MỞ ĐẦU

Lý do chọn đề tài:

Trong bối cảnh giáo dục hiện đại, vai trò của Cố vấn học tập ngày càng trở nên quan trọng trong việc hỗ trợ sinh viên xây dựng kế hoạch học tập, giải quyết các vấn đề học thuật và định hướng nghề nghiệp. Tuy nhiên, các công cụ quản lý hồ sơ hiện tại thường được thực hiện bằng phương pháp thủ công hoặc sử dụng những hệ thống chưa tối ưu, dẫn đến việc mất nhiều thời gian, khó khăn trong việc lưu trữ, tìm kiếm và chia sẻ thông tin. Vì vậy, việc xây dựng một ứng dụng quản lý hồ sơ số hóa, hiện đại và bảo mật là cần thiết để hỗ trợ các giảng viên thực hiện nhiệm vụ một cách hiệu quả hơn.

Mục đích:

Mục tiêu của đề tài là phát triển một ứng dụng quản lý hồ sơ Cố vấn học tập nhằm:

- Số hóa quy trình quản lý hồ sơ.
- Tối ưu hóa việc lưu trữ tìm kiếm và truy xuất thông tin.
- Cung cấp giao diện thân thiện, dễ sử dụng cho giảng viên.
- Đảm bảo tính bảo mật và quyền riêng tư của dữ liệu người dùng.

Đối tượng nghiên cứu:

Ứng dụng được thiết kế dành riêng cho các giảng viên làm nhiệm vụ Cố vấn học tập thuộc khoa Kỹ thuật và Công nghệ, đồng thời có khả năng mở rộng cho các đơn vị khác nếu cần.

Phạm vi nghiên cứu:

- Ứng dụng tập trung vào việc quản lý các loại báo cáo, biểu mẫu liên quan đến nhiệm vụ của Cố vấn học tập.
- Phạm vi phát triển bao gồm giao diện người dùng, hệ thống quản lý dữ liệu, và cơ chế bảo mật thông tin.
- Giới hạn trong việc triển khai thử nghiệm tại khoa Kỹ thuật và Công nghệ, chưa mở rộng ra toàn trường.

CHƯƠNG 1: TỔNG QUAN

1.1 Bối cảnh và lý do chọn đề tài

Hiện nay, công tác quản lý hồ sơ của các Cố vấn học tập tại khoa Kỹ thuật và Công nghệ đang gặp nhiều khó khăn do phương pháp quản lý chủ yếu là thủ công, với việc lưu trữ thông tin trên máy cá nhân . Cụ thể, việc lưu trữ dữ liệu trên máy tính cá nhân gây ra các vấn đề như:

- **Khó khăn trong lưu trữ và tìm kiếm:** Hệ thống hồ sơ không được tổ chức khoa học, làm tăng thời gian và công sức trong việc truy xuất thông tin khi cần thiết.
- **Phân tán dữ liệu:** Các báo cáo và biểu mẫu thường không được lưu trữ tập trung, gây khó khăn trong quản lý và tổng hợp thông tin.
- **Thiếu tính bảo mật:** Việc không áp dụng các công cụ hỗ trợ bảo mật khiến dữ liệu dễ bị thất thoát hoặc xâm nhập trái phép, ảnh hưởng đến sự an toàn và độ tin cậy của thông tin.

Trước những hạn chế này, câu hỏi đặt ra là làm thế nào để tối ưu hóa việc tổ chức lưu trữ, truy xuất và chia sẻ thông tin, đồng thời đảm bảo tính bảo mật cao nhất. Việc thiết kế một ứng dụng quản lý hồ sơ cố vấn học tập hiệu quả và an toàn sẽ góp phần giải quyết triệt để các vấn đề nêu trên, tạo điều kiện thuận lợi cho công tác quản lý và hỗ trợ hoạt động cố vấn học tập.

1.2 Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu là xây dựng một ứng dụng quản lý hồ sơ hiện đại, đáp ứng các yêu cầu thực tiễn trong công tác cố vấn học tập. Cụ thể, ứng dụng được thiết kế nhằm:

- Hỗ trợ lưu trữ các báo cáo và biểu mẫu của Cố vấn học tập một cách khoa học và có tổ chức, giúp quản lý thông tin một cách hệ thống.
- Đảm bảo khả năng truy xuất dữ liệu nhanh chóng, đồng thời tích hợp các tính năng bảo mật để bảo vệ thông tin người dùng và dữ liệu quan trọng.
- Tạo ra một giao diện thân thiện, dễ sử dụng, phù hợp với nhu cầu và khả năng sử dụng của đối tượng chính là các giảng viên.

Ứng dụng này không chỉ giải quyết các vấn đề hiện tại mà còn mang lại sự thuận tiện và hiệu quả cao trong việc hỗ trợ các hoạt động cố vấn học tập.

1.3 Phạm vi nghiên cứu

Đối tượng nghiên cứu: Nghiên cứu tập trung vào các giảng viên thuộc khoa Kỹ thuật và Công nghệ, những người đang đảm nhận nhiệm vụ Cố vấn học tập. Đây là nhóm đối tượng chính sẽ trực tiếp sử dụng ứng dụng để quản lý hồ sơ và thực hiện các công việc liên quan.

Phạm vi nội dung:

Nghiên cứu hướng đến việc xây dựng các chức năng quản lý hồ sơ phục vụ công tác cố vấn học tập, bao gồm:

- Quản lý các loại hồ sơ, báo cáo và biểu mẫu liên quan.
- Các chức năng như tải lên, tải xuống, tìm kiếm, và tổ chức lưu trữ dữ liệu được tích hợp nhằm hỗ trợ tối ưu hóa công việc.

Phạm vi công nghệ:

Dự án sử dụng các công nghệ hiện đại để phát triển ứng dụng:

- **Next.js**: Framework JavaScript mạnh mẽ để xây dựng giao diện người dùng, mang lại trải nghiệm mượt mà và thân thiện.
- **Firebase**: Nền tảng backend toàn diện hỗ trợ lưu trữ, xử lý dữ liệu, và tích hợp bảo mật cho hệ thống.

Những công nghệ này được lựa chọn nhằm đảm bảo ứng dụng có hiệu suất cao, dễ triển khai, và đáp ứng tốt các yêu cầu về bảo mật dữ liệu.

1.4 Phương pháp nghiên cứu

Phân tích vấn đề:

- Khảo sát thực trạng: Tiến hành khảo sát và đánh giá quy trình quản lý hồ sơ hiện tại của các Cố vấn học tập tại khoa Kỹ thuật và Công nghệ.
- Xác định yêu cầu hệ thống: Thu thập ý kiến từ giảng viên và các bên liên quan để xây dựng danh sách các yêu cầu chức năng và phi chức năng cần thiết cho ứng dụng.

Thiết kế giải pháp:

- Lập kế hoạch phát triển ứng dụng, bao gồm thiết kế kiến trúc hệ thống và giao diện người dùng.
- Lựa chọn công nghệ Next.js để phát triển giao diện người dùng và Firebase làm nền tảng lưu trữ và xử lý dữ liệu, đảm bảo tính hiện đại, bảo mật và hiệu quả.

Thực nghiệm:

- Xây dựng ứng dụng dựa trên kế hoạch đã thiết lập.
- Tiến hành kiểm thử để đánh giá tính chính xác và hiệu quả của các chức năng.
- Triển khai ứng dụng trong phạm vi nhỏ tại khoa Kỹ thuật và Công nghệ để thu thập phản hồi và điều chỉnh trước khi mở rộng phạm vi sử dụng.

Quy trình nghiên cứu này đảm bảo rằng ứng dụng được phát triển đáp ứng đúng nhu cầu thực tế, vận hành hiệu quả và có khả năng áp dụng trong thực tế.

1.5 Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học: Nghiên cứu ứng dụng các công nghệ hiện đại, cụ thể là Next.js và Firebase, vào lĩnh vực quản lý hồ sơ trong giáo dục, góp phần mở rộng phạm vi áp dụng của các công nghệ này. Đóng góp tài liệu tham khảo hữu ích cho các nghiên cứu tương lai liên quan đến việc số hóa quy trình quản lý và tự động hóa trong môi trường giáo dục, đặc biệt trong công tác cố vấn học tập.

Ý nghĩa thực tiễn: Giúp giảng viên khoa Kỹ thuật và Công nghệ tiết kiệm thời gian và công sức trong việc lưu trữ, tìm kiếm và quản lý hồ sơ cố vấn học tập, từ đó nâng cao hiệu quả làm việc. Đảm bảo tính bảo mật và minh bạch trong xử lý dữ liệu, góp phần tạo sự tin cậy và an toàn trong công tác quản lý thông tin quan trọng.

Nghiên cứu này không chỉ mang lại giá trị học thuật mà còn có tác động tích cực đến công việc thực tế của các giảng viên, hướng đến một môi trường giáo dục hiện đại và chuyên nghiệp hơn.

1.6 Cấu trúc của đồ án

Tổng quan về cấu trúc của đồ án :

- Chương 1: Tổng quan – Trình bày bối cảnh, mục tiêu, phạm vi và phương pháp nghiên cứu.
- Chương 2: Nghiên cứu lý thuyết – Giới thiệu cơ sở lý thuyết và nền tảng công nghệ.
- Chương 3: Hiện thực hóa nghiên cứu – Mô tả quá trình phát triển ứng dụng.
- Chương 4: Kết quả nghiên cứu – Trình bày các kết quả đạt được và đánh giá hiệu quả.
- Chương 5: Kết quả đạt được

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Framework Next.js

Next.js là một framework mã nguồn mở được phát triển dựa trên thư viện React, cung cấp các tính năng mở rộng giúp tối ưu hóa hiệu suất ứng dụng, bao gồm Render phía máy chủ (Server-Side Rendering - SSR) và tạo trang web tĩnh (Static Site Generation - SSG). Next.js sử dụng React làm nền tảng, mở rộng thêm các tính năng giúp việc xây dựng ứng dụng web trở nên dễ dàng và hiệu quả hơn.

Một trong những tính năng nổi bật của Next.js là SSR, cho phép máy chủ xử lý toàn bộ dữ liệu cần thiết và xử lý JavaScript trước khi gửi trang đã render hoàn chỉnh đến trình duyệt. Điều này giúp các trang web tải nhanh hơn, mang lại trải nghiệm người dùng mượt mà và tăng tốc độ phản hồi. Với SSR, nội dung của trang web được hiển thị ngay lập tức khi người dùng truy cập mà không cần chờ đợi việc tải lại các phần tử riêng lẻ.

Ngoài ra, SSR cũng mang lại lợi ích lớn trong việc tối ưu hóa SEO, giúp các trang web dễ dàng được quét và xếp hạng cao hơn trên các công cụ tìm kiếm. Thông qua việc sử dụng thẻ <head> trong Next.js, bạn có thể dễ dàng quản lý metadata của các trang web, điều này không thể thực hiện được trực tiếp trong React. Thẻ <head> đóng vai trò quan trọng trong việc cải thiện SEO, giúp trang web dễ dàng được nhận diện và đánh giá tốt từ các công cụ tìm kiếm.

2.1.1 Các ưu điểm của Next.js

- **Tốc độ tải trang nhanh hơn:** Next.js giúp cải thiện hiệu suất ứng dụng bằng cách render trang phía server. Điều này làm cho các trang web tải nhanh hơn đáng kể so với các ứng dụng React thông thường, vốn cần phải tải các thành phần JavaScript riêng biệt.
- **Hỗ trợ web tĩnh:** Next.js cung cấp các tính năng mạnh mẽ để xây dựng các trang web tĩnh, giúp tối ưu hóa tốc độ tải trang và giảm tải cho máy chủ.
- **Dễ dàng tiếp cận với người dùng React:** Nếu bạn đã quen thuộc với React, việc làm quen và sử dụng Next.js sẽ trở nên dễ dàng hơn nhiều vì Next.js xây dựng trên nền tảng của React.
- **Tối ưu hóa hiệu suất với code splitting:** Next.js tự động chia nhỏ mã nguồn cho từng trang, giúp cải thiện hiệu suất và tốc độ tải trang bằng cách tải chỉ các mã nguồn cần thiết.

- **API routes tích hợp sẵn:** Next.js hỗ trợ xây dựng các API nội bộ thông qua các API routes, giúp đơn giản hóa quá trình tạo các điểm cuối API.
- **Tích hợp dễ dàng với CSS, JSX và TypeScript:** Next.js hỗ trợ các tính năng hiện đại như CSS-in-JS, JSX, và TypeScript, giúp bạn dễ dàng tích hợp và sử dụng các công nghệ này trong dự án của mình.
- **Tùy chỉnh linh hoạt với plugin:** Next.js cung cấp khả năng tích hợp các plugin, giúp bạn tùy chỉnh và mở rộng framework theo nhu cầu của từng dự án.

2.1.2 Các nhược điểm của Next.js

Khung công cụ cố định: Next.js là một framework có cấu trúc và phương pháp phát triển rõ ràng. Điều này có thể hạn chế sự linh hoạt trong một số trường hợp, khi bạn muốn tùy chỉnh quy trình phát triển hoặc sử dụng công cụ khác ngoài phạm vi của Next.js. Tuy nhiên, các phương pháp và công cụ của Next.js thường đáp ứng tốt yêu cầu của đa số các dự án, đặc biệt là trong phát triển web hiện đại.

2.2 Tailwind CSS

Trong quá trình phát triển ứng dụng web hiện đại, việc thiết kế giao diện người dùng (UI) dễ dàng và hiệu quả là một yếu tố quan trọng quyết định đến trải nghiệm người dùng. Các framework CSS truyền thống thường yêu cầu lập trình viên phải viết một lượng lớn mã CSS tùy chỉnh để đạt được giao diện mong muốn. Tuy nhiên, sự phát triển của các framework CSS mới như **Tailwind CSS** đã mở ra một cách tiếp cận mới, giúp việc xây dựng giao diện web trở nên đơn giản và nhanh chóng hơn.

Hình 1 TailwindCSS

Tailwind CSS là một framework CSS tiện ích (utility-first), giúp các nhà phát triển có thể thiết kế các giao diện đẹp mà không cần phải viết quá nhiều mã CSS tùy chỉnh. Các lớp tiện ích trong Tailwind CSS cho phép bạn thay đổi các thuộc tính như màu sắc, kích thước, bố cục, khoảng cách, và nhiều yếu tố khác chỉ bằng cách sử dụng các lớp CSS có sẵn. Điều này không chỉ giúp tăng tốc quá trình phát triển mà còn giúp mã nguồn trở nên sạch sẽ và dễ bảo trì hơn.

Cùng với sự phát triển mạnh mẽ của **Next.js**, một framework mạnh mẽ của React, việc tích hợp Tailwind CSS vào các dự án Next.js đã trở thành một xu hướng phổ biến. Next.js hỗ trợ tối ưu hóa hiệu suất, cung cấp các tính năng như SSR (Server-Side Rendering) và SSG (Static Site Generation), giúp phát triển ứng dụng web nhanh chóng và hiệu quả. Khi kết hợp với Tailwind CSS, các nhà phát triển có thể xây dựng giao diện động, đẹp mắt mà vẫn đảm bảo hiệu suất cao và khả năng tùy chỉnh linh hoạt.

2.2.1 Tính năng nổi bật của Tailwind CSS

Utility-First Approach (Tiếp cận utility-first): Một trong những điểm mạnh lớn nhất của Tailwind CSS là sự tập trung vào việc cung cấp các lớp utility (tiện ích) thay vì phải viết CSS riêng biệt cho từng phần tử HTML. Điều này có nghĩa là bạn có thể áp dụng các thuộc tính CSS trực tiếp vào các phần tử HTML thông qua các lớp CSS đã được chuẩn bị sẵn. Các lớp này thường rất cụ thể và chỉ thực hiện một công việc duy nhất, giúp giảm thiểu việc phải viết các class CSS phức tạp.

Khả năng tùy chỉnh cao: Tailwind CSS rất linh hoạt và dễ tùy chỉnh. Bạn có thể điều chỉnh các cấu hình trong file tailwind.config.js để thay đổi các thuộc tính như màu sắc, kích thước, font chữ, khoảng cách, v.v. Điều này cho phép bạn tạo ra một hệ thống thiết kế đồng nhất và dễ dàng thay đổi toàn bộ giao diện của trang web mà không cần phải sửa nhiều file CSS.

Tích hợp tốt với các công cụ build: Tailwind CSS tích hợp rất tốt với các công cụ build như Webpack, PostCSS, và Next.js. Bạn có thể dễ dàng cài đặt và cấu hình Tailwind trong các ứng dụng của mình để tận dụng các tính năng tối ưu hóa mã nguồn và giảm kích thước file CSS đầu ra.

- **PurgeCSS:** Tailwind CSS tích hợp với PurgeCSS, giúp loại bỏ các lớp không sử dụng trong quá trình build, giảm kích thước file CSS và tối ưu hóa hiệu suất trang web.
- **JIT (Just-In-Time) Compiler:** Tính năng JIT của Tailwind CSS giúp tạo ra các lớp CSS cần thiết khi trang web được tải, giúp giảm thời gian tải và tối ưu hóa hiệu suất.

Hỗ trợ responsive design dễ dàng: Tailwind CSS cho phép bạn dễ dàng thiết kế giao diện đáp ứng (responsive) nhờ vào các lớp tiện ích dành riêng cho các breakpoint khác nhau. Bạn có thể sử dụng các tiền tố như sm:, md:, lg:, và xl: để áp dụng các lớp khác nhau cho các kích thước màn hình khác nhau mà không cần phải viết mã media query thủ công.

Kết hợp dễ dàng với framework JavaScript khác: Tailwind CSS có thể được kết hợp dễ dàng với các framework JavaScript như React, Vue, Angular, và đặc biệt là Next.js. Nhờ vào việc áp dụng các lớp tiện ích trực tiếp vào JSX hoặc HTML, Tailwind giúp tăng tốc quá trình phát triển UI mà không gặp phải các vấn đề tương thích.

2.2.2 Tailwind CSS và Next.js

Next.js là một framework mạnh mẽ được xây dựng trên React, giúp phát triển các ứng dụng web động và tối ưu hóa cho SEO. Việc tích hợp Tailwind CSS với Next.js mang lại nhiều lợi ích trong việc xây dựng giao diện người dùng (UI) đẹp mắt và dễ sử dụng.

- **Phát triển nhanh chóng:** Với Tailwind CSS, việc thiết kế giao diện trở nên rất nhanh chóng nhờ vào các lớp utility, giúp giảm thiểu thời gian viết CSS.
- **Tối ưu hóa hiệu suất:** Khi kết hợp với Next.js, Tailwind CSS có thể loại bỏ các lớp không sử dụng trong quá trình build, giúp tối ưu hóa dung lượng tệp CSS.
- **Khả năng tùy chỉnh cao:** Tailwind CSS cho phép bạn dễ dàng tùy chỉnh giao diện mà không cần phải thay đổi nhiều trong cấu trúc HTML hoặc CSS.
- **Tính tương thích với React:** Next.js được xây dựng dựa trên React, vì vậy việc sử dụng Tailwind CSS với Next.js rất dễ dàng và linh hoạt, nhờ vào khả năng sử dụng các class utility trực tiếp trong JSX.

2.3 Firebase

Firebase là nền tảng phát triển ứng dụng full-stack của Google, cung cấp một bộ công cụ mạnh mẽ giúp các nhà phát triển xây dựng và triển khai ứng dụng web và di động mà không cần quản lý hạ tầng backend phức tạp. Nền tảng này tích hợp nhiều dịch vụ như cơ sở dữ liệu thời gian thực, xác thực người dùng, và lưu trữ dữ liệu, hỗ trợ các ứng dụng phát triển nhanh chóng và hiệu quả. Các dịch vụ nổi bật của Firebase bao gồm Firebase Authentication, Firebase Firestore, Firebase Realtime Database, và Firebase Cloud Storage, giúp tối ưu hóa việc lưu trữ, đồng bộ hóa và bảo mật dữ liệu người dùng.

2.3.1 Các dịch vụ của Firebase

Firebase cung cấp một loạt các dịch vụ, bao gồm:

- **Firebase Realtime Database**: Đây là một cơ sở dữ liệu NoSQL được tối ưu hóa cho việc đồng bộ dữ liệu thời gian thực giữa người dùng và ứng dụng. Các thay đổi dữ liệu được tự động cập nhật trên mọi thiết bị mà không cần phải tải lại trang.
- **Firebase Firestore**: Firestore là phiên bản mới hơn của Firebase Realtime Database, cung cấp các tính năng mạnh mẽ hơn, bao gồm khả năng truy vấn dữ liệu linh hoạt hơn, bảo mật và phân quyền chi tiết hơn.
- **Firebase Authentication**: Dịch vụ xác thực người dùng của Firebase hỗ trợ nhiều phương thức đăng nhập như email/password, đăng nhập qua mạng xã hội (Google, Facebook, Twitter), và xác thực thông qua số điện thoại.
- **Firebase Cloud Messaging (FCM)**: FCM cho phép gửi thông báo đẩy đến các thiết bị di động và ứng dụng web. FCM hỗ trợ cả thông báo cho người dùng cuối và thông báo hệ thống.
- **Firebase Cloud Storage**: Cung cấp khả năng lưu trữ tệp lớn như hình ảnh, video, và các loại dữ liệu không phải là văn bản. Firebase Cloud Storage tích hợp với Firebase Authentication để đảm bảo chỉ người dùng có quyền truy cập mới có thể tải lên và tải xuống tệp.
- **Firebase Analytics**: Cung cấp các công cụ phân tích mạnh mẽ, giúp các nhà phát triển theo dõi hành vi của người dùng trong ứng dụng, từ đó đưa ra quyết định phát triển ứng dụng thông minh hơn.

- **Firebase Functions**: Dịch vụ tính toán backend cho phép các nhà phát triển viết các hàm JavaScript mà không cần phải triển khai một máy chủ backend. Các hàm này có thể được kích hoạt bởi các sự kiện trong Firebase hoặc từ các API HTTP.

2.3.2 Các tính năng nổi bật của Firebase

- **Đồng bộ thời gian thực**: Firebase Realtime Database và Firestore cho phép các ứng dụng cập nhật và đồng bộ hóa dữ liệu ngay lập tức, đảm bảo sự trải nghiệm mượt mà cho người dùng.
- **Xác thực đơn giản**: Firebase Authentication cung cấp một cách thức xác thực đơn giản và bảo mật cho các ứng dụng mà không cần phải quản lý các chi tiết phức tạp về backend.
- **Tính bảo mật cao**: Firebase sử dụng các cơ chế bảo mật và mã hóa tiên tiến để bảo vệ dữ liệu của người dùng. Các chính sách bảo mật có thể được cấu hình linh hoạt thông qua Firebase Security Rules.
- **Tích hợp với Google Cloud**: Firebase là một phần của hệ sinh thái Google Cloud, giúp các ứng dụng dễ dàng tích hợp với các dịch vụ của Google như Google Analytics, Google Ads, và Google Cloud Storage.
- **Quản lý phân phối ứng dụng dễ dàng**: Firebase App Distribution giúp việc phân phối bản beta của ứng dụng đến người thử nghiệm trở nên đơn giản hơn.
- **Hỗ trợ đa nền tảng**: Firebase hỗ trợ phát triển ứng dụng trên nhiều nền tảng như iOS, Android và Web.

2.3.3 Ưu điểm của Firebase

- **Độ tin cậy dữ liệu mạnh mẽ:** Firebase cung cấp một cơ sở hạ tầng ổn định và bảo mật cao, giúp đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu. Với khả năng đồng bộ hóa dữ liệu thời gian thực, các thay đổi được phản ánh ngay lập tức trên toàn bộ hệ thống.
- **Tích hợp liền mạch với dịch vụ bên thứ ba:** Firebase dễ dàng tích hợp với các dịch vụ và công cụ bên ngoài như Google Analytics, AdMob, và các công cụ phát triển khác. Điều này giúp tối ưu hóa quy trình phát triển ứng dụng và mở rộng chức năng của ứng dụng mà không gặp phải khó khăn trong việc tích hợp.

- **Nền tảng mạnh mẽ và có khả năng mở rộng:** Firebase hỗ trợ khả năng mở rộng linh hoạt, đáp ứng nhu cầu phát triển của các ứng dụng có quy mô lớn và phức tạp. Nền tảng này có thể xử lý lượng người dùng lớn và dữ liệu khổng lồ mà không ảnh hưởng đến hiệu suất.
- **Tích hợp sẵn các tính năng như xác thực người dùng và lưu trữ:** Các dịch vụ như Firebase Authentication và Firebase Cloud Storage giúp đơn giản hóa việc triển khai các tính năng bảo mật và quản lý người dùng mà không cần xây dựng lại từ đầu, giúp tiết kiệm thời gian phát triển.
- **Dễ sử dụng:** Các API và SDK của Firebase được thiết kế dễ hiểu và dễ tích hợp, giúp các nhà phát triển nhanh chóng làm quen và triển khai các tính năng backend mà không gặp phải khó khăn trong việc cấu hình.

2.3.4 Nhược điểm của Firebase

- **Yêu cầu đường học dốc cho các nhà phát triển mới:** Mặc dù Firebase dễ sử dụng đối với các nhà phát triển có kinh nghiệm, nhưng những người mới bắt đầu có thể gặp khó khăn khi làm quen với các tính năng nâng cao như Firebase Functions hoặc các công cụ phức tạp khác. Việc làm quen với các tính năng này có thể yêu cầu một thời gian học hỏi nhất định.
- **Giới hạn trong việc tùy chỉnh cao:** Firebase có thể không phù hợp cho các ứng dụng yêu cầu cấu hình backend phức tạp, vì một số dịch vụ của Firebase thiếu khả năng tùy chỉnh mạnh mẽ như các giải pháp backend truyền thống. Điều này có thể gây hạn chế khi cần triển khai các yêu cầu đặc biệt hoặc các tính năng không được hỗ trợ sẵn.
- **Chi phí:** Mặc dù Firebase cung cấp gói miễn phí, nhưng khi ứng dụng phát triển và lượng người dùng tăng lên, chi phí sử dụng các dịch vụ của Firebase có thể trở nên đắt đỏ, đặc biệt là đối với các dịch vụ như lưu trữ và cơ sở dữ liệu thời gian thực. Các tính năng cao cấp hoặc mức sử dụng vượt qua mức miễn phí sẽ làm tăng đáng kể chi phí duy trì ứng dụng.

Với Firebase, có thể dễ dàng tích hợp và phát triển các tính năng backend cho ứng dụng mà không cần lo lắng về việc quản lý cơ sở hạ tầng, giúp tối ưu hóa quy trình phát triển và quản lý dữ liệu người dùng.

2.4 Tổng quan về quản lý hồ sơ số

2.4.1 Khái niệm quản lý hồ sơ số

Quản lý hồ sơ số là quá trình tổ chức, lưu trữ, xử lý và bảo quản các hồ sơ, tài liệu dưới dạng kỹ thuật số, thay thế hoặc bổ sung cho phương pháp quản lý truyền thống bằng hồ sơ giấy. Hồ sơ số bao gồm các loại tài liệu như văn bản, biểu mẫu, báo cáo, hình ảnh, video, hoặc bất kỳ định dạng thông tin nào được số hóa.

Quá trình quản lý hồ sơ số thường sử dụng các hệ thống phần mềm chuyên dụng, giúp lưu trữ dữ liệu một cách khoa học và có hệ thống. Các dữ liệu này được tổ chức theo cấu trúc rõ ràng, giúp người dùng dễ dàng tìm kiếm, truy cập và chia sẻ khi cần.

Một hệ thống quản lý hồ sơ số hiện đại thường tích hợp nhiều công nghệ tiên tiến, bao gồm:

- **Cơ sở dữ liệu:** Hỗ trợ lưu trữ và xử lý dữ liệu với dung lượng lớn, cho phép quản lý tập trung và truy xuất hiệu quả.
- **Dịch vụ đám mây:** Đảm bảo khả năng truy cập mọi lúc, mọi nơi và tăng cường khả năng mở rộng lưu trữ.
- **Mã hóa dữ liệu:** Bảo vệ dữ liệu khỏi nguy cơ bị truy cập trái phép hoặc thất thoát thông tin.

Quản lý hồ sơ số không chỉ giúp số hóa thông tin để dễ dàng truy xuất mà còn tối ưu hóa việc lưu trữ, nâng cao tính bảo mật và tạo nền tảng cho việc phân tích, khai thác dữ liệu hiệu quả. Đây là một phần quan trọng trong quá trình chuyển đổi số, đáp ứng nhu cầu hiện đại hóa và tối ưu hóa quản lý trong các lĩnh vực khác nhau, đặc biệt trong môi trường giáo dục và tổ chức hành chính.

2.4.2 Vai trò của quản lý hồ sơ sô

Tăng cường hiệu quả lưu trữ và truy cập: Hồ sơ số hóa cho phép lưu trữ một lượng lớn dữ liệu trong không gian nhỏ và được tổ chức một cách khoa học, giúp người dùng dễ dàng phân loại và quản lý thông tin. Các công cụ tìm kiếm tích hợp giúp tìm kiếm, truy cập, và chia sẻ thông tin nhanh chóng, giảm thiểu thời gian và công sức so với phương pháp thủ công truyền thống.

Đảm bảo an toàn và bảo mật dữ liệu: Với sự hỗ trợ của các công nghệ hiện đại như mã hóa dữ liệu, quản lý hồ sơ số giúp bảo vệ dữ liệu khỏi nguy cơ xâm nhập,

thất thoát hoặc sửa đổi trái phép. Ngoài ra, việc phân quyền truy cập đảm bảo tính minh bạch và kiểm soát chặt chẽ quyền truy cập thông tin.

Tăng tính linh hoạt và khả năng cộng tác: Hồ sơ số hóa được lưu trữ trên các nền tảng đám mây hoặc hệ thống trực tuyến, cho phép người dùng truy cập từ xa qua nhiều thiết bị khác nhau. Điều này hỗ trợ tối ưu hóa khả năng cộng tác giữa các cá nhân hoặc nhóm, đặc biệt trong môi trường làm việc từ xa hoặc cần chia sẻ tài liệu liên tục.

Giảm thiểu chi phí và tác động môi trường: So với lưu trữ hồ sơ giấy, việc quản lý hồ sơ số giúp giảm đáng kể chi phí liên quan đến giấy, in ấn và lưu trữ vật lý. Đồng thời, việc số hóa cũng góp phần bảo vệ môi trường bằng cách giảm lượng rác thải từ giấy.

Hỗ trợ phân tích và ra quyết định: Các hệ thống quản lý hồ sơ số thường tích hợp khả năng phân tích dữ liệu, giúp trích xuất thông tin quan trọng từ hồ sơ. Điều này hỗ trợ các nhà quản lý đưa ra quyết định nhanh chóng và chính xác dựa trên dữ liệu thực tế.

Những vai trò trên cho thấy rằng quản lý hồ sơ số không chỉ cải thiện hiệu quả vận hành mà còn mở ra cơ hội ứng dụng công nghệ để tối ưu hóa công tác quản lý trong mọi lĩnh vực, đặc biệt là trong giáo dục.

2.4.3 Một số yêu cầu cơ bản của quản lý hồ sơ

Tính đầy đủ

- Đảm bảo thông tin đầy đủ và chính xác: Các hồ sơ phải chứa đủ thông tin cần thiết để đáp ứng yêu cầu quản lý và truy xuất thông tin, tránh thiếu sót làm ảnh hưởng đến hoạt động của tổ chức.
- Cập nhật thường xuyên: Dữ liệu trong hồ sơ cần được duy trì và cập nhật kịp thời để phản ánh đúng thực trạng.

Tính nhất quán

- Sử dụng chuẩn hóa thông tin: Các định dạng, thuật ngữ, và quy ước đặt tên cần được áp dụng đồng nhất trên toàn hệ thống để tránh nhầm lẫn hoặc sai lệch khi quản lý và sử dụng.
- Tuân thủ các quy định pháp luật: Hồ sơ phải được quản lý theo các tiêu chuẩn và quy định pháp lý hiện hành, đảm bảo tính hợp pháp và minh bạch.

Tính bảo mật

- Phân quyền truy cập: Chỉ những cá nhân hoặc bộ phận được ủy quyền mới được phép truy cập hoặc chỉnh sửa hồ sơ.
- Sao lưu dữ liệu: Hệ thống cần có cơ chế sao lưu định kỳ để phòng ngừa mất mát dữ liệu do lỗi kỹ thuật hoặc tấn công mạng..

Tính khả dụng

- Truy cập dễ dàng: Người dùng cần có khả năng truy cập hồ sơ một cách nhanh chóng, chính xác, không bị gián đoạn bởi các lỗi kỹ thuật hoặc hạn chế hạ tầng.
- Hỗ trợ tìm kiếm thông minh: Hệ thống cần cung cấp chức năng tìm kiếm theo nhiều tiêu chí như tên, ngày tháng, hoặc từ khóa liên quan.

Tính linh hoạt

- Khả năng tích hợp: Hệ thống quản lý hồ sơ phải dễ dàng tích hợp với các phần mềm hoặc ứng dụng khác trong tổ chức.
- Thích ứng với thay đổi: Có khả năng mở rộng hoặc điều chỉnh khi tổ chức phát triển hoặc thay đổi yêu cầu quản lý.

Tính toàn vẹn

- Bảo vệ nguyên bản dữ liệu: Hồ sơ phải được lưu trữ nguyên vẹn, không bị chỉnh sửa hoặc xóa bỏ trái phép.
- Theo dõi lịch sử thay đổi: Hệ thống cần ghi lại mọi thay đổi trong hồ sơ, bao gồm thông tin về người thực hiện và thời điểm thay đổi.

Tính hiệu quả

- Tiết kiệm thời gian: Quản lý hồ sơ cần được thực hiện nhanh chóng, giảm thiểu các thao tác thủ công hoặc lỗi lặp đi lặp lại.
- Tối ưu hóa chi phí: Áp dụng công nghệ và quy trình giúp giảm chi phí vận hành nhưng vẫn đảm bảo hiệu suất.

Tính minh bạch và truy xuất

- Theo dõi và kiểm tra dễ dàng: Mọi hành động liên quan đến hồ sơ (truy cập, chỉnh sửa, xóa) cần được ghi lại để phục vụ mục đích kiểm tra và báo cáo.
- Lưu trữ lâu dài: Đảm bảo hồ sơ có thể được lưu trữ và truy xuất trong thời gian dài mà không bị hư hại hoặc mất mát.

Hồ sơ Cố vấn học tập là tập hợp các tài liệu, báo cáo liên quan đến quá trình cố vấn và hỗ trợ học tập của giảng viên dành cho sinh viên. Các hồ sơ này đóng vai trò quan trọng trong việc theo dõi, đánh giá, và hỗ trợ sinh viên, đồng thời là căn cứ để thực hiện các hoạt động hành chính và báo cáo trong khoa.

2.4.4 Các loại tài liệu trong hồ sơ cố vấn học tập

Hồ sơ Cố vấn học tập là tập hợp các tài liệu và báo cáo quan trọng liên quan đến quá trình cố vấn và hỗ trợ học tập của giảng viên đối với sinh viên. Những hồ sơ này không chỉ giúp theo dõi và đánh giá tiến độ học tập của sinh viên, mà còn đóng vai trò là căn cứ để thực hiện các hoạt động hành chính và báo cáo trong khoa. Các tài liệu trong hồ sơ cố vấn học tập bao gồm nhiều loại, mỗi loại tài liệu đều có vai trò và mục đích riêng biệt, góp phần hỗ trợ giảng viên trong công tác cố vấn.

Một trong những tài liệu quan trọng trong hồ sơ là **lý lịch sinh viên**, cung cấp thông tin cơ bản về sinh viên, giúp giảng viên nắm bắt được nền tảng của sinh viên để hỗ trợ họ một cách cá nhân hóa. Thông qua lý lịch này, giảng viên có thể theo dõi tình hình học tập và phát triển của sinh viên, từ đó đưa ra những hỗ trợ kịp thời.

Biên bản đầu khóa là tài liệu ghi nhận các nội dung trong các buổi họp đầu khóa giữa giảng viên và sinh viên, bao gồm quy chế đào tạo, lộ trình học tập, cũng như các quyền lợi và nghĩa vụ của sinh viên. Biên bản này giúp sinh viên nắm rõ kế hoạch học tập và kỳ vọng từ phía nhà trường ngay từ đầu khóa.

Kế hoạch tiếp đón sinh viên là một tài liệu quan trọng khác, giúp định hướng cho sinh viên mới làm quen với môi trường học tập tại khoa. Tài liệu này giúp tổ chức các hoạt động hỗ trợ đầu khóa một cách có tổ chức và hiệu quả, tạo nền tảng vững chắc cho sinh viên trong suốt năm học.

Kế hoạch cố vấn học tập cả năm là tài liệu xác định các hoạt động và thời gian biểu liên quan đến công tác cố vấn học tập, giúp giảng viên tổ chức công việc một cách khoa học và hiệu quả.

Báo cáo học kỳ là những tài liệu tổng hợp kết quả học tập, rèn luyện của sinh viên trong từng học kỳ, từ đó giúp giảng viên đánh giá tiến độ và đề xuất các biện pháp hỗ trợ sinh viên kịp thời.

Biên bản xét học bổng lưu lại quy trình xét duyệt học bổng, danh sách sinh viên nhận học bổng và các thông tin liên quan, đảm bảo tính minh bạch trong quá trình xét duyệt.

Báo cáo hàng tháng ghi lại tình hình sinh viên và các hoạt động cố vấn hàng tháng, cung cấp thông tin cập nhật về các khó khăn mà sinh viên gặp phải, từ đó giúp giảng viên đưa ra giải pháp hỗ trợ thích hợp.

Tất cả những tài liệu này không chỉ là cơ sở để đánh giá kết quả học tập của sinh viên mà còn giúp giảng viên tổ chức công tác cố vấn học tập một cách có hệ thống và hiệu quả, nâng cao chất lượng giảng dạy và hỗ trợ sinh viên trong suốt quá trình học tập tại trường.

2.4.5 Tầm quan trọng của hồ sơ Cố vấn học tập

Đối với sinh viên: Hồ sơ Cố vấn học tập là công cụ quan trọng giúp họ nhận được sự hỗ trợ kịp thời trong quá trình học tập và phát triển nghề nghiệp. Thông qua các tài liệu trong hồ sơ, giảng viên có thể đánh giá chính xác tình hình học tập của sinh viên và đề xuất các phương án hỗ trợ thích hợp, giúp sinh viên vượt qua khó khăn và phát triển tốt hơn. Bên cạnh đó, hồ sơ cũng cung cấp thông tin minh bạch về tình trạng học tập, quyền lợi cũng như các nghĩa vụ của sinh viên, từ đó giúp sinh viên có cái nhìn rõ ràng và chính xác về quá trình học tập của mình.

Đối với giảng viên: Hồ sơ Cố vấn học tập là một công cụ không thể thiếu giúp họ thực hiện tốt nhiệm vụ cố vấn. Hồ sơ giúp giảng viên theo dõi, đánh giá và hỗ trợ sinh viên một cách có hệ thống và khoa học, từ đó đảm bảo sự phát triển toàn diện của sinh viên. Ngoài ra, hồ sơ còn hỗ trợ giảng viên trong việc báo cáo kết quả công việc cố vấn một cách rõ ràng và minh bạch, tạo điều kiện thuận lợi cho việc thực hiện các trách nhiệm và nghĩa vụ của giảng viên đối với nhà trường và sinh viên.

Đối với nhà trường: Hồ sơ Cố vấn học tập đóng vai trò quan trọng trong việc quản lý chặt chẽ quá trình học tập và rèn luyện của sinh viên. Các tài liệu trong hồ sơ cung cấp thông tin đầy đủ và chính xác về tình trạng học tập của sinh viên, giúp nhà trường thực hiện các công tác quản lý hiệu quả hơn. Hồ sơ cũng tạo cơ sở dữ liệu vững chắc để hỗ trợ công tác thống kê, báo cáo và ra quyết định, từ đó giúp nhà trường đưa ra các chính sách và chiến lược phát triển phù hợp.

Hồ sơ Cố vấn học tập không chỉ là một công cụ quản lý hiệu quả mà còn là cầu nối giữa sinh viên, giảng viên và nhà trường, giúp đảm bảo một môi trường học tập minh bạch, công bằng và hỗ trợ tối đa cho sự phát triển của sinh viên.

CHƯƠNG 3: HIỆN THỰC HOÁ NGHIÊN CỨU

3.1 Mô tả bài toán

- Hệ thống "Quản lý hồ sơ cố vấn học tập cho giảng viên Khoa Kỹ thuật và Công nghệ" được thiết kế nhằm hỗ trợ việc quản lý và truy cập hồ sơ cố vấn học tập một cách hiệu quả, đồng thời đảm bảo quyền hạn truy cập được phân quyền rõ ràng theo từng đối tượng.
- Sinh viên có thể tra cứu thông tin liên hệ của cố vấn học tập như tên, số điện thoại, email, và địa chỉ văn phòng mà không cần đăng nhập vào hệ thống. Sinh viên không có quyền đăng nhập hoặc truy cập các chức năng quản lý khác.
- Giảng viên cố vấn học tập có quyền đăng nhập vào hệ thống để thực hiện các chức năng quản lý hồ sơ, bao gồm: tạo, xóa (tạm thời hoặc vĩnh viễn), và khôi phục thư mục; tải lên, xóa (tạm thời hoặc vĩnh viễn), tải xuống, và khôi phục tệp tin. Giảng viên cũng có thể thay đổi thông tin liên hệ của mình.
- Nhân viên văn phòng khoa có quyền truy cập vào kho lưu trữ của tất cả cố vấn học tập để hỗ trợ công tác quản lý hành chính, nhưng chỉ được phép tải xuống các tệp tin, không có quyền chỉnh sửa, thêm mới hoặc xóa dữ liệu.
- Hệ thống đảm bảo bảo mật dữ liệu, cung cấp cơ chế khôi phục dữ liệu đã xóa và tối ưu hóa giao diện để phù hợp với từng đối tượng người dùng.

3.2 Phân tích yêu cầu

3.1.1 Các yêu cầu về chức năng.

3.1.1.1 Quản lý người dùng

Giảng viên: Giảng viên cần đăng nhập vào hệ thống và truy cập vào các tài liệu liên quan đến lớp học mà mình đang phụ trách. Có thể tạo thư mục lưu trữ và tải lên các tài liệu (báo cáo, biên bản,...).

Nhân viên văn phòng khoa: Nhân viên văn phòng khoa sẽ đăng nhập vào hệ thống để theo dõi và xem các tài liệu mà giảng viên đã tải lên. Tuy nhiên nhân viên văn phòng khoa không có quyền tải lên tài liệu hay xoá các tài liệu của giảng viên nhưng có thể xem các kho lưu trữ của tất cả giảng viên. Để xem của tất cả cần có các chức năng như tìm kiếm, hiển thị danh sách tất cả giảng viên.

3.1.1.2Quản lý hồ sơ cố vấn học tập

- **Lưu trữ tài liệu**: Giảng viên cần có khả năng tải lên các tài liệu liên quan đến cố vấn học tập như báo cáo, biên bản họp, lịch trình cố vấn, biên bản học bổng của sinh viên, lý lịch sinh viên,..
- **Xem tài liệu lưu trữ**: Nhân viên văn phòng khoa có thể truy cập và xem tất cả các tài liệu mà giảng viên đã tải lên, chẳng như báo cáo, biên bản họp, lịch trình cố vấn, biên bản học bổng của sinh viên, lý lịch sinh viên.
- **Cập nhật tài liệu**: Giảng viên có thể xóa các tài liệu đã tải lên, đảm bảo rằng các tài liệu trong hệ thống luôn được cập nhật và chính xác.

3.1.1.3Xác thực và phân quyền người dùng

- **Đăng nhập và xác thực**: Hệ thống sẽ yêu cầu người dùng đăng nhập bằng tài khoản google để sử dụng các tính năng của ứng dụng. Việc sử dụng tài khoản google sẽ tăng tính bảo mật, giảm nhẹ trách nhiệm về quản lý bảo mật tài khoản. Người dùng được lưu vào Firebase thêm các trường thông tin như Bộ môn, số điện thoại ,..
- **Phân quyền**: Phân quyền sẽ được thiết lập rõ ràng giữa giảng viên và nhân viên văn phòng khoa. Cố vấn có quyền quản lý và tải lên tài liệu, trong khi nhân viên văn phòng khoa có quyền kiểm tra tài liệu của tất cả giảng viên trong hệ thống.

3.1.1.3 Giao diện người dùng

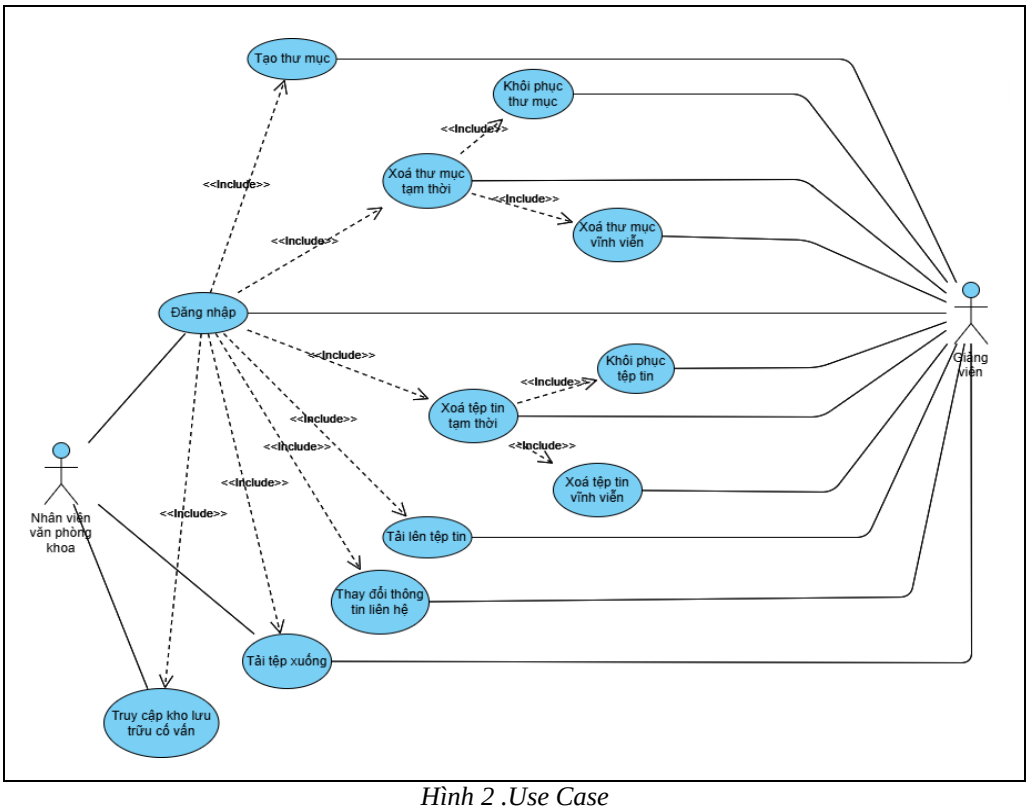
Giao diện thân thiện: Giao diện của hệ thống phải dễ sử dụng và thân thiện với người dùng, đặc biệt đối với giảng viên và nhân viên văn phòng khoa. Các thao tác phải đơn giản, dễ hiểu, và rõ ràng.

3.1.2 Các yêu cầu phi chức năng.

- **Tốc độ tải trang**: Hệ thống cần đảm bảo rằng việc tải trang và tài liệu diễn ra nhanh chóng, ngay cả khi lượng dữ liệu lớn.
- **Khả năng mở rộng**: Hệ thống cần có khả năng mở rộng để phục vụ nhiều người dùng đồng thời mà không làm giảm hiệu suất.
- **Bảo vệ dữ liệu người dùng**: Hệ thống cần bảo vệ dữ liệu cá nhân của người dùng, bao gồm thông tin cá nhân và các tài liệu lưu trữ. Firebase cung cấp các tính năng bảo mật mạnh mẽ để bảo vệ dữ liệu.

- **Mã hóa tài liệu**: Các tài liệu được tải lên cần được mã hóa khi lưu trữ trong Firebase Storage, đảm bảo an toàn cho dữ liệu của người dùng.
- **Quản lý và bảo trì hệ thống**: Hệ thống cần dễ dàng bảo trì và cập nhật. Các bản vá lỗi và tính năng mới có thể được triển khai nhanh chóng mà không làm gián đoạn hoạt động của hệ thống.
- **Tương thích với các trình duyệt web**: Website phải tương thích với các trình duyệt phổ biến như Google Chrome, Mozilla Firefox, Safari, và Microsoft Edge.
- **Tương thích với các thiết bị di động**: Giao diện của hệ thống phải được tối ưu hóa cho các thiết bị di động để giảng viên có thể truy cập dễ dàng từ điện thoại hoặc máy tính bảng.

3.1.3 Sơ đồ Use case trực quan hoá yêu cầu



Hình 2 .Use Case

Mô tả Use Cases

1. Đăng nhập (Giảng viên/ Nhân viên văn phòng khoa)

- Mục tiêu: Cung cấp quyền truy cập vào các chức năng của hệ thống.
- Actor: Giảng viên, Nhân viên văn phòng khoa.
- Tiền điều kiện: Người dùng phải có tài khoản hợp lệ.
- Dòng sự kiện chính:
  + Người dùng nhập thông tin đăng nhập (email và mật khẩu).
  + Hệ thống kiểm tra thông tin đăng nhập.
  + Nếu thông tin hợp lệ, hệ thống cấp quyền truy cập.
- Dòng sự kiện thay thế: Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.
- Hậu điều kiện: Người dùng được đăng nhập thành công hoặc nhận thông báo lỗi.

2. Tạo thư mục (Giảng viên)

- Mục tiêu: Cho phép giảng viên tạo thư mục để tổ chức hồ sơ.
- Actor: Giảng viên.
- Tiền điều kiện: Giảng viên đã đăng nhập thành công.
- Dòng sự kiện chính:
  + Giảng viên chọn chức năng "Tạo thư mục".
  + Hệ thống yêu cầu giảng viên nhập tên thư mục.
  + Giảng viên nhập tên thư mục và xác nhận.
  + Hệ thống tạo thư mục mới trong cơ sở dữ liệu.
- Dòng sự kiện thay thế:
  + Nếu tên thư mục trùng lặp, hệ thống yêu cầu nhập tên khác.
  + Hậu điều kiện: Thư mục được tạo thành công và hiển thị trong danh sách thư mục của giảng viên.

3. Xóa thư mục tạm thời (Giảng viên)

- Mục tiêu: Cho phép giảng viên xóa tạm thời thư mục không cần thiết.
- Actor: Giảng viên.
- Tiền điều kiện: Thư mục tồn tại và giảng viên đã đăng nhập.
- Dòng sự kiện chính:
  + Giảng viên chọn “Xóa tạm thời” thư mục.

+ Hệ thống di chuyển thư mục vào “Thùng rác”.

- Dòng sự kiện thay thế: Nếu thư mục không tồn tại, hiển thị lỗi.
- Hậu điều kiện: Thư mục bị xóa và chuyển vào thùng rác

4. Xóa thư mục vĩnh viễn (Giảng viên)

- Mục tiêu: Cho phép giảng viên xóa hoàn toàn thư mục.
- Actor: Giảng viên.
- Tiền điều kiện: Thư mục nằm trong "Thùng rác".
- Dòng sự kiện chính:
  + Giảng viên chọn "Xóa vĩnh viễn".
  + Hệ thống xác nhận hành động.
  + Thư mục bị xóa hoàn toàn khỏi hệ thống.
- Hậu điều kiện: Thư mục bị xóa vĩnh viễn.

5. Khôi phục thư mục (Giảng viên)

- Mục tiêu: Cho phép giảng viên khôi phục thư mục đã bị xóa tạm thời.
- Actor: Giảng viên.
- Tiền điều kiện: Thư mục phải tồn tại trong "Thùng rác".
- Dòng sự kiện chính:
  + Giảng viên chọn "Khôi phục thư mục".
  + Hệ thống chuyển thư mục từ "Thùng rác" về vị trí ban đầu.
- Hậu điều kiện: Thư mục được khôi phục.

6. Tải lên tệp tin (Giảng viên)

- Mục tiêu: Cho phép giảng viên tải tệp tin lên hệ thống.
- Actor: Giảng viên.
- Tiền điều kiện: Giảng viên đã đăng nhập và chọn thư mục lưu trữ.
- Dòng sự kiện chính:
  + Giảng viên chọn "Tải lên tệp tin".
  + Hệ thống hiển thị giao diện chọn tệp tin.
  + Giảng viên chọn tệp tin và xác nhận.
  + Tệp tin được lưu trong thư mục đã chọn.
- Dòng sự kiện thay thế: Nếu tệp tin vượt dung lượng, hiển thị lỗi.
- Hậu điều kiện: Tệp tin được tải lên.

7. Xóa tệp tin tạm thời (Giảng viên)

- Mục tiêu: Cho phép giảng viên xóa tạm thời tệp tin.
- Actor: Giảng viên.
- Tiền điều kiện: Tệp tin tồn tại trong hệ thống.
- Dòng sự kiện chính**:**
  + Giảng viên chọn "Xóa tạm thời".
  + Tệp tin được di chuyển vào "Thùng rác".
- Hậu điều kiện: Tệp tin nằm trong "Thùng rác".

8. Xóa tệp tin vĩnh viễn (Giảng viên)

- Mục tiêu: Cho phép giảng viên xóa hoàn toàn tệp tin.
- Actor: Giảng viên.
- Tiền điều kiện: Tệp tin nằm trong "Thùng rác".
- Dòng sự kiện chính:
  + Giảng viên chọn "Xóa vĩnh viễn".
  + Hệ thống xác nhận hành động.
  + Tệp tin bị xóa khỏi hệ thống.
- Hậu điều kiện: Tệp tin bị xóa vĩnh viễn.

9. Khôi phục tệp tin (Giảng viên)

- Mục tiêu: Cho phép giảng viên khôi phục tệp tin đã xóa tạm thời.
- Actor: Giảng viên.
- Tiền điều kiện: Tệp tin tồn tại trong "Thùng rác".
- Dòng sự kiện chính:
  + Giảng viên chọn "Khôi phục tệp tin".
  + Tệp tin được chuyển về vị trí ban đầu.
- Hậu điều kiện: Tệp tin được khôi phục.

10. Thay đổi thông tin liên hệ (Giảng viên)

- Mục tiêu: Cho phép giảng viên cập nhật thông tin cá nhân.
- Actor: Giảng viên.
- Tiền điều kiện: Giảng viên đã đăng nhập.
- Dòng sự kiện chính:
  + Giảng viên chọn "Thay đổi thông tin liên hệ".
  + Hệ thống hiển thị thông tin hiện tại.
  + Giảng viên chỉnh sửa và lưu thay đổi.

- Hậu điều kiện: Thông tin liên hệ được cập nhật.

11. Tải xuống tệp tin (Giảng viên/Nhân viên văn phòng khoa)

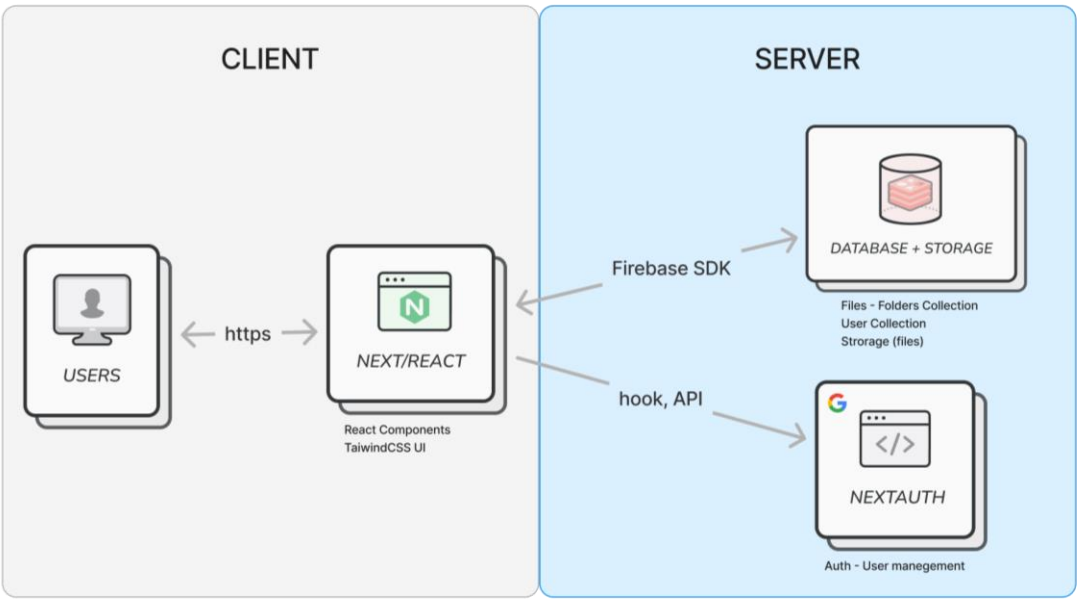
- Mục tiêu: Cho phép tải tệp tin về máy tính cá nhân.
- Actor: Giảng viên, Nhân viên văn phòng khoa.
- Tiền điều kiện: Tệp tin tồn tại và người dùng đã đăng nhập.
- Dòng sự kiện chính:
  + Người dùng chọn tệp tin.
  + Hệ thống tải tệp tin về máy.
- Hậu điều kiện: Tệp tin được tải xuống.

12. Truy cập kho lưu trữ của giảng viên (Nhân viên văn phòng khoa)

- Mục tiêu: Cho phép nhân viên văn phòng khoa truy cập và xem kho lưu trữ của tất cả giảng viên.
- Actor: Nhân viên văn phòng khoa.
- Tiền điều kiện: Nhân viên đã đăng nhập thành công.
- Dòng sự kiện chính:
  + Nhân viên chọn "Kho lưu trữ giảng viên".
  + Hệ thống hiển thị danh sách thư mục và tệp tin của từng giảng viên.
  + Nhân viên chọn tệp tin để tải xuống.
- Hậu điều kiện: Nhân viên tải xuống tệp tin thành công.

3.3 Thiết kế hệ thống

3.2.1 Kiến trúc hệ thống (client – server)



Hình 3. Kiến trúc hệ thống

Dòng dữ liệu và Tương tác giữa các thành phần:

- Frontend → API Routes → Firebase: Khi người dùng thực hiện một thao tác như tải lên một tệp tin mới, tạo thư mục, hoặc di chuyển tệp tin vào thùng rác:
  + Frontend gửi yêu cầu API tới backend (Next.js API Routes).
  + Backend tương tác với Firestore và Firebase Storage để thực hiện thao tác cần thiết (lưu tệp tin, thay đổi dữ liệu thư mục, phục hồi tệp tin, xóa tệp tin).
  + Sau khi thao tác hoàn tất, backend trả về phản hồi thành công hoặc lỗi, frontend sẽ cập nhật lại giao diện người dùng cho phù hợp.
  + Firebase Authentication → Frontend: Firebase Authentication sẽ cung cấp trạng thái đăng nhập người dùng qua NextAuth, frontend nhận và lưu trữ thông tin người dùng để thực hiện các thao tác với Firestore và Firebase Storage một cách an toàn.
- Firestore → Frontend: Firestore sẽ cung cấp dữ liệu động cho các component frontend. Ví dụ: danh sách tệp tin, danh sách thư mục sẽ được frontend lấy từ Firestore qua API, và sau đó hiển thị trong UI.
- Firebase Storage → Frontend: Khi người dùng cần tải xuống tệp tin hoặc hiển thị tệp tin (ví dụ: hình ảnh, tài liệu), frontend sẽ sử dụng URL được Firebase Storage cung cấp để tải xuống tệp từ storage và hiển thị cho người dùng.

Mô tả luồng dữ liệu cho một tình huống cụ thể:

- Đăng nhập người dùng:
  + Người dùng truy cập vào ứng dụng và chọn đăng nhập bằng Google thông qua NextAuth.
  + NextAuth sử dụng Firebase Authentication để xác thực người dùng.
  + Sau khi xác thực thành công, Firebase cung cấp thông tin người dùng
  + Frontend lưu trữ thông tin này và hiển thị giao diện người dùng.
- Tải lên tệp tin:
  + Người dùng chọn tệp tin để tải lên.
  + Frontend gửi yêu cầu API tới Next.js backend.
  + Backend lưu tệp tin vào Firebase Storage và lưu thông tin tệp vào Firestore (bao gồm URL, tên, kích thước,…).
  + Frontend nhận dữ liệu trả về và hiển thị tệp tin mới được tải lên trong UI.

- Xóa tệp tin:
  + Người dùng chọn tệp tin để xóa.
  + Frontend gửi yêu cầu API tới backend để thay đổi trường delete của tệp tin trong Firestore thành true.
  + Backend cập nhật Firestore, và frontend cập nhật lại giao diện người dùng, di chuyển tệp tin vào thùng rác.
- Phục hồi tệp tin từ thùng rác:
  + Người dùng chọn tệp tin trong thùng rác để phục hồi.
  + Frontend gửi yêu cầu API tới backend để thay đổi trường delete của tệp tin thành false.
  + Backend cập nhật Firestore, và frontend cập nhật lại giao diện để phục hồi tệp tin.

3.2.2 Lược đồ cơ sở dữ liệu

Mô hình cơ sở dữ liệu của hệ thống được thiết kế để phù hợp với yêu cầu của dự án, với hai collection chính trong Firebase Firestore:

Users: Lưu trữ thông tin người dùng (Giảng viên) với các trường sau:

STT	Tên trường	Kiểu	Chức năng
1	name	string	Tên
2	email	string	Email
3	emailcontact	string	Email liên hệ
4	image	string	Ảnh
5	phone	string	Số điện thoại
6	faculty	string	Khoa
7	department	string	Bộ môn
8	role	number	Quyền
9	status	string	Trạng thái
10	updatedAt	datetime	Thời gian sửa đổi
11	createdAt	datetime	Thời gian tạo

Bảng 1 . Bảng dữ liệu User

Thư mục: Lưu trữ thông tin thư mục mà giảng viên tạo với các trường sau:

STT	Tên trường	Kiểu	Chức năng
1	id	number	Mã thư mục
2	name	string	Tên thư mục
3	parentFolderId	string	Mã thứ tự thư mục
4	createBy	number	Thời gian tạo
5	delete	boolean	Trạng thái xoá

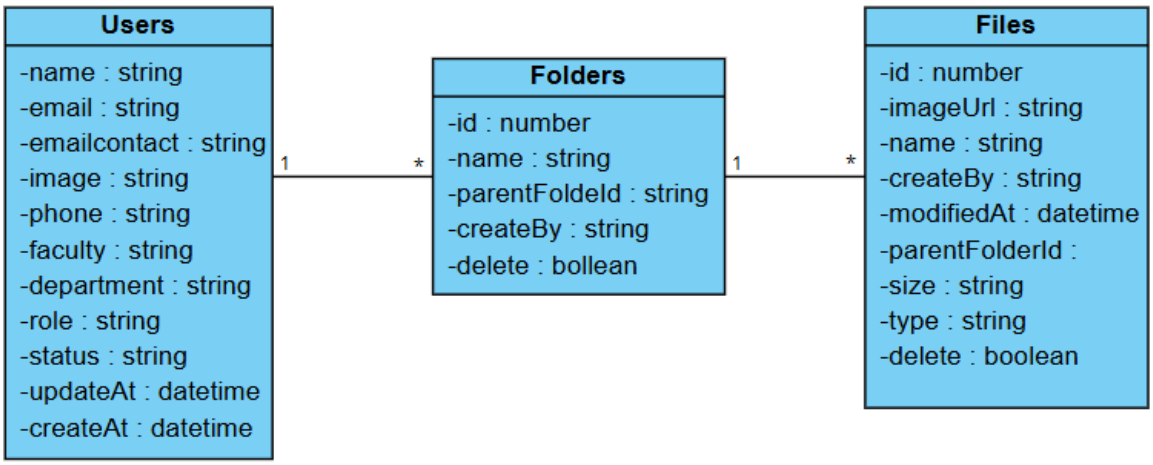
Bảng 2. Bảng dữ liệu Thư mục

Tài liệu: Lưu trữ thông tin các tài liệu báo cáo, biên bản mà giảng viên tải lên, với các trường sau:

STT	Tên trường	Kiểu	Chức năng
1	id	number	Mã tài liệu
2	imageUrl	string	Địa chỉ lưu
3	name	string	Tên tài liệu
4	createdBy	string	Email người tạo
5	modifiedAt	string	Thời gian sửa
6	parentFolderId	string	Mã thư mục cha
7	size	number	Cỡ tài liệu
8	type	string	Kiểu tài liệu
9	delete	boolean	Trang thái xoá

Bảng 3. Bảng dữ liệu Tài liệu

Sơ đồ trực quan hoá cơ sở dữ liệu (Class Diagram)



Hình 4. Sơ đồ lớp trực quan hoá dữ liệu (Class Diagram).

3.4 Cài đặt và triển khai

3.3.1 Cài đặt dự án (Next.js)

Khởi tạo dự án

Dự án được khởi tạo bằng lệnh:

```
npx create-next-app quanlyhoso
```

Các tệp và thư mục ban đầu được tổ chức theo cấu trúc chuẩn của Next.js, bao gồm:

- pages/: Chứa các trang chính.
- public/: Chứa các tài nguyên như hình ảnh, font, v.v.
- styles/: Các tệp CSS mặc định.

Cài đặt các gói phụ thuộc

```
npm install next react react-dom
```

- **next**: Framework hỗ trợ SSR (Server-Side Rendering) và static site generation.
- **react**: Thư viện xây dựng giao diện người dùng.
- **react-dom**: Cung cấp các phương thức để render React components vào DOM.

Cài đặt các gói hỗ trợ thiết kế giao diện

```
npm install tailwindcss postcss autoprefixer
npm install daisyui
npx tailwindcss init
```

- **tailwindcss**: Framework CSS utility-first.
- **postcss** và **autoprefixer**: Công cụ xử lý CSS tự động thêm vendor prefixes.
- **daisyui**: Thư viện các thành phần giao diện tích hợp với TailwindCSS.

Cài đặt Firebase SDK

```
npm install firebase
```

firebase: SDK kết nối với Firebase Firestore, Storage, và Authentication.

Cài đặt các gói quản lý thời gian và phiên đăng nhập.

```
npm install moment next-auth
```

- **moment**: Hỗ trợ xử lý và định dạng thời gian.
- **next-auth**: Hỗ trợ xác thực và phân quyền người dùng trong ứng dụng Next.js.

Cài đặt các gói phát triển.

```
npm install eslint eslint-config-next
```

- **eslint**: Công cụ phân tích và kiểm tra lỗi code.
- **eslint-config-next**: Cấu hình ESLint mặc định cho dự án Next.js.

Sau khi hoàn tất cài đặt các gói được quản lý trong tệp /package.json.

```
{
  "name": "cloud-file-manager",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "buffer": "^6.0.3",
    "firebase": "^10.14.1",
    "moment": "^2.29.4",
    "next": "^13.4.10",
    "next-auth": "^4.24.10",
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "autoprefixer": "^10.4.20",
    "daisyui": "^3.2.1",
    "eslint": "8.45.0",
    "eslint-config-next": "13.4.10",
    "postcss": "^8.4.49",
    "tailwindcss": "^3.4.16"
  }
}
```

Cấu hình tệp .env

```
GOOGLE_CLIENT_ID=39135302522-
kqvvusnu0acj4rtbd9bjirplj9if6u88.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
FIREBASE_API_KEY=AIzaSyD7iIUBlMxTWYchhCPrvPmGjoTMyZ-zCB0
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=your-random-secret
```

- **GOOGLE_CLIENT_ID**: ID ứng dụng Google, dùng để định danh khi tích hợp OAuth2 (Đăng nhập bằng Google).
- **GOOGLE_CLIENT_SECRET**: Mật khẩu bí mật cho ứng dụng Google, dùng để bảo mật các luồng OAuth2.
- **FIREBASE_API_KEY**: Khóa API để kết nối Firebase.
- **NEXTAUTH_URL**: URL gốc của ứng dụng, dùng trong NextAuth để điều hướng xác thực.
- **NEXTAUTH_SECRET**: Chuỗi bí mật để mã hóa token (JWT) trong NextAuth, cần bảo mật tuyệt đối.

3.3.2 Xây dựng các thành phần chức năng (Components)

```
components
│   ListUsers.js
│   SideNavBar.js
│   Toast.js
├───File
│       FileItem.js
│       FileList.js
│       UploadFileModal.js
├───Folder
│       CreateFolderModal.js
│       FolderItem.js
│       FolderItemSmall.js
│       FolderList.js
└───Storage
        Storage.js
        StorageInfo.js
        UserInfo.js
```

Các component chính được thiết kế trong thư mục components:

Quản lý tệp (Files):

- FileItem.js: hiển thị thông tin của từng file, bao gồm tên, kiểu, kích thước file, và các chức năng như :
  + Hiển thị thông tin file: Component này hiển thị tên của file, kiểu file và kích thước file.
  + Tải xuống: Khi người dùng nhấn vào icon tải xuống, hệ thống sẽ mở file lên (sử dụng window.open).
  + Hiển thị các tùy chọn: Khi người dùng nhấn vào biểu tượng "thêm" (các ba dấu chấm), một menu tùy chọn sẽ hiện lên cho phép người dùng chọn các hành động như tải xuống, xóa hoặc khôi phục file.
  + Xóa tạm thời: Nếu file được đánh dấu là đã xóa (delete: true), các tùy chọn sẽ hiển thị để khôi phục hoặc xóa vĩnh viễn file.
  + Khôi phục file: Nếu file bị xóa, người dùng có thể khôi phục lại file này.
  + Xóa vĩnh viễn: Nếu file đã bị xóa và người dùng chọn xóa vĩnh viễn, file sẽ bị xóa hoàn toàn khỏi hệ thống.

```
const handleDownload = () => {
    window.open(file.imageUrl, "_blank");
    setShowOptions(false);
  };

  const handleDelete = async () => {
    try {
      const fileRef = doc(db, "files", file.id.toString());
      await updateDoc(fileRef, { delete: true });
      setShowOptions(false);
    } catch (error) {
      console.error("Lỗi khi xóa file:", error);
    }
  };

  const handleRestore = async () => {
    try {
      const fileRef = doc(db, "files", file.id.toString());
      await updateDoc(fileRef, { delete: false });
      setShowOptions(false);
    } catch (error) {
      console.error("Lỗi khi khôi phục file:", error);
    }
  };
```

- FileList.js: Component này hiển thị danh sách các file, cho phép người dùng tìm kiếm và lọc các file dựa trên tên.

+ Tìm kiếm: Người dùng có thể tìm kiếm các file theo tên bằng cách nhập vào ô tìm kiếm. Component lọc danh sách file dựa trên từ khóa tìm kiếm.
+ Hiển thị danh sách file: Danh sách các file sẽ được hiển thị dưới dạng grid với thông tin về tên, thời gian, kích thước file và các tùy chọn (tải xuống, xóa, v.v.).
+ Lọc file: Mỗi file được hiển thị bằng component FileItem với dữ liệu riêng.

```
const handleClearSearch = () => {
  setSearchTerm('');
};

const filteredFiles = fileList.filter((file) =>
  file.name.toLowerCase().includes(searchTerm.toLowerCase())
);
```

- UploadFileModal.js: cho phép người dùng tải lên các tệp tin mới vào hệ thống. Modal này bao gồm một giao diện người dùng để chọn và tải lên các tệp.
  + Tải lên tệp: Người dùng có thể chọn tệp từ máy tính của họ và tải lên. File sẽ được tải lên Firebase Storage, và sau khi hoàn tất, một URL tải xuống sẽ được lưu trong Firestore.
  + Hạn chế kích thước file: Nếu file vượt quá kích thước tối đa (1MB trong trường hợp này), người dùng sẽ nhận được thông báo lỗi.
  + Lưu thông tin file: Sau khi tệp được tải lên thành công, thông tin file (tên, loại, kích thước, thời gian tạo, người tạo, v.v.) sẽ được lưu vào Firestore trong collection "files".

```
const onFileUpload = async (file) => {
   if (file) {
     if (file?.size > 10000) {
       setShowToastMsg("File is too large")
       return;
     }
     const fileRef = ref(storage, "file/" + file.name);
     uploadBytes(fileRef, file)
       .then(() => {
         return getDownloadURL(fileRef);
       })
       .then(async (downloadURL) => {
         console.log("File available at", downloadURL);
         await setDoc(doc(db, "files", docId.toString()), {
           name: file.name,
           type: file.name.split(".")[1],
           size: file.size,
           modifiedAt: file.lastModified,
           createdBy: session.user.email,
           parentFolderId: parentFolderId,
           imageUrl: downloadURL,
           delete: false,
           id: docId,
         });
         closeModal(true);
         setShowToastMsg("Tải lên thành công!");
       })
       .catch((error) => {
         console.error("Error uploading file:", error);
       });
   }
};
```

- Các xử lý tổng thể :
  + Tải lên file: Người dùng chọn file để tải lên thông qua UploadFileModal. File được tải lên Firebase Storage và URL của file được lưu vào Firestore.
  + Hiển thị file: Sau khi file được tải lên, FileList hiển thị danh sách các file, và mỗi file được hiển thị dưới dạng FileItem với các tùy chọn như tải xuống, xóa, v.v.
  + Tìm kiếm và lọc: Người dùng có thể tìm kiếm các file dựa trên tên.
  + Quản lý file: Người dùng có thể xóa, khôi phục hoặc xóa vĩnh viễn các file đã bị đánh dấu xóa.

Quản lý thư mục

- CreateFolderModal.js: Với chức năng cho phép người dùng tạo thư mục mới.

```
const onCreate=async()=>{
        console.log(folderName)
        await setDoc(doc(db,"Folders",docId),{
            name:folderName,
            id:docId,
            createBy:session.user.email,
            parentFolderId:parentFolderId,
            delete: false,
            createAt: Timestamp.now()
        })
        setShowToastMsg('Folder Created!')
    }
```

- FoldeItem.js và FolderItemSmall.js: Hiển thị thông tin của một thư mục trong giao diện người dùng. Thư mục trong dạng grid với FolderItem.js còn FolderItemSmall.js hiển thị thư mục trong dạng nhỏ gọn hơn dang list.

```
<div className=' flex gap-3 hover:bg-gray-100 p-2 rounded-md
   cursor-pointer'>
    <Image src='/folder.png' alt='folder' width={20} height={20}/>
    <h1>{folder.name}</h1>
</div> //FolderItemSmall

<div className={`w-[120px] flex flex-col justify-center items-center h-full
   border-[1px] rounded-lg p-4 bg-white hover:scale-105 hover:shadow-md
   cursor-pointer `}>
       <Image src='/folder.png' alt='folder' width={50} height={50}/>
       <h2 className='line-clamp-2 text-[12px]
           text-center'>{folder.name}</h2>
</div> //FolderItem
```

- FolderList.js: Component này nhận vào một danh sách các thư mục và hiển thị chúng. Tùy thuộc vào thuộc tính isBig, nó sẽ quyết định hiển thị thư mục theo dạng lớn (grid) hoặc dạng nhỏ (danh sách).
- Các xử lý tổng thể :
  + Tạo thư mục: Người dùng có thể tạo thư mục mới thông qua modal CreateFolderModal. Khi người dùng nhập tên và nhấn "Create", thư mục sẽ được lưu vào Firestore.

+ Hiển thị thư mục: Sau khi thư mục được tạo, danh sách thư mục sẽ được hiển thị trong component FolderList. Mỗi thư mục có thể được hiển thị dưới dạng lớn (với FolderItem) hoặc nhỏ (với FolderItemSmall).
+ Điều hướng thư mục: Khi người dùng nhấn vào một thư mục, ứng dụng sẽ chuyển đến trang con của thư mục đó để hiển thị các tệp bên trong.

Quản lý lưu trữ (Storage):

- StorageInfo.js: Component này hiển thị thông tin về dung lượng lưu trữ đã sử dụng của người dùng. Nó tính toán và hiển thị tổng dung lượng các tệp đã tải lên (theo đơn vị MB).

```
const getAllFiles = async () => {
    const q = query(collection(db, "files"),
      where("createdBy", "==", session.user.email));
    const querySnapshot = await getDocs(q);
    setFileList([])
    querySnapshot.forEach((doc) => {
      totalSize = totalSize + doc.data()['size'];
      setFileList(fileList => ([...fileList, doc.data()]))
    })
    setTotalSizeUsed((totalSize / 1024 ** 2).toFixed(2) + " MB");
  }
```

- UserInfo.js: Component này hiển thị thông tin người dùng hiện tại, bao gồm ảnh đại diện, tên, và email. Nó cũng cung cấp chức năng đăng xuất, biểu mẫu xác nhận đăng xuất cho người dùng.

```
const handleSignOut = () => {
        setShowConfirm(true); // Hiển thị hộp xác nhận
    };
    const confirmSignOut = () => {
        setShowConfirm(false); // Ẩn hộp xác nhận
        signOut(); // Thực hiện đăng xuất
    };
    const cancelSignOut = () => {
        setShowConfirm(false); // Ẩn hộp xác nhận
    };
```

- Storage.js: Hiển thị (Gôm chung) UserInfo và StorageInfo.
- Các xử lý tổng thể:
  + Hiển thị thông tin người dùng và dung lượng lưu trữ: Khi người dùng đã đăng nhập (có session), component Storage sẽ render UserInfo và StorageInfo.

+ Cập nhật dung lượng lưu trữ: Khi component StorageInfo được render, hệ thống sẽ tải danh sách các tệp mà người dùng đã tải lên Firestore. Dung lượng của các tệp này sẽ được tính toán và hiển thị dưới dạng MB.
+ Đăng xuất: Khi người dùng muốn đăng xuất, một hộp xác nhận sẽ được hiển thị, yêu cầu người dùng xác nhận hoặc hủy việc đăng xuất. Nếu người dùng xác nhận, họ sẽ bị đăng xuất khỏi hệ thống.

Các component khác:

- SideNavBar.js: Component này là một thanh điều hướng bên (sidebar), chứa các menu cho phép người dùng điều hướng đến các phần khác nhau của ứng dụng. Ngoài ra, nó còn có các nút để thêm tài liệu hoặc thư mục.Components có các xử lý sau:
  + Hiển thị menu: Các menu trong sidebar được lấy từ menu.list (dữ liệu menu). Mỗi mục menu có thể dẫn đến một trang hay một hành động cụ thể.
  + Điều hướng: Khi người dùng click vào một mục menu, hàm handleMenuClick sẽ sử dụng useRouter từ Next.js để điều hướng người dùng đến trang tương ứng với đường dẫn của menu. Nếu có hàm onMenuClick được truyền vào, nó cũng sẽ được gọi sau khi điều hướng.
  + Thêm tài liệu và thư mục: Có hai nút để mở các modal thêm tài liệu và thư mục. Khi người dùng nhấn vào nút, một modal sẽ xuất hiện, cho phép người dùng thực hiện các tác vụ này.
  + Modals: Hai modal được định nghĩa: CreateFolderModal và UploadFileModal, tương ứng với việc thêm thư mục và tải lên tài liệu.

```
const handleMenuClick = (item) => {
 router.push(item.path); // Điều hướng tới đường dẫn của item
 if (onMenuClick) {
   onMenuClick(item); // Gọi hàm onMenuClick nếu có
 }
};
```

- ListUser.js: Component này hiển thị danh sách người dùng (cố vấn) và cho phép người dùng tìm kiếm theo tên. Nó cũng cho phép chọn một người dùng trong danh sách để thực hiện các hành động khác. Với xử lý như sau:

+ Lọc người dùng: Dữ liệu người dùng (users) được lọc dựa trên từ khóa tìm kiếm (searchTerm). Các người dùng có vai trò là 0 sẽ bị loại bỏ khỏi danh sách, chỉ giữ lại những người có vai trò khác (có thể là cố vấn).
+ Tìm kiếm: Người dùng có thể nhập từ khóa tìm kiếm vào ô input. Khi giá trị của ô tìm kiếm thay đổi (onSearchChange), hàm lọc lại danh sách người dùng và hiển thị các kết quả phù hợp.
+ Chọn người dùng: Khi người dùng click vào một người dùng trong danh sách, hàm onUserClick sẽ được gọi để xử lý sự kiện chọn người dùng. Người dùng được chọn sẽ được làm nổi bật với màu nền khác.
+ Hiển thị người dùng: Danh sách người dùng được hiển thị dưới dạng một danh sách các phần tử <li>, mỗi phần tử chứa ảnh đại diện và tên của người dùng.

```
const filteredUsers = users.filter(user =>
  user.role !== 0 &&
  user.name?.toLowerCase().includes(searchTerm.toLowerCase())
);

<ul className="space-y-2"> {filteredUsers.map((user, index) => (
    <li key={index} className={`flex p-2 rounded-lg border-gray-300
        cursor-pointer h-12
        ${selectedUser?.email === user.email
          ? 'bg-blue-100 border border-blue-500 hover:scale-105'
          : 'bg-white border hover:bg-gray-200 hover:scale-105'}`}
      onClick={() => onUserClick(user)}>
      <div className="flex items-center space-x-3">
        <Image src={user.image || "/default-avatar.png"} alt={user.name}
          className="w-8 h-8 rounded-full" width={32} height={32}/>
        <span className="text-sm font-medium">{user.name}</span>
      </div>
    </li>
  ))}
</ul>
```

- Toast.js: Hiển thông báo ngắn.

3.3.2 Xây dựng các trang chính

Đồ án bao gồm các trang sau:

```
pages
│   admin.js
│   index.js
│   login.js
│   profile.js
│   trash.js
│   _app.js
├───api
│   │
│   └───auth
│           [...nextauth].js
├───auth
│       welcome.js
└───folder
        [folderId].js
```

Các trang đăng nhập:

- Login.js: Chức năng kiểm tra vai trò của người sau khi người dùng đăng nhập thành công. Tìm kiếm người dùng (hiển thị thông tin liên lạc của cố vấn), nút đăng nhập, chân trang.

```
useEffect(() => {
  const checkUserRole = async () => {
    if (session) {
      const userRef = doc(db, 'users', session.user.email);
      const docSnap = await getDoc(userRef);
      if (docSnap.exists()) {
        const userRole = docSnap.data().role;
        if (userRole === 0) {router.push('/admin');
           } else { router.push('/');}
       } else { console.error('User not found in Firestore');
      }
    }
  };
  checkUserRole();
}, [session, router, db]);
```

- […nextauth].js: Trang này cấu hình next-auth cho việc xác thực người dùng qua Google. Nó đảm nhận việc xử lý các callback và lưu trữ dữ liệu người dùng vào Firestore khi người dùng đăng nhập lần đầu.
  + Với xử lý như sau: Khi người dùng đăng nhập thành công qua Google, callback signIn được gọi. Hệ thống sẽ kiểm tra xem người dùng đã có trong Firestore chưa. Nếu chưa, dữ liệu người dùng sẽ được thêm vào Firestore với các thông tin như tên, email, và hình ảnh.

```
export default NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
    }),
  ],
  callbacks: {
    async signIn({ user }) {
      try {
        const userRef = doc(db, "users", user.email);
        const docSnap = await getDoc(userRef);

        if (!docSnap.exists()) {
          await setDoc(userRef, {
            name: user.name || "Unknown",
            email: user.email,
            emailcontact: user.email,
            image: user.image || null,
            phone: null,
            role: 1,
            createdAt: new Date().toISOString(),
            updatedAt: new Date().toISOString(),
            status: "active",
            faculty: null,
            department: null,
          });
        }
        return true;
      } catch (error) {
        console.error("Error saving user data to Firestore:", error);
        return false;
      }
    },
  },
  pages: {
    signIn: '/auth/welcome',},
});
```

Trang thư mục ([folderId].js):

- Hiển thị danh sách thư mục con (subfolders) và danh sách tệp (files) nằm trong thư mục cha. Cập nhật trạng thái theo thời gian thực nhờ **onSnapshot** trong Firestore. Quản lý giao diện tùy thuộc vào vai trò của người dùng . Cho phép người dùng thực hiện các hành động:
  + Xóa thư mục hiện tại (đánh dấu là "đã xóa" trong Firestore).
    + Khôi phục thư mục nếu đã bị xóa.
- Với xử lý như sau:
  + Khởi tạo dữ liệu: Lấy tham số từ URL và xác định vai trò người dùng.
  + Truy vấn dữ liệu: Lấy danh sách thư mục, tệp và trạng thái thư mục từ Firestore.
  + Xử lý tương tác: Xóa hoặc khôi phục thư mục dựa trên hành động người dùng.
  + Cập nhật giao diện: Hiển thị thông tin và tùy chọn tương ứng với trạng thái dữ liệu.

Trang chủ (index.js):

- Chức năng chính:
  + Quản lý thư mục và tệp: Hiển thị danh sách các thư mục gốc (parentFolderId = 0). Hiển thị danh sách tệp thuộc thư mục gốc.
  + Xử lý điều hướng: Nếu người dùng chưa đăng nhập, tự động chuyển hướng về trang đăng nhập. Cung cấp thanh điều hướng bên trái (SideNavBar) để người dùng chọn các chế độ xem hoặc tính năng khác (ví dụ: thùng rác, trang cá nhân).
  + Tích hợp Firebase: Lấy danh sách thư mục và tệp từ Firestore, chỉ hiển thị những mục thuộc về người dùng hiện tại và chưa bị xóa.
  + Cập nhật theo thời gian thực: Sử dụng onSnapshot để theo dõi các thay đổi trong danh sách tệp và cập nhật giao diện.
- Với xử lý như sau:
  + Khởi tạo và kiểm tra đăng nhập: Nếu người dùng chưa đăng nhập (!session), điều hướng về /login bằng router.push.

+ Thiết lập dữ liệu ban đầu và theo dõi thời gian thực: Gọi các hàm getFolderList và getFileList để truy vấn dữ liệu từ Firestore. Sử dụng onSnapshot để tự động cập nhật giao diện khi dữ liệu thay đổi.

```
const getFolderList = async () => {
    const q = query(
      collection(db, 'Folders'),
      where('parentFolderId', '==', 0),
      where('createBy', '==', session.user.email),
      where('delete', '==', false)
    );

    const querySnapshot = await getDocs(q);
    querySnapshot.forEach((doc) => {
      setFolderList((folderList) => [...folderList, doc.data()]);

     });
  };

  const getFileList = async () => {
    const q = query(
      collection(db, 'files'),
      where('parentFolderId', '==', 0),
      where('createdBy', '==', session.user.email),
      where('delete', '==', false)    );
    const unsubscribe = onSnapshot(q, (querySnapshot) => {
      const updatedFileList = [];
      querySnapshot.forEach((doc) => {
        updatedFileList.push({ id: doc.id, ...doc.data() });
      });
      setFileList(updatedFileList);    });

    return unsubscribe;
  };
```

+ Hiển thị giao diện: Thanh điều hướng bên trái (SideNavBar): Cho phép người dùng chọn chế độ xem (ví dụ: thư mục, thùng rác, hồ sơ cá nhân,..). Danh sách thư mục và tệp: Hiển thị các mục gốc của người dùng (thư mục và tệp trong thư mục gốc).
+ Xử lý hành vi người dùng: Khi người dùng chọn các mục trong SideNavBar, cập nhật chế độ hiển thị (currentView) để hiển thị nội dung phù hợp.

Trang dành cho Nhân viên văn phòng khoa (admin.js):

- Chức năng chính:
  + Quản lý người dùng:Hiển thị danh sách tất cả người dùng từ Firestore (bảng users). Cho phép tìm kiếm người dùng dựa trên từ khóa.
  + Hiển thị chi tiết người dùng: Khi chọn một người dùng, hiển thị danh sách thư mục và tệp tin do người dùng đó tạo.
  + Tích hợp Firebase: Lấy danh sách người dùng, thư mục, và tệp tin từ Firestore. Lọc dữ liệu theo email của người dùng và các điều kiện cụ thể (ví dụ: chưa bị xóa).
  + Lưu và tải lại trạng thái: Lưu người dùng được chọn trong localStorage để duy trì trạng thái khi tải lại trang.
- Với xử lý như sau :
  + Tải danh sách người dùng, thư mục và tài liệu: Khi người dùng đăng nhập, gọi fetchUsersAndFolders để truy vấn danh sách người dùng từ Firestore (bảng users).

```
 const fetchUsersAndFolders = async () => {
        const db = getFirestore(app);
        const usersCollection = collection(db, "users");
        const userSnapshot = await getDocs(usersCollection);
        const userList = userSnapshot.docs.map(doc => ({ id:
doc.id, ...doc.data() }));
        setUsers(userList);
        const folderData = {};
        for (const user of userList) {
          const folderQuery = query(
            collection(db, "Folders"),
            where("createBy", "==", user.email),
            where("parentFolderId", "==", 0),
            orderBy("createAt", "desc"),
            limit(1)
          );
          const folderSnapshot = await getDocs(folderQuery);
          folderData[user.email] = folderSnapshot.docs.map(
          doc => doc.data())[0] || null;
        }
        setFolders(folderData);
      };

      fetchUsersAndFolders();
    }

  const fetchUserFoldersAndFiles = async (userEmail) => {
    const db = getFirestore(app);
```

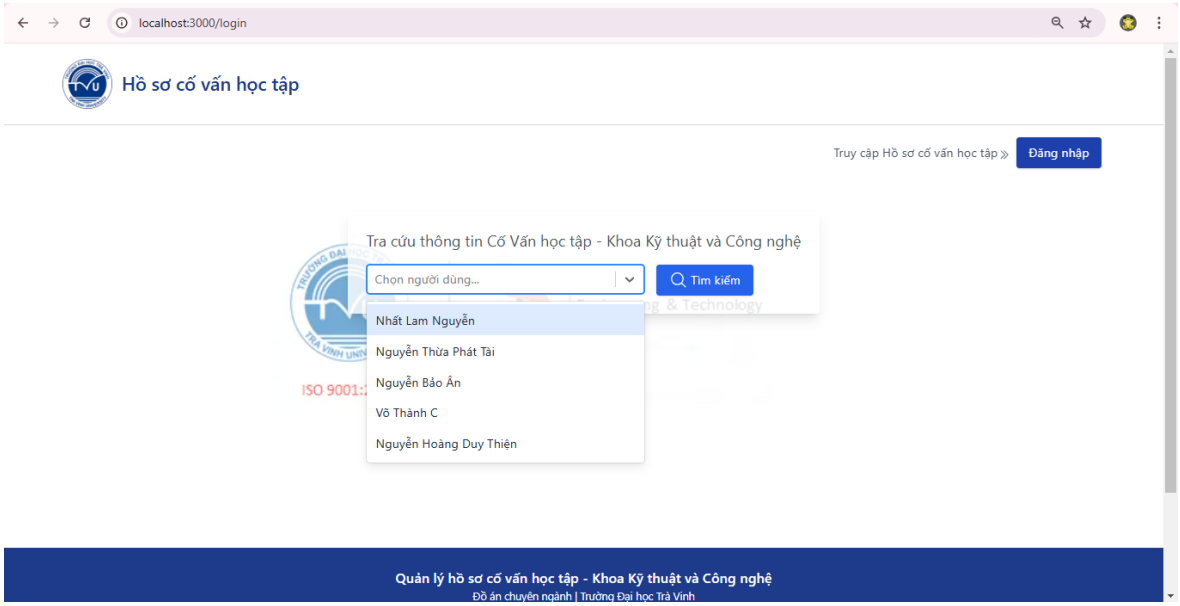
```
    const folderQuery = query(
      collection(db, "Folders"),
      where("createBy", "==", userEmail),
      where("parentFolderId", "==", 0),
      where('delete', '==', false)
    );
    const folderSnapshot = await getDocs(folderQuery);
    const fetchedFolders = folderSnapshot.docs.map(doc =>
doc.data());
    setFolderList(fetchedFolders);
    const fileQuery = query(
      collection(db, "files"),
      where("createdBy", "==", userEmail),
      where("parentFolderId", "==", 0),
      where('delete', '==', false));
    const fileSnapshot = await getDocs(fileQuery);
    const fetchedFiles = fileSnapshot.docs.map(doc =>
doc.data());
    setFileList(fetchedFiles);
  };
```

+ Xử lý chọn người dùng: Gọi fetchUserFoldersAndFiles để lấy danh sách thư mục và tệp gốc của người dùng đó. Lưu trạng thái người dùng đã chọn vào localStorage.
+ Khôi phục trạng thái: Khi tải lại trang, kiểm tra localStorage để khôi phục trạng thái của người dùng đã chọn.
+ Hiển thị giao diện: Danh sách người dùng: Hiển thị trong component ListUsers, bao gồm tên và thư mục gốc gần đây nhất.

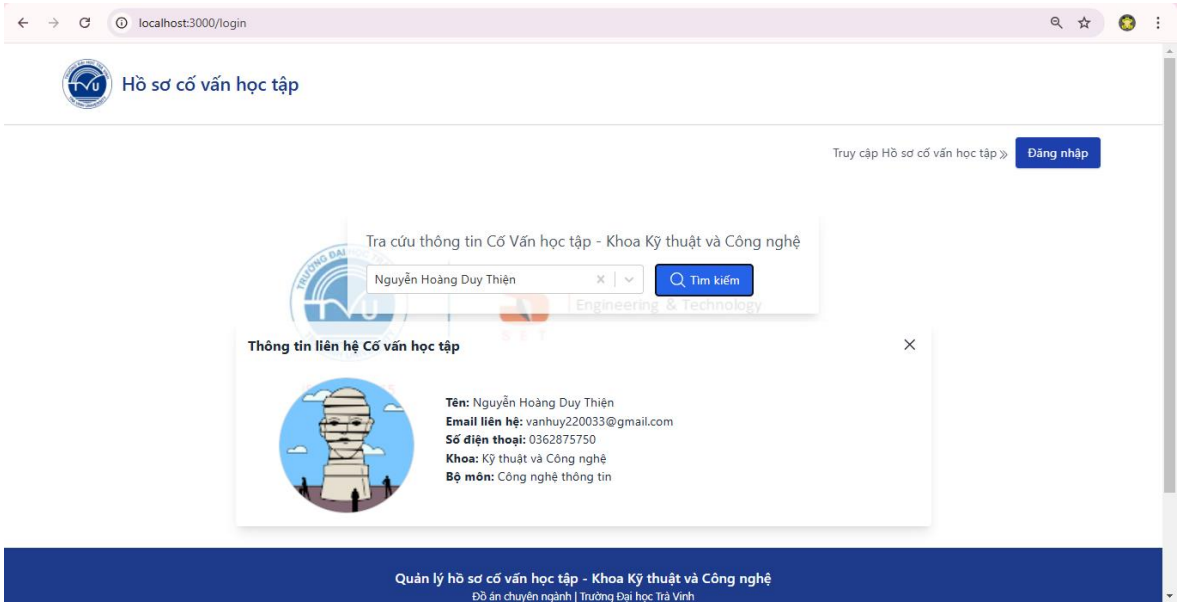
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Giao diện trang (login.js)

- Với chức năng tìm kiếm thông tin liên lạc của Cố vấn học tập
- Được thiết kết với tiêu chí đơn giản, sử dụng 2 màu xanh dương (màu nhận diện Đại học Trà Vinh) cho thành phần như nút bấm, tiêu đề trang web và chân trang và màu trắng làm nền chủ đạo.



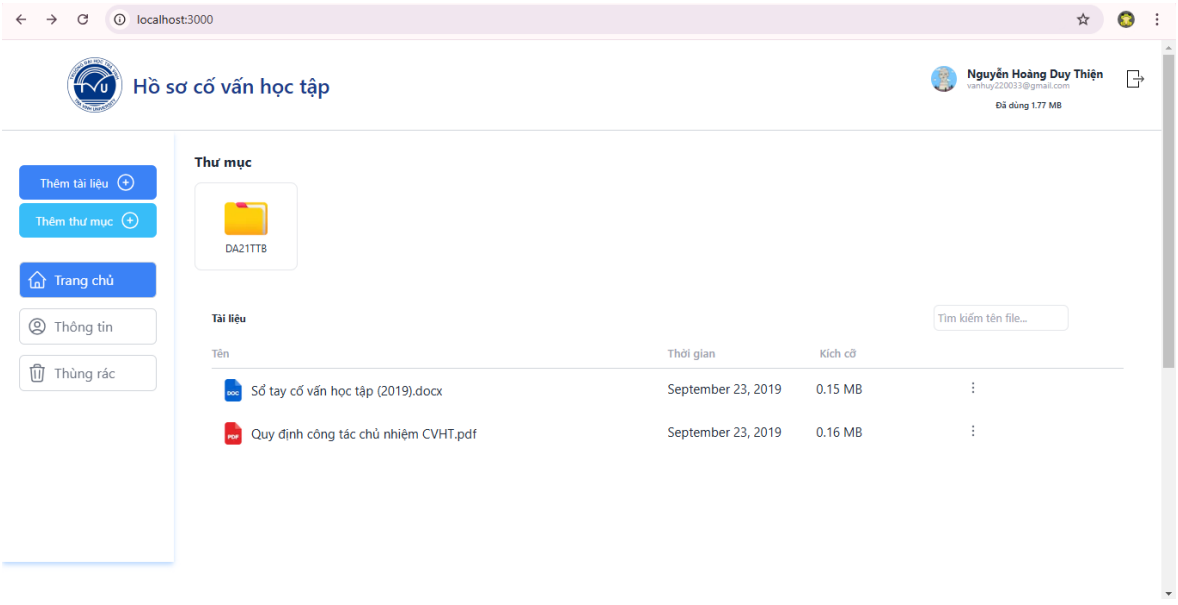
Hình 5. Trang login.js



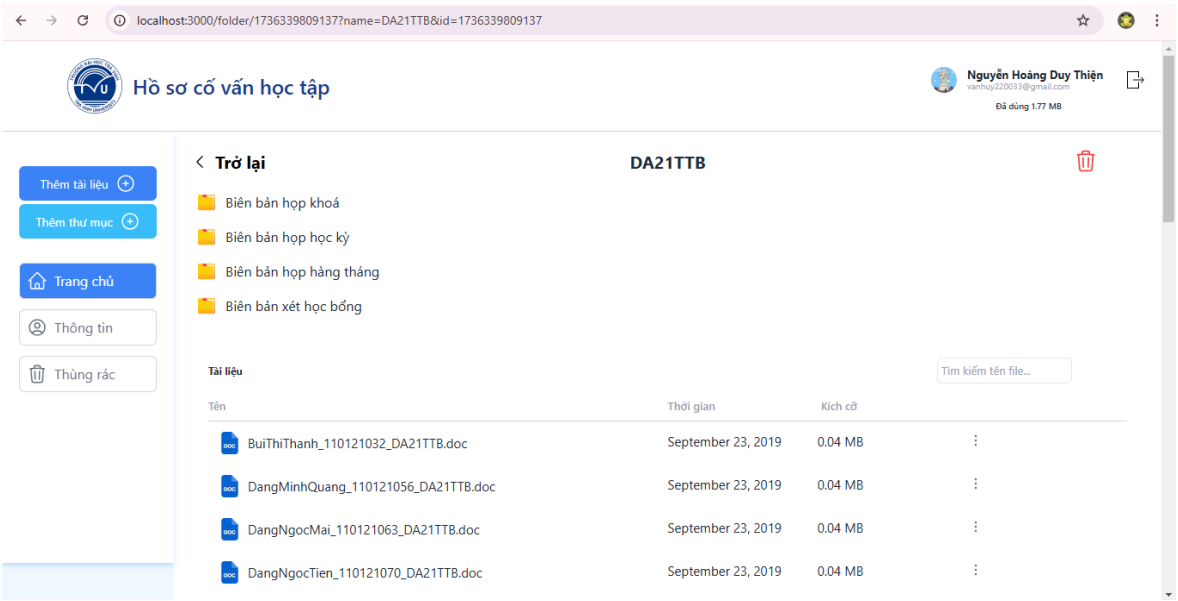
Hình 6. Trang login.js hiển thị kết quả tìm kiếm

4.2 Giao diện trang chủ (index.js)

- Sử dụng bố cục đơn giản với thanh điều hướng đơn bên trái và trang danh sách thư mục, tài liệu bên phải
- Chuyển dạng hiển thị thư mục sang danh sách ở trang thư mục.
- Ở trang này vẫn dùng màu xanh và trắng làm màu chủ đạo để giữ tính đồng bộ cho trang web.



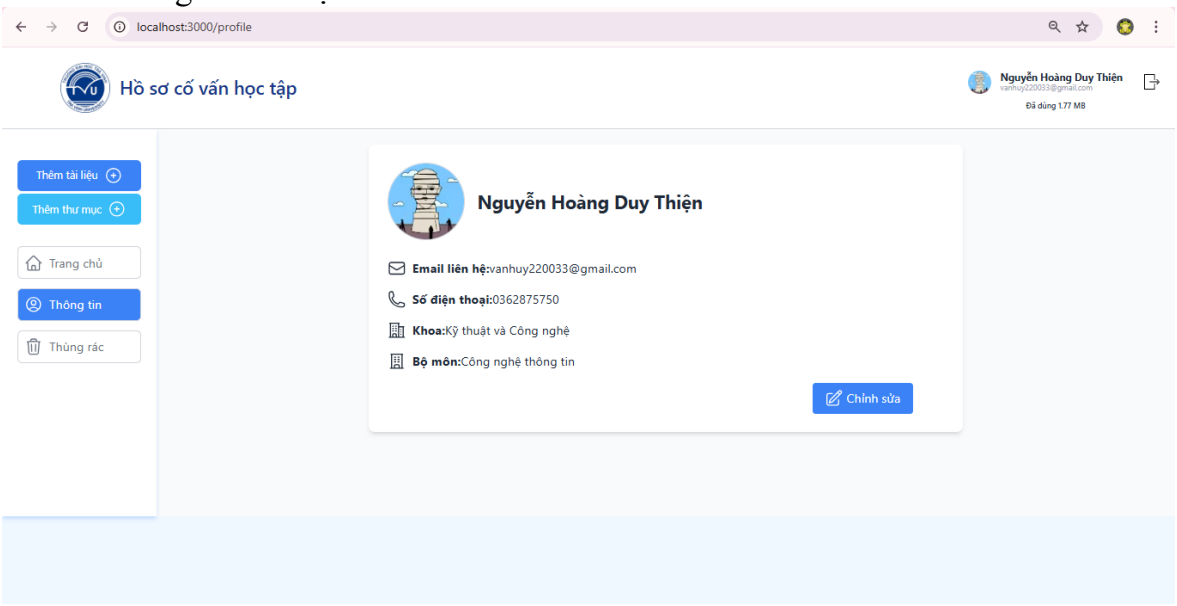
Hình 7. Trang index.js



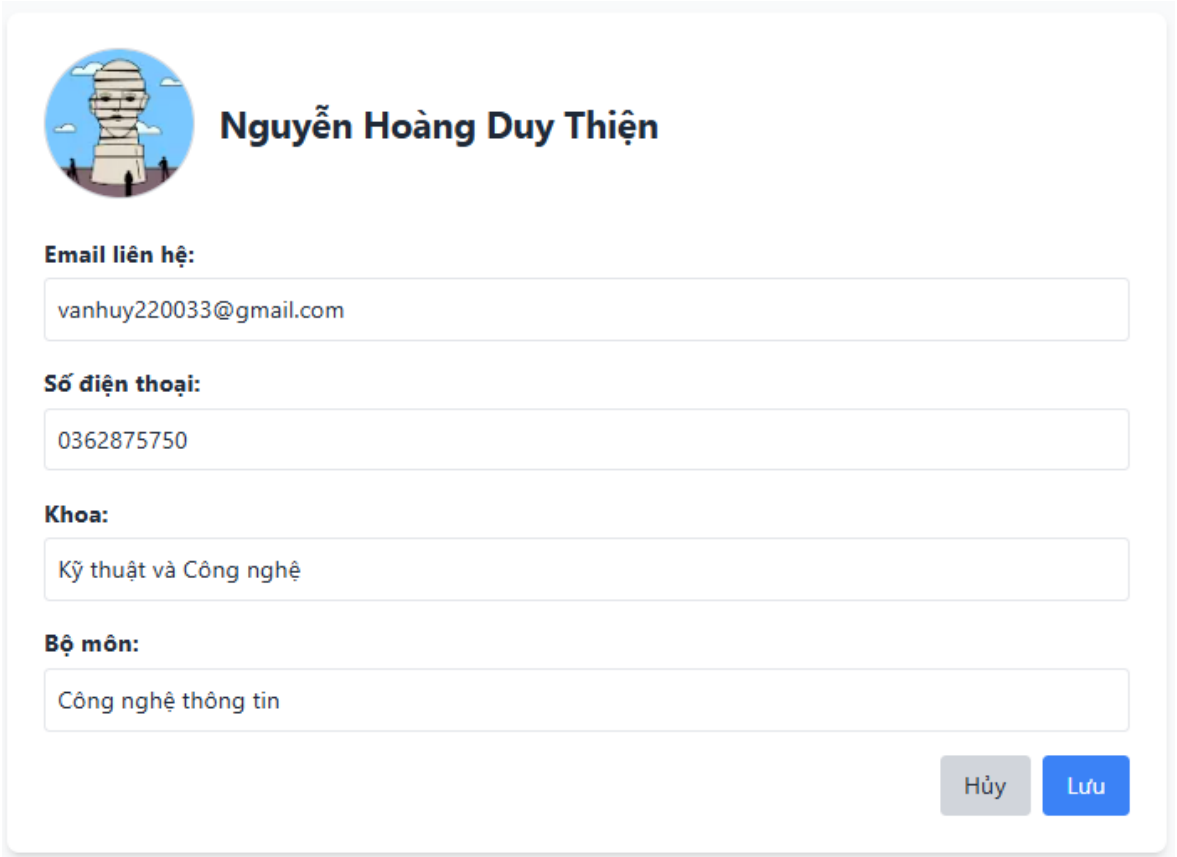
Hình 8. Trang thư mục

4.3 Giao diện trang thông tin (profile.js)

- Trang thông tin: hiển thị thông người dùng dạng ô nổi bậc giữa trang cho phép sửa các thông tin liên lạc.



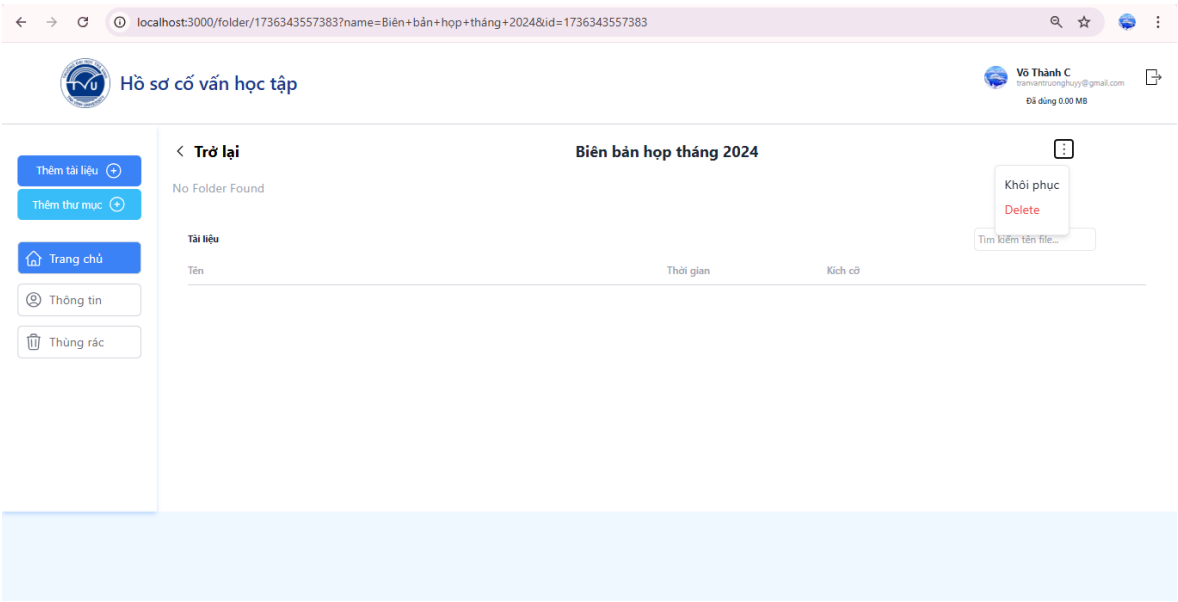
Hình 9. Trang thông tin



Hình 10. Form thay đổi thông tin liên lạc

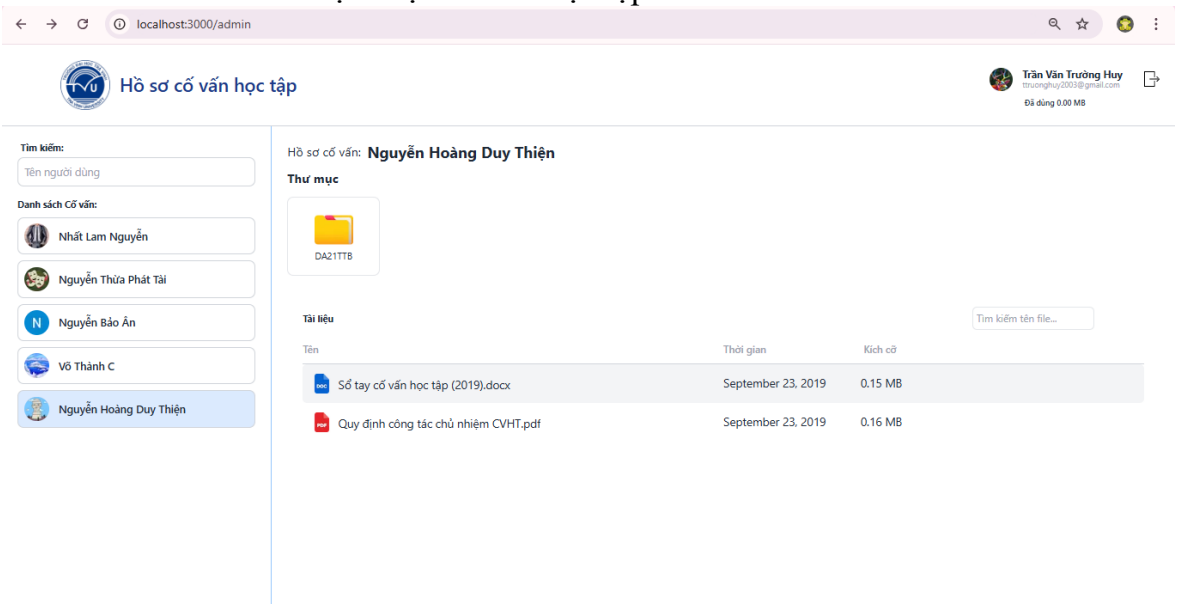
4.4 Giao diện trang thùng rác (trash.js) và trang quản trị (admin.js)

- Trang thùng rác hiển thị các thư mục và tài liệu đã xoá ở trang chủ. Với tùy chọn khôi phục hoặc xoá vĩnh viễn, tùy chọn được xử lý thao tác đóng dễ dàng bằng cách bấm bất kỳ ngoài mục tùy chọn.



Hình 11. Trang thùng rác

- Trang quản trị được thiết kế với bố cục đơn giản và cung cấp công cụ tìm kiếm tối ưu cho việc chọn cố vấn học tập từ danh sách.



Hình 12. Trang quản trị

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Qua quá trình nghiên cứu và phát triển, website quản lý hồ sơ Cố vấn học tập đã hoàn thành các chức năng chính bao gồm:

- Lưu trữ và quản lý hồ sơ học tập của giảng viên theo cách cấu trúc hệ thống, khoa học.
- Hỗ trợ tải lên, tải xuống, xóa, và phục hồi tài liệu, giúp giảng viên đối phó hiệu quả với công tác quản lý tài liệu.
- Xác thực người dùng qua Google OAuth, phân quyền rõ ràng giữa giảng viên và nhân viên văn phòng khoa.
- Cung cấp giao diện người dùng thân thiện, hỗ trợ hiển thị dữ liệu nhanh chóng nhờ Firebase Firestore và Storage.

Hệ thống đã đáp ứng tốt các yêu cầu đặt ra và góp phần tối ưu hóa công việc của giảng viên trong công tác cố vấn học tập. Tuy nhiên, website cũng còn những điểm có thể cải thiện để phát triển hơn nữa trong tương lai.

5.2 Hướng phát triển

Mặc dù đã đạt được những thành tựu nhất định, website vẫn còn nhiều tiềm năng phát triển thêm:

5.2.1 Nâng cao trải nghiệm người dùng

- Cải tiến giao diện: Tối ưu hóa giao diện người dùng để trực quan hơn, bao gồm việc thiết kế lại một số thành phần giao diện như thanh điều hướng, bố cục danh sách tài liệu.
- Tăng tính tương tác: Thêm các hiệu ứng tương tác nhẹ khi thao tác lên giao diện.
- Cá nhân hóa giao diện: Cho phép người dùng tùy chỉnh giao diện theo sở thích cá nhân (chế độ tối, sắp xếp danh mục tài liệu).

5.2.2 Mở rộng chức năng

- Tìm kiếm nâng cao: Bổ sung các tùy chọn tìm kiếm theo ngày tải lên, kích thước tài liệu, hoặc loại tài liệu.
- Thống kê sử dụng: Thêm chức năng thống kê dung lượng đã sử dụng, lượt tải xuống, lượt xem.
- Cải thiện chức năng xửa: Cho phép xác nhận nhiều tài liệu để xóa cùng lúc.

- Phân quyền linh hoạt: Cung cấp các mức phân quyền khác nhau như quản trị viên, giám sát, hoặc nhóm người dùng.

5.2.3 Bảo mật và hiệu suất

- Nâng cấp bảo mật: Sử dụng giao thức HTTPS để đảm bảo truyền tải dữ liệu an toàn.
- Tăng hiệu suất tải trang: Đệ quyết dung lượng tải xuống tối ưu, tích hợp caching.
- Chính sách xác thực nhiều yếu tố (MFA): Thêm cơ chế xác thực qua email hoặc số điện thoại.

5.2.4 Hỗ trợ đa nền tảng

- Tối ưu cho thiết bị di động: Cải thiện giao diện và hiểu năng trên các thiết bị di động như smartphone và tablet.
- Xây dựng Ứng dụng di động (Mobile App): Phát triển ứng dụng di động native để hỗ trợ người dùng linh hoạt trên các hệ điều hành Android và iOS.
- Đám mây lai (Hybrid Cloud): Tích hợp để cung cấp tính năng truy cập linh hoạt giữa các môi trường.

5.3 Lời kết

Quá trình nghiên cứu và phát triển đã hoàn thành những mục tiêu đề ra ban đầu, góp phần cải thiện công tác quản lý hồ sơ của giảng viên. Website là bước tiến quan trọng trong việc đẩy mạnh chuyển đổi số trong giáo dục. Trong tương lai, những hướng phát triển đề ra sẽ góp phần mang lại một hệ thống hoàn thiện hơn, đáp ứng nhu cầu ngày càng cao của người dùng.

DANH MỤC TÀI LIỆU THAM KHẢO

Tìm hiểu TailwindCSS: https://viblo.asia/p/tim-hieu-ve-tailwind-css-924lJp6WKPM

Tìm hiểu Next.js: https://200lab.io/blog/nextjs-la-gi?srsltid=AfmBOoobJ9gbzUxtettBU35V9d1pjdY5vKsSLakOJmc5TK2-t0e3aEZt

Tìm hiểu Firebase: https://viblo.asia/p/tim-hieu-ve-google-firebase-ymwGXVZ4R4p1