TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

ISO 9001:2015

NGUYỄN ĐÌNH TRÍ

XÂY DỰNG HỆ THỐNG TẠO VÀ ĐÁNH GIÁ HỒ SƠ XIN VIỆC TÍCH HỢP AI

KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

VĨNH LONG, NĂM 2025

TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

ISO 9001:2015

XÂY DỰNG HỆ THỐNG TẠO VÀ ĐÁNH GIÁ HỒ SƠ XIN VIỆC TÍCH HỢP

KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

Sinh viên: **Nguyễn Đình Trí**
Lớp: **DA21TTB**
MSSV: **110121119**
GVHD: **TS. Nguyễn Bảo Ân**

VĨNH LONG, NĂM 2025

LỜI CAM ĐOAN

Em xin cam đoan rằng khóa luận tốt nghiệp với đề tài "Xây dựng hệ thống tạo và đánh giá hồ sơ xin việc tích hợp AI" là công trình nghiên cứu độc lập của bản thân em, được hoàn thành dưới sự hướng dẫn tận tình của TS. Nguyễn Bảo Ân.

Tất cả các nội dung, kết quả nghiên cứu trong đồ án này đều trung thực, khách quan và chưa từng được công bố trong bất kỳ nghiên cứu nào trước đây. Các tài liệu tham khảo, thông tin, số liệu và kết quả trích dẫn từ các nguồn khác đều được ghi rõ nguồn gốc theo đúng quy định.

Vĩnh Long, ngày …. tháng .... năm 2025

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

Nguyễn Đình Trí

LỜI CẢM ƠN

Để hoàn thành đồ án tốt nghiệp với đề tài "Xây dựng hệ thống tạo và đánh giá hồ sơ xin việc tích hợp AI", bên cạnh sự nỗ lực cá nhân, em đã nhận được rất nhiều sự hỗ trợ, chỉ dẫn tận tình và động viên quý báu từ quý thầy cô, gia đình và bạn bè.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc và chân thành đến TS. Nguyễn Bảo Ân, người đã trực tiếp hướng dẫn em trong suốt quá trình thực hiện đề tài. Thầy đã không chỉ truyền đạt những kiến thức chuyên môn hữu ích mà còn luôn định hướng, hỗ trợ và tạo điều kiện thuận lợi để em có thể phát triển ý tưởng, từng bước triển khai và hoàn thiện đồ án. Những góp ý tận tình và kịp thời của Thầy đã trở thành nguồn động lực lớn giúp em vượt qua những khó khăn trong quá trình nghiên cứu.

Em cũng xin gửi lời cảm ơn chân thành đến Ban Giám hiệu Trường Đại học Trà Vinh, cùng toàn thể quý thầy cô trong Trường Kỹ thuật và Công nghệ, đặc biệt là các thầy cô thuộc Khoa Công nghệ Thông tin. Những bài giảng, sự truyền đạt kinh nghiệm quý báu trong suốt những năm học qua chính là nền tảng vững chắc giúp em tự tin thực hiện đồ án tốt nghiệp và chuẩn bị hành trang cho con đường sự nghiệp tương lai.

Em xin gửi lời cảm ơn đến các bạn sinh viên lớp DA21TTB, những người đã đồng hành cùng em trong học tập, sẵn sàng chia sẻ kiến thức và luôn động viên em trong suốt quá trình hoàn thành đồ án.

Cuối cùng, con xin dành lời tri ân sâu sắc nhất đến gia đình thân yêu, đặc biệt là cha mẹ, người luôn là điểm tựa tinh thần vững vàng, không ngừng yêu thương, tin tưởng và tạo mọi điều kiện tốt nhất để con yên tâm học tập, rèn luyện và theo đuổi đam mê.

Dù đã cố gắng rất nhiều trong quá trình thực hiện, nhưng do giới hạn về kiến thức và kinh nghiệm thực tế, đồ án vẫn khó tránh khỏi những sai sót. Em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô và các bạn để có thể hoàn thiện đề tài tốt hơn trong tương lai.

NHẬN XÉT

(Của giảng viên hướng dẫn trong đồ án, khóa luận của sinh viên)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

UBND TỈNH VĨNH LONG
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh Phúc

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP

(Của giảng viên hướng dẫn)

Họ và tên sinh viên: Nguyễn Đình Trí MSSV: 110121119

Ngành: Công nghệ Thông tin Khóa: 2021

Tên đề tài: Xây dựng hệ thống tạo và đánh giá hồ sơ xin việc tích hợp AI

Họ và tên Giáo viên hướng dẫn: Nguyễn Bảo Ân .

Chức danh: Phó Trưởng khoa Học vị: Tiến sĩ .

NHẬN XÉT

1. Nội dung đề tài:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

2. Ưu điểm:

.....
.....
.....
.....

3. Khuyết điểm:

.....
.....
.....
.....

.....

4. Điểm mới đề tài:

.....

.....

.....

.....

.....

5. Giá trị thực trên đề tài:

.....

.....

.....

.....

.....

.....

7. Đề nghị sửa chữa bổ sung:

.....

.....

.....

.....

.....

.....

.....

8. Đánh giá:

.....

.....

.....

.....

Vĩnh Long, ngày .... tháng .... năm 2025

Giảng viên hướng dẫn

(Ký & ghi rõ họ tên)

Nguyễn Bảo Ân

NHẬN XÉT

(Của giảng viên chấm đồ án, khóa luận của sinh viên)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên chấm

(Ký và ghi rõ họ tên)

UBND TỈNH VĨNG LONG
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh Phúc

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP

(Của cán bộ chấm đồ án, khóa luận)

Họ và tên người nhận xét: .....
Chức danh: ..... Học vị: .....
Chuyên ngành: .....
Cơ quan công tác: .....
Tên sinh viên: .....
Tên đề tài đồ án, khóa luận tốt nghiệp: .....
.....
.....

I. Ý KIẾN NHẬN XÉT

1. Nội dung:

.....
.....
.....
.....
.....
.....
.....
.....

2. Điểm mới các kết quả của đồ án, khóa luận:

.....
.....
.....

3. Ứng dụng thực tế:

.....
.....
.....
.....
.....

.....

.....

II. CÁC VẤN ĐỀ CẦN LÀM RÕ

(Các câu hỏi của giáo viên phản biện)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

III. KẾT LUẬN

(Ghi rõ đồng ý hay không đồng ý cho bảo vệ đồ án khóa luận tốt nghiệp)

.....

.....

.....

.....

.....

Vĩnh Long, ngày …. tháng …. năm 2025

Người nhận xét

(Ký & ghi rõ họ tên)

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

Từ viết tắt	Ý nghĩa
AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
CLR	Common Language Runtime
IoT	Internet of Things
SEO	Search Engine Optimization
I/O	Input/Output
HTTP/HTTPS	Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
LLM	Large Language Model
URL	Uniform Resource Locator
XML	Extensible Markup Language
SQL	Structured Query Language
UI	User Interface
JWT	JSON Web Token
XSS	Cross-Site Scripting
CSRF	Cross-Site Request Forgery
CV	Curriculum Vitae

DANH MỤC BẢNG, SƠ ĐỒ

DANH MỤC HÌNH ẢNH

TÓM TẮT

Trong bối cảnh cách mạng công nghiệp 4.0 và sự phát triển nhanh chóng của Trí tuệ nhân tạo, việc ứng dụng công nghệ vào đời sống, đặc biệt trong tuyển dụng và tìm việc, đang trở thành xu hướng tất yếu. Thị trường lao động cạnh tranh đòi hỏi ứng viên sở hữu hồ sơ xin việc không chỉ chuyên nghiệp, đầy đủ mà còn ấn tượng và phù hợp vị trí ứng tuyển. Tuy nhiên, việc tự tạo và đánh giá hồ sơ xin việc vẫn là thách thức lớn, nhất là với người thiếu kinh nghiệm hoặc chưa biết cách tối ưu thông tin. Do đó, công cụ thông minh có khả năng gợi ý, chỉnh sửa, đánh giá hồ sơ xin việc một cách khách quan, chính xác là rất cần thiết.

Để đáp ứng nhu cầu đó, đề tài "Xây dựng hệ thống tạo và đánh giá hồ sơ xin việc tích hợp AI" được thực hiện. Đồ án hướng đến việc phát triển một hệ thống cho phép người dùng dễ dàng tạo, chỉnh sửa và quản lý hồ sơ xin việc một cách nhanh chóng thông qua giao diện kéo thả các thành phần được thiết kế sẵn. Điểm nổi bật của hệ thống là tích hợp API của Gemini AI để đánh giá chất lượng hồ sơ xin việc, đưa những phản hồi chuyên sâu và gợi ý cải thiện, giúp ứng viên tối ưu hóa hồ sơ của mình trước khi nộp cho nhà tuyển dụng.

Đồ án đã xây dựng thành công một hệ thống hoàn chỉnh như sau:

- Frontend: Sử dụng Next.js để xây dựng giao diện người dùng hiện đại, thân thiện và linh hoạt. Giao diện này cho phép người dùng kéo thả các thành phần để tạo CV một cách trực quan, cùng với các chức năng quản lý CV và tài khoản.

- Backend: Được xây dựng bằng ASP.NET Core API theo kiến trúc Clean Architecture. Chịu trách nhiệm xử lý logic nghiệp vụ, quản lý dữ liệu người dùng, đồng thời giao tiếp với Gemini API để thực hiện các chức năng đánh giá và gợi ý.

- Cơ sở dữ liệu: Sử dụng MongoDB để lưu trữ các thông tin về hồ sơ xin việc, tài khoản người dùng và các dữ liệu liên quan khác một cách hiệu quả.

Kết quả của đề tài là một hệ thống hoạt động ổn định, cung cấp một công cụ mạnh mẽ hỗ trợ người dùng trong quá trình tìm việc. Hệ thống không chỉ giúp tiết kiệm thời gian và công sức khi tạo hồ sơ xin việc, mà còn đóng vai trò như một "chuyên gia" ảo, đưa ra những nhận xét khách quan và chính xác để nâng cao chất lượng hồ sơ, qua đó tăng cơ hội trúng tuyển cho ứng viên.

CHƯƠNG 1. TỔNG QUAN

1.1 Đặt vấn đề và bối cảnh

Trong thời đại số hóa mạnh mẽ, quá trình tuyển dụng không chỉ đòi hỏi ứng viên có năng lực chuyên môn, mà còn cần thể hiện được sự chuyên nghiệp và nổi bật ngay từ bản hồ sơ xin việc. Tuy nhiên, việc tạo ra một hồ sơ xin việc ấn tượng, logic và phù hợp với từng vị trí công việc là điều không dễ, đặc biệt đối với sinh viên mới ra trường hoặc người chưa có nhiều kinh nghiệm ứng tuyển.

Mặt khác, không phải ai cũng có cơ hội tiếp cận với các chuyên gia tư vấn nghề nghiệp hay công cụ đánh giá hồ sơ xin việc chuyên sâu. Điều này dẫn đến tình trạng nhiều ứng viên bị loại từ vòng sơ tuyển chỉ vì hồ sơ chưa đạt yêu cầu, dù năng lực có thể phù hợp với công việc.

Từ thực trạng đó, việc ứng dụng AI vào quá trình tạo và đánh giá hồ sơ xin việc là một hướng tiếp cận đầy tiềm năng. Một hệ thống thông minh có thể hoạt động như một "cố vấn nghề nghiệp ảo", hỗ trợ người dùng xây dựng và hoàn thiện hồ sơ của mình một cách chủ động. Bằng cách tận dụng sức mạnh của các công nghệ AI, hệ thống có thể đánh giá mức độ phù hợp của hồ sơ xin việc, gợi ý các điểm cần cải thiện và giúp ứng viên chuẩn bị một hồ sơ tốt nhất để tự tin tiếp cận thị trường lao động.

Chính vì vậy, đề tài này được thực hiện nhằm mục tiêu xây dựng một hệ thống hoàn chỉnh, hỗ trợ hiệu quả cho người dùng trong quá trình tạo và tối ưu hóa hồ sơ xin việc.

1.2 Nhu cầu tạo và đánh giá hồ sơ xin việc trong bối cảnh hiện đại

Việc tạo ra một hồ sơ xin việc ấn tượng và chuyên nghiệp là bước đi đầu tiên, đóng vai trò then chốt trong hành trình tìm kiếm việc làm. Tuy nhiên, nhiều ứng viên đặc biệt là sinh viên mới tốt nghiệp hoặc người đang chuyển hướng nghề nghiệp thường gặp một số khó khăn sau:

- Thiếu công cụ linh hoạt: Các nền tảng thiết kế hồ sơ truyền thống đang có giao diện cứng nhắc, ít tùy chọn cá nhân hóa, khiến người dùng khó thể hiện sự sáng tạo và nét riêng.

- Thiếu đánh giá khách quan: Người tìm việc thường không biết liệu hồ sơ của mình đã đủ thuyết phục hay chưa, cũng như chưa rõ điểm mạnh, điểm yếu so với yêu cầu tuyển dụng.

- Khó tối ưu hóa: Nhiều hồ sơ không được chuẩn hóa để vượt qua các hệ thống lọc tự động và khó thu hút ánh nhìn của nhà tuyển dụng.

Trong bối cảnh đó, một hệ thống hỗ trợ tạo và đánh giá hồ sơ xin việc tích hợp AI sẽ đóng vai trò như một "chuyên gia" ảo, giúp ứng viên khắc phục những hạn chế trên. Hệ thống không chỉ cung cấp công cụ trực quan, dễ sử dụng (ví dụ như kéo thả bố cục), mà còn tận dụng trí tuệ nhân tạo để phân tích nội dung, đưa ra gợi ý điều chỉnh hợp lý, từ đó nâng cao chất lượng hồ sơ và gia tăng cơ hội trúng tuyển.

1.3 Mục tiêu, đối tượng và phạm vi nghiên cứu

1.3.1 Mục tiêu

Mục tiêu chính của đề tài là xây dựng một hệ thống hiện đại, giúp người dùng dễ dàng tạo và tối ưu hóa hồ sơ xin việc với sự hỗ trợ của trí tuệ nhân tạo. Để đạt được mục tiêu này, đề tài tập trung vào các nhiệm vụ cụ thể sau:

- Phát triển nền tảng linh hoạt: Xây dựng một hệ thống cho phép người dùng tạo, đọc, cập nhật, xóa hồ sơ xin việc một cách trực quan, sử dụng cơ chế kéo thả các thành phần có sẵn để tiết kiệm thời gian và mang lại trải nghiệm tùy chỉnh cao.

- Tích hợp công nghệ hiện đại: Ứng dụng công nghệ tiên tiến như Next.js cho giao diện người dùng, ASP.NET Core API cho hệ thống backend và MongoDB để quản lý cơ sở dữ liệu, đảm bảo hệ thống hoạt động ổn định, hiệu năng cao và dễ dàng mở rộng.

- Nâng cao chất lượng hồ sơ bằng AI: Tích hợp thành công Gemini API để phát triển chức năng đánh giá hồ sơ xin việc thông minh. Hệ thống sẽ phân tích nội dung, đưa ra các nhận xét chuyên sâu và gợi ý cải thiện dựa trên yêu cầu công việc hoặc thông tin có trong hồ sơ xin việc, giúp người dùng nâng cao khả năng trúng tuyển.

- Hoàn thiện hệ thống chức năng: Triển khai đầy đủ các tính năng chính bao gồm tạo và quản lý hồ sơ xin việc, đánh giá hồ sơ bằng AI. Đồng thời, phát triển các chức năng phụ như quản lý tài khoản và gợi ý việc làm phù hợp.

1.3.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài tập trung vào việc ứng dụng các công nghệ hiện đại để giải quyết bài toán thực tế. Cụ thể, đề tài sẽ đi sâu tìm hiểu về kiến trúc Clean Architecture trong ASP.NET Core API, các framework frontend như Next.js, cùng với cơ sở dữ liệu phi quan hệ MongoDB. Bên cạnh đó, đề tài cũng sẽ nghiên cứu về khả năng của Gemini API trong việc xử lý ngôn ngữ tự nhiên để phân tích, đánh giá nội dung hồ sơ xin việc và đưa ra các gợi ý có ý nghĩa. Song song với đó, việc tìm hiểu nhu cầu và hành vi của người dùng, đặc biệt là sinh viên và người tìm việc, là rất quan trọng để có thể thiết kế một hệ thống thân thiện, dễ sử dụng và mang lại giá trị cao.

1.3.3 Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài được giới hạn như sau: Về chức năng, hệ thống sẽ tập trung vào việc tạo, quản lý và đánh giá hồ sơ xin việc, trong đó chức năng đánh giá sẽ sử dụng API của Gemini. Về công nghệ, đề tài sẽ chỉ sử dụng các công nghệ đã được liệt kê ban đầu như Next.js, ASP.NET Core API, MongoDB, Gemini API và các công nghệ liên quan để đảm bảo tính khả thi trong phạm vi đồ án. Về đối tượng, hệ thống xây dựng nhằm hỗ trợ người dùng nói chung, đặc biệt là các ứng viên chưa có kinh nghiệm, giúp họ có được một hồ sơ xin việc chuyên nghiệp và hiệu quả hơn.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Tìm hiểu Next.js

2.1.1 Tổng quan Next.js

Next.js là một framework dựa trên React, dùng để xây dựng các ứng dụng web full-stack. Bạn sử dụng React Components để xây dựng giao diện người dùng, và Next.js cung cấp thêm các tính năng và tối ưu hóa. Nó cũng tự động cấu hình các công cụ cấp thấp như bundler và compiler [1].

2.1.2 Ưu điểm Next.js

Hiệu suất và SEO vượt trội:

Next.js hỗ trợ Server-Side Rendering (SSR) và Static Site Generation (SSG), giúp nội dung được xử lý từ server hoặc tạo sẵn dưới dạng file tĩnh trước khi gửi đến người dùng. Điều này giúp trang web tải nhanh hơn và được các công cụ tìm kiếm index hiệu quả hơn. Ngoài ra, tính năng Incremental Static Regeneration (ISR) cho phép cập nhật nội dung tĩnh mà không cần build lại toàn bộ ứng dụng, tiết kiệm thời gian triển khai và cải thiện trải nghiệm người dùng.

Routing tự động và đơn giản:

Next.js cung cấp cơ chế định tuyến dựa trên cấu trúc thư mục pages, cho phép tạo trang mới chỉ bằng cách thêm file mà không cần cấu hình phức tạp. Framework này cũng hỗ trợ định tuyến động, giúp xử lý các đường dẫn có tham số một cách linh hoạt. Điều này giúp lập trình viên giảm thời gian thiết lập và dễ dàng quản lý hệ thống điều hướng trong ứng dụng.

Tích hợp API Routes:

Next.js cho phép tạo các endpoint API ngay trong dự án mà không cần cài đặt backend riêng biệt. Điều này rất hữu ích với các ứng dụng vừa và nhỏ, giúp xử lý các tác vụ như nhận form, truy vấn cơ sở dữ liệu hoặc gọi API bên thứ ba. Nhờ đó, quá trình phát triển trở nên nhanh chóng và thống nhất giữa frontend và backend.

Tối ưu hình ảnh và tài nguyên:

Next.js tích hợp sẵn tính năng tối ưu hình ảnh thông qua component next/image, hỗ trợ nén ảnh, lazy loading và tự động điều chỉnh kích thước phù hợp với từng thiết bị.

Ngoài ra, framework còn hỗ trợ preload font, chia nhỏ bundle và chỉ tải mã nguồn khi cần thiết, giúp giảm dung lượng tải xuống và tăng tốc độ phản hồi của trang web.

2.1.3 Nhược điểm Next.js

Cấu hình phức tạp hơn React thuần:

Do tích hợp nhiều tính năng nâng cao, dự án Next.js có thể phức tạp hơn so với React thuần, đặc biệt khi tùy chỉnh Webpack, Babel hoặc thiết lập server tùy biến. Kích thước build cũng có thể lớn hơn nếu không tối ưu đúng cách. Điều này đòi hỏi lập trình viên phải có kiến thức sâu hơn về cấu hình và tối ưu hóa ứng dụng.

Hạn chế khi xử lý backend phức tạp:

API Routes của Next.js phù hợp cho các tác vụ xử lý nhẹ hoặc trung bình, nhưng khi cần xử lý dữ liệu phức tạp, thời gian chạy dài hoặc tính năng real-time, vẫn cần sử dụng backend độc lập như Node.js, .NET hay Java. Điều này khiến Next.js chưa thể thay thế hoàn toàn hệ thống backend chuyên biệt.

Phụ thuộc vào môi trường Node.js:

Next.js yêu cầu môi trường Node.js để chạy, điều này có thể gây hạn chế khi triển khai trên các nền tảng không hỗ trợ Node. Một số tính năng như tối ưu ảnh hoặc SSR cần server chuyên dụng, nên không thể triển khai hoàn toàn trên các dịch vụ hosting tĩnh đơn giản như GitHub Pages hoặc Netlify ở chế độ static export.

2.1.4 Một số tính năng nổi bật của Next.js

- Server-Side Rendering (SSR): Cho phép render trang ngay trên server trước khi gửi đến trình duyệt, giúp cải thiện tốc độ tải trang và tối ưu SEO. SSR đặc biệt hữu ích với các trang cần hiển thị dữ liệu động ngay khi người dùng truy cập.

- Static Site Generation (SSG): Tạo các trang tĩnh ngay tại thời điểm build, giúp trang web tải cực nhanh và giảm tải cho server. SSG phù hợp với các trang có nội dung ít thay đổi, như blog, trang giới thiệu sản phẩm hoặc tài liệu.

- Incremental Static Regeneration (ISR): Cho phép cập nhật từng trang tĩnh một cách tự động mà không cần build lại toàn bộ dự án. Điều này giúp tiết kiệm thời gian triển khai và đảm bảo nội dung luôn được làm mới.

- Routing tự động và linh hoạt: Next.js tạo route dựa trên cấu trúc thư mục pages và hỗ trợ định tuyến động, giúp lập trình viên quản lý đường dẫn dễ dàng. Không cần cài đặt thư viện routing riêng như React Router.

- API Routes tích hợp: Hỗ trợ xây dựng các endpoint API trực tiếp trong dự án mà không cần backend riêng. Điều này giúp xử lý các yêu cầu như lấy dữ liệu, xác thực hoặc kết nối dịch vụ bên thứ ba ngay trong ứng dụng.

- Tối ưu hình ảnh với next/image**:** Cung cấp tính năng nén, resize và lazy loading cho hình ảnh, giúp giảm dung lượng tải xuống và tăng hiệu suất hiển thị. Tự động lựa chọn kích thước phù hợp với từng thiết bị người dùng.

- Hỗ trợ TypeScript sẵn: Next.js tích hợp TypeScript ngay từ đầu, giúp lập trình viên viết code an toàn hơn và giảm lỗi khi phát triển các dự án lớn.

- Middleware: Cho phép chạy logic tùy chỉnh trước khi request được xử lý, hữu ích cho các tác vụ như xác thực, phân quyền hoặc chuyển hướng người dùng.

2.2 Tìm hiểu .NET

2.2.1 Tổng quan .NET

.NET là một nền tảng phát triển phần mềm miễn phí, mã nguồn mở và đa nền tảng, được thiết kế để xây dựng nhiều loại ứng dụng như web, dịch vụ cloud, desktop, mobile, game và IoT. Nền tảng .NET cho phép chạy các chương trình viết bằng nhiều ngôn ngữ (C# là phổ biến nhất) trên nhiều nền tảng như Windows, Linux và macOS. Nền tảng này cung cấp một runtime hiệu suất cao, vốn đang được sử dụng trong nhiều hệ thống quy mô lớn, kèm theo một bộ thư viện tiêu chuẩn (class library) phong phú hỗ trợ các thao tác cơ bản và nâng cao, từ quản lý bộ nhớ đến xử lý luồng và thao tác dữ liệu [2].

2.2.2 Ưu điểm của .NET

Dễ phát triển:

Các nhà phát triển thích sử dụng .NET vì nền tảng này chứa nhiều công cụ giúp họ làm việc dễ dàng hơn. Ví dụ: nhờ sử dụng bộ Visual Studio, các nhà phát triển có thể viết mã nhanh hơn, cộng tác hiệu quả, đồng thời kiểm thử và sửa mã của họ một cách hiệu quả. Khả năng tái sử dụng mã giữa các kiểu triển khai giúp giảm chi phí phát triển.

Ứng dụng hiệu năng cao:

Các ứng dụng .NET cung cấp thời gian phản hồi nhanh hơn và đòi hỏi công suất điện toán thấp hơn. Những ứng dụng này sở hữu các biện pháp bảo mật tích hợp mạnh mẽ và thực hiện hiệu quả các tác vụ phía máy chủ như truy cập cơ sở dữ liệu.

Hỗ trợ cộng đồng:

.NET là nền tảng nguồn mở, tức là bất kỳ ai cũng có thể truy cập để tự do sử dụng, đọc và sửa đổi .NET. Một cộng đồng gồm các nhà phát triển hoạt động tích cực trong việc duy trì và cải thiện phần mềm .NET. .NET Foundation là tổ chức phi lợi nhuận độc lập, được thành lập để hỗ trợ cộng đồng .NET. Tổ chức này cung cấp các tài nguyên học tập, dự án .NET nguồn mở và nhiều sự kiện khác nhau dành cho các nhà phát triển .NET.

***Hỗ trợ đa nền tảng (Cross-platform)*:**

Với .NET Core / .NET 5+, ứng dụng có thể chạy trên Windows, Linux và macOS mà không cần viết lại toàn bộ mã nguồn, giúp mở rộng khả năng triển khai và tiết kiệm công sức phát triển.

Bảo mật tích hợp:

.NET hỗ trợ nhiều cơ chế bảo mật sẵn có như xác thực (Authentication), phân quyền (Authorization), mã hóa dữ liệu, chống tấn công XSS, CSRF… giúp bảo vệ ứng dụng khỏi các lỗ hổng phổ biến.

Khả năng mở rộng (Scalability):

Nền tảng .NET có thể đáp ứng từ các ứng dụng nhỏ cho đến hệ thống quy mô lớn, hỗ trợ mở rộng dễ dàng khi nhu cầu người dùng và dữ liệu tăng.

Hệ sinh thái thư viện và công cụ phong phú:

Kho NuGet chứa hàng trăm nghìn gói thư viện, giúp tích hợp nhanh các chức năng mà không cần viết lại từ đầu, rút ngắn thời gian phát triển.

Tích hợp tốt với Azure và các dịch vụ Microsoft:

.NET hoạt động mượt mà với các dịch vụ đám mây như Azure, Office 365, Power BI…, giúp triển khai và mở rộng ứng dụng dễ dàng.

2.2.3 Nhược điểm của .NET

Bên cạnh nhiều những ưu điểm nổi bật kể trên, .Net cũng tồn tại một số hạn chế sau:

Phụ thuộc vào hệ sinh thái Microsoft:

Dù hiện nay .NET đã hỗ trợ đa nền tảng, nhưng phần lớn công cụ, thư viện và dịch vụ tối ưu nhất vẫn nằm trong hệ sinh thái Microsoft, khiến một số tính năng khó tích hợp với công nghệ ngoài.

Kích thước ứng dụng lớn:

Các ứng dụng .NET, đặc biệt là khi publish dạng self-contained, thường có dung lượng file triển khai lớn hơn so với một số nền tảng khác.

Chi phí bản quyền cho một số công cụ:

Dù .NET runtime miễn phí, nhưng các công cụ mạnh như Visual Studio bản Professional/Enterprise hoặc một số thư viện thương mại kèm có thể tốn phí đáng kể.

Độ phức tạp khi học ban đầu:

Hệ sinh thái .NET rộng lớn với nhiều framework (ASP.NET, WPF, MAUI, Blazor…) và khái niệm (CLR, GC, async/await…) có thể khiến người mới khó nắm bắt toàn diện.

Khả năng tương thích phiên bản:

Khi chuyển từ .NET Framework sang .NET Core hoặc .NET 5+, một số API và thư viện cũ không còn tương thích, yêu cầu chỉnh sửa hoặc thay thế.

2.2.4 Các thành phần của kiến trúc .NET

.NET có kiến trúc dạng mô-đun và được tối ưu hóa. Người dùng có thể chọn các thành phần khác nhau để đáp ứng nhu cầu phát triển phần mềm của họ.

Đây là ba thành phần chính của .NET:

- **Ngôn ngữ .NET:** .NET hỗ trợ nhiều ngôn ngữ lập trình, như C#, F#, và Visual Basic. Các ngôn ngữ này cho phép nhà phát triển viết mã cho các ứng dụng trên nền tảng .NET. Chúng được thiết kế để làm việc mượt mà với .NET, cung cấp cú pháp hiện đại, dễ học, và tính linh hoạt cao.

- **Khung mô hình ứng dụng:** Đây là các thư viện và công cụ được cung cấp để hỗ trợ phát triển các loại ứng dụng khác nhau, như:

+ ASP.NET Core: Dành cho các ứng dụng web và API.

+ Windows Forms và WPF: Dành cho ứng dụng desktop.

+ Xamarin/MAUI: Dành cho ứng dụng di động đa nền tảng.

Các khung mô hình ứng dụng giúp giảm thiểu thời gian phát triển bằng cách cung cấp các thành phần sẵn có, như quản lý giao diện, kết nối cơ sở dữ liệu, và xử lý dữ liệu.

- Thời gian chạy .NET (CLR): Thời gian chạy .NET là môi trường thực thi cho các ứng dụng .NET. Nó quản lý việc thực thi mã, đảm bảo hiệu suất, bảo mật và xử lý các lỗi. Các chức năng chính bao gồm:

+ Biên dịch JIT (Just-In-Time): Biến mã trung gian (IL) thành mã máy thực thi.

+ Quản lý bộ nhớ: Tự động thu hồi bộ nhớ (Garbage Collection).

+ Xử lý ngoại lệ: Quản lý lỗi trong quá trình chạy ứng dụng.

2.2.5 Các framework .NET

Các framework là tập hợp các công cụ và thư viện dành cho nhà phát triển, hỗ trợ việc phát triển dự án .NET một cách nhanh chóng và hiệu quả. Có nhiều framework khác nhau dành cho các loại ứng dụng khác nhau, được liệt kê dưới đây:

- Ứng dụng web: Framework ASP.NET mở rộng nền tảng .NET dành riêng cho việc xây dựng các ứng dụng dựa trên web. Nó hỗ trợ các công nghệ web như REST API, HTML, CSS, và JavaScript. ASP.NET cung cấp cơ sở dữ liệu người dùng tích hợp với xác thực đa yếu tố và xác thực bên ngoài. Framework này hỗ trợ các giao thức xác thực tiêu chuẩn ngành và tích hợp cơ chế bảo mật nhằm bảo vệ ứng dụng .NET của bạn khỏi các cuộc tấn công mạng.

- Ứng dụng di động: Bạn có thể sử dụng Xamarin/Mono để chạy các ứng dụng .NET trên tất cả các hệ điều hành di động chính, bao gồm iOS và Android. Xamarin đi kèm với Xamarin.Forms, một framework giao diện người dùng di động mã nguồn mở. Các nhà phát triển .NET sử dụng Xamarin.Forms để tạo ra trải nghiệm người dùng nhất quán trên các nền tảng di động. Nhờ đó, tất cả ứng dụng .NET có thể trông giống nhau, ngay cả trên các thiết bị di động khác nhau.

- Ứng dụng desktop: Bạn có thể sử dụng Xamarin để phát triển ứng dụng desktop. Ngoài ra, Universal Windows Platform (UWP) mở rộng khả năng phát triển ứng dụng trên Windows 10. Các framework khác như Windows Presentation Foundation (WPF)

và Windows Forms cũng được sử dụng để thiết kế giao diện người dùng trên nền tảng Windows.

- Các ứng dụng khác: Với ML.NET, bạn có thể phát triển và tích hợp các mô hình học máy tùy chỉnh vào ứng dụng .NET của mình. Bạn cũng có thể sử dụng .NET IoT Libraries để phát triển ứng dụng trên các cảm biến và thiết bị thông minh khác. Đối với các giải pháp chưa có trong các framework, bạn có thể tìm thấy rất nhiều thư viện chức năng cụ thể trên kho công khai NuGet. NuGet cho phép bạn tạo, chia sẻ, và sử dụng nhiều thư viện .NET cho hầu hết mọi mục đích.

2.3 Tìm hiểu ASP.NET Core Web API

2.3.1 Tổng quan về ASP.NET Core Web API

ASP.NET Core:

ASP.NET Core là một framework phát triển ứng dụng web mã nguồn mở, được thiết kế theo hướng mô-đun và linh hoạt. Đây là phiên bản tái cấu trúc hoàn toàn của ASP.NET, hợp nhất hai mô hình lập trình riêng biệt trước đây — ASP.NET MVC và ASP.NET Web API — thành một nền tảng thống nhất. Mặc dù được xây dựng mới trên nền tảng web hiện đại, ASP.NET Core vẫn duy trì sự tương đồng về mặt khái niệm so với ASP.NET truyền thống.

Một ưu điểm nổi bật của ASP.NET Core là khả năng chạy song song nhiều phiên bản (side-by-side versioning), cho phép trên cùng một máy tính, các ứng dụng khác nhau có thể sử dụng các phiên bản ASP.NET Core khác nhau. Đây là tính năng mà các phiên bản ASP.NET cũ chưa hỗ trợ. Ban đầu, ASP.NET Core hoạt động được trên cả .NET Framework (Windows-only) và .NET đa nền tảng, nhưng từ phiên bản ASP.NET Core 3.0, hỗ trợ cho .NET Framework đã được loại bỏ.

ASP.NET Core Web API:

ASP.NET Core Web API là nền tảng xây dựng các dịch vụ web RESTful hiệu năng cao và dễ mở rộng dựa trên ASP.NET Core. Công nghệ này cho phép lập trình viên phát triển các API linh hoạt, đáp ứng nhu cầu của nhiều loại client khác nhau như ứng dụng web, ứng dụng di động, ứng dụng desktop hoặc dịch vụ từ bên thứ ba.

2.3.2 Điểm mạnh của ASP.NET Core Web API

Hiệu suất vượt trội:

ASP.NET Core nổi tiếng với khả năng xử lý nhanh và hiệu suất cao, thích hợp cho việc phát triển các API phục vụ lượng truy cập lớn và khối lượng dữ liệu đáng kể. So với phiên bản tiền nhiệm ASP.NET, ASP.NET Core được tối ưu hóa ở mức thấp hơn của hệ thống, tận dụng tốt hơn tài nguyên phần cứng và hỗ trợ lập trình bất đồng bộ (asynchronous programming), giúp giảm thời gian phản hồi và nâng cao khả năng mở rộng hệ thống.

Hỗ trợ đa nền tảng:

Một điểm mạnh nổi bật của ASP.NET Core Web API là khả năng chạy trên nhiều hệ điều hành khác nhau, bao gồm Windows, Linux và macOS. Điều này giúp các nhà phát triển linh hoạt hơn trong triển khai ứng dụng, phù hợp với môi trường hạ tầng và yêu cầu dự án, đồng thời tiết kiệm chi phí khi không bị ràng buộc vào một hệ điều hành duy nhất.

Thiết kế mô-đun linh hoạt:

ASP.NET Core được xây dựng theo kiến trúc mô-đun, cho phép chỉ nạp và sử dụng những thành phần cần thiết. Điều này giúp ứng dụng nhẹ hơn, tối ưu tài nguyên, đồng thời dễ bảo trì và mở rộng khi có yêu cầu bổ sung tính năng mới.

Tích hợp các tính năng phát triển web hiện đại:

Framework này hỗ trợ nhiều tính năng tiên tiến như Dependency Injection (tiêm phụ thuộc) giúp quản lý vòng đời và phụ thuộc của đối tượng một cách rõ ràng; Middleware (phần mềm trung gian) cho phép xử lý linh hoạt các yêu cầu HTTP; cùng với lập trình bất đồng bộ giúp cải thiện hiệu suất trong các tác vụ I/O.

Bảo mật mạnh mẽ:

ASP.NET Core Web API tích hợp sẵn các cơ chế bảo mật như xác thực (Authentication), phân quyền (Authorization), mã hóa dữ liệu và bảo vệ chống các kiểu tấn công phổ biến (XSS, CSRF, SQL Injection). Điều này giúp bảo đảm an toàn cho dữ liệu và người dùng mà không cần triển khai quá nhiều giải pháp bên ngoài.

Khả năng tích hợp linh hoạt:

ASP.NET Core Web API dễ dàng kết nối với các framework front-end như Angular, React, Vue.js, cũng như tích hợp với nhiều hệ quản trị cơ sở dữ liệu (SQL Server, PostgreSQL, MongoDB, MySQL…) và các dịch vụ bên thứ ba, từ đó mở rộng khả năng của ứng dụng.

2.3.3 Một số đặc điểm cơ bản trong ASP.NET Core API

Routing trong ASP.NET Core API:

- Automatic Routing: ASP.NET Core API tự động xác định các route dựa trên tên của các controller và action methods. Ví dụ, nếu bạn tạo một controller có tên là ProductsController, ASP.NET Core tự động tạo route /api/products cho controller này.

- Custom Routing: Bạn có thể định nghĩa các route tùy chỉnh thông qua thuộc tính [Route] trên controller hoặc action method. Ví dụ, /api/products/{id} sẽ cho phép bạn truy vấn thông tin sản phẩm theo id.

- Route Parameters: ASP.NET Core API hỗ trợ route động bằng cách sử dụng tham số trong URL. Ví dụ, /api/products/{id} giúp lấy thông tin chi tiết của một sản phẩm dựa trên id.

- Query Parameters: ASP.NET Core API cho phép bạn xử lý các query parameters trong URL. Ví dụ: /api/products?category=electronics cho phép bạn tìm kiếm sản phẩm theo thể loại.

- Attribute Routing: Bạn có thể sử dụng attribute routing để gắn các route cho từng phương thức cụ thể. Ví dụ, [HttpGet("search/{keyword}")] sẽ tạo ra route /api/products/search/{keyword}.

Response Types và Status Codes trong ASP.NET Core API:

- JSON Response: ASP.NET Core API mặc định trả về dữ liệu ở định dạng JSON. Bạn có thể sử dụng các phương thức như Ok(), BadRequest(), NotFound() để trả về các mã trạng thái HTTP cùng với dữ liệu JSON.

- XML Response: Bạn có thể cấu hình ASP.NET Core API để trả về dữ liệu ở định dạng XML bằng cách sử dụng middleware hoặc cấu hình dịch vụ.

- Custom Response Formats: Bạn có thể tùy chỉnh các định dạng phản hồi khác nhau cho các client khác nhau (ví dụ: JSON, XML, hoặc các định dạng khác) thông qua cấu hình dịch vụ trong Startup.cs

- Status Codes**:** ASP.NET Core API hỗ trợ các mã trạng thái HTTP tiêu chuẩn để xác định kết quả của một yêu cầu, ví dụ như: 200 OK: Phản hồi thành công khi yêu cầu được thực hiện thành công và có dữ liệu trả về. Trả về danh sách sản phẩm hoặc chi tiết sản phẩm thì dùng return Ok(products);

- Custom Status Codes: Bạn có thể trả về mã trạng thái tùy chỉnh dựa trên các tình huống đặc biệt. Ví dụ, nếu bạn cần trả về mã lỗi 422 cho yêu cầu không hợp lệ nhưng không phải do lỗi của người dùng, bạn có thể sử dụng StatusCode(422).

Error Handling trong ASP.NET Core API

- Global Error Handling: ASP.NET Core API cho phép bạn cấu hình global error handling để xử lý các lỗi không mong muốn ở một nơi. Bạn có thể sử dụng middleware như UseExceptionHandler() để xử lý ngoại lệ toàn cục.

- Custom Error Responses: Bạn có thể tạo các phản hồi lỗi tùy chỉnh với thông điệp cụ thể cho người dùng hoặc các mã trạng thái HTTP đặc biệt như 400 (Bad Request) hoặc 500 (Internal Server Error).

- Validation Errors: ASP.NET Core API hỗ trợ validation cho các input dữ liệu. Nếu có lỗi xác thực, bạn có thể trả về thông báo lỗi chi tiết cho client.

Authentication và Authorization trong ASP.NET Core API

- JWT Authentication: ASP.NET Core API hỗ trợ xác thực qua JSON Web Tokens (JWT), giúp bảo mật các endpoint API. Bạn có thể dễ dàng cấu hình middleware để kiểm tra token trong yêu cầu API.

- Role-based Authorization: Bạn có thể sử dụng các policy và roles trong ASP.NET Core API để cấp quyền truy cập cho các user cụ thể. Ví dụ, bạn có thể tạo một policy yêu cầu người dùng phải có quyền Admin để truy cập vào một endpoint.

- OAuth Authentication: ASP.NET Core API hỗ trợ tích hợp OAuth để xác thực người dùng qua các dịch vụ như Google, Facebook, v.v.

2.4 Tìm hiểu MongoDB

2.4.1 Tổng quan về MongoDB

MongoDB là hệ quản trị cơ sở dữ liệu phi quan hệ, mã nguồn mở, được phát triển nhằm đáp ứng nhu cầu của các ứng dụng hiện đại có dữ liệu phong phú và phức tạp. Điểm nổi bật của MongoDB nằm ở khả năng lưu trữ dữ liệu dưới dạng BSON (Binary JSON) — một định dạng nhị phân tương tự như JSON — cho phép định nghĩa cấu trúc dữ liệu một cách linh hoạt mà không bị giới hạn bởi schema cứng nhắc như các hệ truyền thống.

Hệ thống này hỗ trợ mở rộng theo chiều ngang, giúp dễ dàng gia tăng dung lượng và hiệu suất khi lượng dữ liệu hoặc số lượng truy cập tăng lên, phù hợp với các ứng dụng cần khả năng xử lý dữ liệu lớn và mở rộng linh hoạt. MongoDB còn được đánh giá cao nhờ khả năng truy vấn phức tạp và hệ thống chỉ mục tối ưu, góp phần cải thiện tốc độ xử lý dữ liệu.

Ngoài ra, MongoDB tích hợp tốt với nhiều ngôn ngữ lập trình thông qua bộ công cụ và thư viện phong phú, giúp việc phát triển ứng dụng trở nên thuận tiện. Với cộng đồng người dùng đông đảo cùng sự hậu thuẫn từ MongoDB, Inc., công nghệ này đã trở thành một lựa chọn phổ biến trong giới phát triển phần mềm và doanh nghiệp, đặc biệt trong các lĩnh vực như ứng dụng web, phân tích dữ liệu, và các dự án yêu cầu sự linh hoạt cũng như khả năng mở rộng mạnh mẽ.

2.4.2 Lịch sử hình thành của MongoDB

MongoDB được khởi nguồn vào năm 2007 bởi công ty phần mềm 10gen (Mỹ), ban đầu là một phần của nền tảng "Platform as a Service" mà công ty dự định triển khai. Đến năm 2009, 10gen chuyển sang phát triển MongoDB theo mô hình mã nguồn mở, đồng thời bắt đầu cung cấp các dịch vụ thương mại và hỗ trợ kỹ thuật cho khách hàng. Năm 2013, công ty chính thức đổi tên thành MongoDB Inc.

Ngày 20/10/2017, MongoDB Inc. trở thành công ty đại chúng và được niêm yết trên sàn NASDAQ với mã chứng khoán MDB, giá chào bán cổ phiếu lần đầu (IPO) là 24 USD/cổ phiếu.

Ngày 8/11/2018, cùng với bản phát hành ổn định 4.0.4, giấy phép sử dụng của MongoDB được thay đổi từ AGPL 3.0 sang SSPL.

Ngày 30/10/2019, MongoDB Inc. hợp tác với Alibaba Cloud để cung cấp dịch vụ MongoDB dưới dạng giải pháp quản lý sẵn (Managed Service), cho phép khách hàng của Alibaba Cloud triển khai và sử dụng MongoDB từ các trung tâm dữ liệu toàn cầu của Alibaba.

2.4.3 Ưu điểm của MongoDB

Schema linh hoạt:

MongoDB không bắt buộc phải xác định schema cố định trước khi lưu trữ dữ liệu. Nhờ vậy, cấu trúc dữ liệu có thể thay đổi, bổ sung hoặc mở rộng theo thời gian mà không cần chỉnh sửa trực tiếp cấu trúc của cơ sở dữ liệu.

Dữ liệu dạng JSON-like (BSON):

Dữ liệu trong MongoDB được lưu dưới dạng BSON (Binary JSON) — một định dạng nhị phân tương tự JSON, cho phép lưu trữ đa dạng loại dữ liệu và mang lại sự thuận tiện khi thao tác, đặc biệt với các ứng dụng hiện đại.

Khả năng mở rộng ngang (Horizontal Scalability):

MongoDB hỗ trợ mở rộng hệ thống bằng cách bổ sung thêm máy chủ, giúp tăng khả năng xử lý, chịu tải và đáp ứng tốt khi lượng dữ liệu hoặc số lượng truy cập tăng cao.

Truy vấn nâng cao và indexing hiệu quả:

MongoDB cung cấp khả năng thực hiện các truy vấn phức tạp kết hợp với hệ thống chỉ mục tối ưu, từ đó rút ngắn thời gian truy vấn và nâng cao hiệu suất tìm kiếm dữ liệu.

Hỗ trợ dữ liệu lớn và tài liệu phức tạp:

Được thiết kế để xử lý khối lượng dữ liệu lớn, MongoDB cho phép lưu trữ các tài liệu có cấu trúc phong phú, chứa nhiều trường lồng nhau và mảng, mang lại sự linh hoạt trong cách biểu diễn dữ liệu.

Tích hợp đa ngôn ngữ:

MongoDB cung cấp driver cho nhiều ngôn ngữ lập trình khác nhau, giúp dễ dàng kết nối và tích hợp vào các ứng dụng đang phát triển.

Cộng đồng mạnh và hỗ trợ chính thức:

MongoDB sở hữu cộng đồng người dùng đông đảo, cung cấp nhiều tài nguyên học tập và giải pháp. Đồng thời, công ty MongoDB Inc. bảo trợ và phát triển liên tục, đảm bảo tính ổn định lâu dài.

Hỗ trợ ACID và giải pháp sao lưu:

Hệ thống tuân thủ các thuộc tính ACID (Atomicity, Consistency, Isolation, Durability) nhằm đảm bảo tính toàn vẹn và an toàn của dữ liệu. Ngoài ra, MongoDB còn cung cấp nhiều cơ chế sao lưu giúp bảo vệ dữ liệu trước rủi ro mất mát.

2.4.4 Nhược điểm của MongoDB

Mở rộng ngang phức tạp khi tải tăng đột biến:

Dù hỗ trợ khả năng mở rộng ngang, MongoDB có thể gặp khó khăn khi phải xử lý khối lượng dữ liệu tăng nhanh trong thời gian ngắn. Quá trình phân phối dữ liệu và quản lý giao dịch trong môi trường phân tán đòi hỏi kế hoạch và triển khai cẩn thận để tránh rủi ro phức tạp.

Tiêu tốn bộ nhớ lớn:

Để đạt hiệu suất tối ưu, MongoDB thường yêu cầu dung lượng bộ nhớ đáng kể. Điều này có thể trở thành hạn chế nếu hệ thống được triển khai trên máy chủ với tài nguyên hạn chế.

Hạn chế về transactions trong các phiên bản cũ:

Trước bản phát hành 4.0, MongoDB không hỗ trợ transactions gốc (native transactions) trên nhiều document, gây khó khăn cho các ứng dụng cần thực hiện giao dịch phức tạp.

Thiếu công cụ quản trị và thống kê mạnh mẽ:

So với các hệ quản trị cơ sở dữ liệu quan hệ, MongoDB không có nhiều công cụ tích hợp sẵn cho việc quản lý và phân tích thống kê nâng cao, khiến việc giám sát và tối ưu hiệu suất trở nên khó khăn hơn.

Truy vấn phức tạp gặp hạn chế:

Mặc dù xử lý tốt các truy vấn cơ bản, nhưng khi cần thực hiện các truy vấn quy mô lớn và logic phức tạp, MongoDB có thể bị hạn chế và yêu cầu cấu hình chỉ mục kỹ lưỡng để đảm bảo hiệu suất.

Rủi ro bảo mật nếu cấu hình sai:

Việc triển khai MongoDB mà không thiết lập đúng các cơ chế bảo mật có thể dẫn đến nguy cơ rò rỉ hoặc mất dữ liệu. Cần áp dụng các biện pháp bảo vệ chặt chẽ để đảm bảo an toàn cho hệ thống.

2.5 Tìm hiểu Clean Architecture

2.5.1 Tổng quan Clean Architecture

Clean Architecture là một mô hình kiến trúc phần mềm hiện đại, được đề xuất bởi Robert C. Martin nhằm giải quyết vấn đề phân tách rõ ràng giữa logic nghiệp vụ và các chi tiết triển khai như cơ sở dữ liệu, giao diện người dùng hoặc framework. Mục tiêu cốt lõi của mô hình là xây dựng hệ thống có tính độc lập cao, dễ bảo trì và dễ kiểm thử. Clean Architecture là sự kế thừa và tổng hợp các mô hình trước đó như Hexagonal Architecture, Onion Architecture, Screaming Architecture và DCI.

2.5.2 Ưu điểm của Clean Architecture

Bảo trì dễ dàng:

Clean Architecture phân tách rõ ràng giữa các tầng và thành phần, giúp việc sửa lỗi hoặc cập nhật hệ thống trở nên đơn giản. Khi một phần thay đổi, tác động đến các phần khác là tối thiểu.

Khả năng mở rộng linh hoạt:

Cấu trúc cho phép bổ sung hoặc điều chỉnh chức năng mà không phá vỡ các thành phần hiện có. Điều này đặc biệt hữu ích khi dự án cần phát triển lâu dài và mở rộng theo nhu cầu thực tế.

Độc lập về công nghệ:

Mã nguồn không bị ràng buộc với một nền tảng hay framework cụ thể, cho phép chuyển đổi sang công nghệ mới dễ dàng. Ngoài ra, việc thay đổi hoặc cập nhật thư viện, cơ sở dữ liệu hay môi trường triển khai sẽ ít ảnh hưởng đến phần lõi của hệ thống.

Độc lập về giao diện người dùng:

Logic nghiệp vụ được tách rời khỏi giao diện, cho phép thay đổi cách hiển thị (web, mobile, desktop) mà không cần chỉnh sửa phần xử lý chính.

Hỗ trợ kiểm thử hiệu quả:

Việc phân tách rõ ràng giữa logic nghiệp vụ và các yếu tố bên ngoài giúp dễ viết test tự động (unit test, integration test), tăng tính ổn định và giảm rủi ro phát sinh lỗi khi nâng cấp hệ thống.

2.5.3 Nhược điểm của Clean Architecture

Cấu trúc phức tạp và tốn thời gian:

Yêu cầu nhiều lớp và tầng hơn so với kiến trúc đơn giản, khiến khối lượng mã tăng đáng kể. Đối với các ứng dụng nhỏ, ít chức năng hoặc vòng đời ngắn, việc áp dụng có thể gây tốn công sức mà lợi ích không tương xứng.

Mức độ trừu tượng cao:

Việc tách biệt thông qua nhiều interface và tầng trung gian giúp linh hoạt hơn, nhưng lại làm giảm hiệu suất thực thi và khiến việc phát triển nhanh gặp khó khăn. Các bước triển khai phải tuân thủ đầy đủ nguyên tắc, tránh "code nhanh, bỏ qua chuẩn" vì sẽ phá vỡ kiến trúc.

Yêu cầu nhân lực có kinh nghiệm:

Để triển khai hiệu quả, đội ngũ phát triển cần hiểu rõ nguyên tắc Dependency Inversion và mô hình kiến trúc. Nếu thiếu kiến thức hoặc áp lực thời gian, rất dễ vi phạm các nguyên tắc, dẫn đến hệ thống trở nên rối rắm và khó bảo trì.

Chi phí học tập và áp dụng cao:

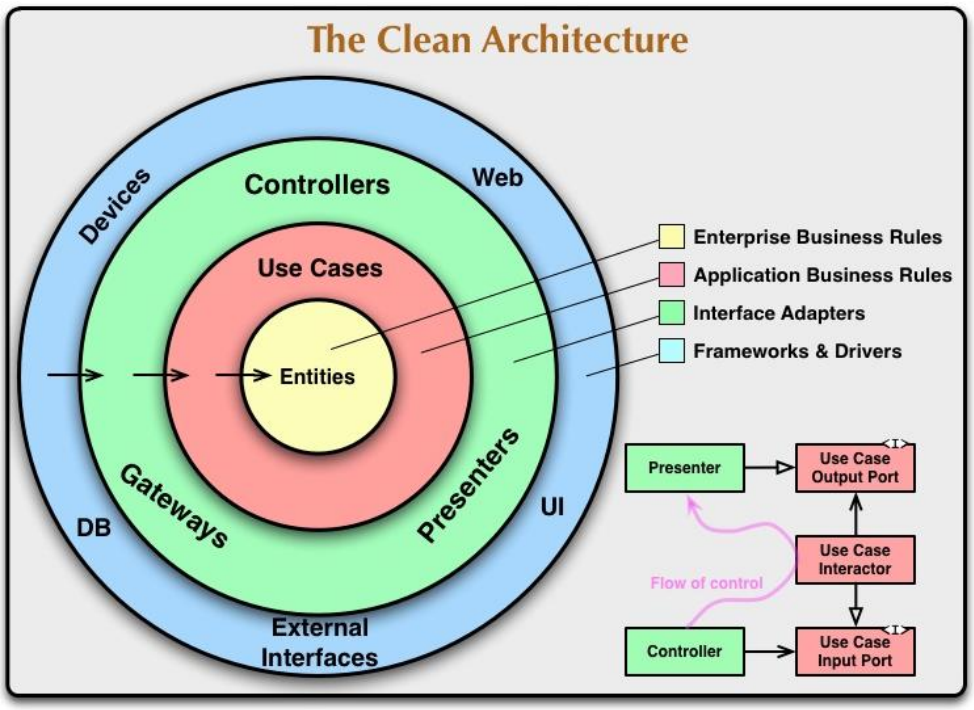
Đối với những nhóm phát triển mới làm quen, thời gian để nghiên cứu và áp dụng đúng Clean Architecture có thể kéo dài, làm tăng chi phí đào tạo và triển khai ban đầu.

2.5.4 Nguyên lý cốt lõi

Mô hình được trình bày dưới dạng các vòng tròn đồng tâm. Quy tắc phụ thuộc (The Dependency Rule) quy định rằng mọi phụ thuộc trong mã nguồn chỉ được phép hướng vào trong, tức là các lớp nằm ở vòng trong không được phép biết gì về các lớp ở vòng

ngoài. Điều này giúp đảm bảo rằng logic nghiệp vụ luôn nằm độc lập với giao diện người dùng, cơ sở dữ liệu hay bất kỳ công nghệ nào được sử dụng ở lớp ngoài [3].

2.5.5 Cấu trúc các tầng



Hình 1: Clean architecture

Tầng Entities

Tầng Entities là nơi tập trung các quy tắc nghiệp vụ mang tính toàn cục trong hệ thống. Các thực thể tại tầng này có thể được biểu diễn dưới dạng các đối tượng với phương thức xử lý, hoặc đơn giản là các cấu trúc dữ liệu kết hợp với các hàm xử lý liên quan. Hình thức thể hiện không quan trọng, miễn là các thực thể đó có thể được tái sử dụng trong nhiều ứng dụng khác nhau trong một tổ chức/doanh nghiệp.

Trong trường hợp hệ thống chỉ bao gồm một ứng dụng đơn lẻ, thì các entities chính là những đối tượng nghiệp vụ cốt lõi. Chúng bao gồm các quy tắc chung và cấp cao nhất trong hệ thống, và ít chịu ảnh hưởng bởi các thay đổi từ bên ngoài. Ví dụ, một sự thay đổi về giao diện điều hướng hay hệ thống bảo mật không nên làm ảnh hưởng đến logic của tầng này. Nói cách khác, không một sự thay đổi vận hành nào từ ứng dụng cụ thể có thể tác động trực tiếp đến tầng entity.

Tầng Use Cases

Tầng Use Cases chứa đựng các quy tắc nghiệp vụ đặc thù của ứng dụng. Phần mềm ở tầng này đảm nhiệm việc hiện thực hóa các chức năng cụ thể của hệ thống thông qua việc điều phối luồng dữ liệu giữa các entities, đồng thời kích hoạt các quy tắc nghiệp vụ tại tầng entity để hoàn thành các mục tiêu đề ra.

Các thay đổi xảy ra tại tầng này không được phép ảnh hưởng đến các entities, và ngược lại, tầng này cũng không nên bị tác động bởi các yếu tố bên ngoài như cơ sở dữ liệu, giao diện người dùng (UI), hoặc các framework phổ biến. Tầng use cases được thiết kế để cách ly hoàn toàn với các yếu tố đó.

Tuy nhiên, khi có sự thay đổi trong cách vận hành của ứng dụng – chẳng hạn thay đổi trong quy trình nghiệp vụ – thì tầng use cases có thể bị ảnh hưởng và cần được điều chỉnh tương ứng.

Tầng Interface Adapters

Tầng Interface Adapters đóng vai trò là cầu nối giữa các tầng nghiệp vụ (use cases và entities) với các yếu tố bên ngoài như cơ sở dữ liệu, giao diện người dùng hoặc dịch vụ bên ngoài. Phần mềm tại tầng này thực hiện nhiệm vụ chuyển đổi dữ liệu giữa định dạng thuận tiện cho tầng nghiệp vụ sang định dạng phù hợp với các hệ thống bên ngoài, và ngược lại.

Ví dụ, toàn bộ kiến trúc MVC (Model-View-Controller) trong một giao diện người dùng đồ họa (GUI) thường được đặt trong tầng này. Các thành phần như Presenter, View và Controller đều thuộc tầng Interface Adapters. Các Model trong trường hợp này chủ yếu là các cấu trúc dữ liệu, được truyền từ Controller đến Use Case, sau đó trở lại từ Use Case đến Presenter và View.

Tầng này cũng đảm nhiệm việc chuyển đổi dữ liệu giữa các thực thể nghiệp vụ và hệ quản trị cơ sở dữ liệu, thông qua các adapter hoặc repository. Nếu hệ thống sử dụng cơ sở dữ liệu quan hệ như SQL, thì toàn bộ câu lệnh SQL nên được giới hạn trong tầng này. Không có đoạn mã nào ở các tầng bên trong (Use Cases, Entities) được phép biết về sự tồn tại của cơ sở dữ liệu hay cú pháp SQL.

Ngoài ra, các adapter xử lý việc giao tiếp với dịch vụ bên ngoài hoặc các hệ thống khác cũng được triển khai tại tầng Interface Adapters.

Tầng Frameworks and Drivers

Đây là tầng ngoài cùng của kiến trúc Clean Architecture, nơi chứa đựng các công cụ và framework như hệ quản trị cơ sở dữ liệu, framework web, hệ thống message broker, hoặc các hệ thống bên ngoài khác.

Lượng mã nguồn viết trong tầng này thường không nhiều, chủ yếu là các đoạn mã "kết dính" (glue code) có nhiệm vụ giao tiếp với tầng Interface Adapters kế cận bên trong.

Tầng này được xem là nơi chứa các chi tiết cụ thể của hệ thống. Giao diện web, cơ sở dữ liệu, các driver – tất cả đều là chi tiết cần được cô lập khỏi phần logic cốt lõi. Bằng cách đẩy các chi tiết này ra rìa hệ thống, kiến trúc đảm bảo rằng những thay đổi ở tầng ngoài không ảnh hưởng đến các tầng bên trong, từ đó nâng cao tính linh hoạt và khả năng mở rộng của phần mềm.

2.5.6 Kỹ thuật vượt qua các ranh giới (Crossing Boundaries)

Khi dữ liệu và luồng điều khiển di chuyển giữa các lớp trong Kiến trúc Sạch, chúng ta phải tuân thủ Quy tắc Phụ thuộc (The Dependency Rule), tức là các lớp bên trong không được biết về các lớp bên ngoài. Điều này tạo ra một mâu thuẫn rõ ràng: làm thế nào để lớp bên trong (ví dụ: Use Case) có thể gọi một phương thức của lớp bên ngoài (ví dụ: Presenter)?

Để giải quyết vấn đề này, chúng ta sử dụng Nguyên lý Đảo ngược Phụ thuộc (Dependency Inversion Principle).

- Cách thức hoạt động: Các lớp bên trong (chẳng hạn như Use Case) sẽ định nghĩa một giao diện (interface).
- Triển khai: Lớp bên ngoài (chẳng hạn như Presenter) triển khai giao diện đó.
- Luồng phụ thuộc: Luồng phụ thuộc của mã nguồn lúc này sẽ hướng vào trong (Presenter phụ thuộc vào interface của Use Case), tuân thủ Quy tắc phụ thuộc.

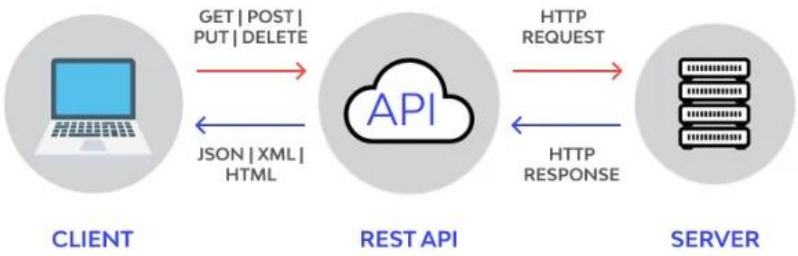
Kỹ thuật này sử dụng đa hình động (dynamic polymorphism) để cho phép luồng điều khiển đi từ trong ra ngoài, trong khi luồng phụ thuộc của mã nguồn vẫn luôn hướng vào trong [3].

2.6 Tìm hiểu RESTful API

2.6.1 Tổng quan RESTful API

RESTful API (Representational State Transfer API) là một phong cách kiến trúc cho hệ thống phân tán, được Roy Fielding giới thiệu trong luận án tiến sĩ năm 2000. Kể từ đó, REST đã trở thành một trong những phương pháp phổ biến nhất để xây dựng các API web (Application Programming Interfaces), nhờ vào cấu trúc đơn giản, khả năng mở rộng cao và dễ triển khai trên giao thức HTTP [4].

RESTful API tổ chức các tài nguyên thông qua những đường dẫn (URL) rõ ràng, dễ đọc và thân thiện với con người. Việc trao đổi dữ liệu giữa máy khách và máy chủ thường sử dụng các định dạng phổ biến như JSON hoặc XML. Nhờ đặc điểm này, RESTful API đảm bảo tính tương thích và khả năng di động cao, cho phép các ứng dụng hoặc hệ thống khác nhau dễ dàng truy cập, chia sẻ và khai thác dữ liệu.



Hình 2: Minh hoạ RESTful API

2.6.2 Các thành phần trong Restful API

API (Application Programming Interface) là tập hợp các quy tắc, chuẩn và phương thức cho phép một ứng dụng hoặc thành phần phần mềm giao tiếp và trao đổi dữ liệu với ứng dụng hoặc thành phần khác. API đóng vai trò như "cầu nối" giúp các hệ thống có thể làm việc cùng nhau. Dữ liệu phản hồi từ API thường được trả về dưới các định dạng phổ biến như JSON hoặc XML, dễ dàng xử lý và tích hợp vào ứng dụng.

REST (Representational State Transfer) là một phong cách kiến trúc được sử dụng để xây dựng API. REST tận dụng các phương thức HTTP như GET, POST, PUT, DELETE... để thao tác với tài nguyên thông qua các URL rõ ràng. Cách tiếp cận này giúp việc giao tiếp giữa máy khách và máy chủ trở nên đơn giản, thống nhất và dễ mở rộng, thay vì phải tạo ra các giao thức phức tạp riêng cho từng loại dữ liệu.

2.6.3 Các phương thức sử dụng trong Restful API

[GET]: Dùng để truy xuất dữ liệu từ một tài nguyên. Khi gửi yêu cầu GET, máy chủ sẽ phản hồi với nội dung thông tin của tài nguyên được chỉ định.

[POST]: Dùng để tạo mới một tài nguyên. Dữ liệu gửi kèm theo yêu cầu sẽ được máy chủ sử dụng để lưu trữ tài nguyên mới.

[PUT]: Được dùng để cập nhật toàn bộ một tài nguyên hiện có hoặc tạo mới nếu tài nguyên đó chưa tồn tại. Yêu cầu này thường kèm theo dữ liệu đầy đủ cần thay đổi.

[DELETE]: Yêu cầu máy chủ xóa bỏ một tài nguyên cụ thể đã được xác định.

[PATCH]: Thực hiện cập nhật một phần dữ liệu của tài nguyên, giúp giảm lượng thông tin cần gửi so với PUT.

[OPTIONS]: Trả về thông tin về các phương thức HTTP được hỗ trợ đối với một tài nguyên hoặc toàn bộ dịch vụ.

[HEAD]: Giống GET nhưng chỉ lấy phần tiêu đề (header) của phản hồi, không bao gồm nội dung. Thường được dùng để kiểm tra tình trạng tài nguyên mà không tải toàn bộ dữ liệu.

2.6.4 Các trạng thái của Restful API

- 200 (OK): Thực hiện thành công.

- 201 (Created): Trả về khi một tài nguyên vừa được tạo thành công.

- 204 (No Content): Trả về khi tài nguyên xoá thành công.

- 304 (Not Modified): Client có thể sử dụng dữ liệu cache.

- 400 (Bad Request): Yêu cầu không hợp lệ.

- 401 (Unauthorized): Yêu cầu cần được xác thực.

- 403 (Forbidden): Bị từ chối không cho phép.

- 404 (Not Found): Không tìm thấy tài nguyên từ URI.

- 405 (Method Not Allowed): Phương thức này không được phép truy cập với người dùng hiện tại.

- 410 (Gone – Resource): Không còn tồn tại, không còn hỗ trợ.

- 415 (Unsupported Media Type): Không hỗ trợ loại tài nguyên này.

- 422 (Unprocessable Entity): Dữ liệu không được xác thực.

- 429 (Too Many Requests): Yêu cầu bị từ chối do bị giới hạn.

2.6.5 Ưu điểm của RESTful API

RESTful API hiện là một trong những giải pháp quan trọng trong phát triển ứng dụng web, giúp việc trao đổi dữ liệu giữa các hệ thống trở nên đơn giản và hiệu quả. Một số lợi thế nổi bật gồm:

Đơn giản trong phát triển:

RESTful API tuân theo các phương thức HTTP chuẩn, giúp quy trình xây dựng ứng dụng trở nên trực quan hơn. Nhờ vậy, lập trình viên có thể tập trung vào chức năng chính thay vì mất thời gian xử lý việc kết nối giữa các thành phần.

Khả năng mở rộng:

Cho phép các hệ thống dễ dàng liên kết và chia sẻ dữ liệu, hỗ trợ bổ sung tính năng mới thông qua việc tích hợp API từ các dịch vụ khác.

Tính độc lập:

Các ứng dụng sử dụng RESTful API có thể hoạt động tách biệt, giúp việc bảo trì và nâng cấp trở nên thuận lợi hơn.

Khả năng tương thích cao:

Có thể tích hợp với nhiều ứng dụng khác nhau mà không cần xây dựng lại API riêng cho từng hệ thống, miễn là tuân thủ cùng tiêu chuẩn giao tiếp.

2.6.6 Nhược điểm của RESTful API

Chi phí triển khai:

Việc xây dựng, vận hành và bảo trì API có thể tốn nhiều nguồn lực, đặc biệt với các hệ thống quy mô lớn.

Nguy cơ bảo mật:

API bổ sung thêm một tầng trong kiến trúc hệ thống, khiến nguy cơ bị tấn công tăng lên. Việc hoạt động chủ yếu trên giao thức HTTP cũng khiến API dễ trở thành mục tiêu của các mối đe dọa bảo mật.

2.7 Tìm hiểu về mô hình ngôn ngữ lớn

2.7.1 Tổng quan hô hình ngôn ngữ lớn

Mô hình ngôn ngữ lớn (LLM) là một dạng mô hình ngôn ngữ được huấn luyện bằng học không giám sát (self-supervised learning) trên khối lượng lớn văn bản, nhằm phục vụ các nhiệm vụ xử lý ngôn ngữ tự nhiên, đặc biệt là sinh văn bản. Các LLM mạnh nhất và phổ biến nhất là các mô hình Generative Pretrained Transformers (GPTs), thường được sử dụng trong các chatbot sinh ngôn ngữ như ChatGPT, Gemini hay Claude. LLM có thể được điều chỉnh cho các nhiệm vụ cụ thể hoặc hướng dẫn thông qua prompt engineering. Những mô hình này nắm bắt được năng lực dự đoán cú pháp, ngữ nghĩa và cấu trúc kiến thức từ ngôn ngữ con người, nhưng đồng thời cũng kế thừa những sai lệch và bất chính xác từ dữ liệu huấn luyện [5].

2.7.2 Các khả năng nổi bật của LLM

- Sinh văn bản: LLM có thể tạo ra nội dung chất lượng cao ở nhiều thể loại khác nhau như bài báo, thơ ca, kịch bản hay thậm chí là mã nguồn lập trình.

- Tóm tắt thông tin**:** Với các văn bản dài, LLM có thể rút gọn thành các đoạn tóm tắt ngắn gọn, vẫn giữ nguyên ý chính và bối cảnh quan trọng.

- Dịch thuật: LLM hỗ trợ dịch từ ngôn ngữ này sang ngôn ngữ khác với độ chính xác cao, đồng thời đảm bảo câu văn mượt mà, tự nhiên.

- Trả lời câu hỏi: Bằng việc khai thác lượng dữ liệu khổng lồ đã học, LLM có thể phản hồi các câu hỏi từ đơn giản đến phức tạp một cách logic và có căn cứ.

- Giao tiếp tự nhiên: LLM có khả năng tham gia đối thoại mạch lạc với con người, đóng vai trò như chatbot thông minh hoặc trợ lý ảo.

2.7.3 Nguyên lý hoạt động

LLM chủ yếu được xây dựng trên kiến trúc Transformer – một loại mạng nơ-ron đặc biệt thích hợp cho dữ liệu tuần tự như văn bản. Cơ chế này giúp mô hình nắm bắt ngữ

cảnh của từ trong câu, từ đó tạo ra phản hồi và nội dung mạch lạc, phù hợp với ngữ nghĩa.

Các yếu tố ảnh hưởng đến hiệu suất LLM

- Quy mô mô hình: Các LLM thường sở hữu hàng tỷ tham số, cho phép chúng học và biểu diễn các mẫu ngôn ngữ phức tạp.

- Dữ liệu huấn luyện: Cả số lượng lẫn chất lượng dữ liệu đều ảnh hưởng trực tiếp đến khả năng và độ chính xác của mô hình.

- Kiến trúc: Mặc dù Transformer là kiến trúc phổ biến nhất, các nghiên cứu đang tiếp tục khám phá những cấu trúc mới nhằm cải thiện hiệu suất và tối ưu chi phí vận hành.

2.8 Tìm hiểu Gemini

2.8.1 Tổng quan Gemini

Gemini API là dịch vụ cung cấp quyền truy cập tới các mô hình AI hiện đại nhất thuộc dòng Gemini do Google DeepMind phát triển. Đây là mô hình AI đa phương thức (multimodal), có khả năng xử lý dữ liệu văn bản, hình ảnh, âm thanh, video và mã nguồn, nhằm hỗ trợ các ứng dụng tạo nội dung như tổng hợp văn bản, dịch vụ chat, phân loại dữ liệu, và truy vấn ngữ cảnh dài [6].

2.8.2 Điểm nổi bật

Đa nhiệm mạnh mẽ:

Gemini được tối ưu để xử lý song song nhiều tác vụ khác nhau trong cùng một phiên làm việc. Ví dụ, nó có thể vừa phân tích một đoạn văn bản dài, vừa dịch nó sang ngôn ngữ khác, đồng thời sinh hình minh họa cho nội dung. Điều này giúp tiết kiệm thời gian và nâng cao hiệu suất làm việc của người dùng.

Kiến thức toàn diện và sâu rộng:

Nhờ được huấn luyện trên tập dữ liệu cực kỳ lớn, bao gồm thông tin văn bản, hình ảnh và mã nguồn, Gemini có khả năng nắm bắt kiến thức đa lĩnh vực, từ khoa học, kỹ thuật đến văn hóa – xã hội. Nó không chỉ "biết" thông tin mà còn hiểu được mối quan hệ và ngữ cảnh giữa các khái niệm.

Khả năng thích nghi và học nhanh:

Gemini có thể tiếp nhận yêu cầu hoặc dữ liệu mới và nhanh chóng điều chỉnh để phù hợp. Chẳng hạn, khi được cung cấp một tập dữ liệu chuyên ngành, mô hình có thể nhanh chóng "làm quen" và tạo ra kết quả phù hợp với phong cách, thuật ngữ và tiêu chuẩn của lĩnh vực đó.

2.8.3 Ứng dụng tiêu biểu

- Tìm kiếm thông minh: Thay vì chỉ trả về danh sách kết quả, Gemini có thể tổng hợp và tóm tắt thông tin từ nhiều nguồn, phân tích ý định của người dùng, rồi đưa ra câu trả lời rõ ràng, kèm ví dụ minh họa nếu cần.

- Sáng tạo nội dung đa dạng: Từ việc viết bài blog, soạn kịch bản phim, tạo thơ văn cho tới thiết kế ý tưởng quảng cáo, Gemini có thể tạo nội dung mang tính sáng tạo cao, giữ đúng giọng điệu và mục tiêu mà người dùng yêu cầu.

- Hỗ trợ lập trình: Cung cấp gợi ý mã (code completion), phát hiện lỗi logic, tối ưu hiệu suất chương trình và thậm chí sinh toàn bộ module phần mềm dựa trên mô tả chức năng của người dùng.

- Tương tác và chăm sóc khách hàng: Tích hợp vào hệ thống hỗ trợ khách hàng để trả lời câu hỏi, hướng dẫn sử dụng sản phẩm, xử lý yêu cầu kỹ thuật hoặc thậm chí giải quyết khiếu nại một cách tự nhiên, không rập khuôn.

- Phân tích và đánh giá dữ liệu: Khả năng xử lý dữ liệu lớn cho phép Gemini phân tích xu hướng, phát hiện bất thường và đề xuất giải pháp chiến lược cho doanh nghiệp.

Ngoài ra, Gemini còn cung cấp API để tích hợp trực tiếp vào ứng dụng, cho phép khai thác khả năng phân tích và đánh giá chất lượng mã nguồn, tự động hóa quy trình dựa trên chuẩn AI.

2.8.4 Tích hợp API Gemini vào Backend ASP.NET Core API

Đăng ký và tạo API Key từ Google AI Studio

Để sử dụng mô hình Gemini do Google cung cấp, trước tiên người dùng cần tạo tài khoản Google và đăng ký API key thông qua Google AI Studio

Sau khi đăng ký và tạo dự án, một khoá API sẽ được cấp phát để sử dụng cho các yêu cầu HTTP đến các mô hình Gemini.

Tìm hiểu cấu trúc API Gemini

Google cung cấp API Gemini thông qua RESTful endpoint, hỗ trợ nhiều mô hình khác nhau (ví dụ: gemini-pro, gemini-1.5-pro, gemini-2.0-pro, gemini-2.0-flash). Tùy vào nhu cầu, bạn có thể chọn mô hình phù hợp:

- gemini-pro: cho các tác vụ văn bản thông thường.
- gemini-1.5-pro: hỗ trợ xử lý ngữ cảnh dài và dữ liệu đa modal.
- gemini-2.0-flash: mô hình nhẹ, tốc độ cao, tối ưu cho chi phí thấp.

Trong bối cảnh dự án này tô sử dụng mô hình gemini-2.0-flash

Quy trình tích hợp vào backend .NET

Việc tích hợp Gemini vào backend .NET thường bao gồm các bước sau:

Bước 1: Cấu hình khóa API

Lưu trữ khóa API trong file cấu hình bảo mật (appsettings.json, hoặc User Secrets) để đảm bảo an toàn. Khóa này sẽ được truyền trong Authorization header khi gọi API.

Bước 2. Gửi yêu cầu tới endpoint Gemini

Backend .NET sử dụng HttpClient để gửi yêu cầu POST tới endpoint Gemini API:

Endpoint chính:
https://generativelanguage.googleapis.com/v1/models/{model_name}:generateContent

Trong đó {model_name} là tên mô hình, ví dụ: gemini-2.0-flash.

Bước 3. Định dạng yêu cầu

Nội dung gửi yêu cầu phải theo cấu trúc JSON do Google định nghĩa, bao gồm:

Trường contents, mô tả nội dung đầu vào người dùng.

Cấu hình tùy chọn như temperature, topK, topP, maxOutputTokens.

Bước 4. Xử lý phản hồi

API trả về kết quả dạng JSON chứa phần nội dung được sinh bởi mô hình. Backend cần phân tích dữ liệu phản hồi để sử dụng trong hệ thống (ví dụ: hiển thị ra giao diện, lưu trữ vào DB, v.v.).

2.9 Tổng quan về một số công nghệ sử dụng khác

Frontend:

- *TypeScript*: Đây là một phần mở rộng của JavaScript được phát triển bởi Microsoft, bổ sung khả năng khai báo kiểu tĩnh. TypeScript cung cấp các công cụ như suy luận kiểu (type inference), định nghĩa giao diện (interface), liệt kê (enum), và lập trình tổng quát (generic), giúp lập trình viên phát hiện lỗi ngay trong quá trình biên dịch. Sử dụng TypeScript mang lại sự ổn định, dễ đọc và khả năng mở rộng cao cho dự án. Sau khi viết, mã TypeScript được chuyển đổi (transpile) thành JavaScript để có thể hoạt động trong trình duyệt hoặc môi trường Node.js [7].

- *Tailwind CSS*: Là một bộ công cụ CSS hướng "utility-first", Tailwind cho phép lập trình viên xây dựng giao diện bằng cách áp dụng trực tiếp các lớp tiện ích như flex, mt-4, text-center,... vào mã HTML. Phương pháp này giảm nhu cầu viết CSS riêng biệt, từ đó đẩy nhanh tiến độ phát triển và đảm bảo tính thống nhất. Tailwind hỗ trợ các tính năng như thiết kế responsive, trạng thái tương tác (hover, focus), chế độ tối (dark mode), và có thể tùy biến mạnh mẽ thông qua tệp cấu hình tailwind.config.js. Kể từ phiên bản 3, Tailwind sử dụng chế độ biên dịch Just-in-Time (JIT) nhằm tối ưu hiệu suất và giảm dung lượng file CSS đầu ra.

Backend:

- *JSON Web Token (JWT):* là một chuẩn mở (RFC 7519) được thiết kế để truyền tải thông tin giữa các thực thể dưới dạng đối tượng JSON một cách an toàn và nhỏ gọn. JWT thường được sử dụng trong các hệ thống xác thực và ủy quyền hiện đại, đặc biệt phổ biến trong các ứng dụng web sử dụng kiến trúc RESTful hoặc các dịch vụ vi mô (microservices). Một JWT bao gồm ba phần chính: Header, Payload, và Signature.

+ Header chứa thông tin về thuật toán ký (ví dụ: HS256 hoặc RS256) và loại token (thường là "JWT").
+ Payload chứa các thông tin (claims) mà hệ thống muốn truyền tải như userId, thời gian hiệu lực (exp), vai trò (role),...

+ Signature được tạo ra bằng cách mã hóa header và payload với một khóa bí mật (hoặc cặp khóa công khai - riêng tư), nhằm đảm bảo tính toàn vẹn và xác thực của token.

Đóng gói triển khai:

- Docker: là một nền tảng mã nguồn mở giúp tự động hoá việc triển khai, đóng gói và phân phối ứng dụng trong các container – một môi trường nhẹ, cô lập và có thể chạy được ở bất kỳ đâu. Không giống như máy ảo truyền thống, container của Docker chia sẻ cùng một nhân hệ điều hành, giúp giảm thiểu tài nguyên và tăng tốc độ khởi động. Docker cho phép các nhà phát triển đóng gói toàn bộ mã nguồn, thư viện phụ thuộc và cấu hình môi trường vào một image duy nhất. Từ image này, có thể tạo ra nhiều container hoạt động nhất quán trên nhiều môi trường khác nhau, từ máy phát triển đến máy chủ sản xuất. Ngoài ra, Docker còn cung cấp các công cụ hỗ trợ mạnh mẽ như Docker Compose (quản lý nhiều container cùng lúc), Docker Hub (lưu trữ và chia sẻ image), giúp đơn giản hóa quá trình phát triển và triển khai phần mềm theo hướng hiện đại, linh hoạt và dễ mở rộng.

CHƯƠNG 3. HIỆN THỰC HOÁ NGHIÊN CỨU

3.1 Mô tả bài toán

Hệ thống được xây dựng nhằm hỗ trợ người dùng tạo, quản lý và tối ưu hóa hồ sơ xin việc một cách nhanh chóng, trực quan và chuyên nghiệp. Ứng dụng tích hợp nhiều chức năng thông minh, từ thiết kế bằng giao diện kéo thả (drag-and-drop) cho đến đánh giá và gợi ý cải thiện nội dung dựa trên trí tuệ nhân tạo (AI).

Người dùng có thể đăng ký tài khoản bằng email và mật khẩu hoặc sử dụng hình thức đăng nhập nhanh thông qua tài khoản Google. Sau khi đăng nhập, giao diện sẽ hiển thị các chức năng chính của hệ thống.

Hệ thống cung cấp công cụ tạo hồ sơ xin việc với nhiều thành phần dựng sẵn như thông tin cá nhân, học vấn, kinh nghiệm làm việc, kỹ năng, chứng chỉ, giải thưởng, dự án,… Mỗi thành phần có thể được tùy chỉnh nội dung, bố cục và màu sắc, giúp người dùng dễ dàng tạo ra một hồ sơ xin việc mang dấu ấn cá nhân mà không cần kỹ năng thiết kế. Các hồ sơ xin việc đã tạo được lưu trữ trên hệ thống và có thể chỉnh sửa hoặc xóa bất cứ lúc nào.

Ngoài việc tạo mới, người dùng có thể tải lên file PDF hồ sơ xin việc sẵn có để hệ thống tự động trích xuất thông tin. AI sẽ phân tích và sắp xếp dữ liệu vào các mục tương ứng trên giao diện chỉnh sửa, giúp tiết kiệm thời gian nhập liệu thủ công. Sau khi hoàn thiện, có thể được xuất ra file PDF hoặc công khai thông qua một đường dẫn chia sẻ với tùy chọn hiển thị ở chế độ công khai hoặc riêng tư.

Hệ thống còn hỗ trợ dịch hồ sơ xin việc giữa các ngôn ngữ (tiếng Việt và tiếng Anh) với khả năng lựa chọn từ ngữ và diễn đạt mượt mà, phù hợp ngữ cảnh chuyên môn. Người dùng có thể tận dụng tính năng này để chuẩn bị hồ sơ ứng tuyển ở môi trường quốc tế.

Một trong những điểm nổi bật của hệ thống là chức năng đánh giá hồ sơ xin việc thông minh. Thông qua API Gemini, hệ thống phân tích và chấm điểm trên thang điểm 0–9, dựa trên các tiêu chí như mức độ phù hợp với mô tả công việc, độ đầy đủ thông tin, chất lượng ngôn ngữ, chính tả và sự xuất hiện của các từ khóa quan trọng. Khi người dùng cung cấp JD (Job Description), hệ thống sẽ so sánh và đánh giá mức độ tương thích

giữa hồ sơ xin việc và yêu cầu tuyển dụng, đồng thời đề xuất các cải thiện cụ thể để giúp hồ sơ trở nên nổi bật hơn.

Với những chức năng trên, hệ thống không chỉ đóng vai trò là công cụ thiết kế hồ sơ xin việc mà còn là một trợ lý cá nhân hỗ trợ người dùng tối ưu hóa hồ sơ xin việc, tăng cơ hội thành công trong quá trình ứng tuyển.

3.2 Phân tích yêu cầu hệ thống

Trước khi thiết kế, cần phân tích rõ những yêu cầu đặt ra đối với hệ thống hệ tạo và đánh giá hồ sơ xin việc. Yêu cầu được chia thành hai loại: yêu cầu chức năng (các tính năng mà hệ thống phải có) và yêu cầu phi chức năng (các tiêu chí về hiệu suất, trải nghiệm, giới hạn kỹ thuật).

3.2.1 Yêu cầu chức năng

Hệ thống cần đáp ứng các chức năng chính sau:

- *Tạo tài khoản người dùng*: Hệ thống cho phép người dùng thực hiện đăng ký và đăng nhập bằng email và mật khẩu, đặt biệt còn có hỗ trợ đặc nhập nhanh qua tài khoản Google, giúp đơn giản hóa quá trình xác thực và mang lại trải nghiệm sử dụng thuận tiện, linh hoạt.

- *Tạo hồ sơ xin việc*: Người dùng có thể tạo hồ sơ xin việc bằng cách sử dụng giao diện kéo thả (drag-and-drop) các thành phần có sẵn như: thông tin cá nhân, học vấn, kinh nghiệm làm việc, kỹ năng, chứng chỉ,… Các thành phần này có thể được tùy chỉnh nội dung bên trong, tỷ lệ bố cục một cách dễ dàng mà không yêu cầu người dùng có kỹ năng thiết kế.

- *Quản lý các hồ sơ xin việc đã tạo:* Hệ thống cho phép người dùng lưu trữ, chỉnh sửa, xóa hồ sơ đã tạo. Người dùng có thể cập nhật nội dung hoặc bố cục của hồ sơ bất cứ lúc nào nhằm đảm bảo thông tin luôn được duy trì chính xác và phù hợp.

- *Import hồ sơ xin việc từ file PDF*: Hệ thống cho phép người dùng tải lên file PDF hồ sơ xin việc hiện có. AI sẽ hỗ trợ trích xuất, tự động sắp xếp vào các mục tương ứng trên giao diện chỉnh sửa để người dùng có thể tiếp tục tùy chỉnh.

- *Xuất PDF hồ sơ xin việc*: Sau khi hoàn thiện, hồ sơ xin việc có thể được xuất ra dưới định dạng PDF để lưu trữ lại.

- *Công khai hồ sơ xin việc*: Người dùng có thể công khai hồ sơ của mình qua một đường dẫn (public link) để chia sẻ trên Internet. Hệ thống hỗ trợ cấu hình chế độ hiển thị (public/private)

- *Dịch hồ sơ sang các ngôn ngữ khác*: có thể dịch từ tiếng Việt sang tiếng Anh hoặc ngược lại. Nhờ khả năng hiểu ngữ cảnh và lựa chọn từ ngữ phù hợp của AI, bản dịch sẽ mượt mà, tự nhiên và giữ nguyên sắc thái chuyên môn, giúp hồ sơ trở nên chuyên nghiệp hơn so với các phương pháp dịch tự động thông thường.

- *Đánh giá hồ sơ sinh việc*: Một trong những điểm nổi bật của hệ thống là khả năng đánh giá hồ sơ xin việc thông qua tích hợp API Gemini – một công cụ AI mạnh mẽ. Hệ thống sẽ tiến hành phân tích và chấm điểm hồ sơ theo thang điểm từ 0 đến 9 dựa trên nhiều tiêu chí như: mức độ phù hợp với mô tả công việc, độ đầy đủ thông tin, từ ngữ sử dụng, chính tả và sự xuất hiện của các từ khóa quan trọng. Ngoài ra, người dùng có thể cung cấp thêm JD (Job Description) để hệ thống so sánh và đánh giá mức độ tương thích giữa hồ sơ xin việc và yêu cầu tuyển dụng. AI cũng sẽ đề xuất các cải thiện cụ thể giúp hồ sơ trở nên chuyên nghiệp và thu hút hơn.

3.2.2 Yêu cầu phi chức năng

Bên cạnh các chức năng, hệ thống cần thỏa mãn các tiêu chí phi chức năng sau:

- *Hiệu năng (Performance):* Hệ thống cần đảm bảo tốc độ phản hồi nhanh và ổn định ở cả frontend và backend. Các thao tác như tạo hồ sơ xin việc, chỉnh sửa, lưu trữ, xuất file PDF, hoặc gửi yêu cầu đánh giá cần được xử lý trong thời gian ngắn. Giao diện người dùng phải phản hồi mượt mà, không bị giật, trong khi backend phải xử lý dữ liệu hiệu quả và phản hồi kết quả một cách kịp thời. Hệ thống cũng cần đảm bảo hiệu năng ổn định khi có nhiều người dùng truy cập cùng lúc.

- *Tính thân thiện, dễ sử dụng*: Giao diện người dùng phải trực quan, đơn giản, dễ tiếp cận đối với cả những người không có kiến thức kỹ thuật chuyên sâu. Giao diện tạo hồ sơ xin việc nên sử dụng cơ chế kéo thả dễ hiểu. Phần đánh giá được bố trí rõ ràng và dễ hiểu, các nút bấm và điều hướng có nhãn rõ ràng, dễ nhận biết.

- *Khả năng mở rộng (Scalability):* Mặc dù hệ thống ban đầu được thiết kế để phục vụ cho một nhóm người dùng giới hạn (ví dụ như một nhóm sinh viên công nghệ thông tin), kiến trúc cần đảm bảo khả năng mở rộng trong tương lai để phục vụ số lượng người

dùng lớn hơn. Việc thiết kế theo hướng module hóa, tách biệt frontend, backend và các dịch vụ AI, giúp dễ dàng mở rộng quy mô hoặc thay thế các thành phần khi cần thiết.

- *Bảo mật và riêng tư*: Do có chức năng đăng nhập, hệ thống phải đảm bảo lưu trữ thông tin xác thực một cách an toàn, sử dụng các cơ chế mã hóa phù hợp. Khi tích hợp và gửi dữ liệu đến các API bên ngoài (ví dụ như Gemini), cần đảm bảo chỉ truyền các thông tin cần thiết, tránh rò rỉ thông tin nhạy cảm.

- *Dễ bảo trì:* Code của hệ thống cần rõ ràng, tài liệu đầy đủ để người khác có thể tiếp tục phát triển. Cấu trúc dự án phải được tổ chức khoa học. Cũng nên có log ghi lại các hoạt động quan trọng để dễ debug khi có sự cố.

3.2.3 Sơ đồ Use-case

Hệ thống có hai tác nhân: người dùng vãng lai, người dùng đăng nhập.

- Người dùng vãng lai: Người truy cập vào trang website chưa đăng nhập.

- Người dùng đăng nhập: Người truy cập vào trang website đã đăng nhập.

Đối với khách vãng lai có thể:

- Đăng ký, đăng nhập tài khoản.

- Xem trang chủ.

- Xem, tải xuống hồ sơ sinh việc công khai.

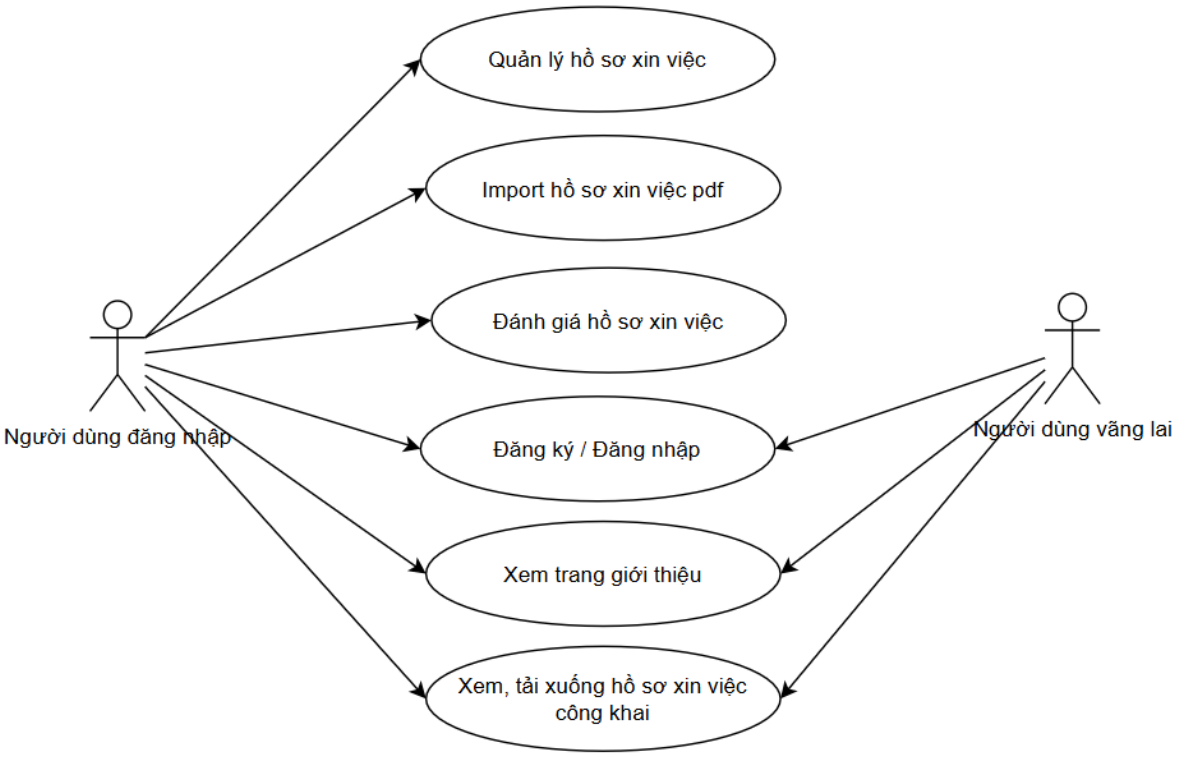
Đối với khách hàng đã đăng nhập hoặc đã đăng ký có thể:

- Có các chức năng của khách vãng lai.

- Quản lý hồ sơ xin việc (xem, thêm, xoá, sửa).

- Import hồ sơ xin việc pdf.

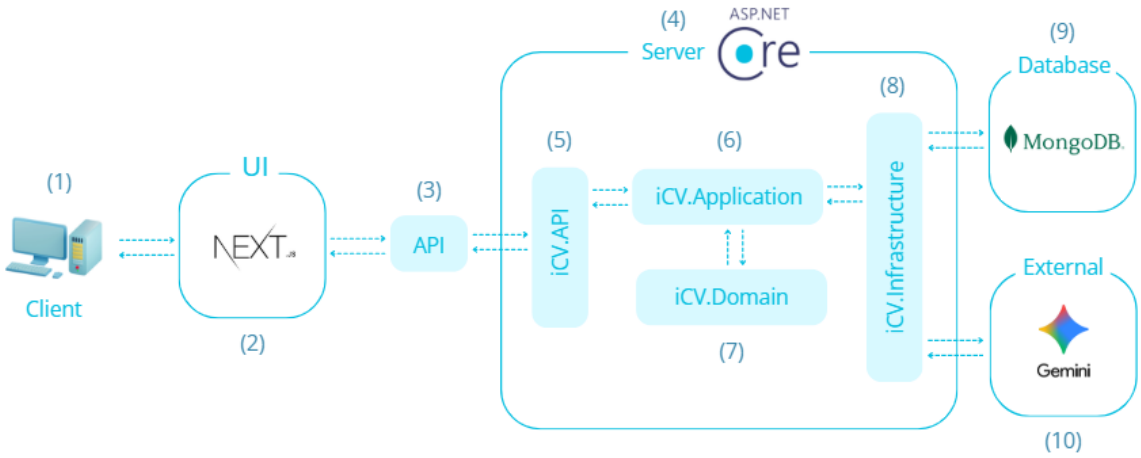
- Đánh giá hồ sơ xin việc.



Hình 3: Sơ đồ use-case của hệ thống

3.3 Kiến trúc hệ thống

Dựa trên các yêu cầu đã phân tích, dưới đây là sơ đồ chi tiết kiến trúc của hệ thống tạo và đánh giá hồ sơ xin việc tích hợp AI bao gồm các mô tả thành phần và luồng tương tác giữa chúng:



Hình 4: Sơ đồ kiến trúc hệ thống

Trong đó:

(1): Các trình duyệt của người dùng.

(2): Giao diện người dùng, nơi người dùng tương tác với hệ thống

(3): Các API từ server

(4): Máy chủ ứng dụng ASP.NET Core Web API xử lý các yêu cầu từ client.

(5): Đây là lớp chịu trách nhiệm cho các controller, middleware,… định tuyến và xử lý yêu cầu.

(6): Lớp chứa logic nghiệp vụ, các use case và luồng công việc đặc thù của ứng dụng.

(7): Lớp chứa các thực thể, logic nghiệp vụ và mô hình cốt lõi.

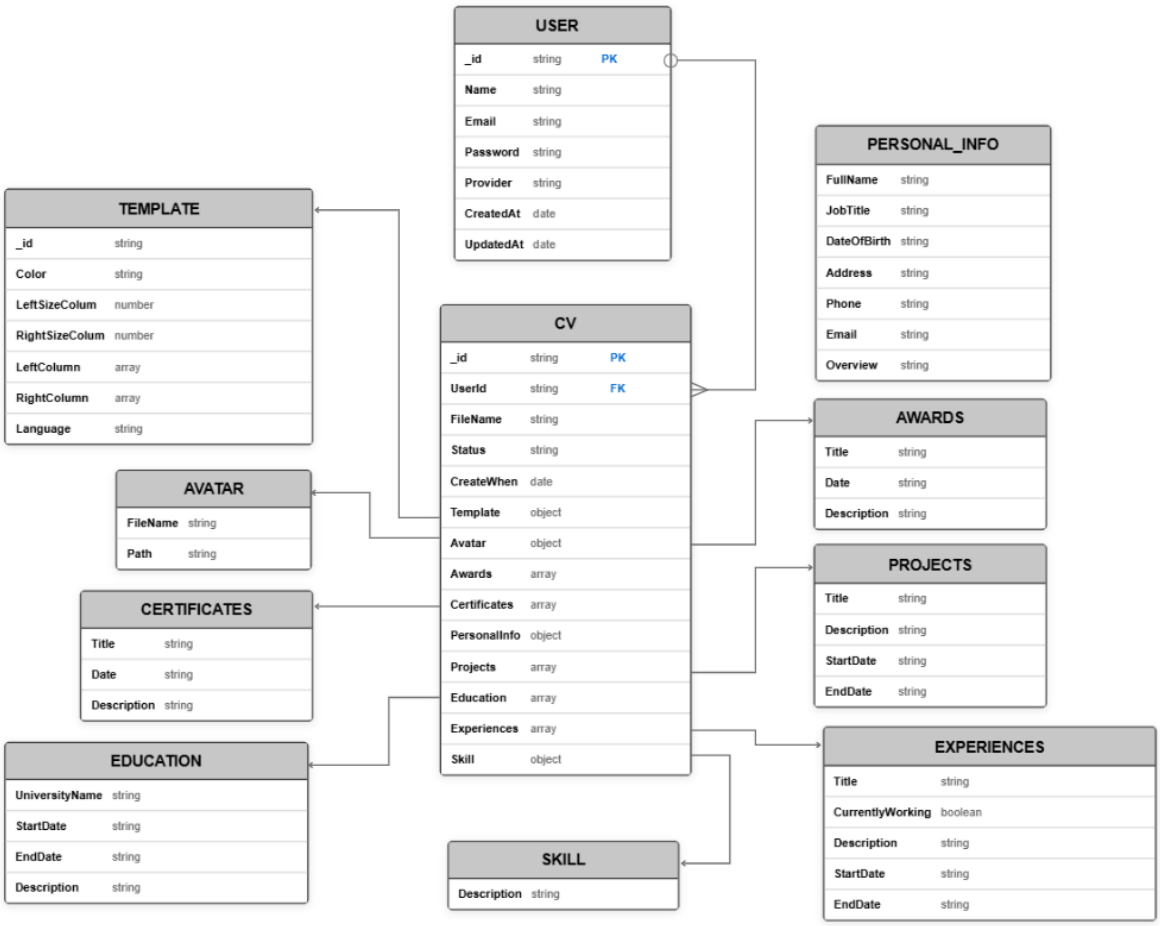
(8): Lớp này chứa các truy cập dữ liệu, dịch vụ và tích hợp với các hệ thống bên ngoài, chẳng hạn như giao tiếp với API của bên thứ ba và các thao tác cơ sở dữ liệu.

(9): Cơ sở dữ liệu nơi dữ liệu của ứng dụng được lưu trữ.

(10): Các dịch vụ bên ngoài như Gemini (thông qua api) mà ứng dụng tương tác, thông qua lớp Infrastructure.

3.4 Thiết kế dữ liệu

3.4.1 Lượt đồ cơ sở dữ liệu



Hình 5: Lược đồ cơ sở dữ liệu

3.4.2 Thiết kế cơ sở dữ liệu

a. Collection: user

Lưu trữ thông tin người dùng tham gia vào hệ thống.

Các trường chính:

- _id (ObjectId): Mã định danh duy nhất của mỗi người dùng.
- Name (String): Họ và tên đầy đủ của người dùng.
- Email (String): Địa chỉ email duy nhất, được sử dụng để đăng nhập vào hệ thống.
- Password (String): Mật khẩu được mã hóa.

- Provider (String): Phương thức đăng ký hoặc đăng nhập của người dùng, ví dụ "local" cho đăng ký trực tiếp hoặc "google"cho đăng nhập qua tài khoản google.
- CreatedAt (Date): Ngày tài khoản được tạo.
- UpdatedAt (Date): Ngày thông tin tài khoản được cập nhật.

b. Collection: CV

Lưu trữ thông tin hồ sơ xin việc vào hệ thống.

Các trường chính:

- _id (ObjectId): Mã định danh duy nhất cho mỗi CV.
- UserId (ObjectId/String): Mã định danh của người dùng sở hữu CV, liên kết với collection users.
- FileName (String): Tên định danh của CV (thường theo tên người dùng hoặc tiêu đề CV).
- Status (String): Trạng thái hiển thị của CV, ví dụ "public" (công khai) hoặc "private" (riêng tư).
- CreateWhen (Date): Ngày và giờ CV được tạo.
- Template (Object): Thông tin thiết kế và bố cục của CV.
  + _id (ObjectId/null): Mã định danh mẫu CV (nếu có).
  + Color (String): Màu chủ đạo của CV, định dạng mã HEX.
  + LeftSizeColum, RightSizeColum (Number): Tỉ lệ kích thước cột trái và cột phải trong bố cục CV (theo %).
  + LeftColumn (Array): Danh sách các khối nội dung ở cột trái (ví dụ: ảnh đại diện, tóm tắt, học vấn, kỹ năng, chứng chỉ).
  + RightColumn (Array): Danh sách các khối nội dung ở cột phải (ví dụ: danh thiếp, thông tin cá nhân, kinh nghiệm, dự án, giải thưởng).
  + Language (String): Ngôn ngữ hiển thị của CV, ví dụ "vi" cho tiếng Việt.
- Avatar (Object): Ảnh đại diện trên CV.

+ FileName (String): Tên file ảnh.
+ Path (String): Đường dẫn lưu trữ ảnh.

- Awards (Array): Danh sách giải thưởng đạt được. Mỗi phần tử bao gồm:
  + Title (String): Tên giải thưởng.
  + Date (String): Thời gian nhận giải.
  + Description (String): Mô tả chi tiết giải thưởng (cho phép HTML).
- Certificates (Array): Danh sách chứng chỉ. Mỗi phần tử bao gồm:
  + Title (String): Tên chứng chỉ.
  + Date (String): Thời gian cấp.
  + Description (String): Mô tả chi tiết (cho phép HTML).
- PersonalInfo (Object): Thông tin cá nhân trên CV.
  + FullName, JobTitle, DateOfBirth, Address, Phone, Email (String): Thông tin cá nhân cơ bản.
  + Overview (String): Tóm tắt mục tiêu và năng lực (cho phép HTML).
- Projects (Array): Danh sách dự án đã tham gia. Mỗi phần tử bao gồm:
  + Title (String): Tên dự án.
  + Description (String): Mô tả chi tiết, vị trí, trách nhiệm, công nghệ sử dụng (cho phép HTML).
  + StartDate, EndDate (String): Thời gian thực hiện.
- Education (Array): Danh sách quá trình học tập.
  + UniversityName (String): Tên trường.
  + StartDate, EndDate (String): Thời gian học.
  + Description (String): Mô tả ngành học hoặc thành tích (cho phép HTML).
- Experiences (Array): Danh sách kinh nghiệm làm việc.
  + Title (String): Tên công ty hoặc tổ chức.

+ CurrentlyWorking (Boolean): Đang làm việc tại đây hay không.
+ Description (String): Nội dung công việc, trách nhiệm, công nghệ sử dụng (cho phép HTML).
+ StartDate, EndDate (String): Thời gian làm việc.

- Skill (Object): Danh sách kỹ năng.
+ Description (String): Nội dung mô tả kỹ năng được phân loại theo nhóm (cho phép HTML).

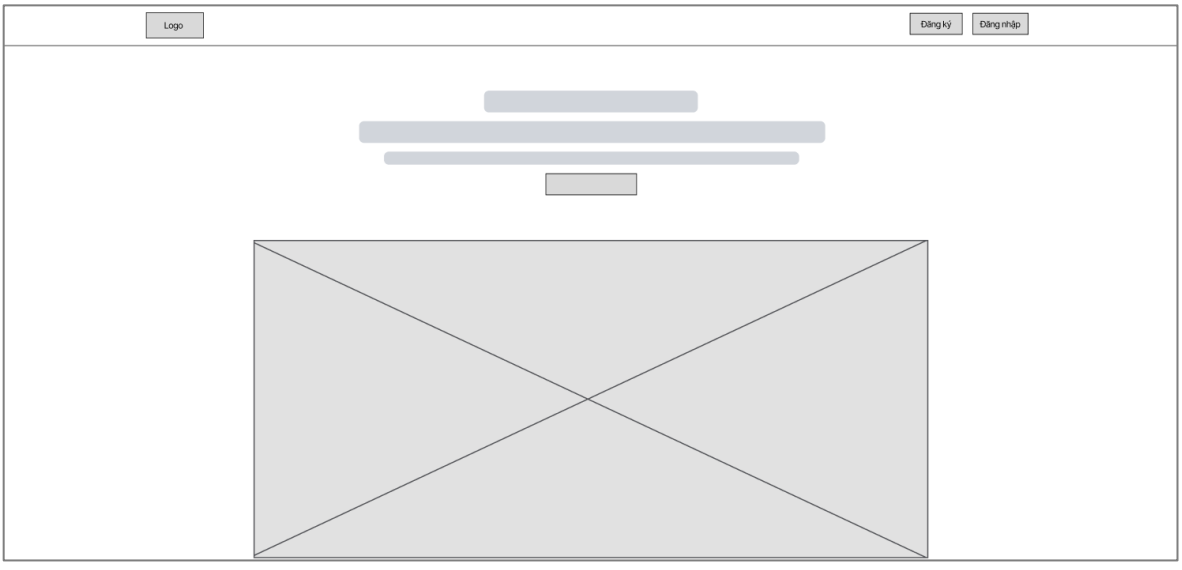
3.5 Thiết kế giao diện

3.5.1 Giao diện Giới thiệu

Giao diện Giới thiệu là trang đầu tiên xuất hiện khi người dùng truy cập hệ thống.

Phần header bao gồm logo của hệ thống cùng các nút Đăng nhập và Đăng ký để người dùng nhanh chóng truy cập hoặc tạo tài khoản.

Phần body chứa các nội dung tóm tắt về mục tiêu, chức năng chính của hệ thống và kèm theo hình ảnh minh họa giao diện, giúp người dùng hình dung được trải nghiệm khi sử dụng.



Hình 6: Bản phác thảo giao diện Giới thiệu

3.5.2 Giao diện Đăng nhập

Giao diện Đăng nhập cung cấp cho người dùng các lựa chọn để truy cập vào hệ thống. Phần đầu giao diện hiển thị văn bản hướng dẫn và các liên kết điều hướng cần thiết. Hai khu vực chính gồm:

- Đăng nhập bằng google: cho phép người dùng sử dụng tài khoản Google để đăng nhập nhanh chóng.

- Đăng nhập bằng email và mật khẩu: cho phép người dùng nhập thông tin tài khoản đã đăng ký để truy cập.

Hình 7: Bản phác thảo giao diện Đăng nhập

3.5.3 Giao diện Đăng ký

Giao diện Đăng ký hỗ trợ người dùng tạo tài khoản mới để sử dụng hệ thống. Phần đầu giao diện hiển thị văn bản hướng dẫn và các liên kết điều hướng. Hai khu vực chính bao gồm:

- Đăng ký bằng google: cho phép người dùng sử dụng tài khoản Google để tạo tài khoản nhanh chóng.

- Đăng ký bằng tên, email và mật khẩu: cho phép người dùng nhập thông tin cá nhân và mật khẩu để tạo tài khoản thủ công.

Hình 8: Bản phác thảo giao diện Đăng ký

3.5.4 Giao diện Quản lý hồ sơ xin việc

Giao diện Quản lý hồ sơ xin việc cho phép người dùng xem, quản lý và thao tác với các hồ sơ đã tạo.

Phần header bao gồm logo hệ thống, thông tin tài khoản người dùng và nút chuyển đổi chế độ sáng/tối. Khi nhấn vào thông tin tài khoản, hệ thống hiển thị modal với các chức năng liên quan đến quản lý tài khoản.

Phần body hiển thị nội dung mô tả chức năng và nút Import hồ sơ PDF để người dùng tải lên hồ sơ xin việc. Bên dưới là danh sách các thẻ hồ sơ xin việc của người dùng kèm các nút thao tác như thêm mới, chỉnh sửa hoặc xóa hồ sơ.

Hình 9: Bản phác thảo giao diện Quản lý hồ sơ xin việc

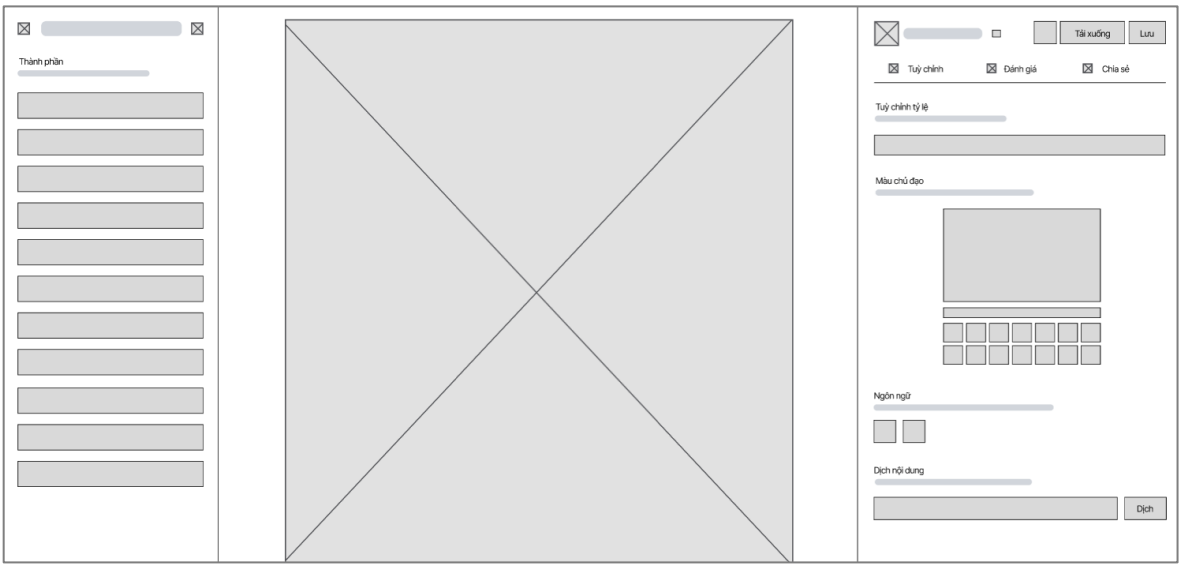
3.5.5 Giao diện Chỉnh sửa hồ sơ xin việc

Giao diện Chỉnh sửa hồ sơ xin việc được thiết kế trực quan, hỗ trợ người dùng dễ dàng thao tác và tùy chỉnh nội dung hồ sơ. Toàn bộ bố cục được chia thành ba khu vực chính: thanh bên trái, khung hồ sơ xin việc, và thanh bên phải, giúp sắp xếp chức năng rõ ràng và thuận tiện khi sử dụng.

- Thanh bên trái hiển thị thông tin định danh của hồ sơ (tên và trạng thái công khai hoặc riêng tư). Bên dưới là danh sách các thành phần của hồ sơ xin việc mà người dùng có thể kéo và thả vào khung hồ sơ.

- Khung hồ sơ xin việc là khu vực hiển thị nội dung chính của hồ sơ. Các thành phần được kéo từ thanh bên trái sẽ xuất hiện tại đây và có thể chỉnh sửa, xóa, hoặc thay đổi vị trí (lên, xuống, trái, phải) để sắp xếp bố cục.

- Thanh bên phải chứa thông tin người dùng cùng nút tải xuống và lưu ở phía trên. Bên dưới là ba thẻ điều hướng: tùy chỉnh, đánh giá và chia sẻ. Trong thẻ tùy chỉnh (mặc định hiển thị khi mở trang) có các tùy chọn như điều chỉnh tỷ lệ bố cục, màu chủ đạo, ngôn ngữ của các thành phần, và nút dịch toàn bộ nội dung hồ sơ sang ngôn ngữ khác.

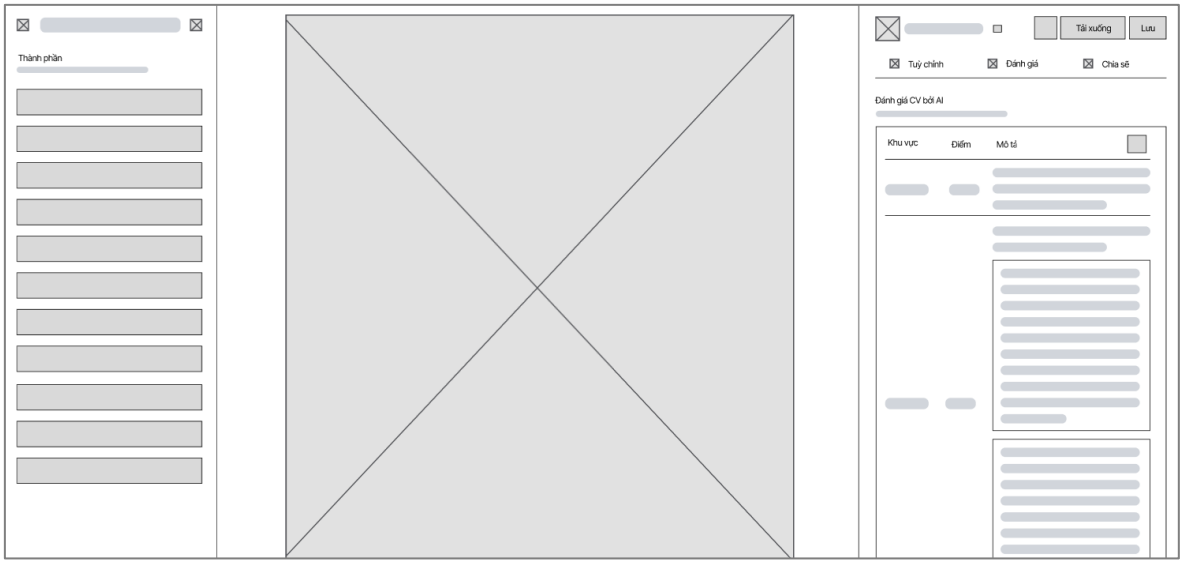


Hình 10: Bản phác thảo giao diện Chỉnh sửa hồ sơ xin việc

3.5.6 Giao diện Đánh giá hồ sơ xin việc

Giao diện Đánh giá hồ sơ xin sử dụnh chung với giao diện Chỉnh sửa hồ sơ xin việc, bao gồm thanh bên trái, khung hồ sơ xin việc ở giữa và thanh bên phải. Điểm khác biệt chính là nó là thẻ đánh trong ba thẻ đề cập ở giao diện Chỉnh sửa hồ sơ xin việc.

Bên trong thẻ Đánh giá có phần hướng dẫn cách thực hiện đánh giá, kèm nút tải lại đánh giá. Nội dung chính của thẻ là bảng đánh giá, gồm ba cột: khu vực, điểm và mô tả.



Hình 11: Bản phác thảo giao diện Đánh giá hồ sơ xin việc

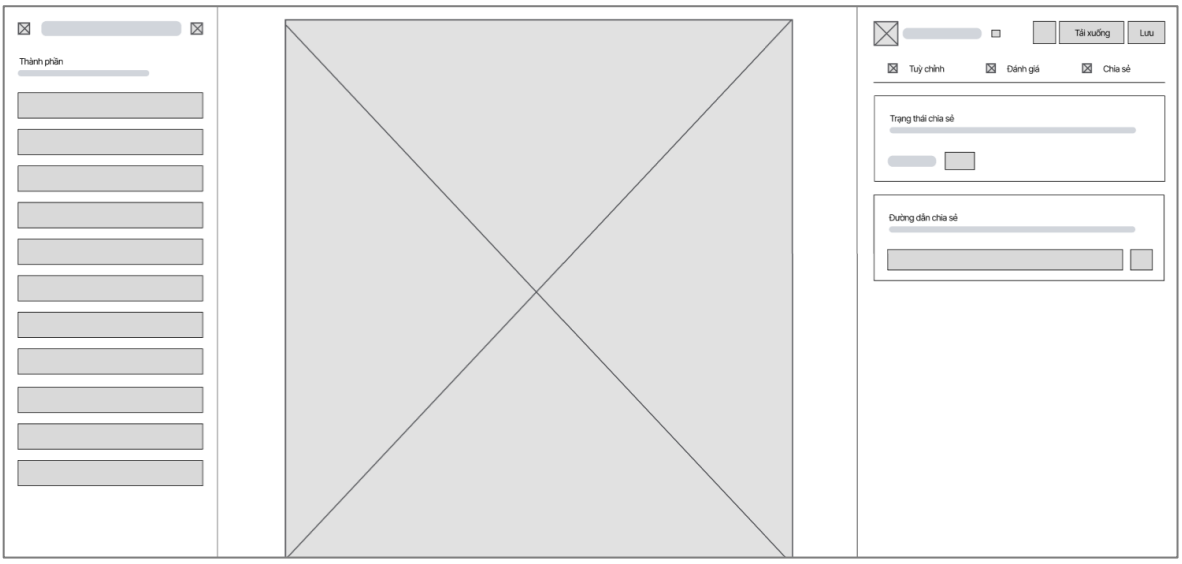
3.5.7 Giao diện Chỉnh sửa trạng thái hồ sơ xin việc

Giao diện Chỉnh sửa trạng thái hồ sơ xin việc sử dụng chung bố cục với giao diện Chỉnh sửa hồ sơ xin việc, gồm thanh bên trái, khung hồ sơ xin việc ở giữa và thanh bên phải. Điểm khác biệt chính là tại thanh bên phải, hệ thống hiển thị thẻ chia sẻ trong ba thẻ đã đề cập ở phần giao diện Chỉnh sửa hồ sơ xin việc.

Thẻ Chia sẻ được chia thành hai vùng: trạng thái chia sẻ và đường dẫn chia sẻ.

- Trạng thái chia sẻ: hiển thị thông tin hướng dẫn, trạng thái hiện tại của hồ sơ xin việc, cùng nút thay đổi trạng thái.

- Đường dẫn chia sẻ: chỉ xuất hiện khi trạng thái được đặt thành công khai, bao gồm liên kết công khai và nút sao chép.



Hình 12: Bản phác thảo giao diện Chỉnh sửa trạng thái hồ sơ xin việc

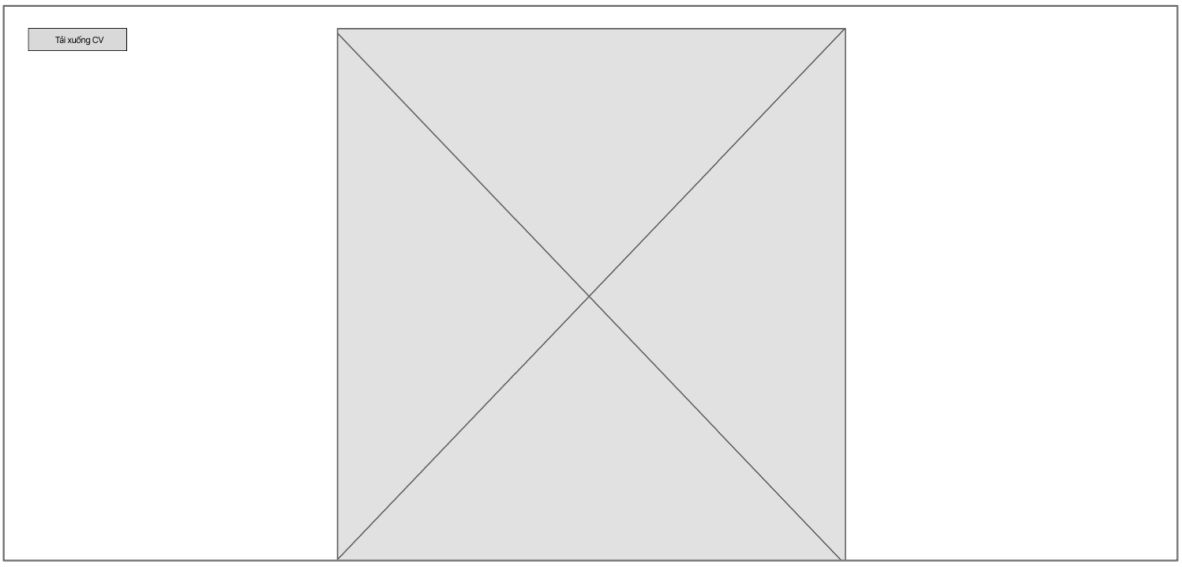
3.5.8 Giao diện Hồ sơ xin việc công khai

Giao diện Hồ sơ xin việc công khai được thiết kế đơn giản, tập trung vào việc hiển thị đầy đủ nội dung của hồ sơ xin việc đã được đặt ở trạng thái công khai. Thành phần chính gồm:

- Khu vực hiển thị hồ sơ: trình bày toàn bộ nội dung hồ sơ xin việc công khai.

- Nút Tải xuống: cho phép người xem tải hồ sơ dưới dạng tệp PDF.

Giao diện này chỉ khả dụng đối với những hồ sơ đã được chia sẻ ở chế độ công khai.



Hình 13: Bản phác thảo giao diện Hồ sơ xin việc công khai

3.6 Thiết kế API phía máy chủ (Back-end)

3.6.1 Các API xác thực

Bảng 1: Mô tả API đăng ký

Phương thức	POST	
Endpoint	{API_DOMAIN}api/auth/sign-up	
Thuộc tính	name	Họ và tên người dùng
	passWord	Mật khẩu đăng nhập.
	email	Địa chỉ email của người dung.
Diễn giải	API cho phép người dùng đăng ký tài khoản mới.	

Bảng 2: Mô tả API đăng nhập

Phương thức	POST	
Endpoint	{API_DOMAIN}api/auth/sign-in	
Thuộc tính	passWord	Mật khẩu đăng nhập.
	email	Địa chỉ email của người dung.
Diễn giải	API cho phép người dùng đăng nhập vào hệ thống.	

3.6.2 Các API hồ sơ xin việc

Bảng 3: Mô tả API lấy danh sách hồ sơ xin việc

Phương thức	GET
Endpoint	{API_DOMAIN}api/CV
Diễn giải	API cho phép lấy danh sách hồ sơ xin việc của người dung trên hệ thống.

Bảng 4: Mô tả API lấy thông tin chi tiết một hồ sơ xin việc

Phương thức	GET
Endpoint	{API_DOMAIN}api/CV/{id}

Diễn giải	API cho phép lấy thông tin chi tiết một hồ sơ xin việc của người dung trên hệ thống.

Bảng 5: Mô tả API tạo hồ sơ xin việc

Phương thức	POST	
Endpoint	{API_DOMAIN}api/CV	
Thuộc tính	userId	ID của người sở hữu.
	fileName	Tên tệp hồ sơ xin việc.
	status	Trạng thái hiển thị (public hoặc private).
	createWhen	Thời điểm tạo (timestamp).
	template	Cấu hình mẫu gồm màu sắc, bố cục cột trái/phải, danh sách các khối nội dung, ngôn ngữ hiển thị.
	avatar	Thông tin ảnh đại diện gồm tên tệp và đường dẫn ảnh.
	awards	Danh sách các giải thưởng, gồm tiêu đề, ngày nhận và mô tả.
	certificates	Danh sách chứng chỉ, gồm tiêu đề, ngày cấp và mô tả.
	personalInfo	Thông tin cá nhân: họ tên, chức danh, ngày sinh, địa chỉ, số điện thoại, email và phần giới thiệu bản thân.
	projects	Danh sách các dự án đã tham gia, gồm tiêu đề, mô tả chi tiết, vai trò, trách nhiệm, công nghệ sử dụng, thời gian bắt đầu và kết thúc.
	education	Thông tin học vấn gồm tên trường, thời gian học và mô tả.
	experiences	Danh sách kinh nghiệm làm việc, gồm tên công ty, trạng thái đang làm việc hay không, mô tả nhiệm vụ, thời gian bắt đầu và kết thúc.

	skill	Danh sách kinh nghiệm làm việc, gồm tên công ty, trạng thái đang làm việc hay không, mô tả nhiệm vụ, thời gian bắt đầu và kết thúc.
Diễn giải	API cho phép tạo mới hồ sơ xin việc cho người dùng.	

Bảng 6: Mô tả API cập nhật hồ sơ xin việc

Phương thức	PUT	
Endpoint	{API_DOMAIN}api/CV/{id}	
Thuộc tính	id	ID của hồ sơ xinh việc
	userId	ID của người sở hữu.
	fileName	Tên tệp.
	status	Trạng thái hiển thị (public hoặc private).
	createWhen	Thời điểm tạo (timestamp).
	template	Cấu hình mẫu gồm màu sắc, bố cục cột trái/phải, danh sách các khối nội dung, ngôn ngữ hiển thị.
	avatar	Thông tin ảnh đại diện gồm tên tệp và đường dẫn ảnh.
	awards	Danh sách các giải thưởng, gồm tiêu đề, ngày nhận và mô tả.
	certificates	Danh sách chứng chỉ, gồm tiêu đề, ngày cấp và mô tả.
	personalInfo	Thông tin cá nhân: họ tên, chức danh, ngày sinh, địa chỉ, số điện thoại, email và phần giới thiệu bản thân.
	projects	Danh sách các dự án đã tham gia, gồm tiêu đề, mô tả chi tiết, vai trò, trách nhiệm, công nghệ sử dụng, thời gian bắt đầu và kết thúc.
	education	Thông tin học vấn gồm tên trường, thời gian học và mô tả.

	experiences	Danh sách kinh nghiệm làm việc, gồm tên công ty, trạng thái đang làm việc hay không, mô tả nhiệm vụ, thời gian bắt đầu và kết thúc.
	skill	Danh sách kinh nghiệm làm việc, gồm tên công ty, trạng thái đang làm việc hay không, mô tả nhiệm vụ, thời gian bắt đầu và kết thúc.
Diễn giải	API cho sửa thông tin hồ sơ xin việc cho người dùng.	

Bảng 7: Mô tả API xoá một hồ sơ xin việc

Phương thức	DELETE
Endpoint	{API_DOMAIN}api/CV/{id}
Diễn giải	API cho phép xoá một hồ sơ xin việc trên hệ thống.

Bảng 8: Mô tả API đánh giá hồ sơ xin việc

Phương thức	GET
Endpoint	{API_DOMAIN}api/CV/{id}/evaluation
Diễn giải	API cho phép lấy thông tin đánh giá một hồ sơ xin việc.

Bảng 9: Mô tả API import hồ sơ xin việc PDF

Phương thức	POST	
Endpoint	{API_DOMAIN}api/CV/import-pdf	
Thuộc tính	pdfFile	File pdf cần import.
	userId	ID của người dùng.
Diễn giải	API cho phép import hồ sơ xin việc PDF vào hệ thống.	

Bảng 10: Mô tả API lấy hồ sơ xin việc công khai

Phương thức	GET
Endpoint	{API_DOMAIN}api/CV/{id}/public
Diễn giải	API cho phép lấy thông tin của hồ sơ xin việc công khai.

Bảng 11: Mô tả API dịch hồ sơ xin việc

Phương thức	POST	
Endpoint	{API_DOMAIN}api/CV/translate	
Thuộc tính	id	ID của hồ sơ xin việc.
	targetLanguage	Ngôn ngữ đích cần dịch.
Diễn giải	API cho phép dịch nội dung hồ sơ xin việc từ ngôn ngữ này sang ngôn ngữ khác (tiếng Anh và tiếng Việt)	

CHƯƠNG 4. CÀI ĐẶT HỆ THỐNG VÀ KẾT QUẢ THỬ NGHIỆM

4.1 Lựa chọn công nghệ

Bảng 12: Các công nghệ được lựa chọn sử dụng

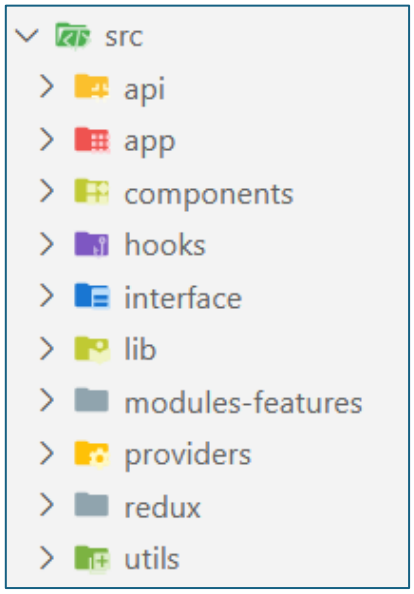
STT	Module	Công nghệ/Giải pháp	Diễn giải
1	Front-end	Next.js	Framework React hỗ trợ kết hợp giữa render phía máy chủ (SSR) và phía máy khách (CSR), tối ưu SEO và hiệu năng tải trang.
		TypeScript	Ngôn ngữ lập trình mở rộng từ JavaScript, bổ sung kiểu tĩnh, giúp code rõ ràng, dễ bảo trì và giảm lỗi runtime.
		TailwindCSS	Framework CSS tiện lợi, cho phép xây dựng giao diện nhanh chóng thông qua các class tiện ích.
		Mantine UI	Bộ thư viện UI React hiện đại với nhiều thành phần có sẵn, hỗ trợ tuỳ biến cao và tích hợp tốt với TypeScript.
		TanStack	Thư viện quản lý truy vấn dữ liệu bất đồng bộ trong React, hỗ trợ cache, tự động đồng bộ và tối ưu số lần gọi API.
		Axios	Thư viện JavaScript hỗ trợ gửi yêu cầu HTTP, dễ sử dụng, hỗ trợ Promise và xử lý

			request/response interceptor.
		dnd-kit	Thư viện kéo thả (drag-and-drop) cho React, hỗ trợ tương tác trực quan và tùy biến linh hoạt trong giao diện người dùng.
2	Back-end	ASP.NET Core Web API	Nền tảng phát triển ứng dụng web đa nền tảng của Microsoft, hiệu năng cao, bảo mật tốt, hỗ trợ xây dựng RESTful API dễ dàng và tích hợp tốt với cơ sở dữ liệu.
		Entity Framework Core	Framework ORM (Object–Relational Mapping) của Microsoft, giúp thao tác cơ sở dữ liệu bằng C# thay vì viết SQL trực tiếp.
		MediatR	Thư viện .NET triển khai mẫu thiết kế Mediator Pattern, giúp giảm sự phụ thuộc giữa các thành phần trong ứng dụng và tối ưu luồng xử lý request/response.
		JWT Bearer Authentication	Cơ chế xác thực dựa trên chuẩn JSON Web Token (JWT), giúp bảo mật API bằng cách mã hóa và xác minh thông tin người dùng trong mỗi yêu cầu (request).
3	Database	MongoDB	Hệ quản trị cơ sở dữ liệu

			NoSQL dạng tài liệu (document-based), lưu trữ dữ liệu dưới dạng BSON, linh hoạt và dễ mở rộng.
4	LLM	Gemini	Mô hình ngôn ngữ lớn được cung cấp miễn phí bởi Google, cho phép tương tác và tạo sinh thông qua Gemini API.

4.2 Xây dựng Frontend

Frontend sử dụng Next.js và được xây dựng với cấu trúc theo module với thư mục rõ ràng, phân tách hợp lý giữa các thành phần giao diện, logic xử lý và tiện ích, giúp dễ dàng quản lý, mở rộng và bảo trì. Cấu trúc được mô tả như sau:



Hình 14: Cấu trúc thư mục Frontend

api/ – Chứa các hàm, service hoặc cấu hình liên quan đến gọi API, bao gồm:

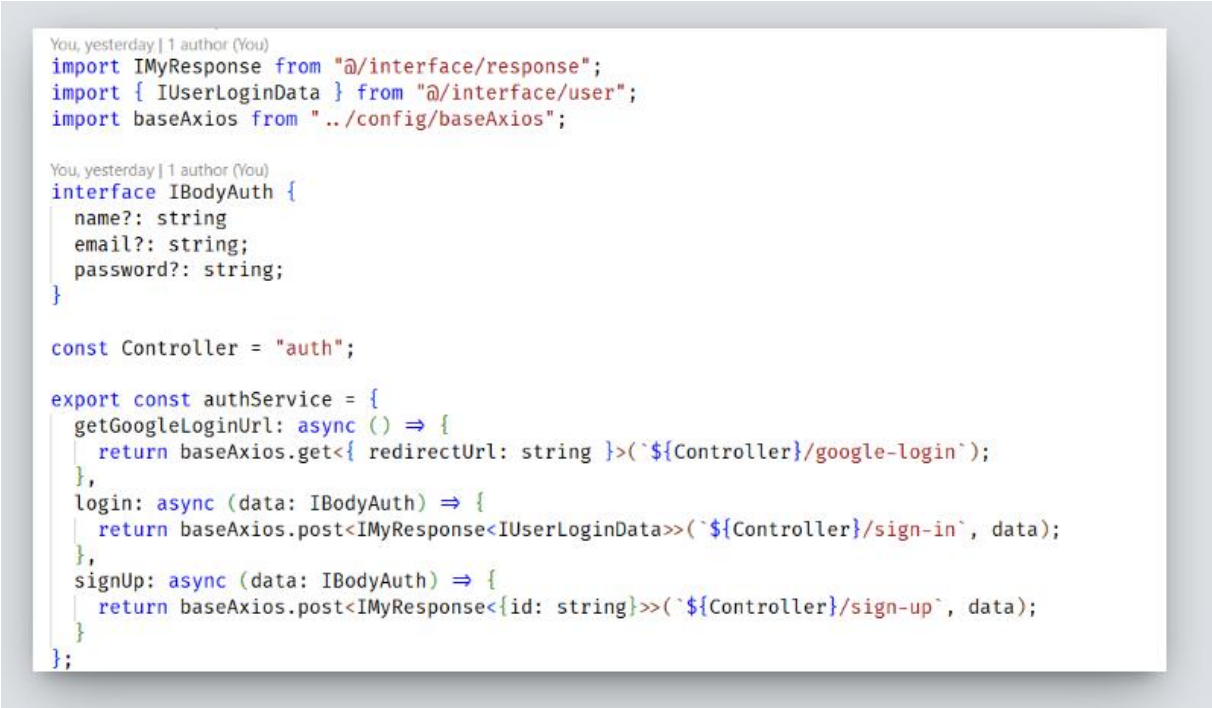
- *Tập tin baseAxios.js:* Chịu trách nhiệm xử lý các yêu cầu HTTP giữa client và server trong ứng dụng. Nó khởi tạo một instance Axios với baseURL được lấy từ biến môi trường, thiết lập header Content-Type: application/json và cho phép gửi kèm thông tin xác thực. Trước mỗi request, interceptor sẽ kiểm tra localStorage để lấy token người dùng và tự động thêm vào header Authorization dưới dạng Bearer Token. Nếu token

không tồn tại hoặc có lỗi xảy ra khi cấu hình request, yêu cầu sẽ bị từ chối. Cách tổ chức này giúp quản lý tập trung việc gửi token, giảm trùng lặp cấu hình cho từng request, đồng thời đảm bảo các yêu cầu API đều được xác thực và gửi dữ liệu đúng định dạng, từ đó nâng cao tính bảo mật và khả năng bảo trì của ứng dụng.

```
You, 2 hours ago | 1 author (You)
import axios from "axios";  62.8k (gzipped: 23.3k)

const baseAxios = axios.create({
  baseURL: `${process.env.NEXT_PUBLIC_API}/api/`,
  headers: {
    "Content-Type": "application/json",
  },
  withCredentials: true,
});

baseAxios.interceptors.request.use(
  (config) => {
    const tokenData = localStorage.getItem("authToken");
    if (tokenData) {
      config.headers.Authorization = `Bearer ${tokenData}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

export default baseAxios;
```

Hình 15: Cấu hình bên trong baseAxios.js

- *Các tập tin service:* Chịu trách nhiệm xử lý các tương tác giữa client và server trong ứng dụng thông qua các API endpoint. Mỗi service tập trung một nhóm chức năng riêng, sử dụng baseAxios để gửi request, truyền dữ liệu và nhận phản hồi từ server theo định dạng chuẩn hóa. Cách tổ chức này giúp quản lý logic nghiệp vụ tập trung, tái sử dụng code, đảm bảo xác thực và xử lý dữ liệu nhất quán, đồng thời nâng cao tính bảo mật, khả năng bảo trì và trải nghiệm người dùng.



```
You, yesterday | 1 author (You)
import IMyResponse from "@/interface/response";
import { IUserLoginData } from "@/interface/user";
import baseAxios from "../config/baseAxios";

You, yesterday | 1 author (You)
interface IBodyAuth {
  name?: string
  email?: string;
  password?: string;
}

const Controller = "auth";

export const authService = {
  getGoogleLoginUrl: async () => {
    return baseAxios.get<{ redirectUrl: string }>(`${Controller}/google-login`);
  },
  login: async (data: IBodyAuth) => {
    return baseAxios.post<IMyResponse<IUserLoginData>>(`${Controller}/sign-in`, data);
  },
  signUp: async (data: IBodyAuth) => {
    return baseAxios.post<IMyResponse<{id: string}>>(`${Controller}/sign-up`, data);
  }
};
```

Hình 16: Cấu hình bên trong authService

app/ – Thường chứa phần khởi tạo ứng dụng, cấu hình global, routes, layout chính cho ứng dụng.

- Mỗi thư mục con trong app/ tương ứng với một route trong ứng dụng, chịu trách nhiệm quản lý giao diện và logic của trang đó. Khi người dùng truy cập vào một đường dẫn nhất định, hệ thống sẽ tự động render các component bên trong thư mục tương ứng, bao gồm layout, thành phần giao diện và các hook liên quan.

- *Tập tin page.tsx:* trong mỗi thư mục con của app/ chịu trách nhiệm định nghĩa component chính của route đó. Đây là điểm bắt đầu để render giao diện khi người dùng truy cập vào đường dẫn tương ứng.

components/ – Các component dùng chung (UI component, button, modal,…).

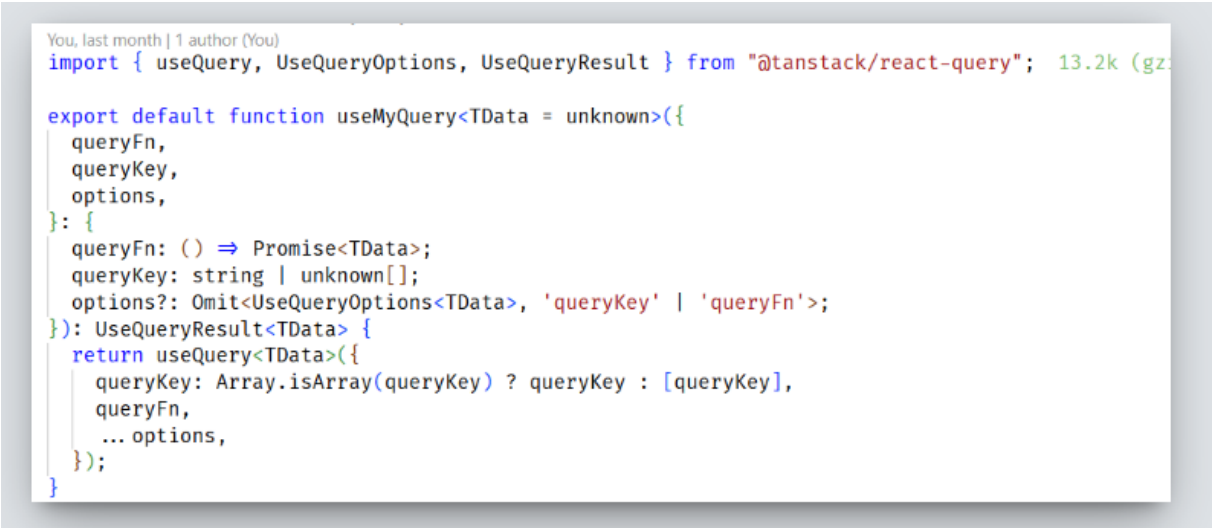
Các tập tin component dùng chung được thiết kế độc lập, có thể nhận props và callback để linh hoạt trong nhiều ngữ cảnh khác nhau. Cách tổ chức này giúp giảm trùng lặp code, đảm bảo sự nhất quán về giao diện và trải nghiệm người dùng, đồng thời dễ dàng bảo trì và mở rộng khi thêm các tính năng mới.

```
You, 3 months ago | 1 author (You)
import {
  ActionIcon,
  useComputedColorScheme,
  useMantineColorScheme,
} from "@mantine/core";  31.5k (gzipped: 9.6k)
import { IconMoon, IconSun } from "@tabler/icons-react";

export function MySwitchTheme() {
  const { setColorScheme } = useMantineColorScheme();
  const computedColorScheme = useComputedColorScheme("light", {
    getInitialValueInEffect: true,
  });

  return (
    <ActionIcon
      onClick={() =>
        setColorScheme(computedColorScheme === "light" ? "dark" : "light")
      }
      variant="default"
      size="lg"
      radius={"md"}
      aria-label="Toggle color scheme"
    >
      <IconSun
        stroke={1.5}
        style={{ display: computedColorScheme === "light" ? "block" : "none" }}
      />
      <IconMoon
        stroke={1.5}
        style={{ display: computedColorScheme === "dark" ? "block" : "none" }}
      />
    </ActionIcon>
  );
}
```

Hình 17: Component dùng chung MySwitchTheme

hooks/ – Các custom React hooks để tái sử dụng logic

Mỗi tập tin là custom React hooks được viết riêng nhằm tái sử dụng logic trong toàn bộ ứng dụng. Mỗi hook thường đóng gói một chức năng cụ thể, ví dụ như gọi API, quản lý trạng thái, hoặc xử lý dữ liệu, giúp tách biệt logic khỏi component giao diện



```
You, last month | 1 author (You)
import { useQuery, UseQueryOptions, UseQueryResult } from "@tanstack/react-query";  13.2k (gz

export default function useMyQuery<TData = unknown>({
  queryFn,
  queryKey,
  options,
}: {
  queryFn: () => Promise<TData>;
  queryKey: string | unknown[];
  options?: Omit<UseQueryOptions<TData>, 'queryKey' | 'queryFn'>;
}): UseQueryResult<TData> {
  return useQuery<TData>({
    queryKey: Array.isArray(queryKey) ? queryKey : [queryKey],
    queryFn,
    ...options,
  });
}
```

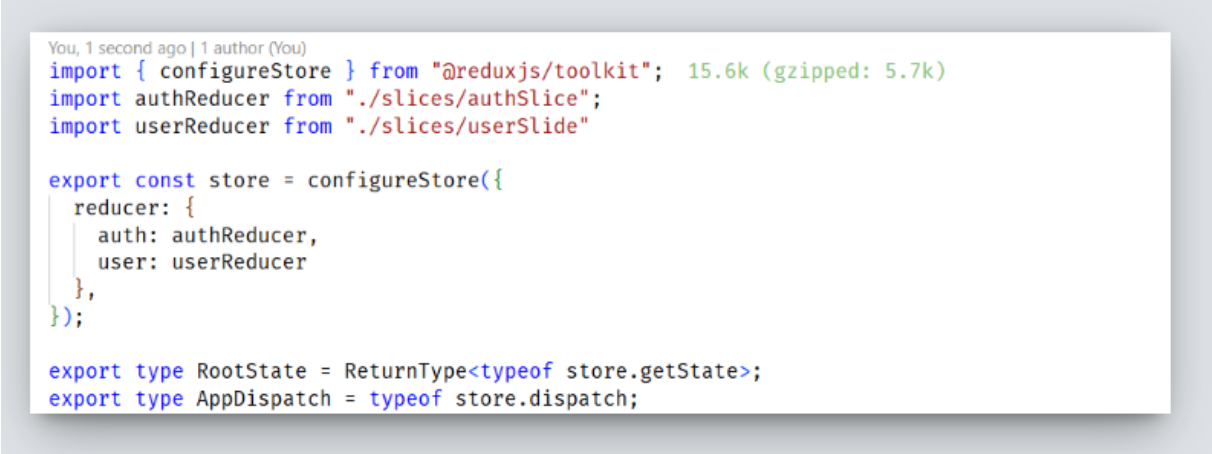
Hình 18: Hook useMyQuery

interface/ – Khai báo interface/type TypeScript cho dữ liệu và props.

lib/ – Các thư viện helper hoặc module hỗ trợ, có thể chứa logic không thuộc riêng module nào (vd: format date, parse data).

modules-features/ – Chứa các tính năng module lớn của app, mỗi module thường gồm component, service, logic riêng.

Mỗi module được tổ chức độc lập với component, gọi service và logic riêng. Bên trong mỗi module thường bao gồm các component chuyên biệt để hiển thị giao diện, các service xử lý nghiệp vụ và gọi API, cùng các custom hook hoặc helper function hỗ trợ quản lý trạng thái, sự kiện và dữ liệu riêng của module. Một số module còn kèm theo style hoặc asset đặc thù phục vụ cho giao diện.



Hình 19: Cấu hình store của Redux

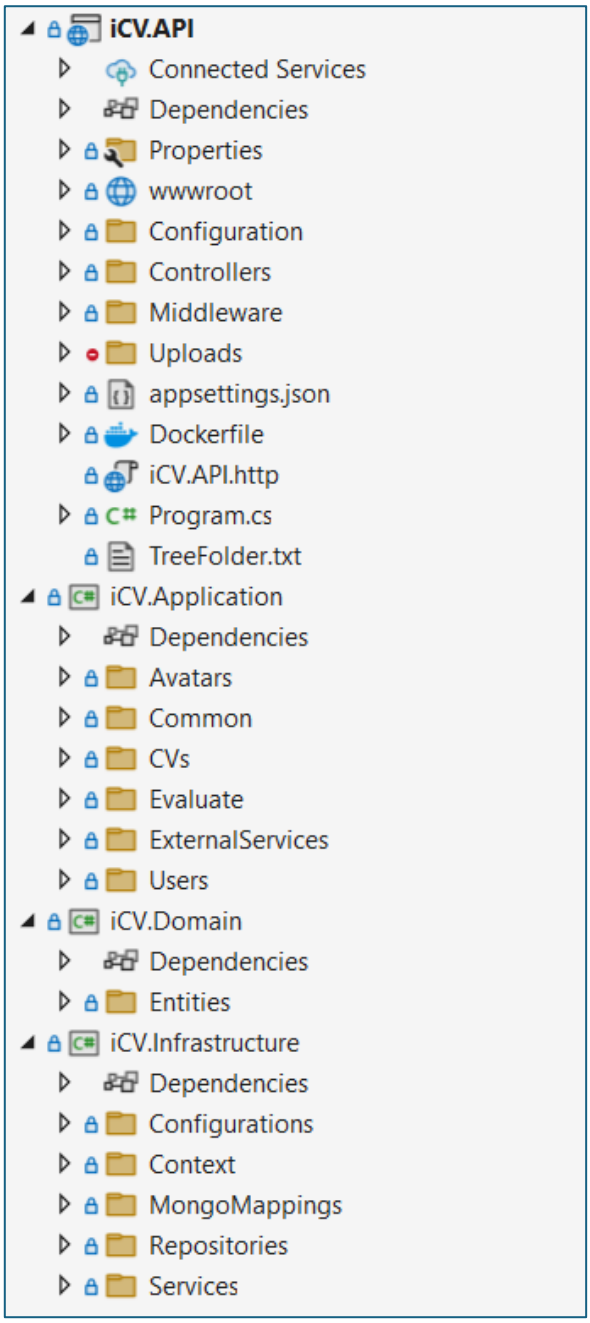
utils/ – Các hàm tiện ích (utility functions) dùng chung toàn dự án.

providers/ – Các React Context Providers hoặc dependency injection để cung cấp dữ liệu/config cho toàn bộ app.

redux/ – Cấu hình store Redux, slice, action, reducer.

4.3 Xây dựng Backend

Backend được xây dựng bằng ASP.NET Core Web API, áp dụng mô hình Clean Architecture nhằm đảm bảo tính tách biệt giữa các tầng, dễ bảo trì và mở rộng. Kiến trúc gồm bốn lớp chính được mô tả như sau:



Hình 20: Cấu trúc thư mục Backend

Domain Layer (iCV.Domain)

Domain Layer là lớp nội tâm nhất của hệ thống, chứa các thực thể cốt lõi (entities) như User, CV, Block, Education, Experience, và Skill. Lớp này chịu trách nhiệm định nghĩa cấu trúc dữ liệu và logic nghiệp vụ trung tâm, đảm bảo tính toàn vẹn của các quy tắc nghiệp vụ. Đặc điểm nổi bật là Domain Layer hoàn toàn độc lập, không phụ thuộc vào bất kỳ lớp nào khác, giúp duy trì tính thuần khiết của mô hình miền (Domain Model).

Ví dụ: Đây là một ví dụ về một entity đơn giản trong tầng Domain, đại diện cho một khối CV, chỉ chứa các thuộc tính mà không có bất kỳ phụ thuộc nào vào framework hay các tầng khác.

```
namespace iCV.Domain.Entities
{
    public class CV
    {
        public string Id { get; set; }

        public string? UserId { get; set; }
        5 references
        public string? FileName { get; set; }
        4 references
        public string? Status { get; set; }
        6 references
        public DateTime? CreateWhen { get; set; }
        6 references
        public Template? Template { get; set; }
        7 references
        public Avatar? Avatar { get; set; }
        6 references
        public List<Award>? Awards { get; set; }
        6 references
        public List<Certificate>? Certificates { get; set; }
        6 references
        public PersonalInfo? PersonalInfo { get; set; }
        6 references
        public List<Project>? Projects { get; set; }
        6 references
        public List<Education>? Education { get; set; }
        6 references
        public List<Experience>? Experiences { get; set; }
        6 references
        public Skill? Skill { get; set; }
    }
}
```

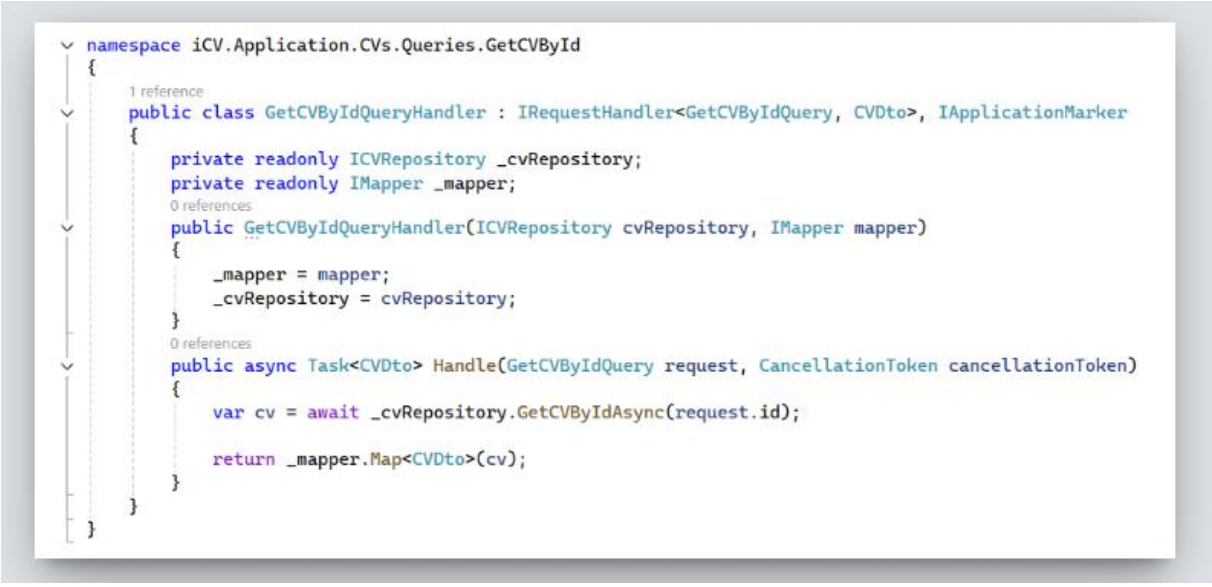
Hình 21: Entity CV trong tầng Domain

Application Layer (iCV.Application)

Application Layer triển khai các use case của ứng dụng thông qua mô hình CQRS (Commands/Queries). Lớp này định nghĩa các giao diện (interfaces) cho repository và service, đồng thời thực hiện xử lý logic nghiệp vụ và kiểm tra dữ liệu (validation). Application Layer sử dụng thư viện MediatR để hiện thực pattern CQRS và chứa các DTO (Data Transfer Objects). Lớp này chỉ phụ thuộc vào Domain Layer, giúp tách biệt logic nghiệp vụ với các lớp hạ tầng.

Vi dụ: Đoạn code này minh họa một query handler trong mô hình CQRS, xử lý use case lấy danh sách hồ sơ xin việc. Handler sử dụng interface ICVRepository (được định

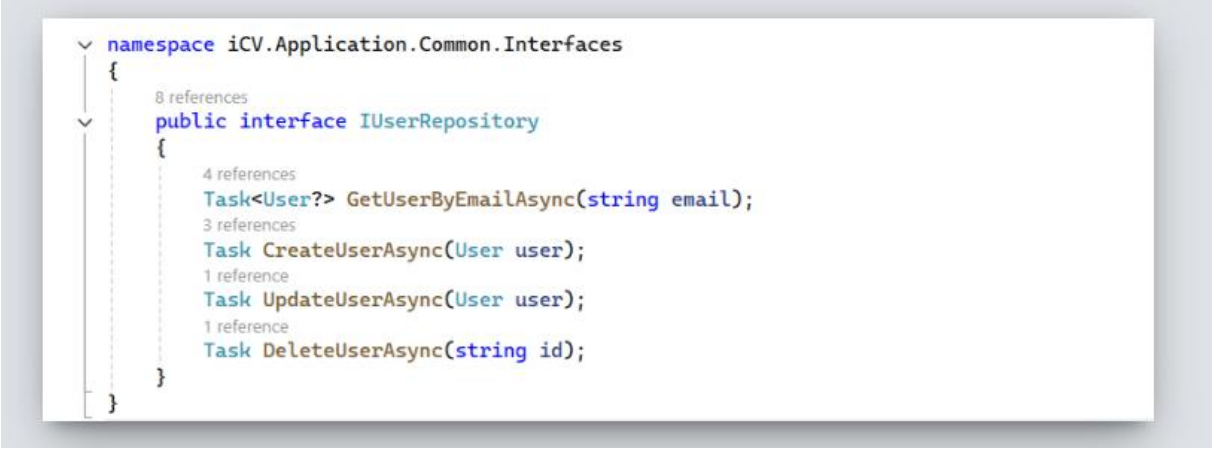
nghĩa trong tầng Application) để tương tác với cơ sở dữ liệu mà không cần biết chi tiết triển khai.



```
namespace iCV.Application.CVs.Queries.GetCVById
{
    1 reference
    public class GetCVByIdQueryHandler : IRequestHandler<GetCVByIdQuery, CVDto>, IApplicationMarker
    {
        private readonly ICVRepository _cvRepository;
        private readonly IMapper _mapper;
        0 references
        public GetCVByIdQueryHandler(ICVRepository cvRepository, IMapper mapper)
        {
            _mapper = mapper;
            _cvRepository = cvRepository;
        }
        0 references
        public async Task<CVDto> Handle(GetCVByIdQuery request, CancellationToken cancellationToken)
        {
            var cv = await _cvRepository.GetCVByIdAsync(request.id);

            return _mapper.Map<CVDto>(cv);
        }
    }
}
```

Hình 22: Một handler xử lý lấy danh sách hồ sơ xin việc

Ví dụ: Interface này định nghĩa các phương thức để thao tác với thực thể User, cho phép tầng Application tương tác với dữ liệu người dùng mà không cần biết chi tiết về cách lưu trữ (có thể là MongoDB, SQL Server, hay bất kỳ cơ sở dữ liệu nào khác). Và tầng Infrastructure sẽ triển khai theo interface này.



```
namespace iCV.Application.Common.Interfaces
{
    8 references
    public interface IUserRepository
    {
        4 references
        Task<User?> GetUserByEmailAsync(string email);
        3 references
        Task CreateUserAsync(User user);
        1 reference
        Task UpdateUserAsync(User user);
        1 reference
        Task DeleteUserAsync(string id);
    }
}
```

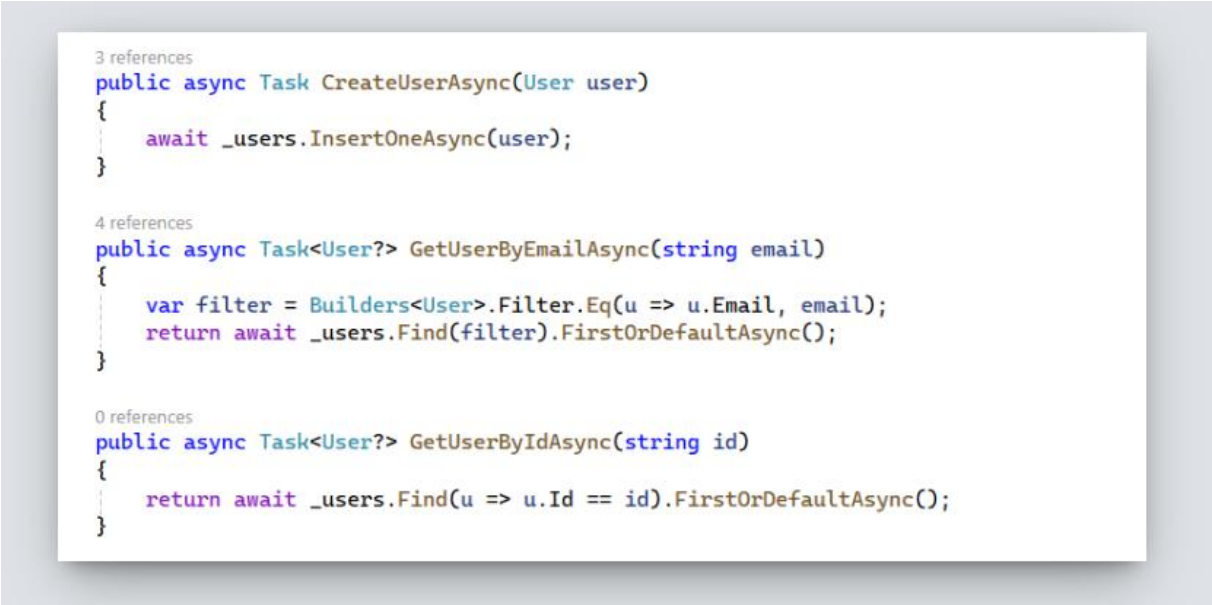
Hình 23: Interface IUserRepository của tầng Application

Infrastructure Layer (iCV.Infrastructure)

Infrastructure Layer đảm nhận việc triển khai các giao diện được định nghĩa trong Application Layer. Lớp này tương tác trực tiếp với cơ sở dữ liệu MongoDB, tích hợp với các dịch vụ bên ngoài như Gemini AI, và xử lý các thao tác với file.

Bên trong chứa các repository cụ thể và các dịch vụ như GeminiEvaluationService và PdfCVImportService. Infrastructure Layer phụ thuộc vào cả Application Layer và Domain Layer, đóng vai trò là cầu nối giữa logic nghiệp vụ và các công nghệ, dịch vụ thực thi.

Ví dụ: Đây là minh về một UserRepository trong tầng Infrastructure triển khai interface IUserRepository từ tầng Application. Các hàm trong repository này dùng để tương tác với cơ sở dữ liệu.



```
3 references
public async Task CreateUserAsync(User user)
{
    await _users.InsertOneAsync(user);
}

4 references
public async Task<User?> GetUserByEmailAsync(string email)
{
    var filter = Builders<User>.Filter.Eq(u => u.Email, email);
    return await _users.Find(filter).FirstOrDefaultAsync();
}

0 references
public async Task<User?> GetUserByIdAsync(string id)
{
    return await _users.Find(u => u.Id == id).FirstOrDefaultAsync();
}
```

Hình 24: Một số hàm trong User Repository

Presentation Layer (iCV.API)

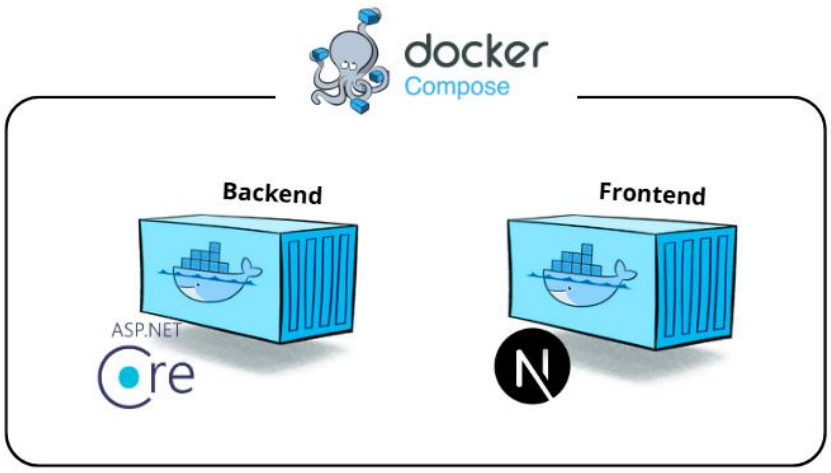
Presentation Layer xử lý tương tác với người dùng thông qua Web API. Lớp này cung cấp các endpoint để nhận và phản hồi các request, thực hiện cấu hình dependency injection và cơ chế xác thực (authentication). Presentation Layer được xây dựng bằng ASP.NET Core, tích hợp Swagger để hỗ trợ kiểm thử và tài liệu hoá API. Lớp này phụ thuộc vào Application Layer và Infrastructure Layer, đảm bảo luồng giao tiếp giữa người dùng và hệ thống được thực hiện trơn tru, bảo mật và hiệu quả.

Ví dụ: Controller này cung cấp endpoints để lấy danh sách CV sử dụng MediatR để gửi commands/queries đến các handlers tương ứng trong tầng Application, xử lý kết quả và trả về response phù hợp cho client.

```
[Authorize]
[HttpGet]
0 references
public async Task<IActionResult> GetCVs()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (string.IsNullOrEmpty(userIdClaim)) return Unauthorized("Missing user ID in token.");
    var query = new GetCVsQuery { UserId = userIdClaim };
    var result = await _mediator.Send(query);
    return Ok(new
    {
        isSuccess = true,
        message = "CVs retrieved successfully.",
        data = result
    }
    );
}
```

Hình 25: Hàm lấy danh sách hồ sơ xin việc trong CV Controller

4.4 Đóng gói triển khai



Hình 26: Đóng gói hệ thống bằng Docker Compose

Trong dự án, hệ thống bao gồm Frontend được phát triển bằng Next.js và Backend được xây dựng bằng ASP.NET Core Web API. Để đơn giản hóa quá trình triển khai, đảm bảo tính nhất quán giữa các môi trường (phát triển, kiểm thử, sản xuất) và nâng cao khả năng tự động hóa, công nghệ Docker và Docker Compose đã được nghiên cứu và áp dụng.

Sử dụng Docker

Docker cho phép đóng gói toàn bộ ứng dụng cùng các thư viện và cấu hình cần thiết vào một image duy nhất. Điều này đảm bảo rằng ứng dụng có thể chạy ổn định trên mọi máy chủ có cài đặt Docker, bất kể sự khác biệt về hệ điều hành hoặc môi trường cài đặt.

- Backend (.NET): Một tập tin Dockerfile được xây dựng để biên dịch và đóng gói ứng dụng ASP.NET Core Web API. Image được tạo bao gồm runtime của .NET và toàn bộ các thư viện phụ thuộc, đảm bảo backend có thể khởi chạy ngay lập tức trong container mà không cần cài đặt bổ sung.

Frontend (Next.js): Một Dockerfile riêng được xây dựng để thực hiện quá trình build ứng dụng Next.js sang mã tĩnh hoặc mã sẵn sàng chạy server-side rendering.

Sử dụng Docker Compose

Docker Compose được sử dụng để quản lý và khởi chạy nhiều container của hệ thống một cách đồng bộ. Một tập tin docker-compose.yml được định nghĩa để mô tả toàn bộ các service trong hệ thống, bao gồm:

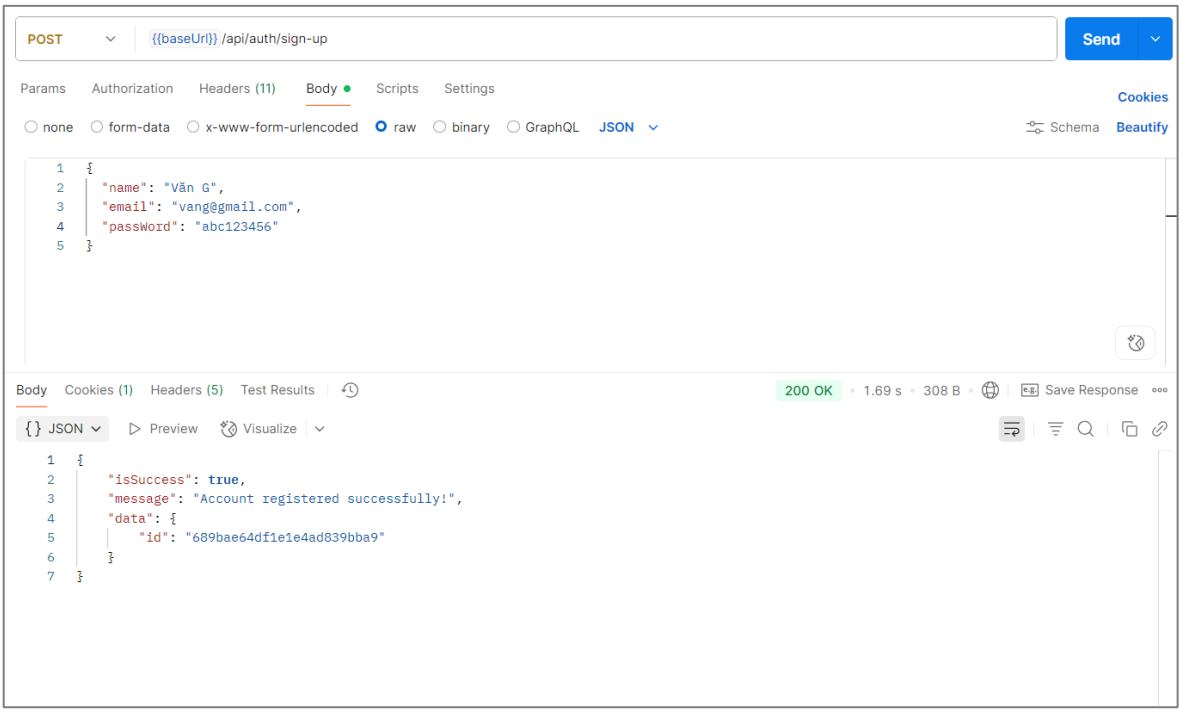
- Service Backend: Chạy container từ image ASP.NET Core Web API đã được build, đảm nhận xử lý logic nghiệp vụ và cung cấp API cho frontend.

- Service Frontend: Chạy container từ image Next.js đã được build, đảm nhận hiển thị giao diện và tương tác với người dùng cuối.

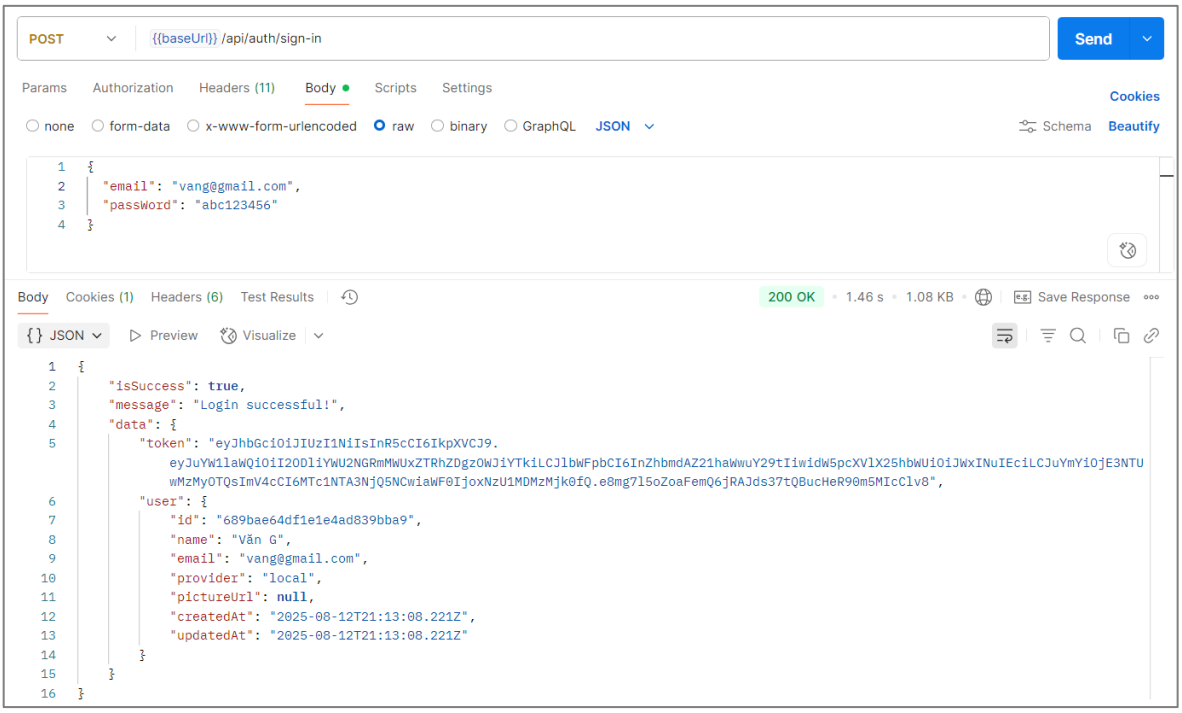
Thông qua Docker Compose, các container được kết nối bằng một network nội bộ, cho phép các service giao tiếp với nhau bằng tên dịch vụ thay vì địa chỉ IP cố định. Ngoài ra, các biến môi trường quan trọng cũng được định nghĩa tập trung, giúp dễ dàng thay đổi mà không cần chỉnh sửa mã nguồn.

4.5 Kết quả thử nhiệm API

4.5.1 Các API liên quan xác thực

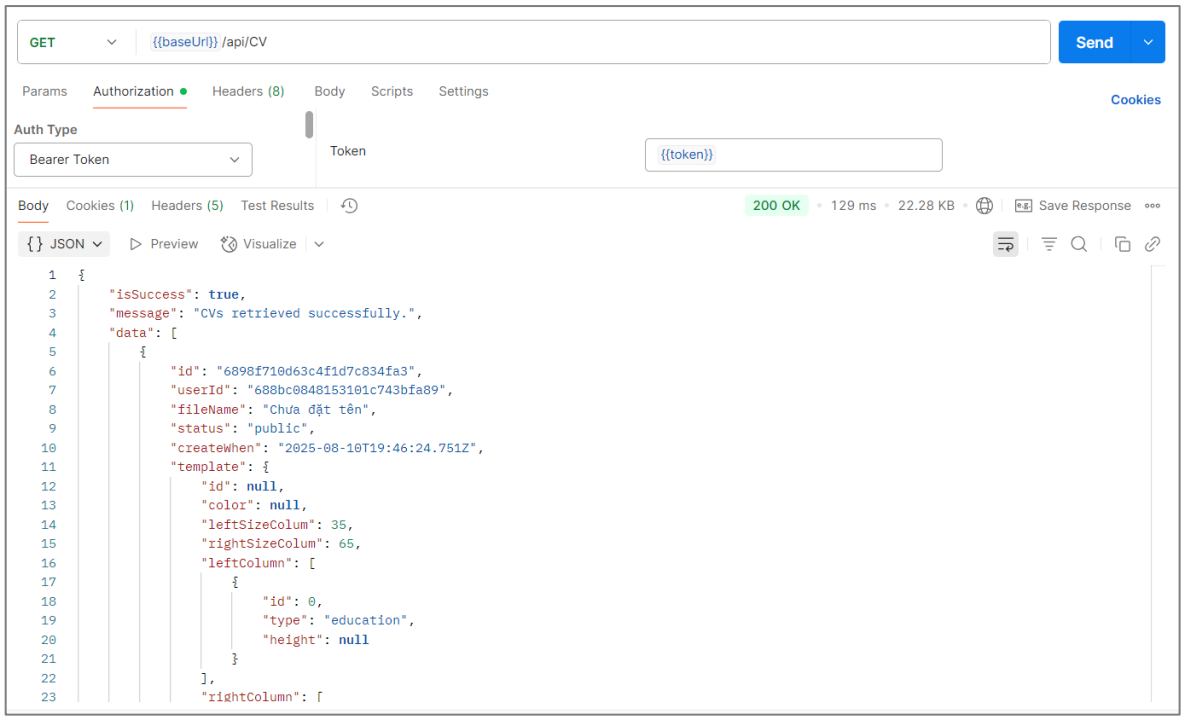


Hình 27: Kiểm thử API đăng ký

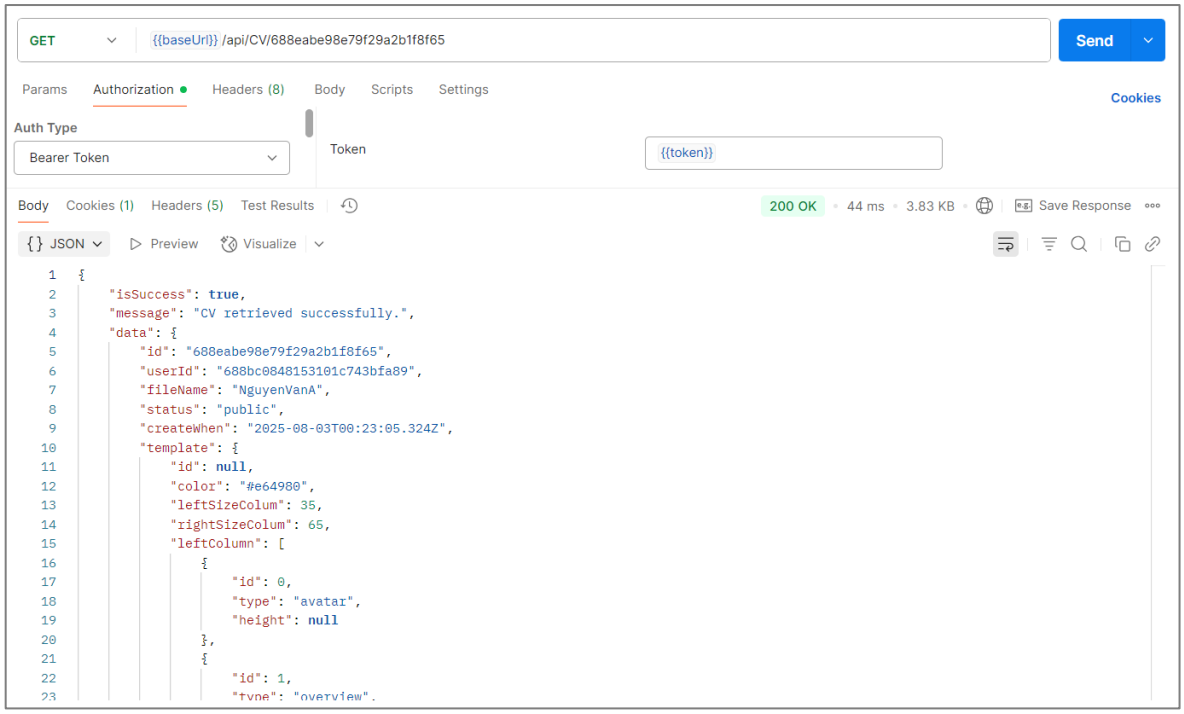


Hình 28: Kiểm thử API đăng nhập

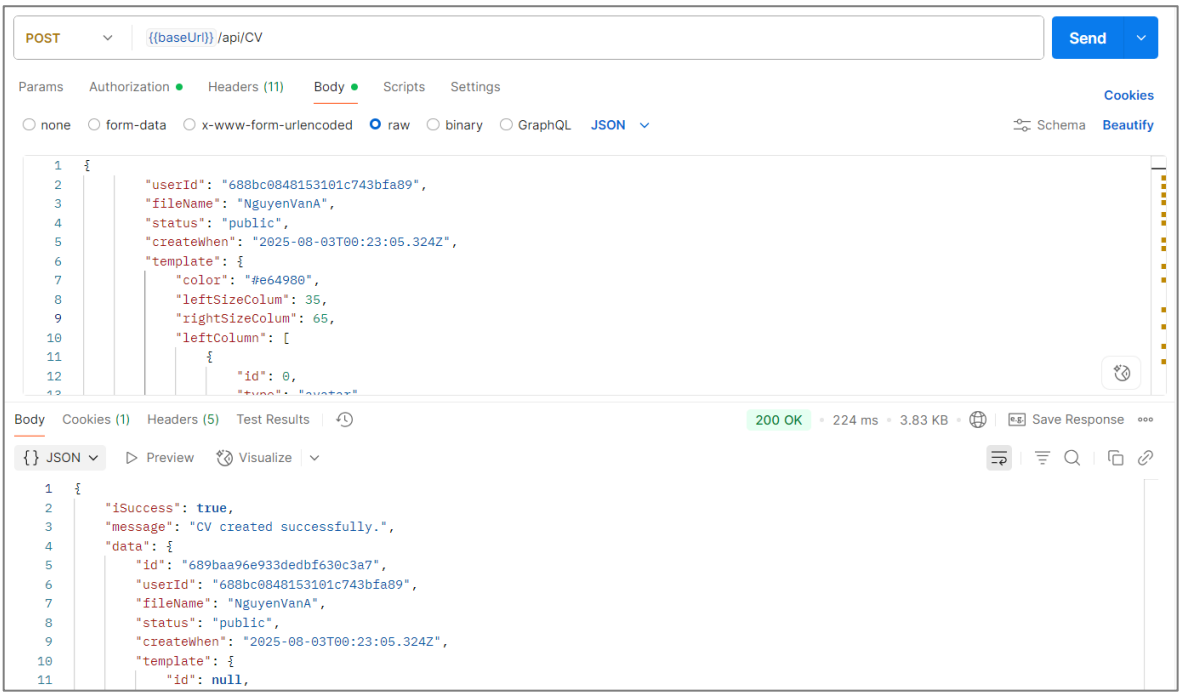
4.5.2 Các API liên quan đến hồ sơ xin việc



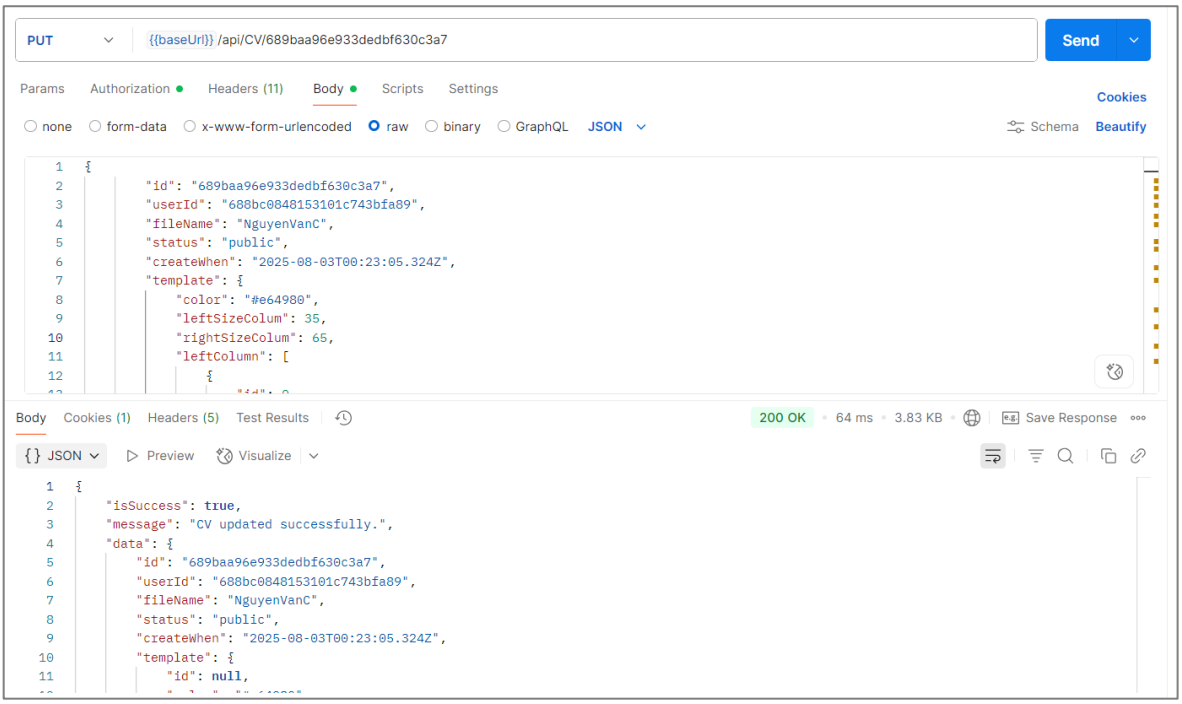
Hình 29: Kiểm thử API lấy danh sách hồ sơ xin việc



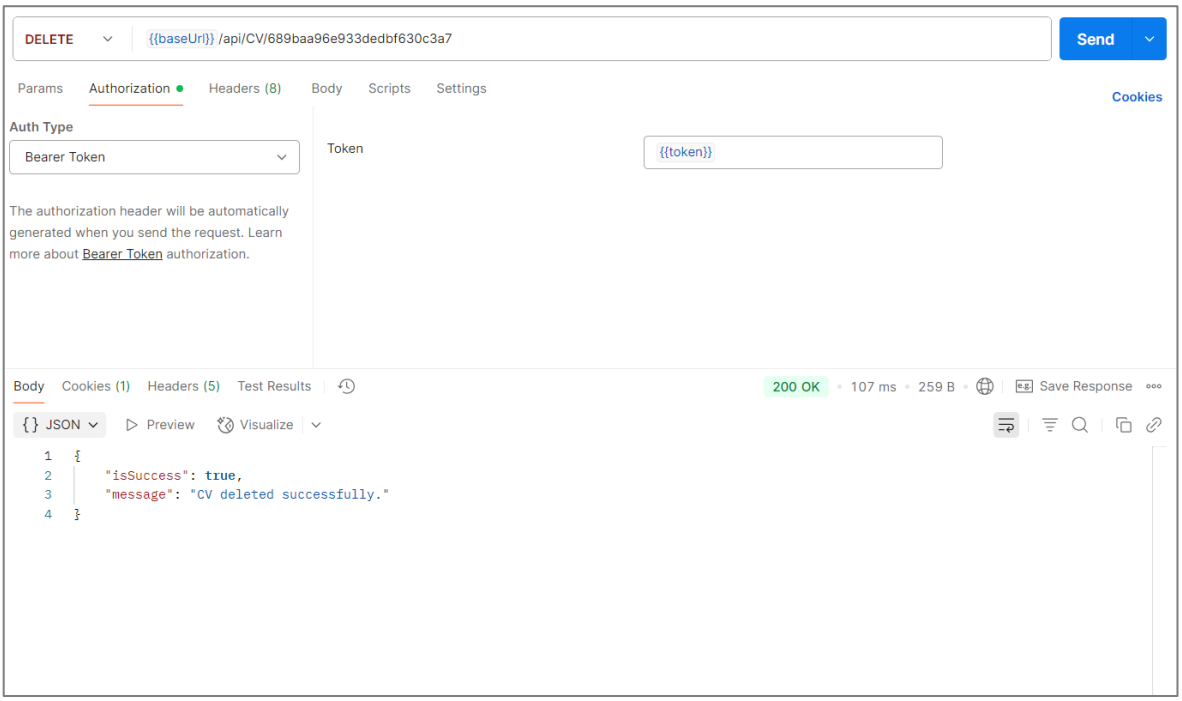
Hình 30: Kiểm thử API lấy thông tin một hồ sơ xin việc



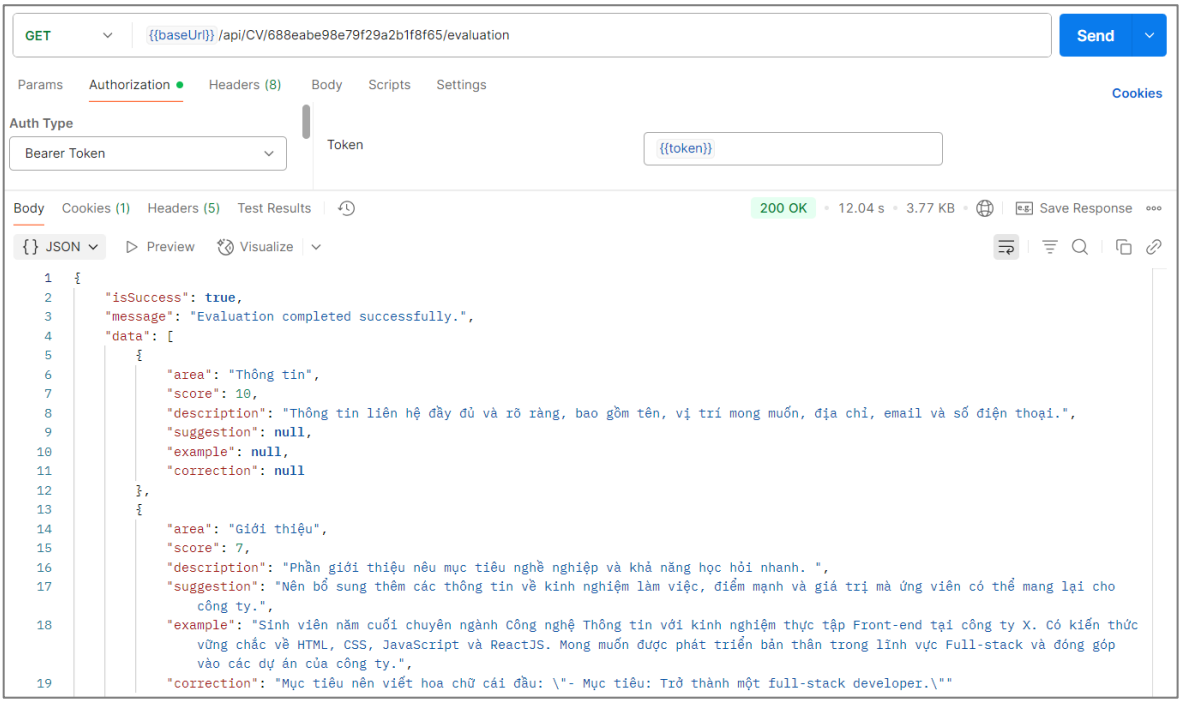
Hình 31: Kiểm thử API tạo hồ sơ xin việc



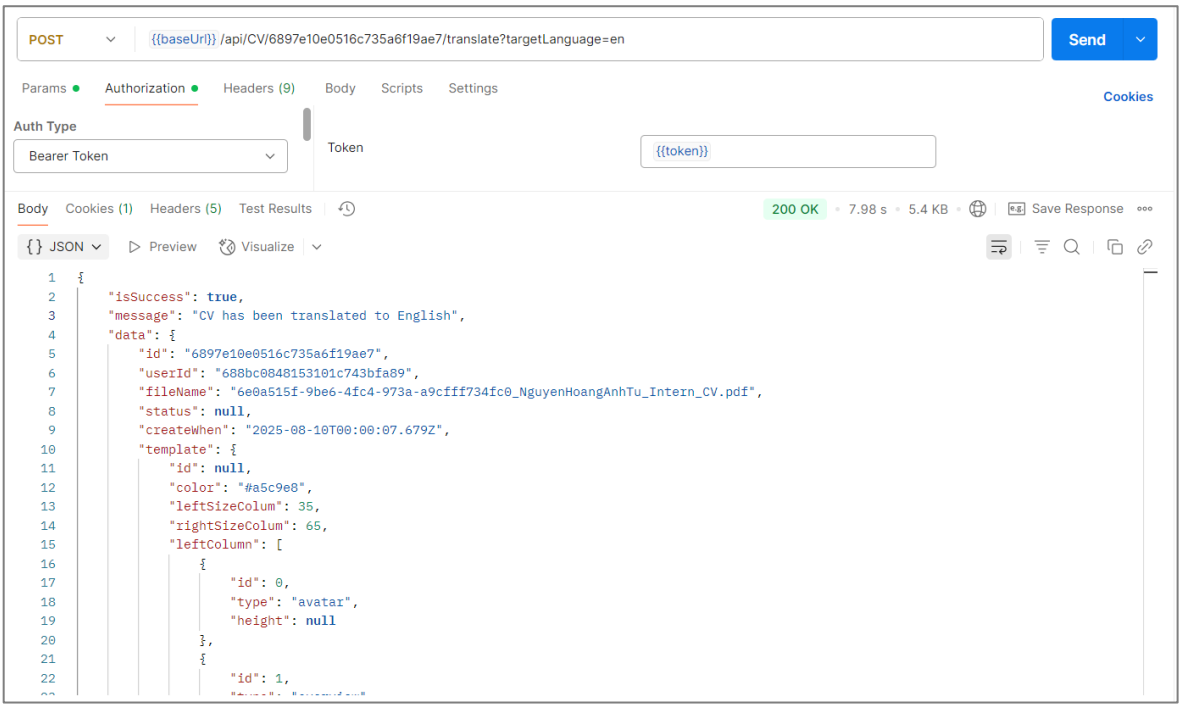
Hình 32: Kiểm thử API cập nhật hồ sơ xin việc



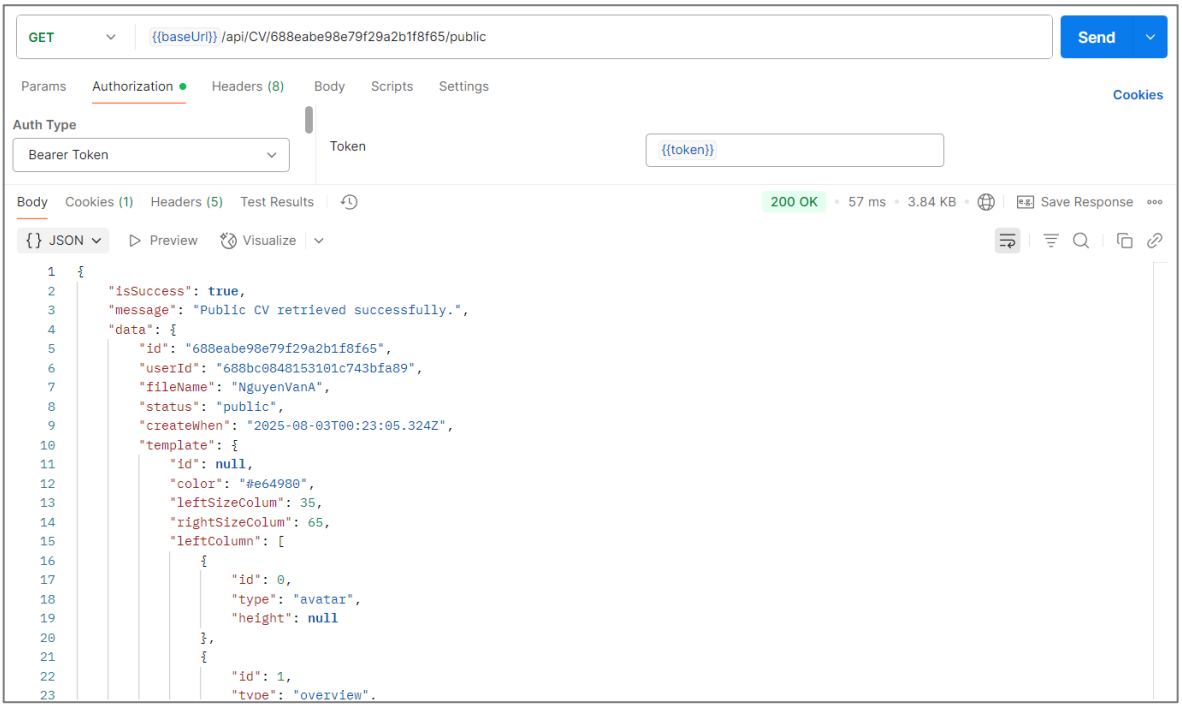
Hình 33: Kiểm thử API xoá hồ sơ xin việc



Hình 34: Kiểm thử API đánh giá hồ sơ xin việc



Hình 35: Kiểm thử API dịch nội dung hồ sơ xin việc

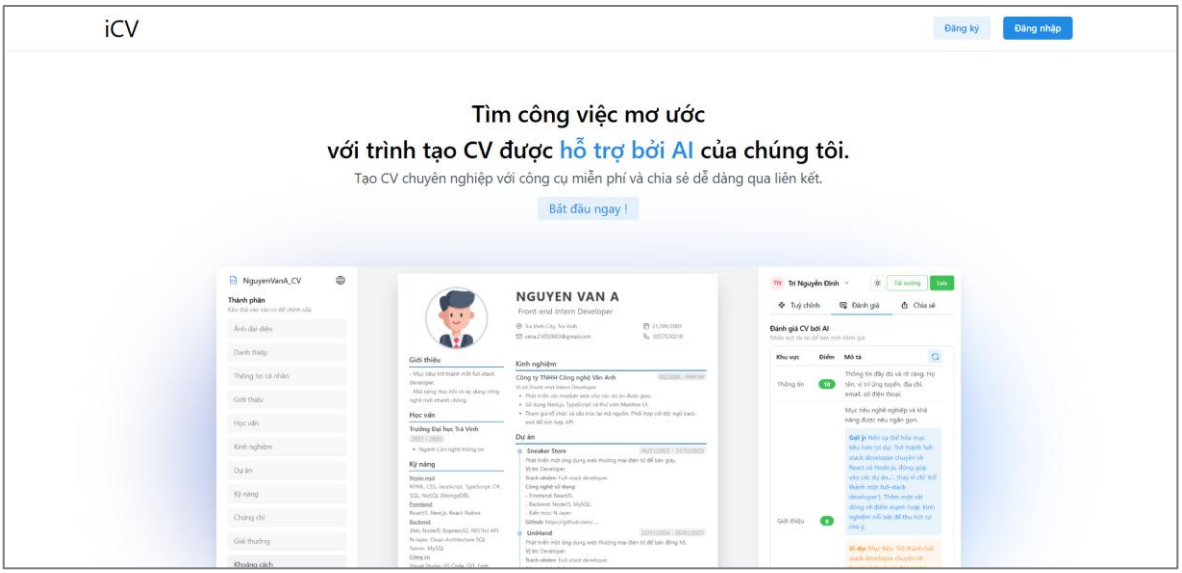


Hình 36: Kiểm thử API lấy thông tin hồ sơ xin việc công khai

4.6 Kết quả thử nghiệm chức năng

4.6.1 Chức năng Giới thiệu

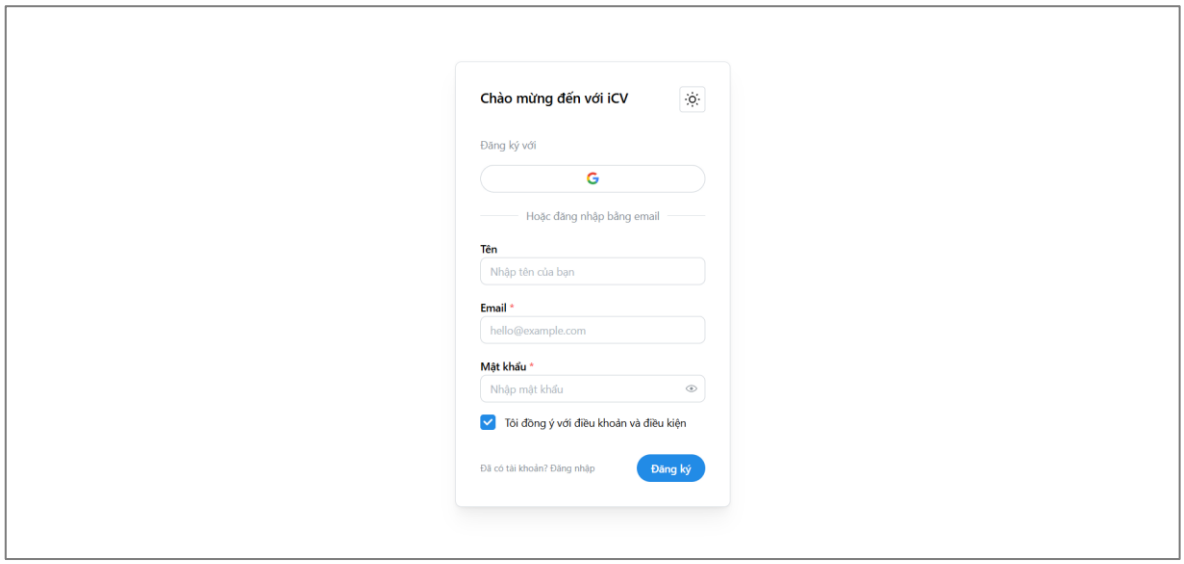
Ở trang chủ, hệ thống sẽ giới thiệu một cách ngắn gọn nhưng lôi cuốn về mục tiêu, lợi ích và các tính năng nổi bật, giúp người truy cập nhanh chóng nắm bắt được giá trị mà sản phẩm mang lại. Bố cục trang được thiết kế trực quan, kết hợp hình ảnh minh họa sinh động và mô tả trực tiếp từ giao diện thật của hệ thống, tạo ấn tượng ban đầu mạnh mẽ và khuyến khích người dùng khám phá sâu hơn.



Hình 37: Giao diện thức tế chức năng giới thiệu

4.6.2 Chức năng Đăng ký

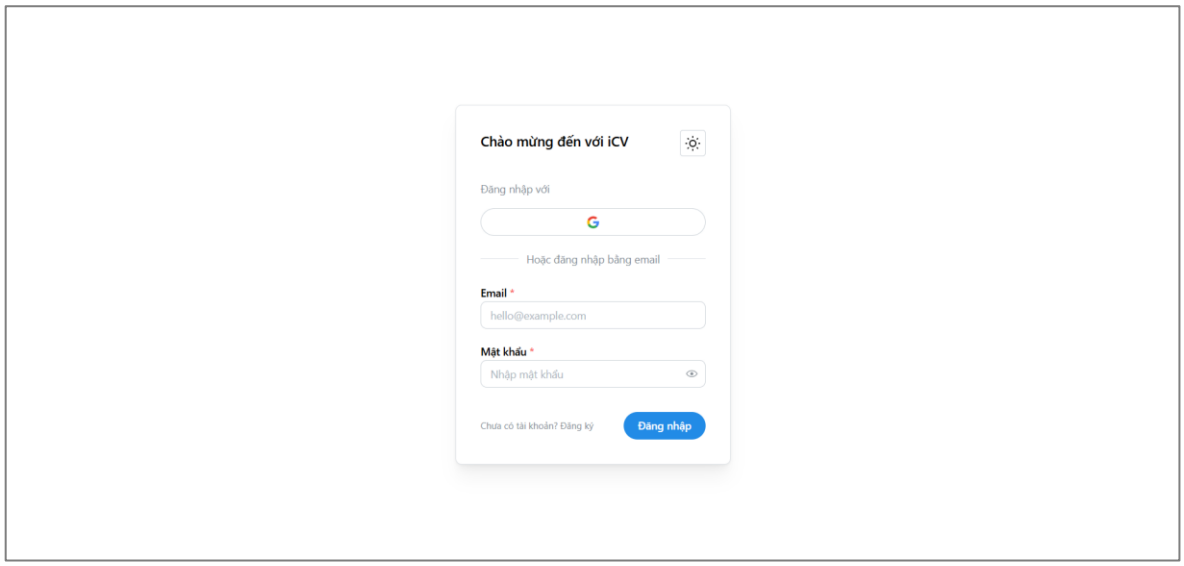
Hệ thống cung cấp giao diện đăng ký thân thiện, cho phép người dùng tạo tài khoản mới bằng cách nhập các thông tin cơ bản gồm họ và tên, địa chỉ email, số điện thoại và mật khẩu. Bên cạnh phương thức đăng ký truyền thống, hệ thống còn hỗ trợ đăng ký nhanh thông qua tài khoản Google, giúp rút ngắn thời gian thao tác. Cách tiếp cận linh hoạt này vừa đáp ứng nhu cầu của người dùng ưa thích quy trình tiêu chuẩn, vừa phù hợp với những người muốn trải nghiệm nhanh chóng và tiện lợi. Ngoài ra nếu đã có tài khoản có thể nhấn vào "Đã có tài khoản? Đăng nhập" để điều hướng đến trang đăng nhập.



Hình 38: Giao diện thực tế chức năng đăng ký

4.6.3 Chức năng Đăng nhập

Hệ thống cho phép người dùng truy cập bằng tài khoản đã đăng ký trước đó thông qua việc nhập địa chỉ email và mật khẩu. Tương tự đăng ký, hệ thống còn hỗ trợ đăng nhập nhanh bằng tài khoản Google. Ngoài ra nếu chưa có tài khoản có thể nhấn vào "Chưa có tài khoản? Đăng ký" để điều hướng đến trang đăng ký.

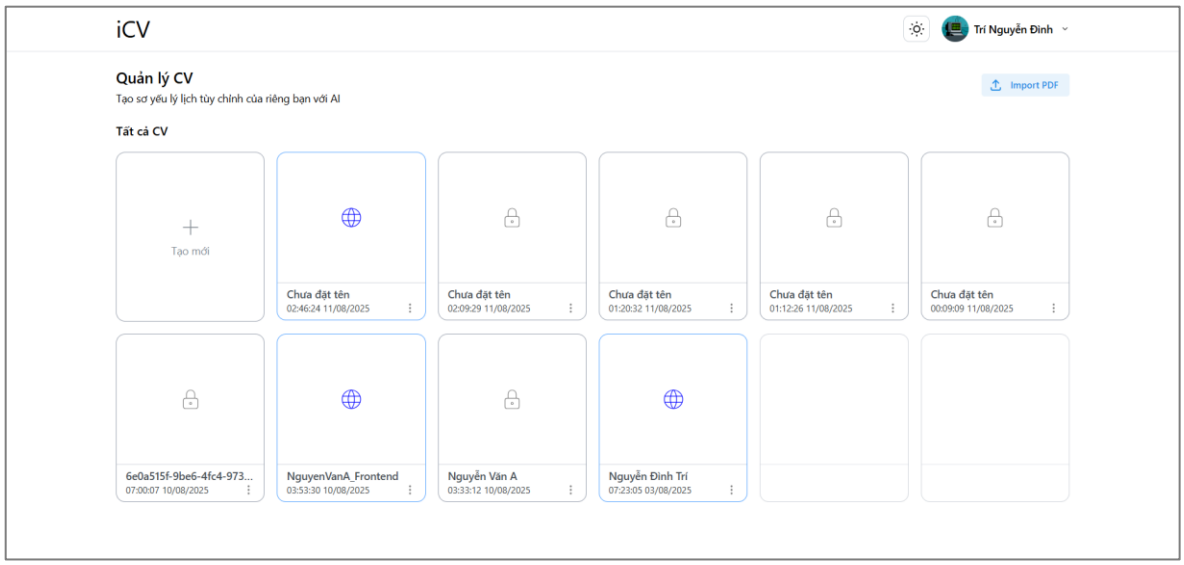


Hình 39: Giao diện thực tế chức năng đăng nhập

4.6.4 Chức năng Quản lý hồ sơ xin việc

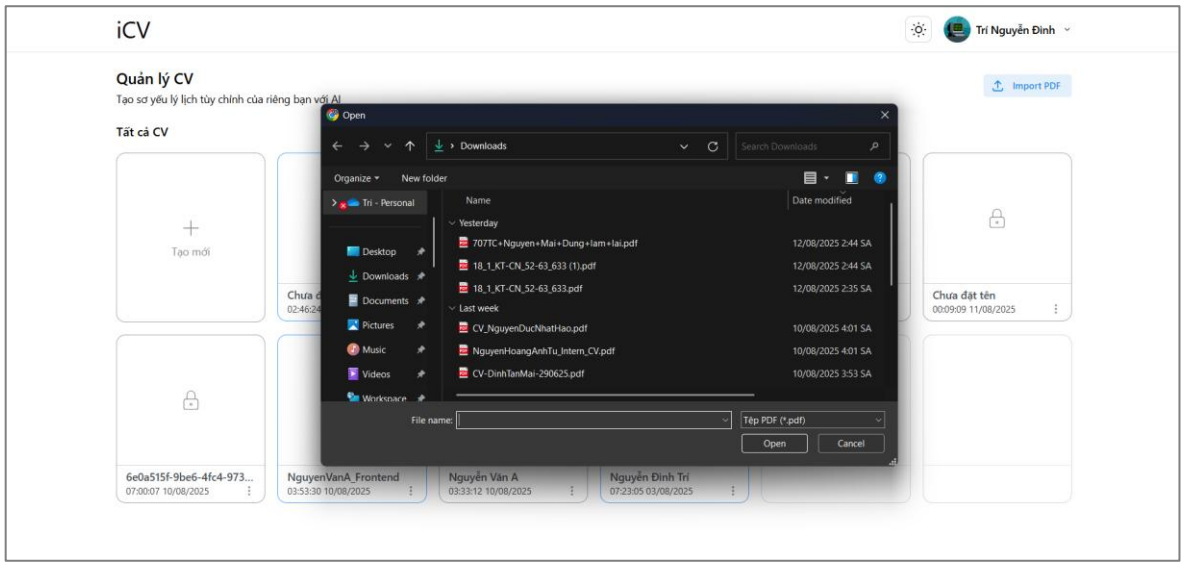
Hệ thống cung cấp một giao diện quản lý tập trung, trình bày danh sách các hồ sơ xin việc dưới dạng thẻ trực quan và dễ theo dõi. Mỗi thẻ hiển thị những thông tin rút gọn quan trọng của hồ sơ, bao gồm tên file, ngày tạo, và trạng thái (công khai hoặc riêng

tư). Người dùng có thể nhanh chóng thực hiện các thao tác như tạo mới hồ sơ, xóa hồ sơ, nhấn vào mỗi thẻ để chỉnh sửa hoặc import hồ sơ xin việc từ file PDF trực tiếp trên giao diện này.



Hình 40: Giao diện thực tế chức năng quản lý hồ sơ xin việc

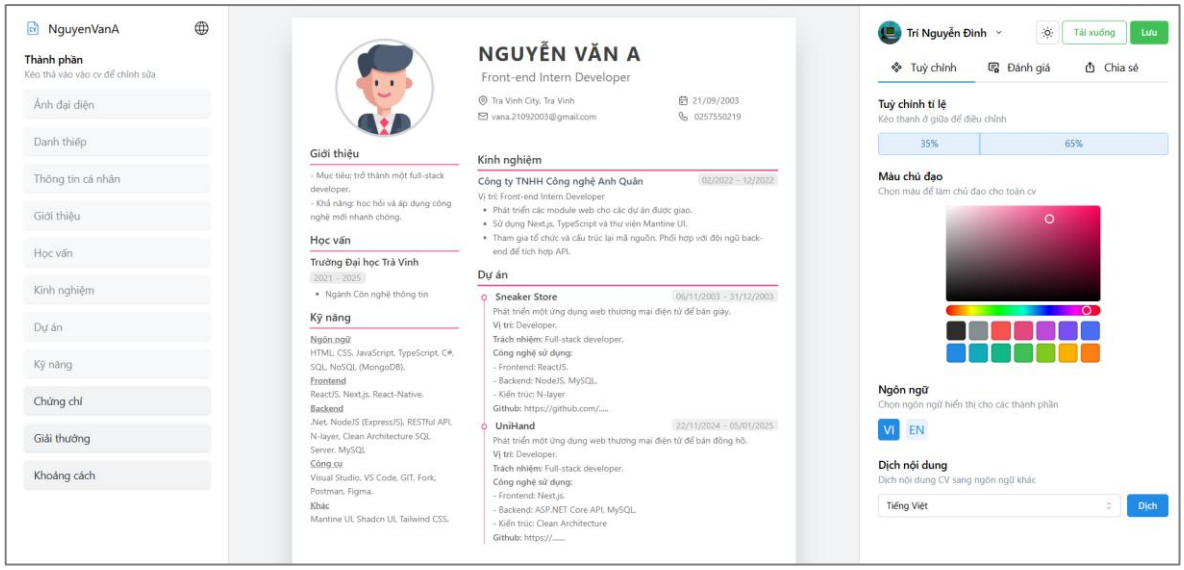
Khi người dùng nhấn vào chức năng Import, hệ thống sẽ mở hộp thoại chọn tệp (File Explorer trên Windows hoặc trình quản lý tệp tương ứng của hệ điều hành) để người dùng duyệt và chọn file CV ở định dạng PDF từ máy tính. Ngay sau khi tệp được tải lên, AI sẽ tự động phân tích nội dung, trích xuất thông tin và phân loại thành các mục tương ứng như thông tin cá nhân, học vấn, kinh nghiệm làm việc, kỹ năng, chứng chỉ, v.v. Các dữ liệu này sẽ được tự động nhập vào trình tạo CV và hình thành một hồ sơ mới trên hệ thống. Người dùng chỉ cần tinh chỉnh lại bố cục hoặc nội dung nếu muốn, thay vì phải nhập thủ công từ đầu, giúp tiết kiệm đáng kể thời gian và công sức.



Hình 41: Giao diện thực tế chắc năng import hồ sơ xin việc pdf

4.6.5 Chức năng Chỉnh sửa hồ sơ xin việc

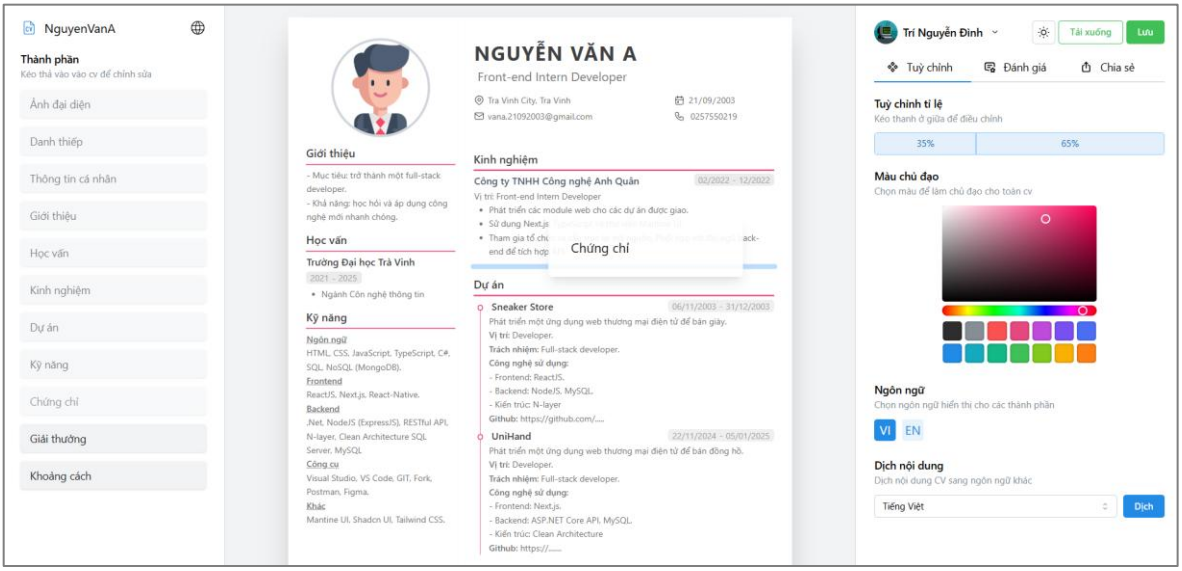
Hệ thống cung cấp trình chỉnh sửa hồ sơ xin việc trực quan, cho phép người dùng thay đổi tên file, tùy chỉnh tỉ lệ bố cục, lựa chọn màu sắc chủ đạo và ngôn ngữ hiển thị. Mỗi thành phần trong CV có thể được thêm mới, sắp xếp lại hoặc xóa bỏ tùy nhu cầu. Các thao tác được thiết kế đơn giản, trực quan, giúp người dùng nhanh chóng cá nhân hóa hồ sơ xin việc theo phong cách riêng.



Hình 42: Giao diện thực tế chức năng chỉnh sửa hồ sơ xin việc

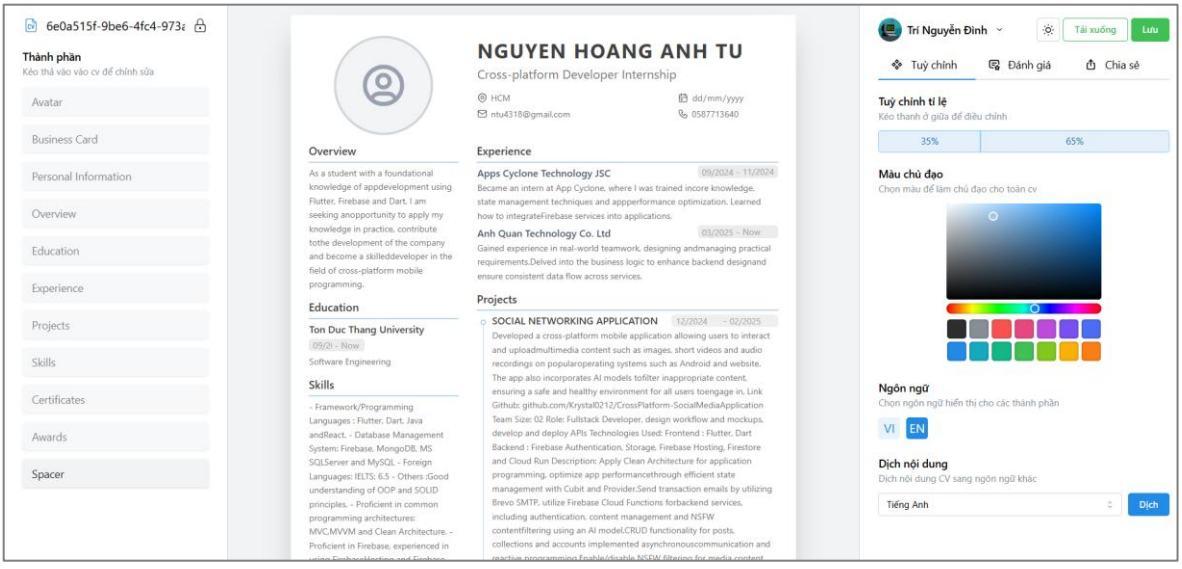
Giao diện chỉnh sửa hỗ trợ kéo-thả (drag-and-drop), cho phép người dùng thêm mới hoặc thay đổi vị trí của các thành phần trong hồ sơ xin việc một cách trực quan. Mỗi thành phần đều có thể được chỉnh sửa nội dung trực tiếp, thay đổi tiêu đề, mô tả, thông

tin chi tiết, hoặc xóa bỏ nếu không cần thiết. Người dùng cũng có thể tùy chỉnh tỉ lệ bố cục để cân đối hiển thị giữa các cột, đồng thời thay đổi màu sắc chủ đạo nhằm tạo dấu ấn cá nhân và nâng cao tính thẩm mỹ của hồ sơ xin việc.



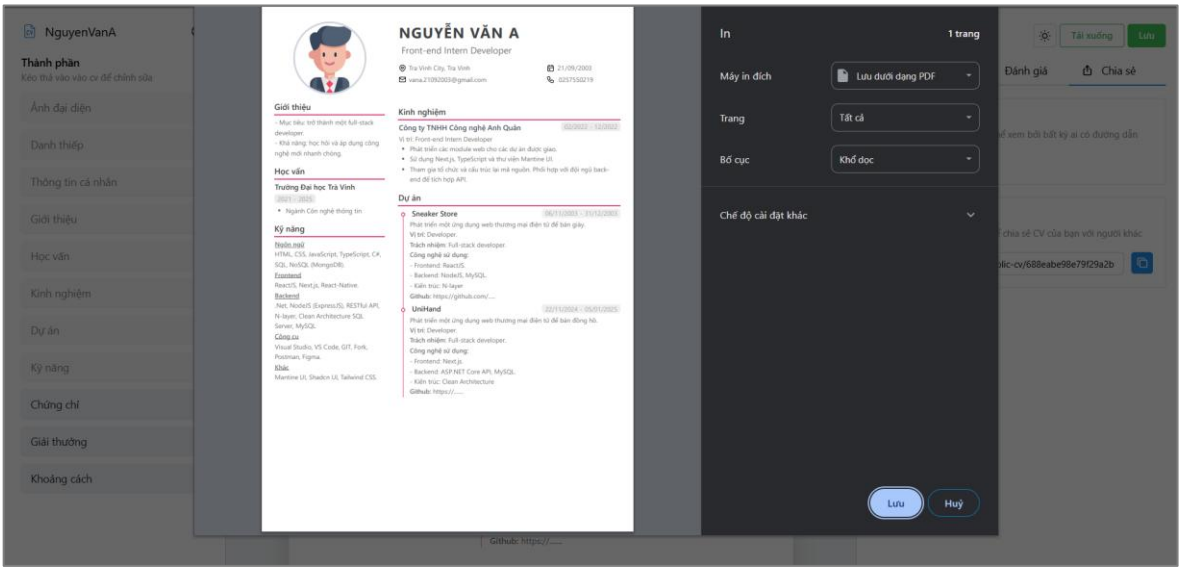
Hình 43: Giao diện thực tế chức kéo thả thành phần hồ sơ xin việc

Người dùng có thể sử dụng chức năng dịch tích hợp AI (Gemini API) để chuyển đổi ngôn ngữ, ví dụ từ tiếng Việt sang tiếng Anh hoặc ngược lại. AI sẽ phân tích ngữ cảnh, giữ nguyên định dạng và thuật ngữ chuyên môn, đảm bảo bản dịch mượt mà, tự nhiên và phù hợp với văn phong hồ sơ xin. Điều này giúp người dùng dễ dàng tạo ra nhiều phiên bản hồ sơ xin cho các thị trường lao động khác nhau mà không cần dịch thủ công.



Hình 44: Giao diện thực tế chức năng dịch nội dung hồ sơ xin việc

Ngoài ra, trong giao diện chỉnh sửa, hệ thống cũng tích hợp nút "Tải xuống" cho phép người dùng xuất hồ sơ xin việc hiện tại thành tệp PDF. Khi nhấn nút này, một cửa sổ tải xuống sẽ xuất hiện, giúp người dùng lưu nhanh về thiết bị của mình với định dạng và bố cục giống hệt phiên bản hiển thị trên màn hình.



Hình 45: Giao diện thực tế chức năng tải xuống hồ sơ xin việc pdf

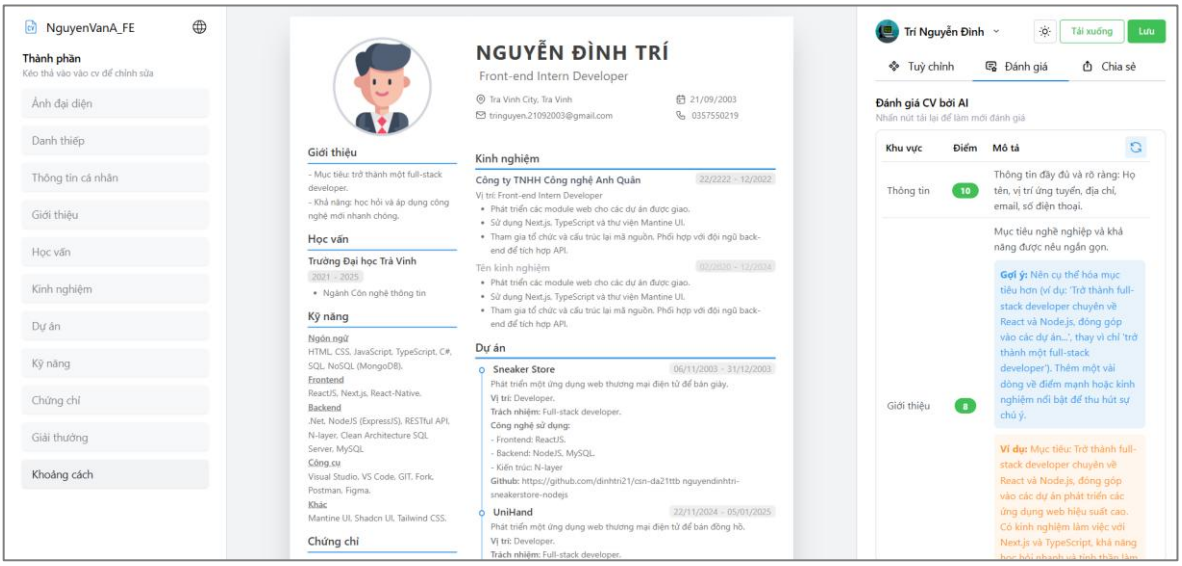
4.6.6 Chức năng Đánh giá hồ sơ xin việc

Khi người dùng yêu cầu đánh giá, hệ thống sẽ tiến hành phân tích nội dung hồ sơ xin việc và chấm điểm theo thang điểm từ 0 đến 9 dựa trên nhiều tiêu chí khác nhau. Các tiêu chí này bao gồm mức độ phù hợp với mô tả công việc (Job Description), độ đầy đủ và chính xác của thông tin, chất lượng ngôn ngữ, ngữ pháp và chính tả, cùng với việc tối ưu hóa sự xuất hiện của các từ khóa quan trọng. Người dùng cũng có thể tải lên JD cụ thể để hệ thống so sánh và đánh giá mức độ tương thích giữa hồ sơ xin việc và yêu cầu tuyển dụng.

Kết quả đánh giá sẽ được trình bày dưới dạng bảng gồm 3 cột, bao gồm: Khu vực – tên phần hoặc nội dung được đánh giá; Điểm số – số điểm từ 0 đến 9 mà AI chấm cho từng khu vực; và Mô tả – nhận xét chi tiết của AI về điểm mạnh, điểm yếu hoặc những vấn đề cần cải thiện. Cách trình bày này giúp người dùng nhanh chóng nắm bắt tình trạng tổng quan và chi tiết của từng phần trong hồ sơ xin việc.

Ngoài việc chấm điểm và đưa ra nhận xét, hệ thống còn có khả năng cung cấp gợi ý, ví dụ minh họa và điều chỉnh cụ thể cho từng nội dung. Ví dụ, AI có thể gợi ý thêm kỹ năng liên quan đến yêu cầu công việc, điều chỉnh câu mô tả kinh nghiệm để ngắn

gọn và súc tích hơn, hoặc đề xuất bổ sung số liệu minh chứng nhằm tăng tính thuyết phục. Nhờ đó, người dùng không chỉ biết hồ sơ của mình đang ở mức nào, mà còn nhận được định hướng rõ ràng để nâng cấp và hoàn thiện hồ sơ xin việc, giúp tăng khả năng gây ấn tượng với nhà tuyển dụng.



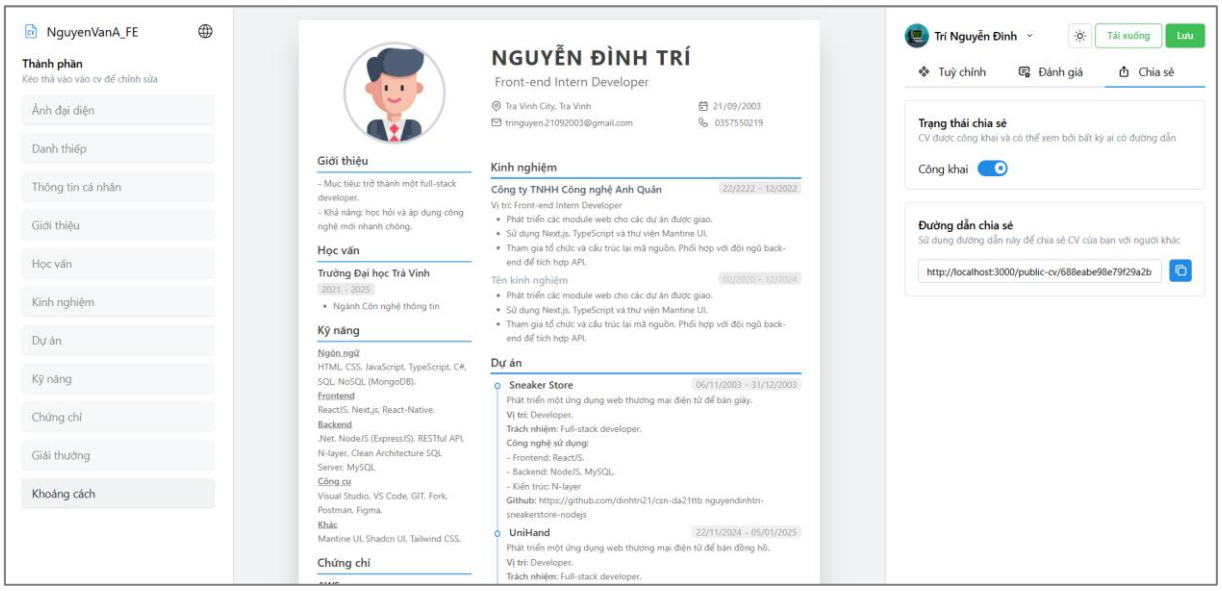
Hình 46: Giao diện thực tế chức năng đánh giá hồ sơ xin việc

4.6.7 Chức năng Chỉnh sửa trạng thái hồ sơ xin việc

Chức năng chỉnh sửa trạng thái hồ sơ xin việc cho phép người dùng linh hoạt quản lý quyền truy cập hồ sơ xin việc của mình. Trong giao diện, mỗi hồ sơ sẽ có tùy chọn chuyển đổi giữa hai chế độ hiển thị: Công khai (Public) và Riêng tư (Private).

Khi người dùng chọn chế độ công khai, hệ thống sẽ tự động tạo một đường dẫn chia sẻ duy nhất. Đường dẫn này sẽ được hiển thị ngay trên giao diện, kèm theo nút Sao chép liên kết để người dùng dễ dàng copy và gửi cho nhà tuyển dụng, đăng trên mạng xã hội hoặc chia sẻ qua email. Bất kỳ ai có liên kết này đều có thể truy cập và xem hồ sơ mà không cần đăng nhập.

Ngược lại, khi chuyển về chế độ Riêng tư, đường dẫn chia sẻ sẽ bị vô hiệu hóa, đảm bảo rằng hồ sơ chỉ có thể được xem khi người sở hữu hồ sơ xin việc đăng nhập vào hệ thống. Điều này giúp người dùng chủ động kiểm soát quyền riêng tư và phạm vi tiếp cận của hồ sơ xin việc, tùy theo từng giai đoạn ứng tuyển hoặc mục đích sử dụng.



Hình 47: Giao diện thực tế chức năng chỉnh sửa trạng thái hồ sơ xin việc

4.6.8 Chức năng Hiển thị hồ sơ xin việc công khai

Chức năng hiển thị hồ sơ xin việc công khai cho phép bất kỳ ai có đường dẫn chia sẻ do người dùng tạo trước đó truy cập và xem toàn bộ nội dung hồ sơ xin việc trực tiếp trên trình duyệt. Khi truy cập vào liên kết, hệ thống sẽ hiển thị trang chuyên biệt với bố cục rõ ràng, trình bày đầy đủ các thành phần của hồ sơ xin việc.

Hình 48: Giao diện thực tế chức năng hiển thị hồ sơ xin việc công khai

Trên trang này, người xem còn có thể sử dụng nút "Tải xuống CV" để tải phiên bản PDF của hồ sơ về thiết bị của mình. Tệp PDF được xuất trực tiếp từ dữ liệu trong hệ thống, đảm bảo nội dung và định dạng giống với hồ sơ xin việc gốc mà người dùng đã thiết kế.

Tính năng này giúp việc chia sẻ hồ sơ trở nên tiện lợi và chuyên nghiệp hơn: nhà tuyển dụng hoặc đối tác có thể xem nhanh hồ sơ trực tuyến, đồng thời tải bản PDF nếu cần lưu trữ hoặc sử dụng cho các bước tuyển chọn tiếp theo.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết quả đạt được của đề tài

Đề tài đã nghiên cứu và xây dựng thành công một hệ thống quản lý và chỉnh sửa hồ sơ xin việc trực tuyến, tích hợp trí tuệ nhân tạo để hỗ trợ người dùng tạo, chỉnh sửa, dịch thuật và đánh giá hồ sơ xin việc một cách nhanh chóng, trực quan. Hệ thống đáp ứng đầy đủ các yêu cầu cốt lõi: giao diện thân thiện, cho phép tùy chỉnh bố cục, màu sắc, ngôn ngữ; hỗ trợ nhập liệu thủ công hoặc tự động thông qua tính năng import từ file PDF; đồng thời cung cấp khả năng chia sẻ hồ sơ công khai qua đường dẫn.

Kết quả thử nghiệm thực tế cho thấy hệ thống vận hành ổn định, khả năng trích xuất thông tin từ hồ sơ xin việc PDF đạt độ chính xác cao, tính năng kéo-thả bố cục hoạt động mượt mà và phản hồi nhanh. Việc tích hợp API AI (Gemini) giúp tự động dịch nội dung và đánh giá hồ sơ theo nhiều tiêu chí, góp phần nâng cao chất lượng và tính chuyên nghiệp của hồ sơ xin việc. Tính năng tải xuống PDF trực tiếp từ giao diện chỉnh sửa cũng mang lại trải nghiệm liền mạch và tiện lợi cho người dùng.

5.2 Hạn chế của hệ thống và nghiên cứu

Bên cạnh những kết quả đạt được, hệ thống vẫn tồn tại một số hạn chế cần được cải thiện trong các nghiên cứu và phát triển tiếp theo:

- Về tính đa dạng mẫu hồ sơ xin việc: Hệ thống hiện mới cung cấp một số lượng hạn chế mẫu hồ sơ xin việc, chủ yếu phù hợp với nhóm đối tượng lập trình viên. Chưa có nhiều mẫu thiết kế chuyên biệt cho các lĩnh vực hoặc đối tượng khác nhau như nhà thiết kế, chuyên gia marketing. Điều này phần nào hạn chế khả năng cá nhân hóa và đáp ứng nhu cầu đa dạng của người dùng.

-Về kỹ thuật trích xuất từ PDF: Quá trình phân tích nội dung từ file PDF đôi khi gặp khó khăn với các định dạng phức tạp hoặc bố cục không chuẩn, dẫn đến thông tin bị thiếu hoặc sắp xếp sai vị trí. Bên cạnh đó, việc phân tách dữ liệu theo vùng cố định có thể làm mất tính liền mạch của ngữ cảnh.

- Về đánh giá và phản hồi người dùng: Hệ thống mới chủ yếu được kiểm tra qua các tiêu chí kỹ thuật như tốc độ phản hồi, độ chính xác trích xuất và tính ổn định. Chưa có nhiều khảo sát hoặc thu thập phản hồi định tính từ người dùng cuối để đánh giá trải nghiệm thực tế, tính thân thiện giao diện và mức độ đáp ứng nhu cầu.

5.3 Đề xuất hướng phát triển

Để hoàn thiện và nâng cao chất lượng hệ thống, một số hướng phát triển trong tương lai được đề xuất như sau:

- Mở rộng và cá nhân hóa mẫu hồ sơ xin việc: Bổ sung nhiều mẫu hồ sơ xin việc đa dạng, được thiết kế chuyên biệt cho từng nhóm đối tượng như sinh viên mới tốt nghiệp, nhân sự cấp quản lý, chuyên gia kỹ thuật, hoặc các ngành sáng tạo. Cho phép người dùng tùy chỉnh bố cục sâu hơn, ví dụ thay đổi vị trí các khối thông tin, chọn nhiều phong cách màu sắc, và chèn các yếu tố đồ họa cá nhân hóa.

- Tích hợp công nghệ AI nâng cao: Ứng dụng các mô hình AI hiện đại để gợi ý cải thiện nội dung hồ sơ xin việc, phát hiện lỗi ngữ pháp và tối ưu từ khóa cho từng lĩnh vực nghề nghiệp. Mở rộng tính năng dịch tự động sang nhiều ngôn ngữ khác nhau, đảm bảo giữ nguyên định dạng và thuật ngữ chuyên ngành.

- Tối ưu hiệu năng và trải nghiệm người dùng: Rút ngắn thời gian tải và xử lý dữ liệu, đồng thời cải thiện giao diện để thao tác trực quan hơn, hỗ trợ cả trên thiết bị di động. Thực hiện khảo sát người dùng ở nhiều nhóm khác nhau để thu thập phản hồi và điều chỉnh hệ thống phù hợp nhu cầu thực tế.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] “Introduction,” Next.js Documentation. [Online]. Available: https://nextjs.org/docs. Accessed: 17 June 2025.

[2] “Introduction to .NET,” Microsoft Learn, 10 January 2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/introduction. Accessed: 17 August 2025.

[3] R. C. Martin (Uncle Bob), “The Clean Architecture,” Clean Code Blog, 13 August 2012. [Online]. Available: blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. Accessed: 20 June 2025.

[4] “What is REST?: REST API Tutorial,” RESTfulapi.net. [Online]. Available: https://restfulapi.net/. Accessed: 12 June 2025.

[5] “Large language model,” Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Large_language_model. Accessed: 17 June 2025.

[6] “Files API,” Gemini API, Google AI for Developers. [Online]. Available: https://ai.google.dev/gemini-api/docs/files. Accessed: 17 June 2025.

[7] Tailwind CSS Documentation, “Get started with Tailwind CSS,” [Online]. Available: https://tailwindcss.com/docs. Accessed: 17 June 2025.