chức năng này hiển thị trạng thái xử lý hồ sơ của sinh viên, bao gồm các bước: Đang xử lý, hoàn tất, và thất bại. Người dùng có thể theo dõi chi tiết trạng thái hiện tại và lý do nếu hồ sơ bị từ chối. Ngoài ra, nút hủy hồ sơ được cung cấp để sinh viên hủy yêu cầu nếu cần.



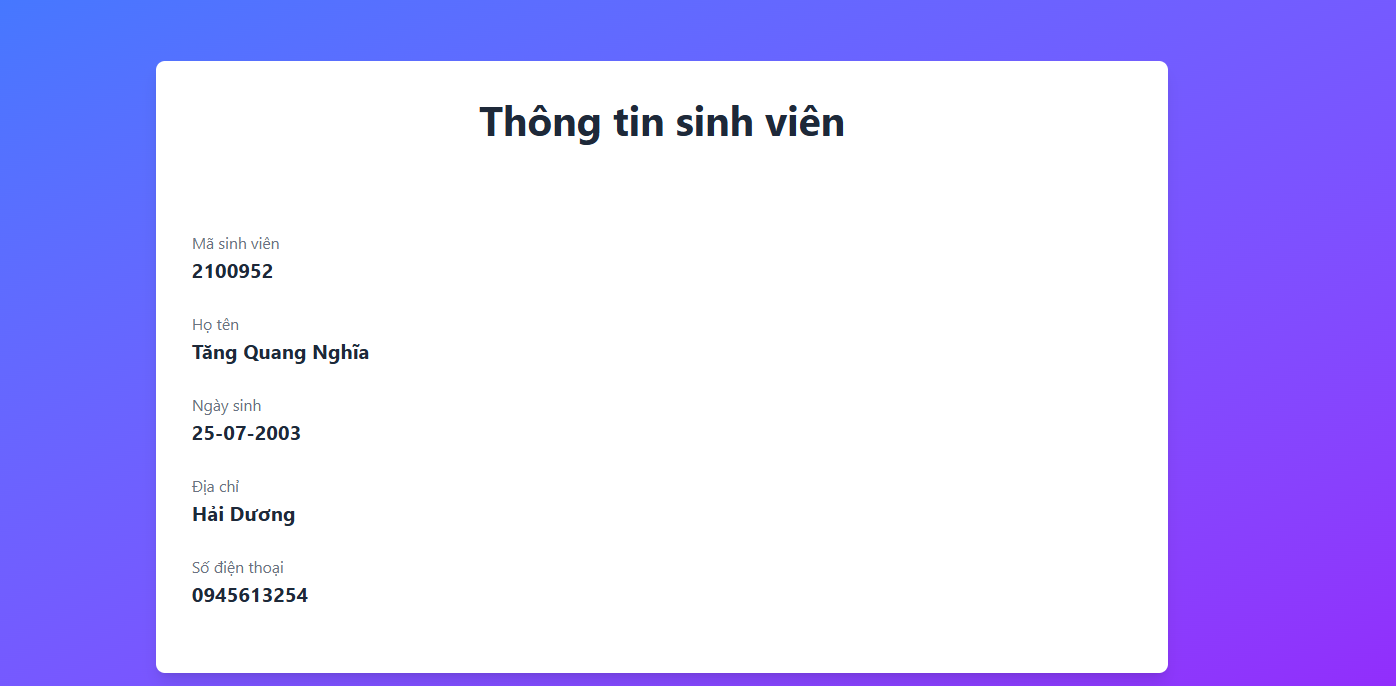
Hình 3.18: Giao diện xử lý yêu cầu nhưng chưa có yêu cầu

Giao diện tiến trình xử lý hồ sơ sinh viên khi chưa có yêu cầu nào được nộp. Chức năng này thông báo rằng hiện tại không có hồ sơ hoặc yêu cầu nào để hiển thị, giúp người dùng dễ dàng nhận biết trạng thái trống của hệ thống.



Hình 3.19: Giao diện tạo hồ sơ

Giao diện tạo hồ sơ yêu cầu trong hệ thống hỗ trợ một cửa cho sinh viên. Chức năng này cho phép người dùng nhập thông tin cá nhân như họ và tên, ngày sinh, mã số sinh viên, lớp, hộ khẩu thường trú, và nội dung yêu cầu. Ngoài ra, người dùng có thể tải lên ảnh căn cước công dân để bổ sung hồ sơ. Sau khi hoàn tất, nhấn nút tạo hồ sơ để gửi yêu cầu.



Hình 3.20: Giao diện thông tin sinh viên

Giao diện thông tin sinh viên trong hệ thống hỗ trợ một cửa. Chức năng này hiển thị các thông tin cá nhân của sinh viên, bao gồm mã sinh viên, họ tên, ngày sinh, địa chỉ, và số điện thoại. Giao diện giúp người dùng dễ dàng kiểm tra và xác nhận thông tin cá nhân của mình.

# KẾT LUẬN

# 

## 1. Kết quả đạt được

* Áp dụng hệ thống thực tế hỗ trợ sinh viên 1 cửa để xây dựng website.
* Thiết kế được giao diện dễ nhìn, đẹp mắt.
* Xây dựng được cơ sở dữ liệu chặt chẽ.

## 2. Hạn chế

* Tốc độ khởi chạy chương trình còn chậm, chưa tối ưu.

## 3. Hướng phát triển

* Tối ưu mã nguồn tốc độ chạy của chương trình
* Thiết kế giao diện dễ nhìn, đẹp mắt hơn.
* Tích hợp vào hệ thống thực tế để quản lí.

# TÀI LIỆU THAM KHẢO

[1] ReactJS Documentation: https://react.dev/

[2] Ant Design Documentation: https://ant.design/

[3] TailwindCss Documentation: https://tailwindcss.com/docs/installation/using-vite

[4] NodeJs Documentation: https://nodejs.org/docs/latest/api/

[5] SQL Server Documentation: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16

[6] Redux Documentation: https://redux-toolkit.js.org/tutorials/quick-start