**TRƯỜNG ĐẠI HỌC SAO ĐỎ**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP LỚN ĐỒ ÁN CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI : Xây dựng hệ thống hỗ trợ một cửa cho sinh viên Trường đại học Sao Đỏ**

|  |  |
| --- | --- |
| **Nhóm sinh viên thực hiện:** | **Nguyễn Chí Thọ**  **Tăng Quang Nghĩa** |
| **Lớp:** | **DK12 - CNTT2** |
| **Khóa:** | **12** |
| **Giảng viên hướng dẫn** | **ThS. Phạm Thị Hường** |

**LỜI NÓI ĐẦU**

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, sự giúp đỡ dù ít hay nhiều, dù là trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập, nhóm em đã nhận được rất nhiều sự quan tâm, giúp đỡ của cô, gia đình và bạn bè.

Nhóm em xin chân thành cám ơn cô đã tận tâm hướng dẫn nhóm em qua từng buổi học trên lớp cũng như những buổi nói chuyện, thảo luận về môn học. Trong thời gian được học tập và thực hành dưới sự hướng dẫn của cô,nhóm em không những thu được rất nhiều kiến thức bổ ích, mà còn được truyền sự say mê và thích thú đối với bộ môn “Đồ án Công nghệ phần mềm” những lời hướng dẫn, dạy bảo của cô thì nhóm em nghĩ bài tập lớn này của em rất khó có thể hoàn thành được.

Mặc dù đã rất cố gắng hoàn thiện bài tập lớn với tất cả sự nỗ lực, tuy nhiên, do bước đầu đi vào thực tế, tìm hiểu và xây dựng đề tài trong thời gian có hạn, và kiến thức còn hạn chế, nhiều bỡ ngỡ, nên đề tài “Xây dựng hệ thống hỗ trợ một cửa cho sinh viên Trường đại học Sao Đỏ” chắc chắn sẽ khôn thể tránh khỏi những thiếu sót. Nhóm em rất mong nhận được sự quan tâm, thông cảm và những đóng góp quý báu của các cô và các bạn để đề tài này ngày càng hoàn thiện hơn.

Sau cùng, chúng em xin kính chúc các thầy cô trong Khoa Công Nghệ Thông Tin dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Em xin trân thành cảm ơn cô!

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc194438565)

[1. Lí do chọn đề tài 1](#_Toc194438566)

[2. Mục đích chọn đề tài 1](#_Toc194438567)

[3. Đối tượng và phạm vi nghiên cứu 1](#_Toc194438568)

[4. Phương pháp nghiên cứu 1](#_Toc194438569)

[5. Ý nghĩa thực tiễn của đề tài 1](#_Toc194438570)

[Chương 1: CƠ SỞ LÝ THUYẾT 3](#_Toc194438571)

[1.1. Khái quát đề tài 13](#_Toc194438572)

[1.1.1*.* Giới thiệu 13](#_Toc194438573)

[1.1.2. Tính cấp thiết của đề tài: 13](#_Toc194438574)

[1.1.3. Phương pháp triển khai 14](#_Toc194438575)

[1.2. Giới thiệu về Javascript 3](#_Toc194438576)

[1.2.1. Lịch sử phát triển của Javascript 3](#_Toc194438577)

[1.2.2. Đặc điểm kĩ thuật 4](#_Toc194438578)

[1.2.3. Các thành phần cốt lõi của Javascript 4](#_Toc194438579)

[1.2.4. AJAX/Fetch API 4](#_Toc194438580)

[1.2.5. JavaScript trong phát triển Full-Stack 5](#_Toc194438581)

[1.3. Giới thiệu về ReactJs 5](#_Toc194438582)

[1.3.1. Đặc điểm nổi bật 6](#_Toc194438583)

[1.3.2. Kiến trúc và nguyên lý hoạt động 6](#_Toc194438584)

[1.3.3. Lợi ích của ReactJs trong hệ thống một cửa 7](#_Toc194438585)

[1.4. Giới thiệu về TailWind 8](#_Toc194438586)

[1.4.1. Đặc điểm nổi bật 8](#_Toc194438587)

[1.4.2. Lợi ích của Tailwind CSS 9](#_Toc194438588)

[1.4.3. So sánh với các framework truyền thống 9](#_Toc194438589)

[1.4.4. Ứng dụng Taiwind trong hệ thống một cửa 9](#_Toc194438590)

[1.5. Giới thiệu về SQL Server 10](#_Toc194438591)

[1.5.1. Kiến trúc và thành phần chính của SQL Server 10](#_Toc194438592)

[1.5.2. Các tính năng nâng cao của SQL Server 10](#_Toc194438593)

[1.6. Kết luận chương 1 12](#_Toc194438594)

[Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 13](#_Toc194438595)

[2.1. Tổng quan về bài toán 13](#_Toc194438596)

[2.1.1. Bối cảnh và thực trạng hiện tại: 13](#_Toc194438597)

[2.1.2. Yêu cầu đặt ra đối với hệ thống mới 14](#_Toc194438598)

[2.1.3. Đối tượng sử dụng và lợi ích mang lại 16](#_Toc194438599)

[2.2. Mô hình xác minh chữ ký Error! Bookmark not defined.](#_Toc194438600)

[2.2.1. Mô hình mạng song song Error! Bookmark not defined.](#_Toc194438601)

[2.2.2. Kiến trúc của mô hình Error! Bookmark not defined.](#_Toc194438602)

[2.3. Training và kết quả Error! Bookmark not defined.](#_Toc194438603)

[2.3.1. Dữ liệu Error! Bookmark not defined.](#_Toc194438604)

[2.3.2. Tiền xử lý Error! Bookmark not defined.](#_Toc194438605)

[2.3.3. Huấn luyện – Train Error! Bookmark not defined.](#_Toc194438606)

[2.3.4. Kết quả Error! Bookmark not defined.](#_Toc194438607)

[Chương 3: THIẾT KẾ HỆ THỐNG 18](#_Toc194438608)

[3.1. Lựa chọn công nghệ 18](#_Toc194438609)

[3.2. Cài đặt các thư viện cần thiết 18](#_Toc194438610)

[3.2.1. Cài đặt python 18](#_Toc194438611)

[3.2.2. Cài đặt các thư viện 18](#_Toc194438612)

[3.3. Thiết kế chương trình 18](#_Toc194438613)

[3.3.1. Cấu trúc chương trình 18](#_Toc194438614)

[3.3.2. Chương trình chạy 19](#_Toc194438615)

[3.3.3.Giao diện chương trình 23](#_Toc194438616)

[3.4. Tổng quan quá trình thiết kế 24](#_Toc194438617)

[3.4.1. Quá trình cài đặt và khởi chạy 24](#_Toc194438618)

[3.4.2. Hướng phát triển 25](#_Toc194438619)

[KẾT LUẬN 26](#_Toc194438620)

[TÀI LIỆU THAM KHẢO 27](#_Toc194438621)

# **MỞ ĐẦU**

## 1. Lí do chọn đề tài

Hiện tại, 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 chọn 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 và phạm vi nghiên cứu

1. 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.

1. 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. Phương pháp nghiên cứu

1. Phương pháp khảo sát:

* Thu thập ý kiến sinh viên và cán bộ về khó khăn khi sử dụng phương pháp truyền thống, nhu cầu về hệ thống.
* Phân tích yêu cầu chức năng từ các phòng ban.

1. Phương pháp:

* 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).
* Kiểm thử chức năng và hiệu năng bằng Jest, Postman.

## 5. 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.

# **Chương 1: CƠ SỞ LÝ THUYẾT**

## 1.1. Giới thiệu về 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. Lịch sử phát triển của Javascript

Hành trình phát triển của JavaScript bắt đầu từ năm 1995, khi Brendan Eich - một lập trình viên tại Netscape Communications - được giao nhiệm vụ tạo ra một ngôn ngữ kịch bản cho trình duyệt Netscape Navigator. Đáng kinh ngạc là ông đã hoàn thành phiên bản đầu tiên của JavaScript chỉ trong vòng 10 ngày. Ban đầu, ngôn ngữ này có tên là Mocha, sau đó đổi thành LiveScript, và cuối cùng chính thức trở thành JavaScript - một cái tên được chọn để tận dụng sự nổi tiếng của Java lúc bấy giờ, mặc dù hai ngôn ngữ này không hề có mối quan hệ họ hàng.

Năm 1997 đánh dấu một bước ngoặt quan trọng khi JavaScript được chuẩn hóa bởi ECMA International (Tổ chức tiêu chuẩn châu Âu về công nghệ thông tin và truyền thông) thành đặc tả ECMAScript (ES). Quá trình chuẩn hóa này tạo nền tảng cho sự phát triển thống nhất của JavaScript, giúp nó tránh được tình trạng phân mảnh giữa các trình duyệt khác nhau.

Trong những năm gần đây, JavaScript đã trải qua nhiều bước tiến vượt bậc với các phiên bản ECMAScript mới. Đặc biệt, ES6 (ECMAScript 2015) được coi là một cuộc cách mạng khi bổ sung hàng loạt tính năng hiện đại như arrow functions, classes, modules, promises... Các phiên bản tiếp theo như ES7 (2016) đến ES14 (2023) tiếp tục hoàn thiện ngôn ngữ với async/await, optional chaining, nullish coalescing và nhiều cải tiến khác, giúp JavaScript ngày càng mạnh mẽ và biểu cảm hơn.

### 1.1.2. Đặc điểm kĩ thuật

JavaScript sở hữu nhiều đặc điểm kỹ thuật độc đáo khiến nó trở thành một ngôn ngữ lập trình đặc biệt linh hoạt:

1. Ngôn ngữ thông dịch:

Khác với các ngôn ngữ biên dịch như C++ hay Java, JavaScript là ngôn ngữ thông dịch, có nghĩa là mã nguồn được thực thi trực tiếp mà không cần biên dịch trước. Điều này mang lại ưu điểm lớn về tốc độ phát triển và khả năng sửa lỗi nhanh chóng. Tuy nhiên, nhược điểm là hiệu nất thường thấp hơn so với ngôn ngữ biên dịch. May mắn thay, sự ra đời của các engine hiệu năng cao như V8 (Google) đã giúp thu hẹp đáng kể khoảng cách này.

1. Dyamic Typing: Kiểu dữ liệu được tự động trong runtime.

JavaScript sử dụng kiểu dữ liệu động, nghĩa là kiểu của biến được xác định tự động trong quá trình thực thi chương trình (runtime) chứ không cần khai báo trước. Điều này giúp mã nguồn ngắn gọn và linh hoạt hơn, nhưng cũng tiềm ẩn nguy cơ khó phát hiện lỗi kiểu dữ liệu trong các ứng dụng lớn. Giải pháp cho vấn đề này là sử dụng TypeScript - một siêu tập của JavaScript bổ sung khả năng kiểm tra kiểu tĩnh.

1. Hỗ trợ lập trình hướng đối tượng (OOP) và hàm (Functional Programming):

JavaScript hỗ trợ cả lập trình hướng đối tượng (OOP) và lập trình hàm (Functional Programming). Ở khía cạnh OOP, từ ES6 trở đi, JavaScript đã bổ sung cú pháp class, inheritance chính thức cùng các tính năng hướng đối tượng khác. Về Functional Programming, JavaScript hỗ trợ mạnh mẽ các khái niệm như higher-order functions (hàm bậc cao), closure, cùng các hàm xử lý mảng tiện lợi như map, filter, reduce.

### 1.1.3. Các thành phần cốt lõi của Javascript

1. Biến (Variable):

JavaScript cung cấp ba cách khai báo biến: var (phạm vi hàm), let (phạm vi khối, có thể thay đổi giá trị) và const (phạm vi khối, hằng số không thể thay đổi). Sự ra đời của let và const trong ES6 giúp quản lý biến chặt chẽ hơn, tránh các vấn đề như "hoisting" hay "variable leakage" thường gặp khi dùng var.

1. Hàm (Function) là đơn vị cơ bản của logic trong JavaScript. Có hai cách khai báo hàm phổ biến:

* Function declaration: function myFunc() {...}
* Arrow function (từ ES6): const myFunc = () => {...}

Arrow function không chỉ ngắn gọn hơn mà còn giải quyết vấn đề về ngữ cảnh this trong JavaScript.

1. DOM (Document Object Model):

API quan trọng giúp JavaScript tương tác với HTML/CSS. Thông qua DOM, JavaScript có thể truy vấn, thêm, xóa, sửa các phần tử HTML, thay đổi style, hoặc xử lý sự kiện người dùng. Ví dụ, khi sinh viên gửi yêu cầu thành công trên hệ thống một cửa, JavaScript có thể hiển thị thông báo xanh nhấp nháy mà không cần tải lại trang.

### 1.1.4. AJAX/Fetch API

Công nghệ cho phép giao tiếp với máy chủ theo cách bất đồng bộ, giúp ứng dụng web trở nên mượt mà hơn. Thay vì phải tải lại toàn bộ trang khi cần cập nhật dữ liệu, JavaScript có thể gửi yêu cầu và nhận phản hồi từ server một cách âm thầm, sau đó chỉ cập nhật phần nội dung cần thay đổi.

### 1.1.5. JavaScript trong phát triển Full-Stack

1. Front-end:

+ Framework/Library:

* React.js: Xây dựng giao diện người dùng (UI) với các component tái sử dụng. Được phát triển bởi Facebook, sử dụng component-based architecture giúp xây dựng giao diện người dùng phức tạp một cách hiệu quả.
* Vue.js: Framework tiến bộ với learning curve thấp, dễ tích ứng vào các dự án hiện có.
* Angular: Framework toàn diện của Google, phù hợp cho các ứng dụng enterprise-scale.

+ Quản lý trạng thái: Các thư viện như Redux (cho React) hoặc Vuex (cho Vue) giúp quản lý trạng thái ứng dụng một cách tập trung và dễ dàng debug.

1. Back-end (NodeJs)

Node.js là môi trường thực thi JavaScript phía server, xây dựng trên engine V8 của Google Chrome

+ Runtime enviroment: Chạy JavaScript phía server:

+ Framework:

* Express.js: Framework nhẹ, linh hoạt để xây dựng RESTful API.
* NestJs: Framework theo kiến trúc module, hỗ trợ TypeScript, phù hợp cho ứng dụng lớn.
* Kết nối cơ sở dữ liệu: JavaScript có thể làm việc với hầu hết các hệ quản trị CSDL thông qua các thư viện như Mongoose (MongoDB), Sequelize (SQL), hoặc Knex.js.

1. Real-time Comunication:

Thư viện Socket.io cho phép xây dựng ứng dụng real-time như chat, thông báo tức thì - tính năng cực kỳ hữu ích cho hệ thống một cửa khi cần thông báo ngay lập tức khi yêu cầu của sinh viên được xử lý.

1.2. Giới thiệu về ReactJs

ReactJS là một trong những thư viện JavaScript mã nguồn mở hàng đầu hiện nay được phát triển bởi Facebook (nay là Meta). Ra mắt lần đầu vào năm 2013, ReactJS nhanh chóng trở thành công cụ không thể thiếu trong phát triển giao diện người dùng (UI) nhờ những ưu điểm vượt trội về hiệu năng và khả năng bảo trì. Khác với các framework truyền thống, ReactJS tập trung vào việc xây dựng ứng dụng thông qua các component độc lập, có thể tái sử dụng, giúp quá trình phát triển trở nên nhanh chóng và hiệu quả hơn bao giờ hết.

Sức mạnh của ReactJS được chứng minh qua việc nó được sử dụng làm nền tảng cho nhiều ứng dụng quy mô lớn của các tập đoàn công nghệ hàng đầu như Facebook, Instagram, Netflix hay Airbnb. Điều này không chỉ khẳng định độ tin cậy mà còn cho thấy khả năng mở rộng đáng kinh ngạc của ReactJS khi xử lý các ứng dụng với lượng người dùng khổng lồ và yêu cầu nghiệp vụ phức tạp.

1.2.1. Đặc điểm nổi bật

1. Kiến trúc dựa trên Component:

ReactJS đưa ra một cách tiếp cận hoàn toàn mới trong việc xây dựng giao diện người dùng bằng cách chia nhỏ ứng dụng thành các component độc lập. Mỗi component là một đơn vị hoàn chỉnh bao gồm cả logic và giao diện, có thể được tái sử dụng ở nhiều nơi trong ứng dụng. Ví dụ, trong hệ thống một cửa, chúng ta có thể có các component như Header, StudentForm, hay Notification. Cách tiếp cận này mang lại nhiều lợi ích thiết thực

Ưu điểm: Tái sử dụng code, dễ kiểm thử và quản lý. Dễ dàng kiểm thử từng phần độc lập hơn.

1. Virual DOM:

Một trong những vấn đề lớn nhất trong phát triển web truyền thống là việc thao tác trực tiếp với DOM thật thường gây ra hiện tượng reflow và repaint tốn kém tài nguyên. ReactJS giải quyết triệt để vấn đề này bằng cách giới thiệu khái niệm Virtual DOM - một bản sao nhẹ của DOM thật được lưu trữ trong bộ nhớ.

Khi có sự thay đổi dữ liệu, ReactJS sẽ dễ dàng so sánh sự khác biệt giữa Virtual DOM và DOM thật(quá trình diffing). Chỉ cập nhật những phần tử thực sự thay đổi(quá trình reconciliation). Giảm thiểu tối đa các thao tác DOM không cần thiết. Cho ra kết quả là ứng dụng chạy mượt mà ngay cả khi xử lý khối lượng lớn dữ liệu và tương tác phức tạp.

1. JSX(JavaScript XML):

JSX (JavaScript XML) là một phần không thể thiếu trong trải nghiệm phát triển với ReactJS. Nó cho phép lập trình viên viết mã HTML trực tiếp trong JavaScript một cách tự nhiên, tạo nên sự thống nhất giữa logic và giao diện.

JSX sau đó sẽ được chuyển đổi thành các lời gọi hàm React.createElement() thông qua quá trình transpile (thường sử dụng Babel). Cú pháp này giúp code trở nên dễ đọc, dễ bảo trì và giảm thiểu sai sót khi làm việc với giao diện.

1. Luồng dữ liệu một chiều (Unidirectional Data Flow):

ReactJS áp dụng triệt để nguyên tắc luồng dữ liệu một chiều từ trên xuống (top-down). Dữ liệu trong ứng dụng React luôn được truyền từ component cha xuống component con thông qua props. Điều này tạo nên tính nhất quán trong quá trình quản lý trạng thái ứng dụng, dễ dàng theo dõi luồng dữ liệu, giảm thiểu các hiệu ứng phụ khó kiểm soát, thuận lợi cho việc debug và kiểm thử

1. Hỗ trợ Server-Side Rendering (SSR):

Với các framework như Next.js, ReactJS có thể được render trên server trước khi gửi đến client, mang lại nhiều lợi ích quan trọng như cải thiện tốc độ tải trang ban đầu, tối ưu hóa SEO do các công cụ tìm kiếm có thể crawl nội dung dễ dàng hơn, trải nghiệm người dùng tốt hơn trên các thiết bị có cấu hình thấp, tương thích tốt với mạng chậm hoặc không ổn định.

1.2.2. Kiến trúc và nguyên lý hoạt động

1. Vòng đời component(component lifecycle):

Mỗi component trong ReactJS đều trải qua một vòng đời cụ thể với các phương thức quan trọng như:

* **componentDidMount**: Được gọi ngay sau khi component được render lần đầu tiên, thường dùng để gọi API hoặc thiết lập subscription
* **componentDidUpdate**: Được gọi sau mỗi lần component cập nhật, dùng để xử lý các tác vụ phụ thuộc vào sự thay đổi props/state
* **componentWillUnmount**: Được gọi trước khi component bị hủy, dùng để dọn dẹp các resource đang sử dụng

Với sự xuất hiện của Hooks trong React 16.8, các functional component giờ đây cũng có thể sử dụng các tính năng lifecycle thông qua useEffect.

1. State và Props – Hai khái niệm cốt lõi:

State: đại diện cho trạng thái nội bộ của component, có thể thay đổi trong quá trình thực thi. State được quản lý bằng this.setState (class component) hoặc useState hook (functional component).

Props: là các thuộc tính được truyền từ component cha xuống component con, có tính chất read-only(chỉ đọc)

Ví dụ:

1. React Hook

Cho phép sử dụng state và lifecycle trong functional component. Được ra mắt trong React 16.8, Hooks mang đến cách tiếp cận mới để sử dụng state và các tính năng khác mà không cần dùng đến class. Các hook quan trọng bao gồm:

* useState: Quản lý state trong functional component
* useEffect: Xử lý side effects (gọi API, subscription...)
* useContext: Truy cập React context
* useReducer: Quản lý state phức tạp theo mô hình reducer
* useMemo và useCallback: Tối ưu hiệu năng

1.2.3. Lợi ích của ReactJs trong hệ thống một cửa

1. Xây dựng giao diện động và thân thiện:

ReactJS là lựa chọn lý tưởng để tạo ra các form đăng ký dịch vụ phức tạp, bảng hiển thị trạng thái hồ sơ với khả năng cập nhật real-time mà không cần tải lại trang. Các tính năng như validation, autocomplete hay dynamic form có thể được triển khai dễ dàng nhờ vào hệ thống component linh hoạt.

1. Tích hợp dễ dàng với RESTful API/GraphQL:

ReactJS cung cấp nhiều cách để tương tác với backend nhờ sử dụng fetch API có sẵn trong trình duyệt. Hoặc sử dụng thư viện axios với nhiều tính năng nâng cao

1. Quản lý trạng thái phức tạp với Redux/Context API:

Với ứng dụng quy mô lớn như hệ thống một cửa, việc quản lý state tập trung là vô cùng quan trọng như:

* Redux: Cung cấp single source of truth cho toàn bộ ứng dụng, phù hợp cho các state cần truy cập ở nhiều nơi như thông tin sinh viên đăng nhập, trạng thái toàn cục...
* Context API: Giải pháp nhẹ nhàng hơn để chia sẻ dữ liệu xuyên suốt component tree mà không cần truyền props qua nhiều cấp
* React Query: Thư viện chuyên cho việc quản lý server state và caching

1. Hỗ trợ đa nền tảng:

Ưu điểm lớn nhất của ReactJS là khả năng chia sẻ code với React Native – framework để phát triển ứng dụng di động. Điều này có ý nghĩa rất lớn vì nó tiết kiệm thời gian và chi phí phát triển do logic nghiệp vụ đã viết cho web có thể tái sử dụng cho mobile, đồng thời cũng đảm bảo được tính nhất quán giữa các nền tảng.

1. Hệ sinh thái phong phú:

ReactJS được hỗ trợ bởi một hệ sinh thái vô cùng đa dạng:

* Các thư viện UI phổ biến: Materia-UI, Ant Design, Chaka UI.
* Công cụ quản lý form: Formik, React Hook Spring.
* Thư viện animation: Framer Motion, React Spring
* Công cụ kiểm thử: Jest, React Testing Library
* Và hàng nghìn package hữu ích khác trên npm

1.3. Giới thiệu về TailWind

Tailwind CSS đã tạo nên một cuộc cách mạng trong cách tiếp cận styling khi được Adam Wathan và cộng đồng giới thiệu vào năm 2017. Khác biệt hoàn toàn với các framework CSS truyền thống như Bootstrap hay Foundation vốn cung cấp các component có sẵn, Tailwind CSS áp dụng triết lý "utility-first" - tập trung vào việc cung cấp các lớp tiện ích cấp thấp giúp xây dựng giao diện tùy chỉnh hoàn toàn mà không cần rời khỏi file HTML.

1.3.1. Đặc điểm nổi bật

Phương pháp tiếp cận Utility-First của Tailwind mang lại sự linh hoạt chưa từng có trong việc xây dựng giao diện. Hệ thống bao gồm hàng trăm lớp tiện ích được đặt tên một cách hệ thống, cho phép kiểm soát mọi khía cạnh của giao diện từ khoảng cách, màu sắc, kiểu chữ đến các hiệu ứng phức tạp.

Khả năng responsive được tích hợp sâu trong Tailwind thông qua hệ thống breakpoint mặc định, giúp tạo ra các giao diện đáp ứng trên mọi thiết bị một cách dễ dàng. Nhà phát triển chỉ cần thêm các tiền tố breakpoint vào trước các lớp tiện ích để kiểm soát giao diện trên các kích thước màn hình khác nhau.

Một số đặc điểm của TailWind như:

1. Phương pháp Utility-First triệt để(Utility-First Approach):

Tailwind cung cấp một hệ thống phong phú các lớp tiện ích (utility classes) từ cơ bản đến nâng cao, cho phép kiểm soát mọi khía cạnh của styling thông qua việc kết hợp các lớp này trực tiếp trong markup.

1. Hệ thống responsive mạnh mẽ(Responsive Design):

Tailwind áp dụng cách tiếp cận mobile-first với 5 breakpoint mặc định (sm:640px, md:768px, lg:1024px, xl:1280px, 2xl:1536px). Các lớp utility có thể được áp dụng theo breakpoint.

1. Customization mạnh mẽ:

Thông qua file tailwind.config.js, mọi khía cạnh của design system có thể được cá nhân hóa màu sắc, font chữ, spacing…

1. Hỗ trợ JIT(Just-In-Time) Mode:

Được giới thiệu trong Tailwind v3.0, chế độ JIT chỉ sinh CSS cho những utility đang được sử dụng, mang lại những lợi ích như:

* Kích thước file CSS nhỏ hơn 90% so với phương pháp truyền thống
* Thời gian build nhanh hơn đáng kể
* Hỗ trợ các tính năng dynamic như arbitrary variants (vd: md:[&:nth-child(2)]:underline)

1.3.2. Lợi ích của Tailwind CSS

1. Tốc độ phát triển nhanh:

Giảm thời gian chuyển đổi giữa HTML và CSS nhờ viết style trực tiếp trong markup.

Không cần đặt tên class phức tạp theo BEM hay các quy ước phức tạp khác (Ví dụ: .card-container\_\_header--active).

1. Tính nhất quán:

Áp dụng các giá trị được định nghĩa sẵn (ví dụ: spacing scale: 0.25rem, 0.5rem, 1rem…) để đảm bảo UI đồng bộ.

1. Dễ bảo trì:

Thay đổi style tập trung qua file cấu hình thay vì sửa nhiều file CSS.

Tránh “CSS bloat” (CSS thừa) nhờ cơ chế loại bỏ class không sử dụng (PurgeCSS).

1. Tích hợp tốt với JavaScript Frameaworks

Làm việc hiệu quả với React, Vue.js, Angular…

1.3.3. So sánh với các framework truyền thống

Khi so sánh với các framework truyền thống như Bootstrap, Tailwind nổi bật với sự linh hoạt và khả năng tùy biến cao hơn hẳn. Trong khi Bootstrap tập trung vào việc cung cấp các component có sẵn với giao diện tiêu chuẩn, Tailwind trao toàn quyền kiểm soát cho nhà phát triển để tạo ra các giao diện độc đáo. Bảng dưới là sự so sánh giữa 2 framework:

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Tailwind CSS** | **Bootstrap** |
| **Phong cách** | Utility-first, không component có sẵn | Component-based với style mặc định |
| **Tùy biến** | Cao (qua config file và utility classes) | Hạn chế (phải ghi đè CSS) |
| **Kích thước** | Nhỏ hơn nhờ JIT Mode | Lớn do chứa nhiều component không dùng |
| **Phù hợp** | Dự án cần UI độc đáo, tối ưu workflow | Dự án cần prototype nhanh với UI chuẩn |

Về hiệu suất, Tailwind vượt trội nhờ chế độ JIT chỉ sinh ra các CSS thực sự được sử dụng, trong khi các framework truyền thống thường phải bao gồm toàn bộ các style dù có được dùng hay không.

1.3.4. Ứng dụng Taiwind trong hệ thống một cửa

Trong hệ thống một cửa, Tailwind có thể được sử dụng để tạo ra các form đăng ký với giao diện nhất quán và chuyên nghiệp. Khả năng responsive giúp hệ thống hiển thị tối ưu trên mọi thiết bị từ máy tính để bàn đến điện thoại di động.

Tính năng dark mode có thể được triển khai dễ dàng để mang lại trải nghiệm người dùng tốt hơn trong các điều kiện ánh sáng khác nhau. Các hiệu ứng transition mượt mà có thể được áp dụng để nâng cao trải nghiệm tương tác.

Khi kết hợp với các framework JavaScript hiện đại, Tailwind giúp xây dựng các component có tính tái sử dụng cao, đảm bảo tính nhất quán trong khi vẫn giữ được sự linh hoạt trong thiết kế. Điều này đặc biệt quan trọng đối với hệ thống một cửa cần phục vụ nhiều đối tượng người dùng với các yêu cầu giao diện đa dạng.

1.4. Giới thiệu về 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.

1.4.1. Kiến trúc và thành phần chính của SQL Server

1. Database Engine:

Database Engine là thành phần cốt lõi, chịu trách nhiệm lưu trữ dữ liệu, xử lý truy vấn và quản lý giao dịch. Storage Engine quản lý lưu trữ dữ liệu trên đĩa (tables, indexes, logs). Query Optimizer tối ưu hóa truy vấn để đạt hiệu suất cao nhất. Transaction Manager đảm bảo tính toàn vẹn dữ liệu thông qua ACID (Atomicity, Consistency, Isolation, Durability).

1. SQL Server Management Studio (SSMS):

SSMS là công cụ đồ họa giúp quản trị cơ sở dữ liệu, bao gồm tạo và quản lý database, bảng, stored procedures, triggers. Nó cũng hỗ trợ thực thi truy vấn, phân tích hiệu suất và thao tác backup & restore dữ liệu.

1. SQL Server Integration Services (SSIS):

SSIS là công cụ ETL (Extract, Transform, Load), giúp tích hợp dữ liệu từ nhiều nguồn như Excel, CSV, Oracle, MySQL. Nó tự động hóa quy trình nhập liệu vào hệ thống, rất hữu ích khi kết nối với các hệ thống khác của trường như hệ thống điểm danh hoặc học phí.

1. SQL Server Reporting Services (SSRS):

SSRS là hệ thống báo cáo chuyên nghiệp, cho phép thiết kế báo cáo động như tổng hợp thông tin sinh viên hoặc thống kê dịch vụ. Báo cáo có thể xuất ra nhiều định dạng như PDF, Excel, Word và tích hợp với Power BI để phân tích dữ liệu trực quan.

1. SQL Server Analysis Services (SSAS):

SSAS là công cụ phân tích dữ liệu đa chiều (OLAP), giúp xây dựng Data Warehouse cho hệ thống. Nó hỗ trợ phân tích xu hướng, chẳng hạn như số lượng sinh viên sử dụng dịch vụ theo từng kỳ học, từ đó hỗ trợ ra quyết định dựa trên dữ liệu.

1.4.2. Các tính năng nâng cao của SQL Server

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

1. 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.
2. 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.
3. 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.

1.4.3. Các công cụ hỗ trợ phát triển

Ngoài SSMS, SQL Server còn cung cấp bộ công cụ đa dạng từ SQL Server Management Studio cho công tác quản trị truyền thống, đến Azure Data Studio - giải pháp phát triển đa nền tảng hiện đại. Đặc biệt, sự tích hợp sâu với Power Platform cho phép nhà trường nhanh chóng xây dựng các ứng dụng nghiệp vụ mà không cần nhiều kiến thức lập trình.

1.5. 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.

1.5.1. 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.

1.6. Kết luận chương 1

Chương này đã giới thiệu được các ngôn ngữ, thư viện và các framework liên quan khi xây dựng hệ thống hỗ trợ một cửa cho sinh viên Trường đại học Sao Đỏ. Trình bày được tính khả thi và cấp thiết của đề tài. Cũng đã nêu được các hạn chế của phương pháp truyền thống đã gây ra.

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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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).

1. Bảng Notifications (Thông báo)

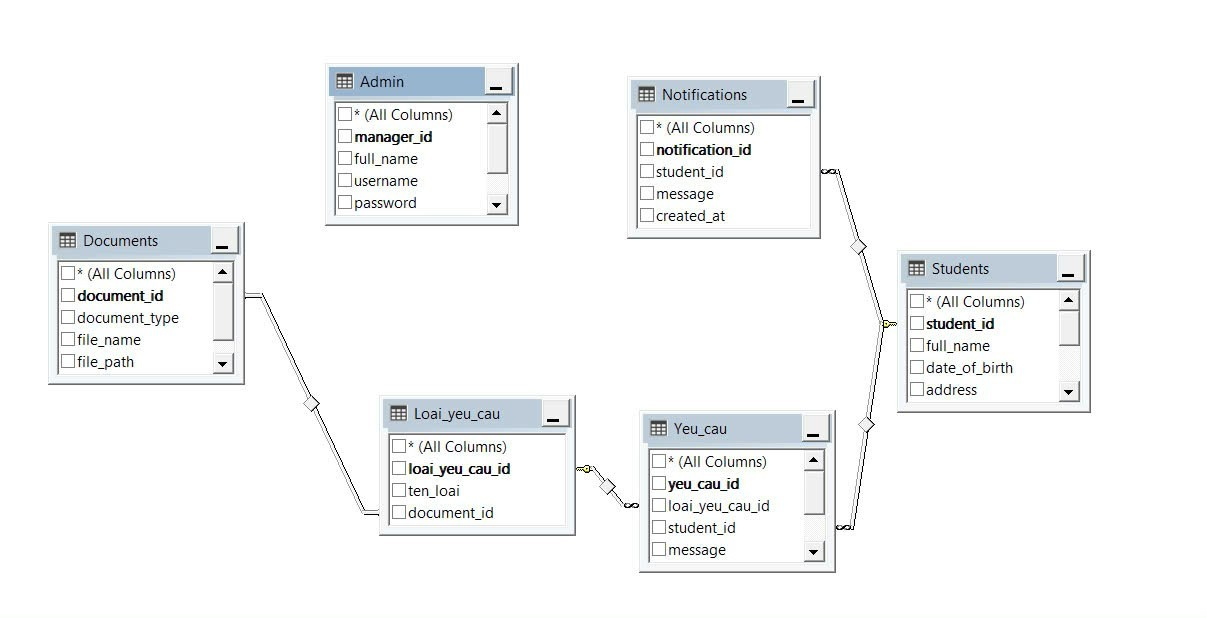
Thông báo push tới sinh viên khi yêu cầu được cập nhật.

Đặc tả chi tiết:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trường** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** | **Giá trị mẫu** |
| notification\_id | INT | PRIMARY KEY, IDENTITY(1,1) | ID tự tăng | 1, 2, 3, ... |
| student\_id | INT | NOT NULL, FOREIGN KET | ID sinh viên nhận thông báo | 2100568 |
| message | NVARCHAR(MAX) | NOT NULL | Nội dung thông báo | “Yêu cầu #2 đã được duyệt” |
| created\_at | DATETIME | DEFAULT GETDATE() | Thời gain gửi | 2025-02-18 08:30:00 |
| is\_read | BIT | DEFAULT 0 | Đã đọc/chưa đọc | 0 |

2.2.2. Sơ đồ quan hệ

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



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:

1. pool.request(): Kết nối tới SQL Server thông qua connection pool.
2. .input('id', id): Truyền tham số an toàn, tránh SQL injection.
3. .query(query): Thực thi câu lệnh SQL.
4. 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.

1. Controllers - Tầng xử lý logic
2. Routes (Định nghĩa API endpoints)

2.4. Thiết kế Front-end

WIP

Chương 3: KẾT QUẢ THỰC HIỆN

3.1.

KẾT LUẬN

Bài toán xác minh chữ kí là vấn đề có ý nghĩa quan trọng đối với chữ kí điện tử hay hoạt động ngân hàng. Mô hình Signet là một kiến trúc mạng nơ-ron mạnh mẽ và hiệu quả cho bài toán xác thực chữ ký tay trực tuyến. Sự kết hợp độc đáo giữa CNNs và RNNs cho phép mô hình nắm bắt được cả đặc trưng không gian và thời gian của chữ ký, mang lại độ chính xác và khả năng khái quát hóa cao. Mô hình Signet đã đóng góp quan trọng vào sự phát triển của lĩnh vực xác thực sinh trắc học và có tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau.

TÀI LIỆU THAM KHẢO

**1.** **Dataset Signature verification:** <https://www.kaggle.com/datasets/robinreni/signature-verification-dataset/data>

**2.** **Báo cáo phân tích hiệu quả của các mô hình mạng học máy**:

<https://jst-ud.vn/jst-ud/article/download/7820/5613/10672>