

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023 - 2024

XÂY DỰNG WEBSITE TÌM KIẾM VIỆC LÀM BẰNG REACTJS

Giáo viên hướng dẫn:
Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Đinh Tấn Mãi
MSSV: 110121063
Lớp: DA21TTB

Trà Vinh, tháng 01 năm 2024

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm
Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành đến quý thầy cô, các bạn bè đã luôn quan tâm, giúp đỡ em trong suốt quá trình học tập và thực hiện đề tài "Xây dựng website tìm kiếm việc làm bằng ReactJS".

Đề tài này là một thử thách lớn đối với em, đòi hỏi em phải có kiến thức chuyên môn vững vàng và kỹ năng lập trình thành thạo. Tuy nhiên, nhờ sự hướng dẫn tận tình của thầy, sự động viên của gia đình và bạn bè, em đã có thể hoàn thành đề tài đúng tiến độ và đạt được kết quả tốt.

Em xin chân thành cảm ơn thầy đã luôn dành thời gian quý báu để hướng dẫn, giải đáp thắc mắc của em trong suốt quá trình thực hiện đề tài. Thầy đã giúp em hiểu rõ hơn về đề tài, từ đó có thể xây dựng được một website tìm kiếm việc làm đáp ứng được nhu cầu của người dùng.

Em cũng xin gửi lời cảm ơn đến các bạn bè trong lớp đã luôn giúp đỡ và động viên em trong suốt quá trình học tập và thực hiện đề tài. Sự giúp đỡ của các bạn đã giúp em có thêm động lực để hoàn thành đề tài một cách tốt nhất.

Đề tài "Xây dựng website tìm kiếm việc làm bằng ReactJS" tuy đã hoàn thành nhưng vẫn còn nhiều thiếu sót. Em rất mong nhận được ý kiến đóng góp của quý thầy cô, các bạn để đề tài được hoàn thiện hơn.

Một lần nữa, em xin gửi lời cảm ơn chân thành đến quý thầy cô, các bạn bè đã luôn quan tâm, giúp đỡ em trong suốt quá trình học tập và thực hiện đề tài.

Em xin chân thành cảm ơn!

Trân trọng,

MỤC LỤC

CHƯƠNG 1	TỔNG QUAN	10
CHƯƠNG 2	NGHIÊN CỨU LÝ THUYẾT	12
2.1	Cấu trúc Web API	12
2.2	Frontend với ReactJS	13
2.3	Backend với Node.js	14
2.4	Xác thực JSON Web Token (JWT)	15
CHƯƠNG 3	HIỆN THỰC HÓA NGHIÊN CỨU	18
3.1	Đặc tả yêu cầu hệ thống	18
3.1.1	Yêu cầu chức năng	18
3.1.2	Yêu cầu phi chức năng	19
3.2	Sơ đồ use-case	20
3.2.1	Sơ đồ use-case của người tìm việc	20
3.2.2	Sơ đồ use-case của người nhà tuyển dụng	21
3.3	Thiết kế dữ liệu	22
3.3.1	Lược đồ cơ sở dữ liệu	22
3.3.2	Chi tiết các thực thể	23
3.4	Thiết kế, kiến trúc ứng dụng	27
3.4.1	Kiến trúc toàn cảnh	27
3.4.2	Cách frontend nhận dữ liệu	28
3.4.3	Thiết kế API	29
3.4.4	Thiết kế frontend	32
3.5	Triển khai	36
CHƯƠNG 4	KẾT QUẢ NGHIÊN CỨU	37
4.1	Giao diện frontend	37
4.1.1	Giao diện chung	37
4.1.2	Giao diện người tìm việc	44
4.1.3	Giao diện nhà tuyển dụng	46
4.2	Giao diện backend	49
CHƯƠNG 5	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	51

5.1	Những kết quả đạt được	51
5.2	Hướng phát triển	52

DANH MỤC BẢNG

Bảng 3.1	Danh sách các thực thể	22
Bảng 3.2	Chi tiết thực thể “ user ”: Ứng viên.....	23
Bảng 3.3	Chi tiết thực thể “ companies ”: Nhà tuyển dụng.....	23
Bảng 3.4	Chi tiết thực thể “ jobs ”: Công việc.....	24
Bảng 3.5	Chi tiết thực thể “ provinces ”: Tỉnh thành	25
Bảng 3.6	Chi tiết thực thể “ fields ”: Ngành nghề.....	25
Bảng 3.7	Chi tiết thực thể “ apply_job ”: Ứng tuyển	26
Bảng 3.8	Chi tiết thực thể “ save_job ”: Lưu việc làm	26
Bảng 3.9	Chi tiết thực thể “ follow_company ”: Theo dõi nhà tuyển dụng.....	27

DANH MỤC HÌNH ẢNH

Hình 2.1. Sơ đồ hoạt động của web API.....	12
Hình 2.2 Hình ảnh cấu trúc của JSON Web Token.....	16
Hình 3.1 Sơ đồ use-case của người tìm việc.	20
Hình 3.2 Sơ đồ use-case nhà tuyển dụng.	21
Hình 3.3 Lược đồ cơ sở dữ liệu.	22
Hình 3.4 Tập component.js phía frontend.....	28
Hình 3.5 Tập index.js từ phía backend	29
Hình 3.6 Cấu trúc tập tin index.js	30
Hình 3.7 Cấu trúc thư mục routes	31
Hình 3.8 Cấu trúc thư mục controllers.....	31
Hình 3.9 Tập tin <i>index.js</i>	32
Hình 3.10 Tập tin <i>job.routes.js</i>	32
Hình 3.11 Tập tin <i>job.controller.js</i>	32
Hình 3.12 Cấu trúc thư mục frontend	33
Hình 3.13 Tập tin <i>Header.js</i>	34
Hình 3.14 Tập tin <i>Home.js</i>	34
Hình 3.15 Tập tin <i>MainLayout.js</i>	35
Hình 3.16 Tập tin <i>App.js</i>	35
Hình 3.17 Kết quả thực hiện cấu trúc thư mục frontend	36
Hình 4.1 Giao diện trang chủ.....	38
Hình 4.2 Giao diện trang ngành nghề/ địa điểm.....	39
Hình 4.3 Giao diện trang tìm kiếm việc làm.....	40
Hình 4.4 Giao diện trang tìm kiếm nhà tuyển dụng	41
Hình 4.5 Giao diện trang chi tiết việc làm	42
Hình 4.6 Giao diện trang chi tiết nhà tuyển dụng	43
Hình 4.7 Giao diện trang chi tiết người tìm việc.....	44
Hình 4.8 Giao diện trang đăng ký người tìm việc	45
Hình 4.9 Giao diện trang đăng nhập nhà tuyển dụng.....	46
Hình 4.10 Giao diện trang đăng ký nhà tuyển dụng.....	47
Hình 4.11 Trang đăng nhập nhà tuyển dụng	48
Hình 4.12 Giao diện trang quản lý đơn ứng tuyển của nhà tuyển dụng	49
Hình 4.13 Giao diện API phía Backend.....	50

TÓM TẮC ĐỒ ÁN CƠ SỞ NGÀNH

Vấn đề nghiên cứu

Đồ án này tập trung vào nghiên cứu ReactJS là một framework JavaScript mã nguồn mở, được phát triển và duy trì bởi Meta và cộng đồng các nhà phát triển và công ty cá nhân. Nhằm xây dựng một website tìm kiếm việc làm có giao diện đơn giản, dễ sử dụng, đáp ứng được nhu cầu của người dùng.

Hướng tiếp cận

Để thực hiện đề tài này tôi thực hiện 2 bước tiếp cận như sau: nghiên cứu ReactJS và thiết kế, xây dựng website. Với bước nghiên cứu ReactJS, giúp tôi nắm vững kỹ năng cần thiết để xây dựng website, như cài đặt và cấu hình ReactJS, sử dụng component, state, props và sử dụng API. Còn với bước thiết kế và xây dựng website, cần thiết kế khoa học, hợp lý, dễ sử dụng, thân thiện với người dùng.

Cách giải quyết vấn đề

Để giải quyết vấn đề của đề tài này tôi phải thực hiện nghiên cứu ReactJS, tìm hiểu về cách nó hoạt động, cách sử dụng component, quản lý state và props, và cách tương tác với API. Thiết kế giao diện một cách cẩn thận, sử dụng các nguyên tắc thiết kế UX/UI để đảm bảo trải nghiệm người dùng tốt nhất. Tận dụng những kỹ thuật và tính năng mà ReactJS cung cấp để tối ưu hóa hiệu suất và tích hợp API một cách linh hoạt. Tích cực tìm hiểu và áp dụng những kiến thức mới, cũng như học hỏi từ các tài liệu, hướng dẫn và dự án thực tế khác.

Kết quả đạt được

Sau khi thực hiện đề tài này tôi được các kết quả như sau, đạt được kiến thức sâu rộng về ReactJS, bao gồm cách cài đặt, sử dụng component, quản lý state và props, tương tác với API, và triển khai ứng dụng. Xây dựng được website tìm kiếm việc làm thành công, cung cấp trải nghiệm người dùng tốt và đáp ứng nhu cầu tìm kiếm công việc một cách thuận tiện. Hiểu biết rõ ràng về quy trình phát triển ứng dụng web, từ việc xác định yêu cầu đến triển khai và duy trì.

MỞ ĐẦU

Trong thời đại công nghệ ngày nay, nhu cầu tìm kiếm việc làm ngày càng tăng cao. Tuy nhiên, việc tìm kiếm việc làm phù hợp với năng lực và sở thích của bản thân lại không hề dễ dàng, và việc phát triển một trang web sử dụng ReactJS sẽ giúp nâng cao trải nghiệm người dùng, từ đó giúp họ dễ dàng tìm thấy cơ hội nghề nghiệp phù hợp. ReactJS là một framework JavaScript phổ biến, được sử dụng để xây dựng các ứng dụng web hiện đại. Việc sử dụng ReactJS sẽ giúp tôi xây dựng website tìm kiếm việc làm có giao diện đẹp mắt, hiệu năng cao và dễ sử dụng. Đề tài này có ý nghĩa thực tiễn, đáp ứng được nhu cầu của người dùng, đồng thời cũng là cơ hội để tôi vận dụng kiến thức và kỹ năng đã học vào thực tế.

Mục đích của đề tài là giúp có cơ hội được học hỏi thêm kiến thức và kỹ năng mới. Bằng cách xây dựng website này bằng ReactJS, tôi mong muốn tạo ra một trải nghiệm người dùng mượt mà, tương tác và đồng thời hỗ trợ nhà tuyển dụng và người tìm việc kết nối một cách thuận lợi. Giúp tôi có cơ hội được rèn luyện các kỹ năng mềm cần thiết cho công việc sau khi ra trường, đồng thời hiểu rõ hơn về cách tích hợp các công nghệ mới nhất vào các ứng dụng web.

Đối tượng nghiên cứu của đề tài chủ yếu là ReactJS, một thư viện JavaScript mạnh mẽ và linh hoạt được phát triển bởi Facebook, bao gồm các tính năng và công nghệ đi kèm với ReactJS nhằm tối ưu hóa trải nghiệm người dùng và quản lý trạng thái ứng dụng.

Phạm vi nghiên cứu của đề tài là tìm hiểu khái quát về ReactJS, nắm được các định nghĩa, khái niệm, cách render, import trong ReactJS và các thuộc tính quan trọng như component, state, props, JSX, ... Đề tài tập trung tập vào cung cấp một nền tảng thuận tiện cho người tìm việc và nhà tuyển dụng. Đối với người tìm việc họ có thể dễ dàng xem danh sách các việc làm phù hợp với mình và thực hiện ứng tuyển một cách thuận tiện. Với nhà tuyển dụng, họ sẽ nhận được các đơn ứng tuyển một cách hiệu quả thông qua hệ thống quản lý đơn ứng tuyển.

CHƯƠNG 1 TỔNG QUAN

Trong thời đại ngày nay, với sự hội nhập mạnh mẽ của kinh tế quốc tế, nhu cầu tìm kiếm việc làm ngày càng trở nên quan trọng và tăng cao. Tuy nhiên, nhiều người đối mặt với thách thức lớn khi cố gắng tìm kiếm công việc phù hợp với năng lực, kinh nghiệm, và sở thích cá nhân của họ. Điều này đặt ra một thách thức không nhỏ khi người lao động cần một cách hiệu quả để tìm kiếm và xin việc làm.

Đối mặt với những thách thức này, tôi đã quyết định xây dựng website tìm kiếm việc làm bằng ReactJS để xây dựng giao diện người dùng đẹp mắt và linh hoạt. Việc này giúp người dùng dễ dàng tương tác với trang web, đồng thời cung cấp trải nghiệm mượt mà. Qua đó, chúng tôi tối ưu hóa hiệu suất và giảm thời gian tải trang, mang lại sự thoải mái khi sử dụng. Đối với phần backend, tôi đã chọn Node.js với tính không đồng bộ mạnh mẽ. Điều này giúp xử lý đồng thời nhiều yêu cầu từ người dùng mà không ảnh hưởng đến hiệu suất. Node.js không chỉ mạnh mẽ mà còn linh hoạt, giúp tôi dễ dàng mở rộng và tích hợp các dịch vụ khác nhau thông qua RESTful API. Việc tích hợp RESTful API giúp tôi tạo ra một giao tiếp mượt mà giữa frontend và backend. Điều này cung cấp sự linh hoạt và khả năng mở rộng, cho phép tôi tích hợp các dịch vụ mới và mở rộng chức năng một cách dễ dàng.

Sau khi thực hiện đề tài xây dựng website tìm kiếm việc làm bằng ReactJS, tôi đã đạt được những kết quả sau: Giao diện người dùng được xây dựng bằng ReactJS mang lại trải nghiệm mượt mà và thân thiện. Cho phép người tìm việc và nhà tuyển dụng tạo tài khoản và đăng nhập vào trang web để sử dụng các tính năng khác. Tìm kiếm và lọc công việc được thiết kế linh hoạt, giúp người tìm việc dễ dàng tìm thấy công việc phù hợp với nhu cầu của mình. Người tìm việc có thể tìm và ứng tuyển công việc phù hợp với bản thân. Quản lý hồ sơ ứng tuyển được thực hiện một cách thuận tiện và hiệu quả.

Để xây dựng đề tài này tôi đã thực hiện các bước sau:

Bước 1: Nghiên cứu công nghệ

Bước đầu tiên là nghiên cứu Reactjs, bao gồm các kiến thức và kỹ năng cần thiết để xây dựng website bằng ReactJS. Các kiến thức và kỹ năng cần nghiên cứu như tổng quan về ReactJS, cài đặt và cấu hình ReactJS, sử dụng các component, các state và props trong ReactJS, sử dụng ReactJS với API. Tiếp theo là nghiên cứu Node.js với framework

Express.js, tìm hiểu và xây dựng API để xử lý các yêu cầu từ người dùng, kết nối với cơ sở dữ liệu MySQL.

Bước 2: Nghiên cứu về website tìm kiếm việc làm.

Sau khi nghiên cứu ReactJS, cần nghiên cứu về website tìm kiếm việc làm, bao gồm nhu cầu và yêu cầu của người dùng đối với website tìm kiếm việc làm. Các nhu cầu và yêu cầu cần nghiên cứu bao gồm: Giao diện website, tính năng website, bảo mật website.

Bước 3: Thiết kế và xây dựng website.

Trên cơ sở nghiên cứu ReactJS và nghiên cứu về website tìm kiếm việc làm, tiến hành thiết kế và xây dựng website. Các công việc cần thực hiện bao gồm: Thiết kế giao diện website, lập trình website, kiểm thử website.

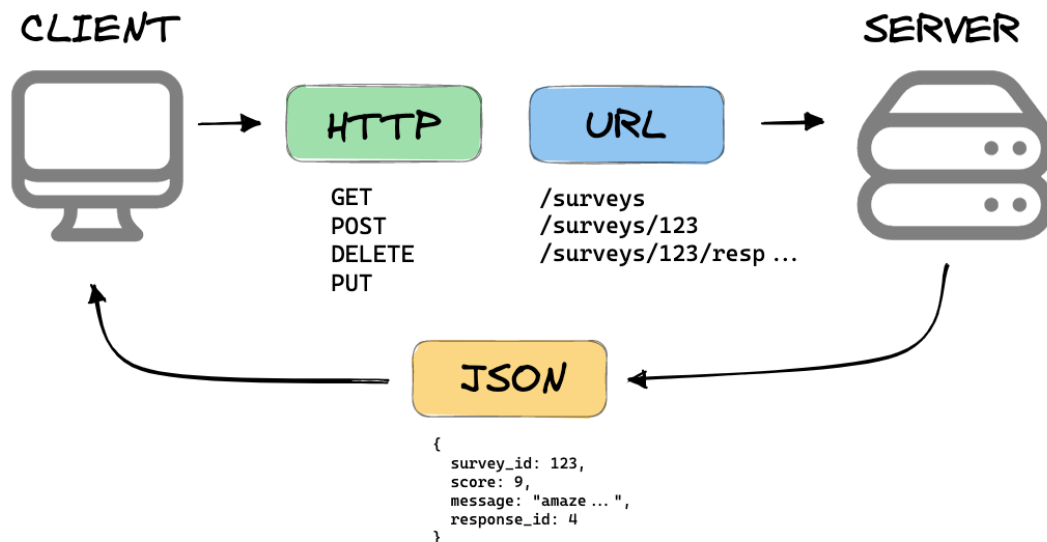
Bước 4: Triển khai website .

Sau khi hoàn thành việc xây dựng website, cần triển khai website lên môi trường sản xuất. Cần xác định rõ nhu cầu và yêu cầu của người dùng trước khi bắt đầu thiết kế và xây dựng website. Cần kiểm thử website thường xuyên để đảm bảo website hoạt động bình thường.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

2.1 Cấu trúc Web API

Web API (viết tắt của Application Programming Interface) [1] là một phương thức được sử dụng để các website hay ứng dụng web khác nhau có thể trao đổi thông tin, dữ liệu qua lại. Web API cung cấp khả năng truy xuất đến một tập các hàm hay dùng và từ đó có thể trao đổi dữ liệu giữa các ứng dụng. Web API thường ứng dụng trong các hệ thống website, cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. Web API hoạt động thông qua giao thức HTTP hoặc HTTPS và trả lại dữ liệu ở dạng JSON hoặc XML. Web API có nhiều ưu điểm như khả năng tích hợp linh động, cập nhật thông tin thời gian thực, có tiêu chuẩn chung dễ sử dụng, và hoạt động trên nhiều thiết bị internet khác nhau như trình duyệt web, ứng dụng di động. Tuy nhiên, Web API cũng có nhược điểm như bảo mật dữ liệu, khả năng xử lý lỗi, và khả năng kiểm soát sử dụng API.



Hình 2.1. Sơ đồ hoạt động của web API

Cấu trúc của một website thường bao gồm hai thành phần chính là:

- Phần frontend của website là phần mà người dùng có thể nhìn thấy và tương tác trực tiếp. Nó bao gồm giao diện, các nút bấm, các trường nhập liệu, các hình ảnh, video, và các hiệu ứng trên trang web. Frontend thường được xây dựng bằng các ngôn ngữ lập trình như HTML, CSS, và JavaScript. Các lập trình viên frontend thường phải có kiến

thức về thiết kế đồ họa, các công nghệ liên quan đến tương tác người dùng, và các công nghệ mới nhất trong lĩnh vực này. Frontend cũng có thể được xây dựng bằng các framework như React, Angular, và Vue.js. Các framework này cung cấp các công cụ và thư viện giúp cho việc phát triển frontend trở nên dễ dàng hơn. Tuy nhiên, việc sử dụng các framework cũng có nhược điểm như tốc độ tải trang chậm và khả năng tùy biến giảm.

- Phần backend của website là một phần quan trọng trong việc xây dựng một trang web hoàn chỉnh. Backend là phần của website mà người dùng không thể nhìn thấy được, nhưng nó đóng vai trò quan trọng trong việc xử lý dữ liệu và tương tác với cơ sở dữ liệu. Backend thường được xây dựng bằng các ngôn ngữ lập trình như Java, Python, Ruby, PHP, và Node.js. Các lập trình viên backend thường phải có kiến thức về cơ sở dữ liệu, các giao thức truyền tải dữ liệu, và các công nghệ liên quan đến bảo mật thông tin. Backend cũng có thể được xây dựng bằng các dịch vụ đám mây như Amazon Web Services, Google Cloud Platform, và Microsoft Azure. Các dịch vụ này cung cấp khả năng lưu trữ dữ liệu, xử lý dữ liệu, và triển khai ứng dụng trên nền tảng đám mây. Tuy nhiên, việc sử dụng dịch vụ đám mây cũng có nhược điểm như chi phí cao và khả năng kiểm soát dữ liệu bị giới hạn.

2.2 Frontend với ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Meta để tạo ra những ứng dụng web hấp dẫn, nhanh và hiệu quả. ReactJS cho phép các doanh nghiệp tạo ra những ứng dụng web với giao diện tốt hơn để nâng cao trải nghiệm người dùng. ReactJS sử dụng mô hình component, nơi mỗi giao diện người dùng được biểu diễn dưới dạng các thành phần độc lập. Điều này không chỉ tăng khả năng tái sử dụng mã, mà còn giúp dễ dàng quản lý, bảo trì mã nguồn và phát triển ứng dụng theo quy mô lớn. ReactJS, với đặc tính linh hoạt và hiệu suất cao, đã trở thành một công cụ quan trọng và ứng dụng rộng rãi trong nhiều lĩnh vực của phát triển website. Dưới đây là một số ứng dụng chính của ReactJS như :

Xây dựng giao diện người dùng linh hoạt: làm cho việc xây dựng giao diện người dùng trở nên linh hoạt và dễ dàng. Thành phần độc lập giúp tái sử dụng mã nguồn, giảm thiểu sự phức tạp của dự án và tăng cường khả năng mở rộng.

Ứng dụng đơn trang (SPA): thích hợp cho việc xây dựng các ứng dụng đơn trang, nơi trang web tải một lần và sau đó tương tác với người dùng mà không cần tải lại trang. Điều này giúp tạo ra trải nghiệm người dùng mượt mà và nhanh chóng.

Giao diện người dùng động và hiệu suất cao: Virtual DOM của ReactJS giúp tối ưu hóa hiệu suất bằng cách giảm tải cho DOM. Sự linh hoạt và khả năng tương tác của giao diện người dùng được cải thiện đáng kể, đặc biệt là trong các ứng dụng đòi hỏi tương tác nhanh và độ mượt mà cao.

Ứng dụng thời gian thực và chat: ReactJS thường được sử dụng trong xây dựng các ứng dụng thời gian thực như chat, nơi dữ liệu phải được cập nhật ngay lập tức và hiệu suất đóng vai trò quan trọng.

Bảng dữ liệu: trong các ứng dụng quản lý dữ liệu và bảng điều khiển, ReactJS giúp tạo ra các giao diện người dùng phức tạp và hiệu suất cao, cho phép người dùng tương tác mạnh mẽ với dữ liệu.

Ứng dụng tìm kiếm và lọc dữ liệu: thư viện tìm kiếm và lọc tích hợp vào ReactJS giúp xây dựng các ứng dụng có khả năng tìm kiếm và lọc linh hoạt, đáp ứng nhanh chóng đối với nhu cầu người dùng.

Hệ thống CMS (Content Management System): ReactJS được sử dụng rộng rãi trong các hệ thống quản lý nội dung (CMS) và trang web đòi hỏi tương tác cao, nhanh chóng và hiệu suất là chìa khóa cho trải nghiệm người dùng tích cực.

Một số ứng dụng cụ thể của ReactJS như là: Facebook, Instagram, Netflix, New York Times, ... đều sử dụng ReactJS để xây dựng giao diện người dùng cho trang web và ứng dụng di động của mình.

2.3 Backend với Node.js

Node.js là một nền tảng xây dựng ứng dụng mạng có hiệu suất cao dựa trên JavaScript runtime của Chrome V8. Nó cho phép lập trình viên sử dụng JavaScript để viết mã cho cả phía server và client, đồng thời cung cấp khả năng xử lý không đồng bộ và lập trình sự kiện, tối ưu cho ứng dụng web thời gian thực. Nó là một nền tảng JavaScript thuần túy điều này có nghĩa là có thể sử dụng tất cả các kiến thức JavaScript để phát triển ứng dụng web. Node.js có thể xử lý nhiều yêu cầu cùng một lúc một cách hiệu quả. Express.js là một framework của Node.js, cung cấp một cách linh hoạt và đơn giản để xây

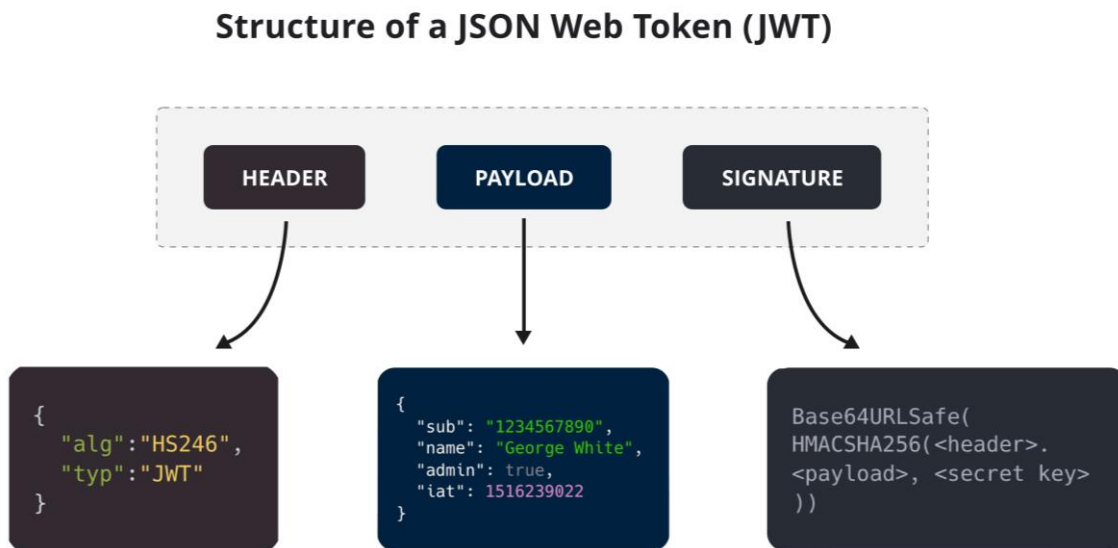
dựng các ứng dụng web và API. Nó giảm đi sự phức tạp trong quá trình phát triển, giúp tăng cường hiệu suất và hiệu quả. Khi sử dụng cùng nhau, Node.js và Express.js tạo thành một cơ sở hạ tầng mạnh mẽ cho việc xây dựng các ứng dụng web và API hiệu quả. Node.js giúp xử lý không đồng bộ và Express.js giúp giảm độ phức tạp, đồng thời cung cấp các tính năng cần thiết cho việc phát triển ứng dụng một cách nhanh chóng và dễ dàng.

Việc sử dụng Node.js và Express.js mang lại nhiều lợi ích, bao gồm:

- Tăng hiệu suất: Node.js có thể xử lý nhiều yêu cầu cùng một lúc một cách hiệu quả, giúp tăng hiệu suất của ứng dụng web.
- Tăng khả năng mở rộng: Node.js có thể được mở rộng dễ dàng để đáp ứng nhu cầu phát triển của ứng dụng web.
- Tăng tính bảo mật: Node.js cung cấp một số tính năng bảo mật tích hợp, giúp bảo vệ ứng dụng web của bạn khỏi các cuộc tấn công.
- Dễ dàng phát triển: Node.js và Express.js là các nền tảng dễ học và sử dụng, giúp dễ dàng bắt đầu phát triển ứng dụng web.

2.4 Xác thực JSON Web Token (JWT)

JSON Web Token (JWT) [2] là một chuỗi mã hóa được sử dụng để truyền tải thông tin giữa các ứng dụng web và mobile. JWT là một chuỗi JSON được mã hóa bằng phương pháp mã hóa, trở thành một chuỗi ký tự lộn xộn. JWT cung cấp khả năng xác thực và đánh dấu tin cậy thông qua “chữ ký” của nó. JWT được sử dụng trong phương thức xác thực dựa trên token (Token-based authentication) để xác minh quyền truy cập vào tài nguyên mà không cần phải cung cấp lại username/password. JWT bao gồm ba phần chính: Header, Payload, và Signature. Header chứa thông tin về loại token và thuật toán đã dùng để mã hóa. Payload chứa thông tin về người dùng, chẳng hạn như tên người dùng, vai trò người dùng và thời hạn hiệu lực của token. Signature được sử dụng để xác minh tính xác thực của token. JWT có nhiều ưu điểm như dễ sử dụng, tích hợp linh động, và hoạt động trên nhiều thiết bị internet khác nhau như trình duyệt web, ứng dụng di động. Tuy nhiên, JWT cũng có nhược điểm như bảo mật dữ liệu, khả năng xử lý lỗi, và khả năng kiểm soát sử dụng API.



Hình 2.2 Hình ảnh cấu trúc của JSON Web Token

Cách thức hoạt động của JWT:

- Khi người dùng đăng nhập vào ứng dụng, máy chủ sẽ xác thực người dùng và tạo một token cho người dùng. Token này sẽ được lưu trữ trong trình duyệt của người dùng.
- Khi người dùng truy cập ứng dụng, họ sẽ gửi token này đến máy chủ. Máy chủ sẽ xác minh tính xác thực của token và cấp quyền cho người dùng truy cập ứng dụng.

Có ba loại lưu trữ token phổ biến như sau:

- Lưu trữ JWT trong Memory (Bộ Nhớ): Khi token được tạo ra, nó chỉ tồn tại trong bộ nhớ của ứng dụng. Token sẽ mất đi khi ứng dụng bị tắt hoặc khi trình duyệt đóng. Ưu điểm là an toàn vì không lưu trên đĩa cứng, không nguy cơ bị lộ thông tin khi trình duyệt đóng. Nhược điểm, không giữ được trạng thái đăng nhập khi làm mới trang hoặc đóng trình duyệt.
- Lưu trữ JWT trong Cookie: Token được gửi về máy khách và lưu trữ trong một cookie. Mỗi lần yêu cầu được gửi đến server, cookie sẽ được đính kèm. Dữ liệu được giữ nguyên giữa các phiên làm việc, có thể được gửi tự động trong các yêu cầu HTTP. Nhược điểm, nguy cơ tấn công Cross-Site Scripting (XSS) nếu cookie không được cấu hình chặt chẽ.

- Lưu trữ JWT trong Local Storage hoặc Session Storage: Token được lưu trữ trong localStorage hoặc sessionStorage của trình duyệt. Dữ liệu được giữ nguyên giữa các phiên làm việc, có thể lưu nhiều thông tin hơn so với cookie. Nhược điểm, nguy cơ bị tấn công XSS và CSRF. LocalStorage có thể truy cập bằng JavaScript từ một số trang web bị tấn công.

Tôi đã chọn lưu trữ JWT trong Cookie vì dữ liệu sẽ được lưu trữ trên máy tính của người dùng giúp cải thiện hiệu suất của ứng dụng, vì nó không cần phải truy cập máy chủ mỗi khi người dùng cần truy cập dữ liệu. Cookies tương đối an toàn. Dữ liệu trong Cookies được mã hóa, giúp bảo vệ khỏi bị truy cập trái phép. Có nhiều thư viện và khung có sẵn để giúp bạn làm việc với Cookies. Điều này có thể giúp tôi dễ dàng triển khai và bảo trì việc sử dụng Cookies trong ứng dụng của mình.

CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Đặc tả yêu cầu hệ thống

3.1.1 Yêu cầu chức năng

Trang tìm kiếm việc làm này phục vụ đồng thời hai đối tượng chính: Người tìm việc và nhà tuyển dụng.

Đối với người tìm việc có các chức năng sau:

- **Đăng ký tài khoản, đăng nhập, đăng xuất:** Người tìm việc có thể đăng nhập vào tài khoản hiện có hoặc đăng ký tài khoản mới để sử dụng các tính năng cá nhân và tận hưởng trải nghiệm tốt hơn. Thực hiện các biện pháp bảo mật như mã hóa mật khẩu để đảm bảo an toàn thông tin cá nhân của nhà tuyển dụng.
- **Tìm kiếm công việc:** Giao diện tìm kiếm cho phép người tìm việc nhập các tiêu chí như địa điểm, ngành nghề, từ khóa. Kết quả tìm kiếm sẽ hiển thị danh sách công việc phù hợp.
- **Lọc và sắp xếp:** Người tìm việc có thể lọc kết quả tìm kiếm dựa trên các tiêu chí khác như ngành nghề, mức lương, địa điểm, loại công việc, kinh nghiệm và trình độ học vấn. Sắp xếp kết quả theo thứ tự mong muốn như bài đăng mới nhất, cũ nhất, ...
- **Xem thông tin chi tiết việc làm:** Người tìm việc có thể xem thông tin chi tiết về từng vị trí việc làm bằng cách nhấp vào việc làm đó. Thông tin chi tiết bao gồm mô tả công việc, yêu cầu công việc, quyền lợi, ...
- **Theo dõi nhà tuyển dụng:** là một công cụ quan trọng giúp người tìm việc duy trì sự kết nối và theo dõi các thông tin việc làm từ nhà tuyển dụng mà họ quan tâm. Người tìm việc có thể xem danh sách các nhà tuyển dụng đã đăng ký trên trang web.
- **Lưu việc làm:** chức năng này giúp người dùng quản lý và theo dõi các vị trí việc làm mà họ quan tâm. Nếu không quan tâm nữa, người dùng có thể hủy lưu việc làm khỏi danh sách.
- **Nộp hồ sơ ứng tuyển:** Người tìm việc có thể nộp hồ sơ ứng tuyển cho các vị trí việc làm bằng cách nhấp vào nút "Ứng tuyển". Người tìm việc cần nhập thông tin ứng tuyển như thông tin cá nhân và đính kèm tập tin cv.

Đối với nhà tuyển dụng có các chức năng sau:

- **Đăng ký tài khoản, đăng nhập, đăng xuất:** Cung cấp giao diện đăng nhập bảo mật để nhà tuyển dụng nhập thông tin đăng nhập của mình, bao gồm email và mật khẩu. Thực hiện các biện pháp bảo mật như mã hóa mật khẩu để đảm bảo an toàn thông tin cá nhân của nhà tuyển dụng.
- **Đăng tin tuyển dụng:** Nhà tuyển dụng có thể đăng thông tin tuyển dụng, đăng mọi chi tiết về công việc của mình, từ mô tả công việc đến các yêu cầu và mong muốn về ứng viên. Điều này giúp tạo ra một hình ảnh rõ ràng về vị trí làm việc và thu hút những ứng viên phù hợp.
- **Quản lý việc làm:** là công cụ được thiết kế đơn giản và tối ưu quá trình tuyển dụng của nhà tuyển dụng. Nó mang lại sự linh hoạt và tiện ích, giúp nhà tuyển dụng dễ dàng theo dõi, quản lý thông tin về các vị trí công việc một cách hiệu quả. Cung cấp tính năng sửa đổi linh hoạt, cho phép nhà tuyển dụng điều chỉnh thông tin chi tiết về công việc một cách dễ dàng. Cho phép xóa hoặc dừng ứng tuyển các công việc.
- **Xem hồ sơ ứng viên:** Hiển thị danh sách hồ sơ ứng viên cho các vị trí việc làm. Nhà tuyển dụng sử dụng để đánh giá và tương tác với ứng viên một cách chi tiết và có hiệu quả. Bằng cách này, họ có thể đảm bảo rằng họ tìm thấy những người có kỹ năng và sở thích phù hợp nhất cho vị trí công việc cụ thể.

3.1.2 Yêu cầu phi chức năng

Giao diện thân thiện, dễ sử dụng: Giao diện website cần được thiết kế đẹp mắt, dễ sử dụng và phù hợp với người dùng. Sử dụng các thành phần UI/UX phổ biến giúp người dùng dễ dàng tiếp cận và sử dụng website. Sử dụng màu sắc, font chữ phù hợp giúp website trở nên đẹp mắt và dễ nhìn. Sử dụng bố cục hợp lý giúp người dùng dễ dàng tìm thấy thông tin cần thiết.

Giao diện responsive: Website cần có giao diện responsive để có thể sử dụng trên các thiết bị di động. Giao diện responsive giúp người dùng có thể sử dụng website một cách thuận tiện trên các thiết bị di động có kích thước màn hình khác nhau.

Hiệu suất cao: Website cần có hiệu suất cao để người dùng có thể sử dụng một cách mượt mà.

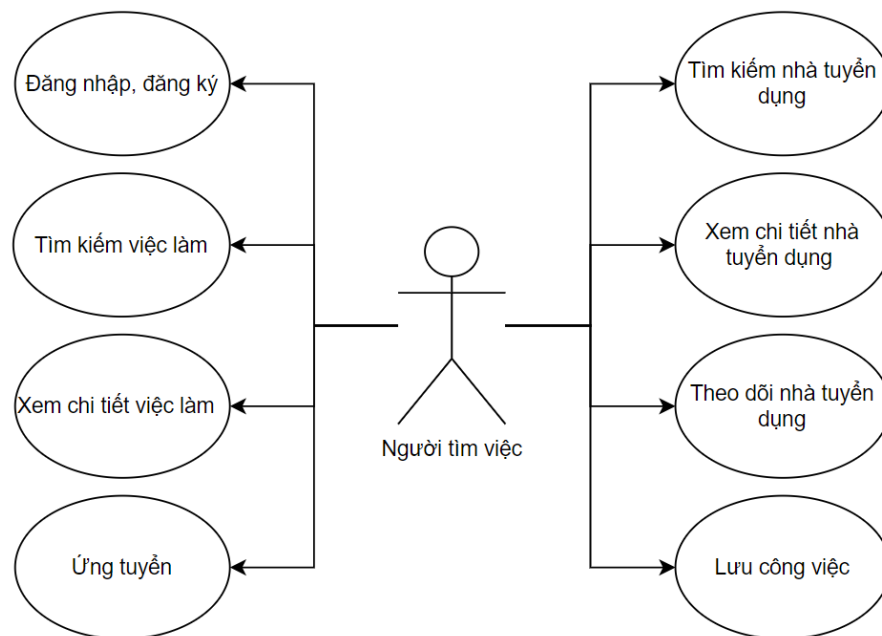
Tính khả dụng: Website cần hoạt động ổn định, không bị gián đoạn. Website cần được kiểm tra và bảo trì thường xuyên để đảm bảo hoạt động ổn định.

Khả năng mở rộng: là một yêu cầu quan trọng của website. Tính khả năng mở rộng cho phép website có thể được mở rộng thêm các tính năng mới hoặc tăng quy mô hoạt động mà không cần thay đổi đáng kể cấu trúc hiện tại của website.

3.2 Sơ đồ use-case

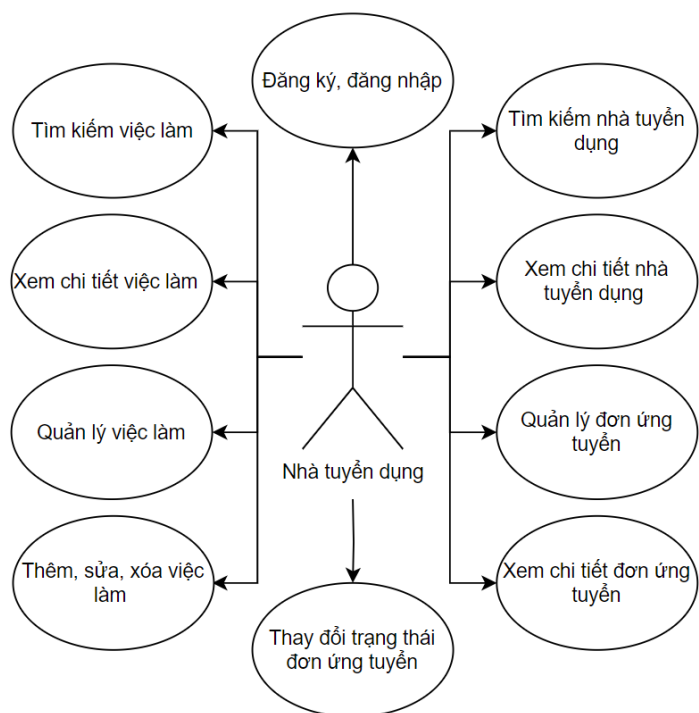
Sơ đồ use-case là một công cụ trong phát triển phần mềm được sử dụng để mô tả cách mà một hệ thống sẽ tương tác với các thành phần khác nhau, bao gồm người dùng và các thành phần hệ thống khác. Nó giúp hiểu rõ các yêu cầu và chức năng của hệ thống thông qua việc biểu diễn các tình huống và tương tác chính giữa các thực thể. Mỗi use-case biểu thị một tác vụ hoặc chức năng cụ thể mà hệ thống cung cấp để đáp ứng nhu cầu của người dùng hoặc hệ thống. Sơ đồ use-case giúp xác định và hiểu rõ cách người dùng và hệ thống tương tác với nhau. Nó là một công cụ hữu ích trong việc xây dựng và hiểu cấu trúc chức năng của một hệ thống phần mềm và làm cơ sở cho quá trình thiết kế và phát triển.

3.2.1 Sơ đồ use-case của người tìm việc



Hình 3.1 Sơ đồ use-case của người tìm việc.

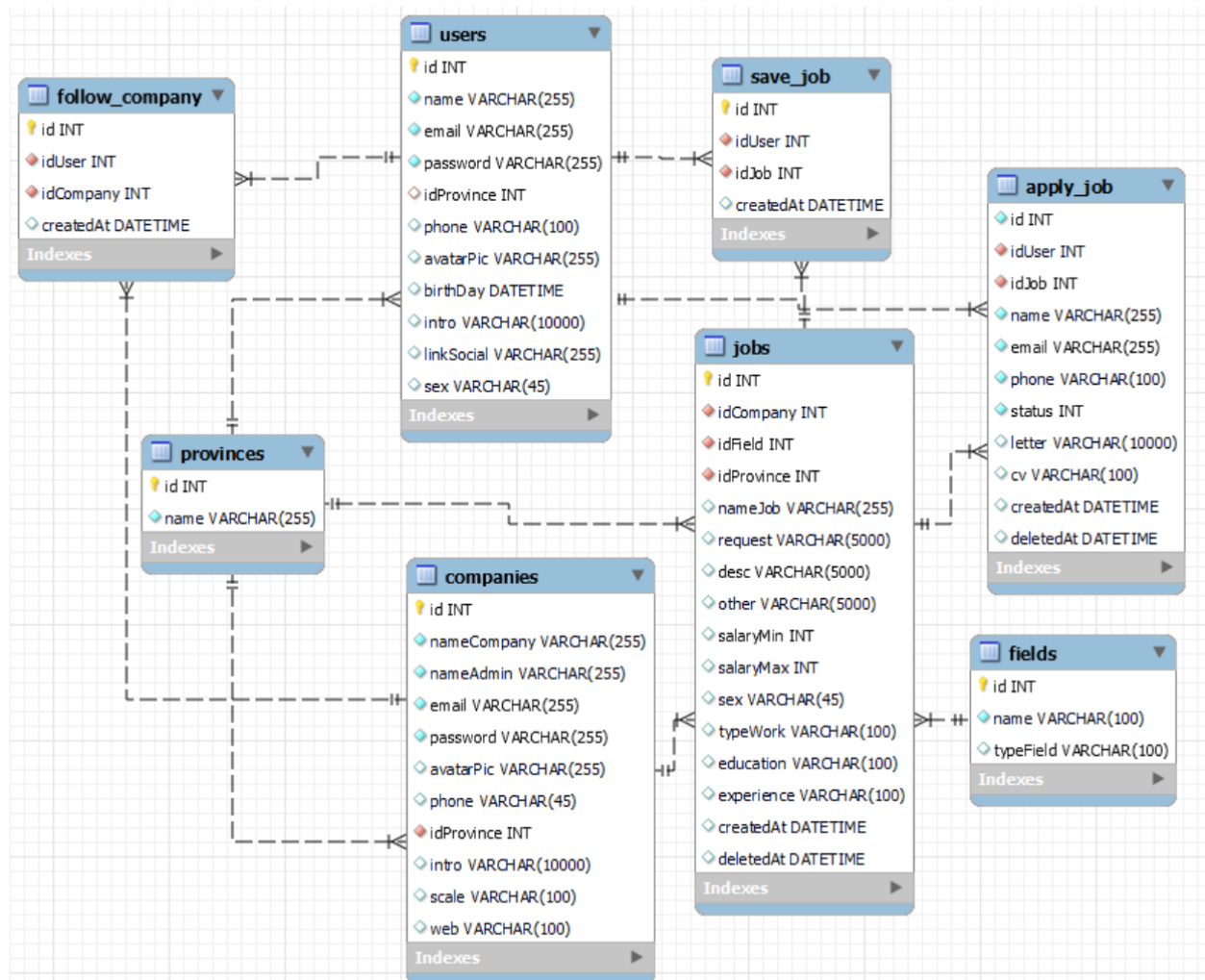
3.2.2 Sơ đồ use-case của người nhà tuyển dụng



Hình 3.2 Sơ đồ use-case nhà tuyển dụng.

3.3 Thiết kế dữ liệu

3.3.1 Lược đồ cơ sở dữ liệu



Hình 3.3 Lược đồ cơ sở dữ liệu.

Bảng 3.1 Danh sách các thực thể

STT	Tên thực thể	Diễn giải
1	users	Ứng viên
2	companies	Nhà tuyển dụng
3	jobs	Công việc
4	provinces	Tỉnh thành
5	fields	Ngành nghề

6	apply_job	Ứng tuyển
7	save_job	Lưu công việc
8	follow_company	Theo dõi nhà tuyển dụng

3.3.2 Chi tiết các thực thể

Tên thực thể: **user**

Mô tả: Lưu trữ thông tin ứng viên

Chi tiết thực thể:

Bảng 3.2 Chi tiết thực thể “**user**”: Ứng viên

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id ứng viên	integer	Khóa chính
2	name	Họ tên	varchar	
3	email	Email	varchar	
4	phone	Số điện thoại	varchar	
5	password	Mật khẩu	varchar	
6	idProvince	Địa chỉ	integer	Khóa ngoại
7	avatarPic	Hình đại diện	varchar	
8	brithday	Ngày sinh	datetime	
9	intro	Giới thiệu	varchar	
10	linkSocial	Địa chỉ trang mạng xã hội	varchar	
11	sex	Giới tính	varchar	

Tên thực thể: **companies**

Mô tả: Lưu trữ thông tin nhà tuyển dụng

Chi tiết thực thể:

Bảng 3.3 Chi tiết thực thể “**companies**”: Nhà tuyển dụng

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id ứng viên	integer	Khóa chính

2	nameCompany	Tên công ty tuyển dụng	varchar	
3	nameAdmin	Tên người tuyển dụng	varchar	
4	email	Email	varchar	
5	password	Mật khẩu	varchar	
6	avatarPic	Hình đại diện	varchar	
7	phone	Số điện thoại	varchar	
8	idProvince	Địa chỉ	integer	Khóa ngoại
9	intro	Giới thiệu	varchar	
10	scale	Quy mô	varchar	
11	web	Địa chỉ trang web nhà tuyển dụng	varchar	

Tên thực thể: **jobs**

Mô tả: Lưu trữ thông tin công việc

Chi tiết thực thể:

Bảng 3.4 Chi tiết thực thể “**jobs**”: Công việc

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id công việc	integer	Khóa chính
2	idCompany	Công ty	integer	Khóa ngoại
3	idField	Ngành nghề	integer	Khóa ngoại
4	idProvince	Địa chỉ	integer	Khóa ngoại
5	nameJob	Tên công việc	varchar	
6	request	Yêu cầu	varchar	
7	desc	Mô tả công việc	varchar	
8	other	Mô tả thêm công việc	varchar	
9	salaryMin	Lương thấp nhất	integer	

10	salaryMax	Lương cao nhất	integer	
11	sex	Giới tính	varchar	
12	typeWork	Thể loại làm việc	varchar	
13	Education	Trình độ học vấn	varchar	
14	experience	Kinh nghiệm	varchar	
15	createdAt	Ngày tạo	datetime	
16	deletedAt	Ngày xóa	datetime	

Tên thực thể: **provinces**

Mô tả: Lưu trữ thông tin tỉnh thành

Chi tiết thực thể:

Bảng 3.5 Chi tiết thực thể “**provinces**”: Tỉnh thành

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id tỉnh thành	integer	Khóa chính
2	name	Tên tỉnh thành	varchar	

Tên thực thể: **fields**

Mô tả: Lưu trữ thông tin ngành nghề

Chi tiết thực thể:

Bảng 3.6 Chi tiết thực thể “**fields**”: Ngành nghề

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id ngành nghề	integer	Khóa chính
2	name	Tên	varchar	
3	typeField	Loại ngành nghề	varchar	

Tên thực thể: **apply_job**

Mô tả: Lưu trữ thông tin ứng tuyển

Chi tiết thực thể:

Bảng 3.7 Chi tiết thực thể “**apply_job**”: Ứng tuyển

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id ngành nghề	integer	Khóa chính
2	idUser	Người dùng	integer	Khóa ngoại
3	idJob	Công việc	integer	Khóa ngoại
4	name	Tên ứng viên	varchar	
5	email	Email ứng viên	varchar	
6	phone	Số điện thoại ứng viên	varchar	
7	status	Trạng thái của đơn ứng tuyển	integer	
8	letter	Thư	varchar	
9	cv	Địa chỉ cv	varchar	
10	createdAt	Ngày tạo	datetime	
11	deletedAt	Ngày xóa	datetime	

Tên thực thể: **save_job**

Mô tả: Lưu trữ thông tin lưu việc làm

Chi tiết thực thể:

Bảng 3.8 Chi tiết thực thể “**save_job**”: Lưu việc làm

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id lưu việc làm	integer	Khóa chính
2	idUser	Người dùng	integer	Khóa ngoại
3	idJob	Công việc	integer	Khóa ngoại
4	createdAt	Ngày tạo	datetime	

Tên thực thể: **follow_company**

Mô tả: Lưu trữ thông tin theo dõi nhà tuyển dụng

Chi tiết thực thể:

Bảng 3.9 Chi tiết thực thể “**follow_company**”: Theo dõi nhà tuyển dụng

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Ràng buộc toàn vẹn
1	id	Id theo dõi nhà tuyển dụng	integer	Khóa chính
2	idUser	Người dùng	integer	Khóa ngoại
3	idCompany	Nhà tuyển dụng	integer	Khóa ngoại
4	createdAt	Ngày tạo	datetime	

3.4 Thiết kế, kiến trúc ứng dụng

3.4.1 Kiến trúc toàn cảnh

Website được xây dựng dựa trên kiến trúc RESTful API, RESTful API [3] (viết tắt của Representational State Transfer API) là một kiểu thiết kế API được sử dụng để xây dựng các ứng dụng web. RESTful API sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để thực hiện các thao tác trên tài nguyên, cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. RESTful API hoạt động thông qua giao thức HTTP hoặc HTTPS và trả lại dữ liệu ở dạng JSON hoặc XML.

Kiến trúc ứng dụng có thể được mô tả như sau:

- Database: Sử dụng cơ sở dữ liệu (MySQL) lưu trữ liên quan đến người tìm việc (user), nhà tuyển dụng, thông tin công việc, và các thông tin khác liên quan. Thiết kế các bảng cho mỗi đối tượng (users, jobs, companies, ...). Quan hệ giữa các bảng được thiết lập một cách hợp lý để hỗ trợ các truy vấn hiệu quả.

- Backend : Sử dụng Express.js, RESTful API xây dựng các API cho việc tương tác giữa frontend và backend. Cung cấp các dịch vụ như đăng nhập, đăng ký, tìm kiếm công việc, đăng tin tuyển dụng, và quản lý hồ sơ người dùng. Sử dụng middleware để xử lý yêu cầu HTTP, kiểm soát quyền truy cập, và thực hiện các chức năng bảo mật như xác thực JWT. Controllers: Các controllers xử lý logic kinh doanh, gọi database để truy vấn và cập nhật dữ liệu. Controllers giữ vai trò chính trong việc điều phối luồng thông tin giữa frontend và database.

- Frontend: Sử dụng ReactJS xây dựng các thành phần để hiển thị giao diện người dùng. Các thành phần này bao gồm trang chủ, trang tìm kiếm, trang chi tiết công việc,

trang đăng ký, đăng nhập, và các thành phần khác. Sử dụng React Router để quản lý định tuyến giữa các trang. Gọi các API từ backend để nhận và gửi dữ liệu giữa frontend và backend. Tích hợp responsive design để đảm bảo trang web hiển thị đẹp mắt và dễ sử dụng trên mọi thiết bị.

Kiến trúc này giúp phân chia rõ ràng giữa các thành phần chính của ứng dụng, tạo sự đồng bộ và linh hoạt trong quá trình phát triển và bảo trì. Frontend và backend hoạt động cùng nhau thông qua các API để cung cấp trải nghiệm người dùng mượt mà và tích hợp với cơ sở dữ liệu để lưu trữ và truy vấn thông tin một cách hiệu quả.

3.4.2 Cách frontend nhận dữ liệu

Để nhận dữ liệu từ backend theo phương thức RESTful API trong ReactJS, có thể sử dụng các phương thức HTTP như GET, POST, PUT hoặc DELETE. Cụ thể, frontend gửi request đến backend thông qua phương thức HTTP và địa chỉ URL của API. Backend sẽ xử lý request và trả về response cho frontend. Dữ liệu được truyền tải giữa frontend và backend thông qua các định dạng dữ liệu như JSON hoặc XML. Sau khi nhận được response từ backend, frontend sẽ xử lý dữ liệu và hiển thị trên giao diện người dùng.

Ví dụ tệp component.js phía frontend (ReactJS):

```
1
2 import React, { useEffect, useState } from "react";
3
4 const Component = () => {
5   const [data, setData] = useState([]);
6
7   useEffect(() => {
8     // Gửi yêu cầu GET đến backend
9     fetch("http://localhost:3000/users")
10      .then((response) => response.json())
11      .then((data) => setData(data));
12   }, []);
13
14   return (
15     <div>
16       {data.map((user) => (
17         <div key={user.id}>{user.name}</div>
18       ))}
19     </div>
20   );
21 };
22
23 export default Component;
```

Hình 3.4 Tệp component.js phía frontend

Ví dụ trên, component sử dụng phương thức *useEffect* để gửi yêu cầu GET đến backend khi component được render. Yêu cầu GET được gửi đến URL *http://localhost:3000/users*. Khi phản hồi được trả về, dữ liệu được chuyển đổi thành JSON và lưu trữ trong biến data. Dữ liệu được sử dụng để tạo các thẻ div trong DOM.

Phía backend, có thể sử dụng hàm *app.get()* hoặc *app.post()* để định nghĩa các tuyến đường xử lý yêu cầu từ frontend.

Ví dụ tệp *index.js* từ phía backend (express.js) :

```
1  const express = require("express");
2
3  const app = express();
4
5  // Định nghĩa tuyến đường /users
6  app.get("/users", (req, res) => {
7    // Lấy dữ liệu người dùng từ cơ sở dữ liệu
8    // ...
9
10   // Trả về dữ liệu người dùng
11   res.send(data);
12 });
13
14 app.listen(3000, () => {
15   console.log("Ứng dụng đang chạy trên cổng 3000");
16 });
```

Hình 3.5 Tệp *index.js* từ phía backend

Trong ví dụ trên, tuyến đường */users* được định nghĩa để xử lý yêu cầu *GET*. Khi máy khách gửi yêu cầu GET đến URL *http://localhost:3000/users*, hàm *app.get()* sẽ được thực thi. Hàm này sẽ lấy dữ liệu người dùng từ cơ sở dữ liệu và trả về dữ liệu dưới dạng JSON.

3.4.3 Thiết kế API

Thiết kế API là một trong những yếu tố quan trọng nhất để tạo ra một ứng dụng hiệu quả và dễ bảo trì. Cấu trúc API sẽ giúp tôi tổ chức các tập tin của mình một cách hợp lý, giúp dễ dàng tìm thấy các tập tin cần tìm và giúp dễ dàng thêm các tính năng mới vào ứng dụng của mình. Cấu trúc API được tôi sử dụng như sau :

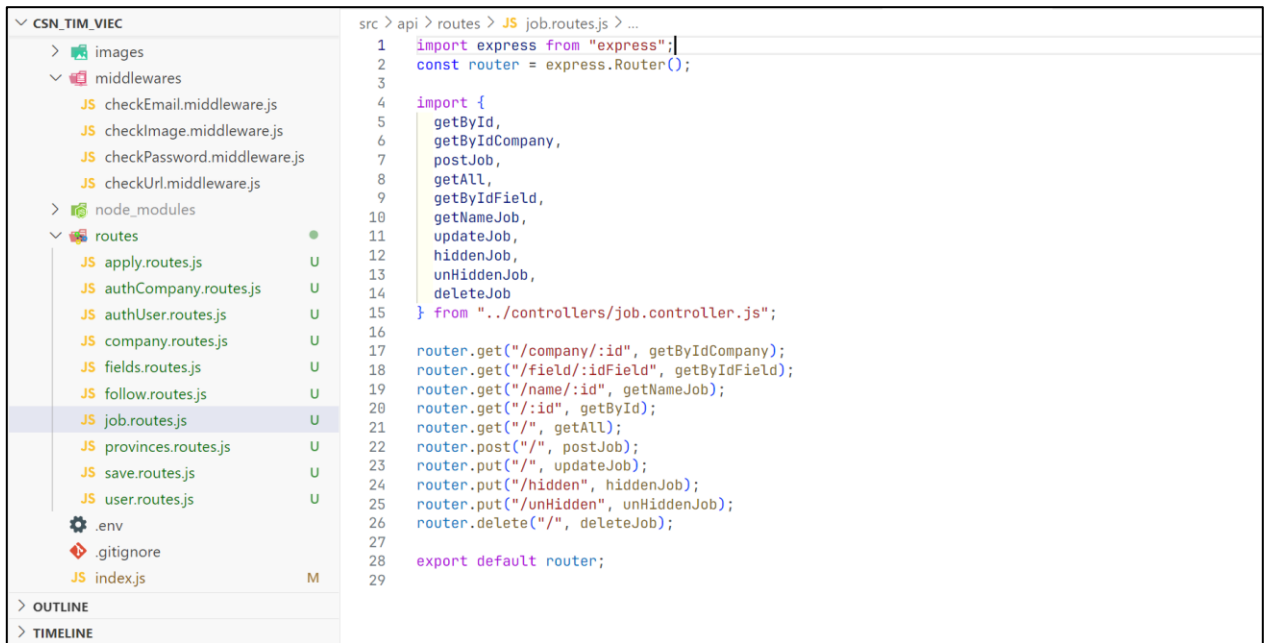
Tập tin *index.js*: các API sẽ được định nghĩa trong tập tin *index.js* để xử lý các yêu cầu từ phía người dùng gửi lên. Các API này sẽ cung cấp các chức năng khác nhau cho người dùng, chẳng hạn như xác thực, truy cập dữ liệu,... Các hàm xử lý yêu cầu sẽ được định nghĩa trong các thư mục *routes* tương ứng. Ví dụ, hàm xử lý yêu cầu cho

`/api/job` sẽ được định nghĩa trong thư mục `routes` với tập tin là `job.routes.js`. Các API được tôi cấu trúc trong tập tin `index.js` như sau:



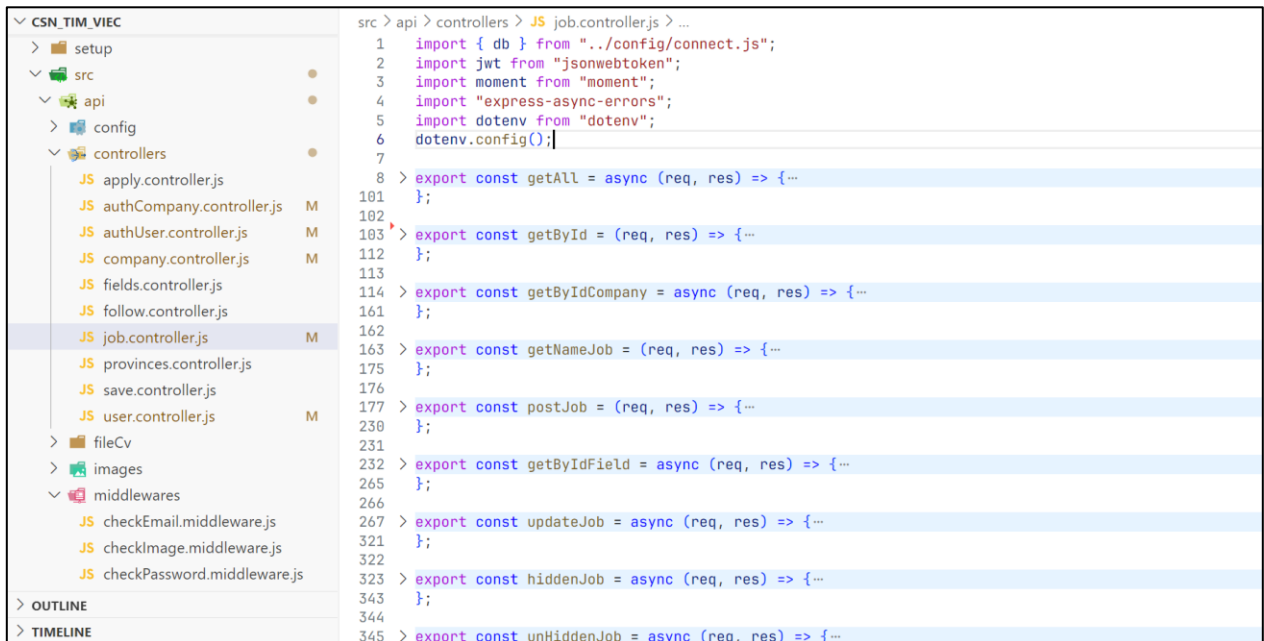
Hình 3.6 Cấu trúc tập tin `index.js`

Thư mục `routes`: Thư mục này chứa các tập tin route. Mỗi tập tin route sẽ định tuyến các URL và các hàm xử lý tương ứng cho các yêu cầu của người dùng. Mỗi tập tin route sẽ định nghĩa các route cho một chức năng cụ thể của ứng dụng, chẳng hạn như lấy dữ liệu, thêm dữ liệu, cập nhật dữ liệu hay xóa dữ liệu. Cấu trúc các tập tin trong thư mục `routes` được tôi sử dụng như sau:



Hình 3.7 Cấu trúc thư mục routes

Thư mục controllers: Thư mục này chứa các file controller. Mỗi file controller sẽ chứa các hàm xử lý các yêu cầu của người dùng liên quan đến một chức năng cụ thể của ứng dụng. Với thư mục controllers được tôi sử dụng cấu trúc như sau:



Hình 3.8 Cấu trúc thư mục controllers

Ví dụ, khi có API được gọi với đường dẫn là `/api/jobs/:id`. Tham số id trong đường dẫn được sử dụng để xác định id của job cần lấy. Tập tin `index.js` sẽ gọi hàm `jobRouter` trong tập tin `job.routes.js` khi đường dẫn này được gọi.

```
1 import express from "express";
2 const app = express();
3 import jobRouter from "../routes/job.routes.js"
4
5 app.use("/api/job", jobRouter);
6
7
```

Hình 3.9 Tập tin *index.js*

Trong tập tin *job.routes.js* sẽ xác định phương thức được gọi, với phương thức GET tập tin sẽ gọi hàm xử lý *getById* trong file controller *job.controller.js*.

```
1 import express from "express";
2 const router = express.Router();
3 import {getById} from "../controllers/job.controller.js";
4
5 router.get("/:id", getById);
6
7 export default router;
8
```

Hình 3.10 Tập tin *job.routes.js*

Hàm *getById* này sẽ sử dụng tham số id trong đường dẫn để thực hiện truy vấn cơ sở dữ liệu để lấy thông tin về job có id đó. Nếu job không tồn tại, hàm sẽ trả về mã trạng thái HTTP 404. Nếu job tồn tại, hàm sẽ trả về thông tin về job dưới dạng JSON.

```
1 import { db } from "../config/connect.js";
2
3 export const getById = (req, res) => {
4   const q = "SELECT * FROM jobs WHERE id = ?";
5
6   db.query(q, req.params.id, (err, data) => {
7     if (err) res.status(404).json({ message: "Job not found" });
8     return res.status(200).json(data[0]);
9   });
10 };
11
```

Hình 3.11 Tập tin *job.controller.js*

3.4.4 Thiết kế frontend

Thiết kế frontend được xây dựng bằng ReactJS. ReactJS sử dụng một mô hình thành phần để tổ chức giao diện, giúp cho việc phát triển và bảo trì trở nên dễ dàng hơn. Thiết kế frontend bằng ReactJS bao gồm các bước sau:

Xác định các thành phần: bước đầu tiên là xác định các thành phần của giao diện. Mỗi thành phần là một đơn vị nhỏ, có thể tái sử dụng, có thể được kết hợp với nhau để tạo thành giao diện hoàn chỉnh. Ví dụ, trang web của tôi có được chia thành các thành phần sau:

- Header: Chứa logo, menu, và các thông tin chung về trang web.
- Main: Chứa nội dung chính của trang web, chẳng hạn như danh sách các công việc, danh sách nhà tuyển dụng,....
- Footer: Chứa thông tin liên hệ, bản quyền, và các liên kết khác.

Tạo các thành phần React: Sau khi đã xác định các thành phần, bước tiếp theo là tạo các thành phần React tương ứng. Mỗi thành phần React là một function, sẽ chứa HTML, CSS, và JavaScript.

Kết nối các thành phần: Các thành phần React được kết nối với nhau bằng các thuộc tính và phương thức. Ví dụ, thành phần Main có thể sử dụng thuộc tính user để nhận danh sách sản phẩm từ phía backend.

Cấu trúc dự án được tổ chức theo cấu trúc hợp lý để dễ dàng quản lý và mở rộng. Cấu trúc được tôi sử dụng như sau:

```
1  src/
2    components/  # Chứa các thành phần có thể tái sử dụng
3      Header.js
4      Footer.js
5      Sidebar.js
6      ...
7    pages/       # Chứa các trang cụ thể
8      Home.js
9      Job.js
10     Search.js
11     ...
12    layout/     # Chứa layout chung
13      MainLayout.js
14     ...
15    App.js      # Thành phần gốc của ứng dụng
```

Hình 3.12 Cấu trúc thư mục frontend

Thư mục **components** sẽ chứa các thành phần của ReactJS có thể tái sử dụng được. Các thành phần này được chia thành các thư mục con theo chức năng hoặc tính năng của chúng. Ví dụ, tập *Header.js* chứa thành phần Header có thể được sử dụng trong nhiều trang khác nhau của ứng dụng.

```
src > components > Header.js > ...
1  import React from "react";
2
3  const Header = () => {
4    return (
5      <header>
6        <h2>Tiêu đề</h2>
7        <nav>
8          <a href="/">Trang chủ</a>
9          <a href="/about">Giới thiệu</a>
10         <a href="/contact">Liên hệ</a>
11        </nav>
12      </header>
13    );
14  };
15
16  export default Header;
17
```

Hình 3.13 Tập tin *Header.js*

Thư mục **pages** sẽ chứa các trang cụ thể của ứng dụng. Ví dụ như *Home.js* sẽ chứa các nội dung của trang chủ.

```
src > pages > Home.js > ...
1  import React from "react";
2
3  const Home = () => {
4    return (
5      <div>
6        <h2>Trang chủ</h2>
7        <p>Nội dung trang chủ</p>
8      </div>
9    );
10  };
11
12  export default Home;
13
```

Hình 3.14 Tập tin *Home.js*

Thư mục **layout** sẽ chứa các layout dùng chung của cả ứng dụng. Layout sẽ bao gồm các thành phần sẽ dùng chung như header, footer, sidebar, ...

```
src > layout > MainLayout.js > ...
1  import React from "react";
2  import Header from "../components/Header";
3
4  const MainLayout = ({ children }) => {
5    return (
6      <div>
7        <Header />
8        {children}
9      </div>
10   );
11 };
12
13 export default MainLayout;
14
```

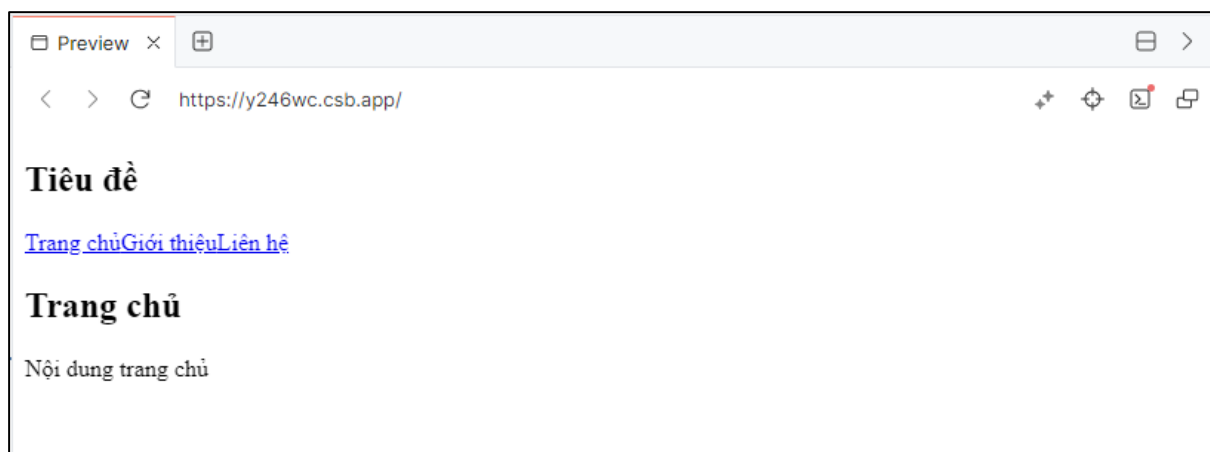
Hình 3.15 Tập tin *MainLayout.js*

Tập tin *App.js* là thành phần gốc của ứng dụng ReactJS. Thành phần này được đặt trong thư mục *src* của ứng dụng. Tập tin này sẽ import thư viện React và các thành phần cần thiết khác. Trong ví dụ này, các thành phần được đưa vào là *MainLayout* và *Home*. *MainLayout* là thành phần sẽ cấu trúc layout của cả trang bao gồm các thành phần dùng chung như *Header*. Trong *MainLayout* sẽ có *Home*, *Home* là thành phần của trang chủ, trang này sẽ nội dung trang chủ.

```
src > App.js > App
1  import "../styles.css";
2  import MainLayout from "../layout/MainLayout.js";
3  import Home from "../pages/Home.js";
4
5  export default function App() {
6    return (
7      <MainLayout>
8        <Home />
9      </MainLayout>
10   );
11 }
12
```

Hình 3.16 Tập tin *App.js*

Kết quả khi thực hiện các ví dụ trên :



Hình 3.17 Kết quả thực hiện cấu trúc thư mục frontend

3.5 Triển khai

Sau khi xây dựng hoàn thành website ở phía nhà phát triển, tôi đã triển khai ứng dụng lên các nền tảng đám mây. Triển khai ứng dụng giúp người dùng có thể trải nghiệm trực tiếp ứng dụng của tôi. Điều này tạo ra môi trường thực tế hóa, giúp tôi nhận phản hồi thực sự từ người dùng và cải thiện trải nghiệm của họ. Môi trường sản xuất có thể mang lại các vấn đề mà môi trường phát triển không thể phát hiện được. Triển khai ứng dụng giúp tôi kiểm thử và gỡ lỗi trực tiếp trên môi trường mà người dùng sẽ sử dụng, giúp tìm ra và sửa các vấn đề có thể xảy ra trong điều kiện thực tế. Biết được hiệu suất thực tế và tối ưu hóa ứng dụng để đảm bảo tải trang nhanh chóng và mượt mà cho người dùng.

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

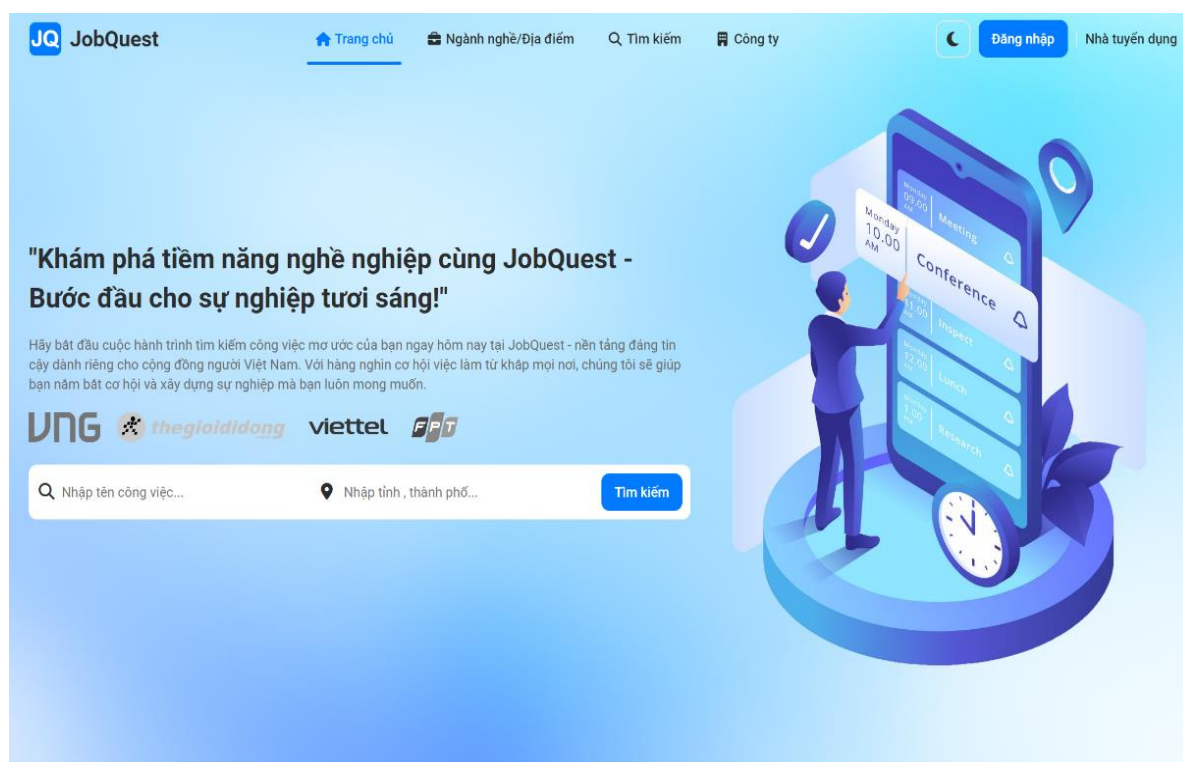
Chương này một chương quan trọng trong đồ án “xây dựng website tìm kiếm việc làm bằng ReactJS”. Chương này sẽ trình bày về kết quả nghiên cứu của tôi về hai phần chính của website là giao diện frontend và giao diện backend. Giao diện frontend là phần giao diện người dùng của website. Giao diện frontend chịu trách nhiệm hiển thị thông tin và tương tác với người dùng. Giao diện backend là phần giao diện máy chủ của website. Giao diện backend chịu trách nhiệm hiển thị các API, giúp tôi dễ dàng tìm thấy các API và cách sử dụng các API, xem thông tin về các tham số, mô hình dữ liệu và các yêu cầu phải đáp ứng.

4.1 Giao diện frontend

4.1.1 Giao diện chung

4.1.1.1 Trang chủ

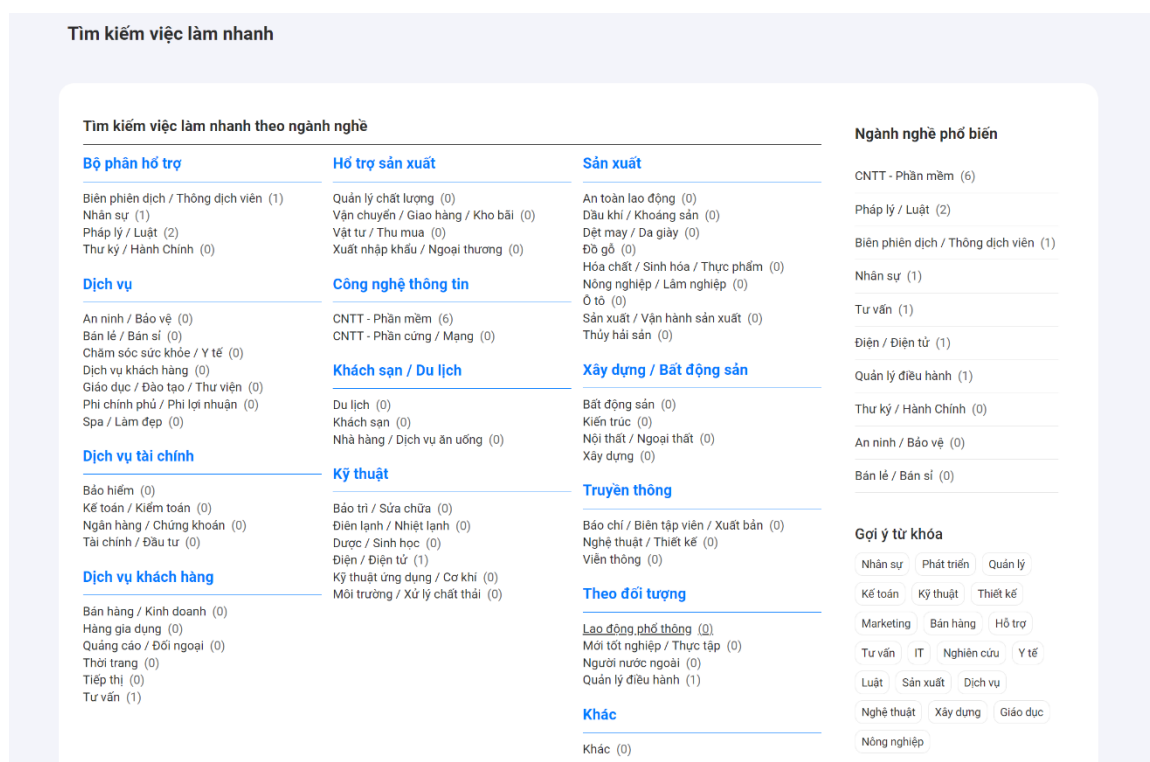
Trang chủ là điểm đầu tiên mà người dùng gặp khi truy cập. Với thiết kế thân thiện, trang chủ tập trung vào việc đơn giản hóa quá trình tìm kiếm việc làm và tạo ra một trải nghiệm người dùng mượt mà. Ô tìm kiếm nhanh tại vị trí trung tâm, kèm theo, giúp người tìm việc bắt đầu tìm kiếm việc làm của họ một cách dễ dàng. Nút đăng nhập giúp người dùng dễ dàng nhìn thấy và đăng nhập hoặc đăng ký tài khoản mới để tạo hồ sơ cá nhân và ứng tuyển công việc.



Hình 4.1 Giao diện trang chủ

4.1.1.2 Trang ngành nghề/ địa điểm

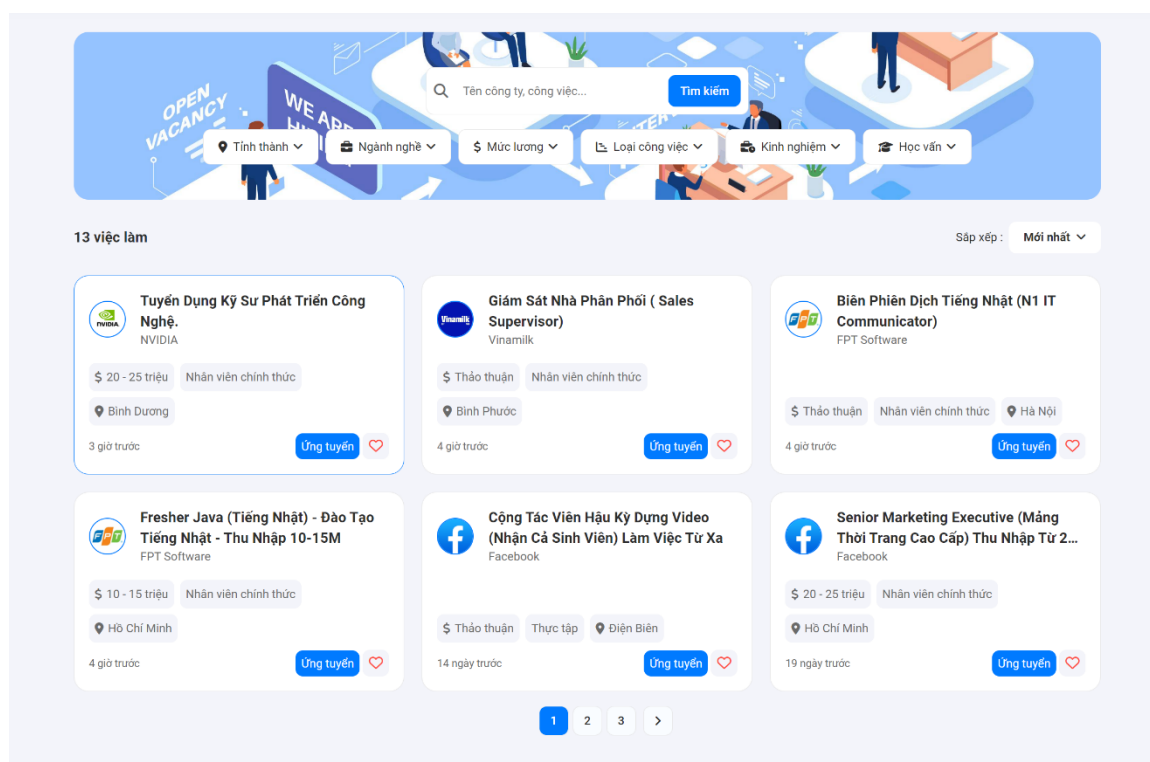
Trang ngành nghề/ Địa điểm hiển thị tất cả các ngành nghề hiện có, được sắp xếp theo loại ngành và cả số lượng việc làm đang tuyển dụng, giúp người dùng có thể dễ dàng tìm thấy ngành nghề phù hợp. Danh sách các địa điểm cũng được sắp xếp theo thứ tự bảng chữ cái, nên việc tìm kiếm cũng rất dễ dàng và nhanh chóng.



Hình 4.2 Giao diện trang ngành nghề/ địa điểm

4.1.1.3 Trang tìm kiếm việc làm

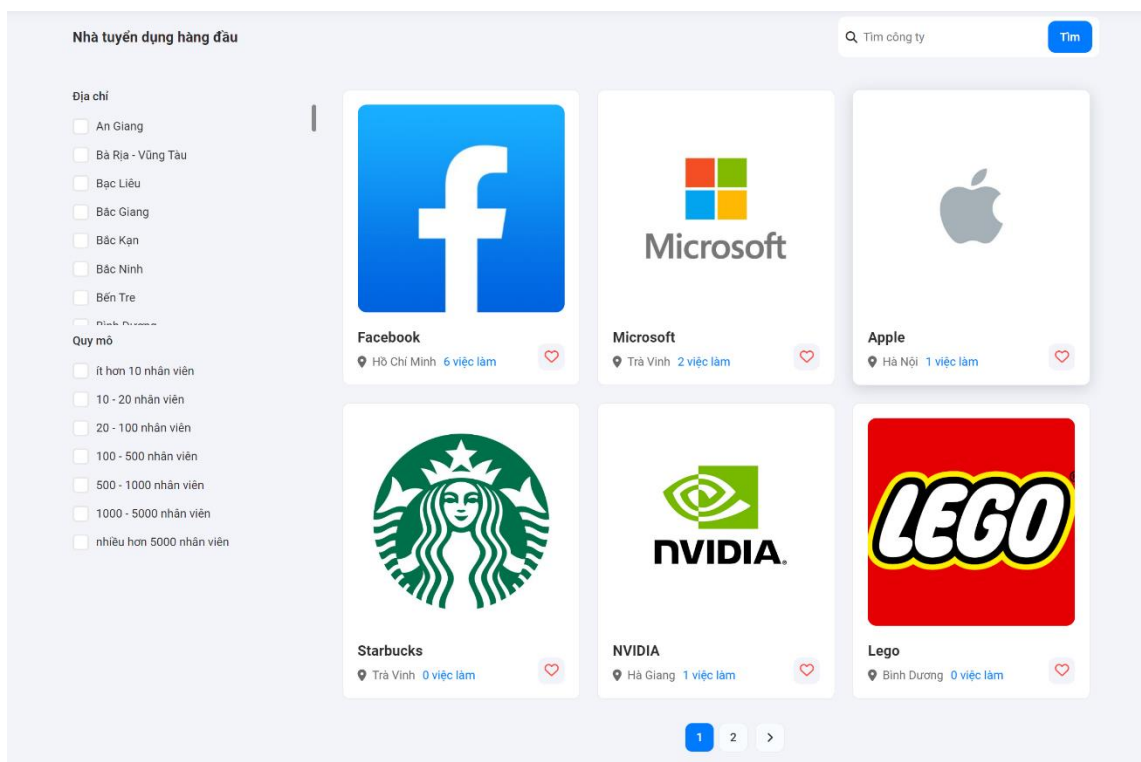
Trang tìm kiếm việc làm là một trong những trang quan trọng nhất của website. Thanh tìm kiếm nổi bật tại vị trí giữa trang, cho phép người tìm kiếm nhập từ khóa hoặc tên công ty để nhanh chóng tìm thấy công việc mong muốn. Trang còn cung cấp một loạt các tùy chọn lọc đa dạng để người tìm kiếm có thể điều chỉnh kết quả tìm kiếm theo mong muốn cá nhân. Có thể lọc theo ngành nghề, địa điểm làm việc, mức lương, loại công việc (toàn thời gian, bán thời gian, thực tập), học vấn, kinh nghiệm, và nhiều tiêu chí khác. Tính năng sắp xếp được tích hợp giúp người tìm kiếm sắp xếp kết quả theo các tiêu chí khác nhau như thời gian đăng tin, mức lương. Điều này giúp họ dễ dàng tìm thấy những công việc phù hợp và hiệu quả.



Hình 4.3 Giao diện trang tìm kiếm việc làm

4.1.1.4 Trang tìm kiếm nhà tuyển dụng

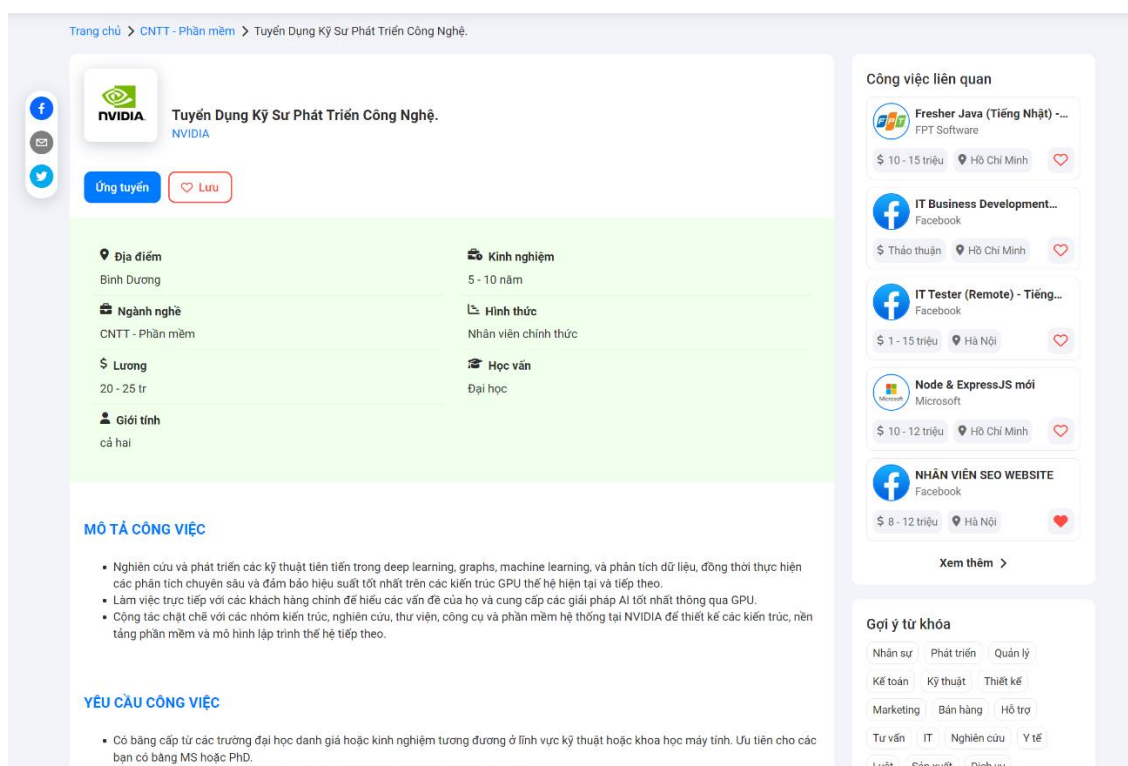
Giao diện tìm kiếm công ty được thiết kế với sự tập trung vào trải nghiệm người dùng, cung cấp nhiều chức năng linh hoạt để người tìm việc có thể dễ dàng lọc và tìm kiếm công ty một cách hiệu quả. Thanh tìm kiếm chính được đặt ở vị trí nổi bật nhất, giúp người tìm việc dễ dàng nhận diện và tìm kiếm. Người dùng có thể lọc các kết quả tìm kiếm theo địa chỉ và quy mô công ty chẳng hạn số lượng nhân viên. Mỗi nhà tuyển dụng được hiển thị với ảnh đại diện, tên, địa điểm và số lượng việc làm đang tuyển dụng của nhà tuyển dụng đó.



Hình 4.4 Giao diện trang tìm kiếm nhà tuyển dụng

4.1.1.5 Trang chi tiết việc làm

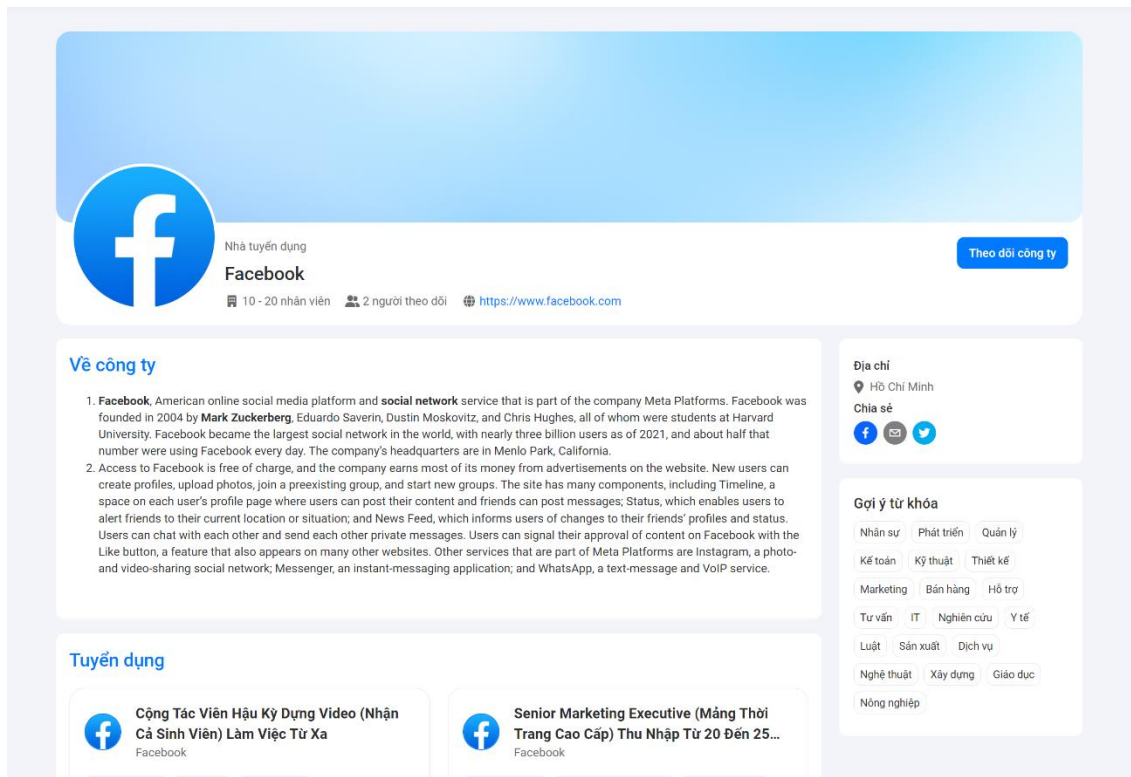
Trang chi tiết việc làm là nơi người tìm việc có thể khám phá mọi chi tiết quan trọng về công việc mà họ quan tâm. Từ mô tả việc làm đến yêu cầu công việc và quyền lợi, mọi thông tin đều được trình bày một cách rõ ràng và hấp dẫn. Nút ứng tuyển được đặt ở vị trí nổi bật để người tìm việc có thể dễ dàng tìm thấy và chủ động bắt đầu quá trình ứng tuyển, sau đó trang sẽ cung cấp biểu mẫu đơn ứng tuyển được thiết kế đơn giản và dễ điền, giảm thiểu thời gian và nỗ lực của người ứng tuyển, cho phép người ứng tuyển tải lên hồ sơ của mình một cách dễ dàng, giúp nhà tuyển dụng có cái nhìn toàn diện về kinh nghiệm và kỹ năng của họ. Trang cũng hiển thị danh sách các việc làm liên quan, giúp người tìm việc khám phá thêm các cơ hội tương tự hoặc có thể phù hợp với sự quan tâm của họ.



Hình 4.5 Giao diện trang chi tiết việc làm

4.1.1.6 Trang chi tiết nhà tuyển dụng

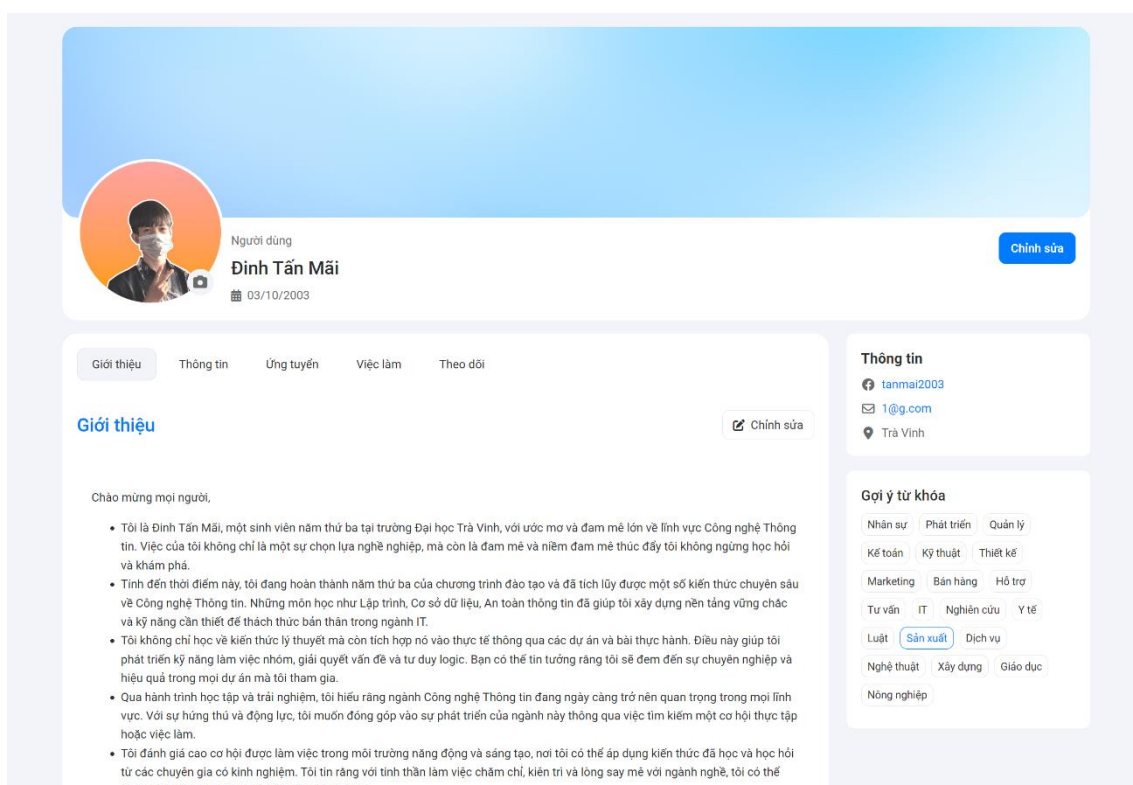
Trang chi tiết nhà tuyển dụng không chỉ là nơi để người tìm việc khám phá về doanh nghiệp mà còn là cổng thông tin chi tiết về các việc làm mà nhà tuyển dụng đang cung cấp. Hiện thị ảnh đại diện của nhà tuyển giúp dễ nhận biết và tạo ra ấn tượng mạnh mẽ. Hiện thị thông tin cơ bản như tên nhà tuyển dụng, quy mô nhân viên, và số lượng người theo dõi nhà tuyển dụng. Cung cấp liên kết trực tiếp đến trang web chính thức của nhà tuyển dụng, giúp người tìm việc có thể tìm hiểu thêm về doanh nghiệp. Kế bên đó là nút theo dõi, giúp người tìm việc có thể lưu những nhà tuyển dụng yêu thích. Hiện thị danh sách các công việc đang tuyển dụng, được lọc theo công việc mới nhất. Mỗi công việc được liệt kê với tiêu đề, ngành nghề, vị trí và các thông tin quan trọng khác.



Hình 4.6 Giao diện trang chi tiết nhà tuyển dụng

4.1.1.7 Trang chi tiết người tìm việc

Trang chi tiết ứng viên là nơi nhà tuyển dụng có thể khám phá về cá nhân, kỹ năng và kinh nghiệm làm việc của ứng viên. Hiển thị ảnh đại diện cá nhân giúp tạo ra ấn tượng chuyên nghiệp. Hiển thị các thông tin cơ bản như tên, ngày sinh, giúp tạo ra sự gần gũi và thân thiện. Cung cấp liên kết đến các trang mạng xã hội cá nhân để nhà tuyển dụng có thể hiểu thêm về sự chuyên nghiệp và sở thích cá nhân của ứng viên. Cho phép nhà tuyển dụng liên hệ trực tiếp với ứng viên thông qua email. Cung cấp phần giới thiệu ngắn gọn về bản thân, kỹ năng và mục tiêu sự nghiệp.



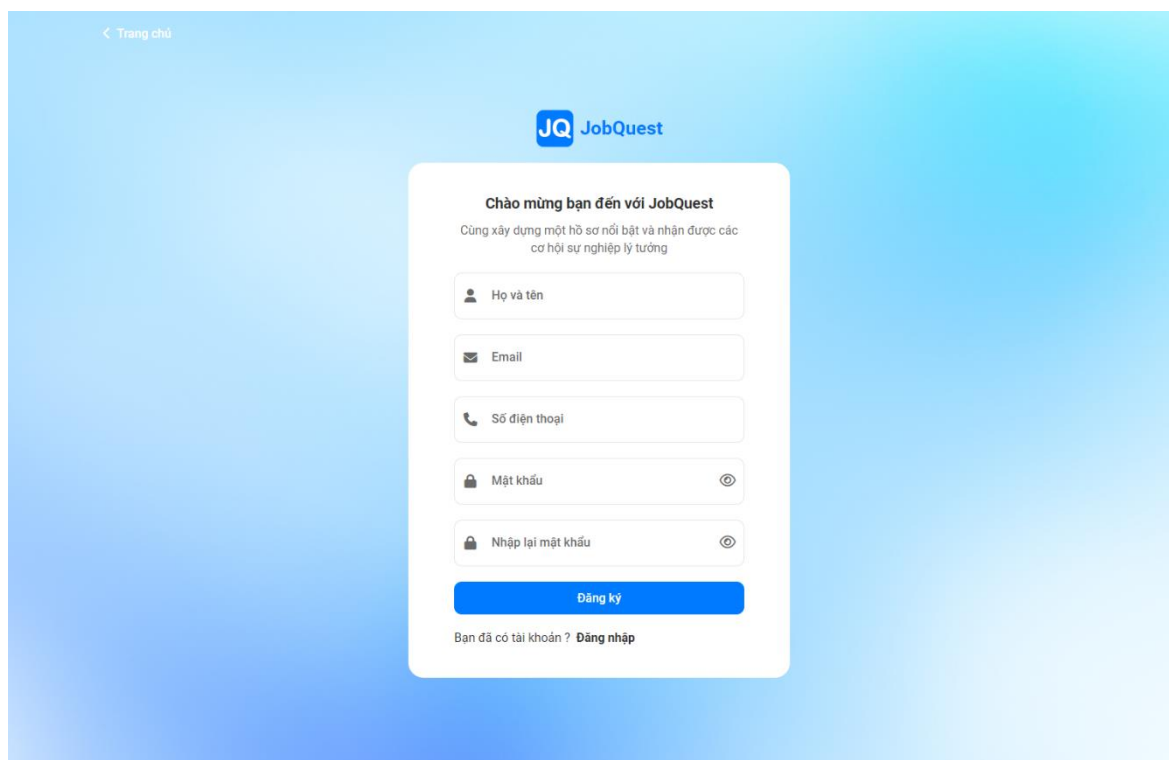
Hình 4.7 Giao diện trang chi tiết người tìm việc

4.1.2 Giao diện người tìm việc

4.1.2.1 Trang đăng ký người tìm việc

Giao diện trang đăng ký người dùng cần được thiết kế để tạo trải nghiệm thuận lợi và an toàn. Dưới đây là mô tả chi tiết:

- Email: Ô nhập dữ liệu để nhập địa chỉ email cho tài khoản mới.
- Mật khẩu: Ô nhập dữ liệu để nhập mật khẩu cho tài khoản mới.
- Họ và tên: Ô nhập dữ liệu để nhập họ và tên của người dùng.
- Số điện thoại: Ô nhập dữ liệu để nhập số điện thoại của người dùng.



< Trang chủ

JQ JobQuest

Chào mừng bạn đến với JobQuest

Cùng xây dựng một hồ sơ nổi bật và nhận được các cơ hội sự nghiệp lý tưởng

Họ và tên

Email

Số điện thoại

Mật khẩu

Nhập lại mật khẩu

Đăng ký

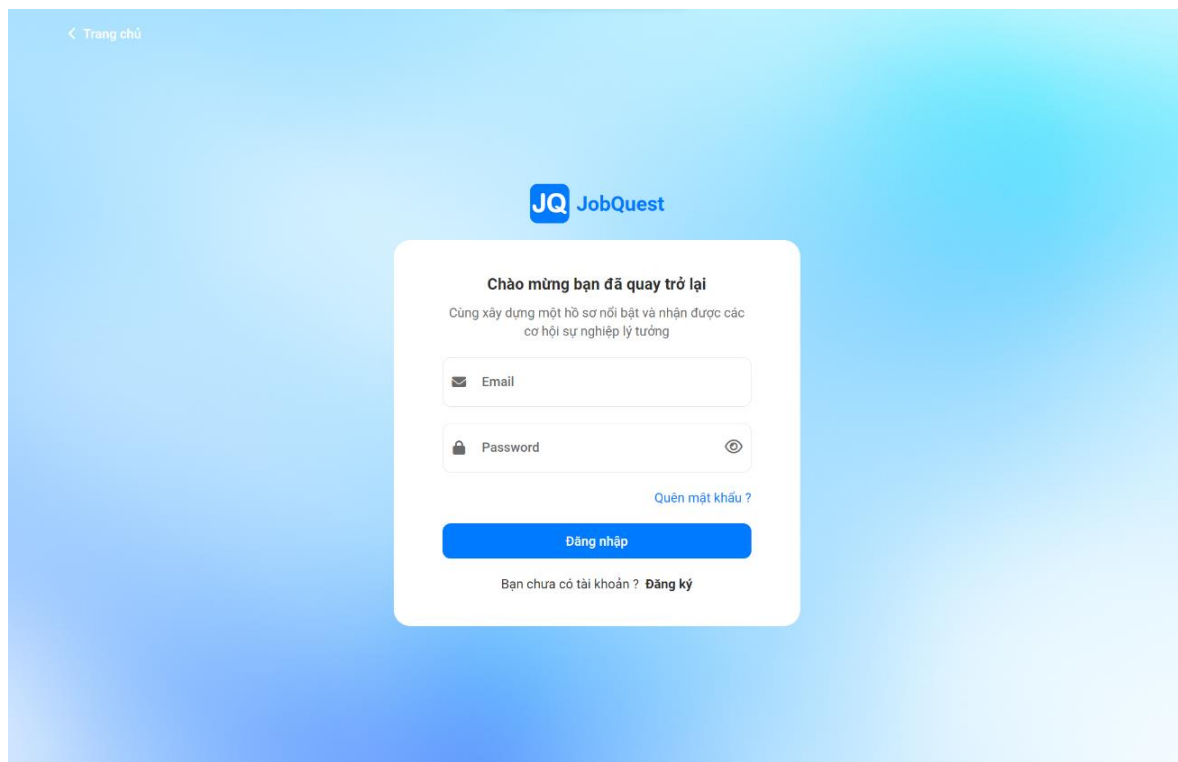
Bạn đã có tài khoản? [Đăng nhập](#)

Hình 4.8 Giao diện trang đăng ký người tìm việc

4.1.2.2 Trang đăng nhập người tìm việc

Chức năng này cho phép người tìm việc đăng nhập vào website bằng email và mật khẩu khi muốn ứng tuyển công việc và thực hiện các chức năng khác của website.

- Email: Ô nhập dữ liệu để nhập địa chỉ email đã đăng ký.
- Mật khẩu: Ô nhập dữ liệu để nhập mật khẩu của tài khoản.
- Nút đăng nhập: Nút nhấn để xác nhận thông tin đăng nhập.
- Liên kết quên mật khẩu: Liên kết dẫn đến trang khôi phục mật khẩu nếu người dùng quên mật khẩu.
- Liên kết đăng ký: Dẫn đến trang đăng ký nếu người dùng chưa có tài khoản.



Hình 4.9 Giao diện trang đăng nhập nhà tuyển dụng

4.1.3 Giao diện nhà tuyển dụng

4.1.3.1 Trang đăng ký nhà tuyển dụng

Giao diện đăng ký cho nhà tuyển dụng được thiết kế sao cho dễ sử dụng, thân thiện với người dùng và đảm bảo thu thập đầy đủ thông tin cần thiết.

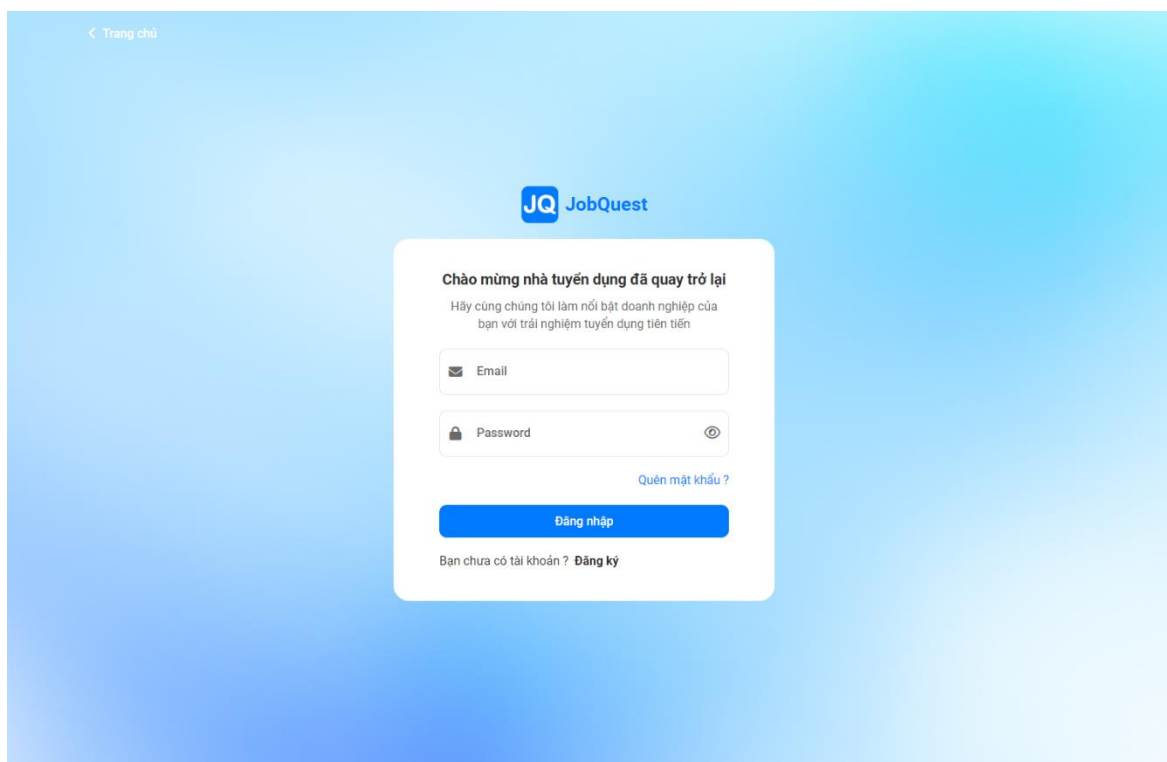
The screenshot shows a web interface for 'JobQuest' with a registration form for employers. The form is titled 'Đăng ký nhà tuyển dụng' (Register Employer) and includes a sub-header 'Hãy cùng chúng tôi xây dựng cơ hội tuyển dụng tốt nhất cho doanh nghiệp của bạn' (Join us in building the best job opportunities for your business). The form is divided into two columns: 'Thông tin đăng nhập' (Login Information) and 'Thông tin công ty' (Company Information). The 'Thông tin đăng nhập' column contains fields for Email, Số điện thoại (Phone Number), Mật khẩu (Password), and a link to 'Nhập lại mật khẩu' (Reset Password). The 'Thông tin công ty' column contains fields for Tên công ty (Company Name), Tên người đại diện (Representative Name), Quy mô (Size), and Địa chỉ (Address). A blue 'Đăng ký' (Register) button is at the bottom of the form. Below the button, there is a link 'Bạn đã có tài khoản? Đăng nhập' (Do you have an account? Log in).

Hình 4.10 Giao diện trang đăng ký nhà tuyển dụng

4.1.3.2 Trang đăng nhập nhà tuyển dụng

Giao diện đăng nhập cho nhà tuyển dụng cần được thiết kế để cung cấp trải nghiệm đơn giản và an toàn. Dưới đây là mô tả cho giao diện này:

- Email: Ô nhập dữ liệu để nhập địa chỉ email đã đăng ký.
- Mật khẩu: Ô nhập dữ liệu để nhập mật khẩu đã đăng ký.
- Quên mật khẩu: Liên kết để chuyển hướng người dùng đến trang khôi phục mật khẩu nếu cần.
- Nút đăng nhập: Nút nhấn để xác nhận thông tin đăng nhập.
- Liên kết đăng ký: Dẫn đến trang đăng ký nếu nhà tuyển dụng chưa có tài khoản.



Hình 4.11 Trang đăng nhập nhà tuyển dụng

4.1.3.3 Trang quản lý đơn ứng tuyển

Trang quản lý đơn ứng tuyển của nhà tuyển dụng thường cung cấp một giao diện tiện ích và hiệu quả để nhà tuyển dụng có thể quản lý thông tin liên quan đến các hồ sơ ứng viên và quá trình tuyển dụng. Dưới đây là mô tả chi tiết về trang quản lý đơn ứng tuyển:

Danh sách đơn ứng tuyển: trang sẽ hiển thị một danh sách các đơn ứng tuyển mới nhất hoặc theo trạng thái, cho phép nhà tuyển dụng theo dõi và quản lý từng hồ sơ một.

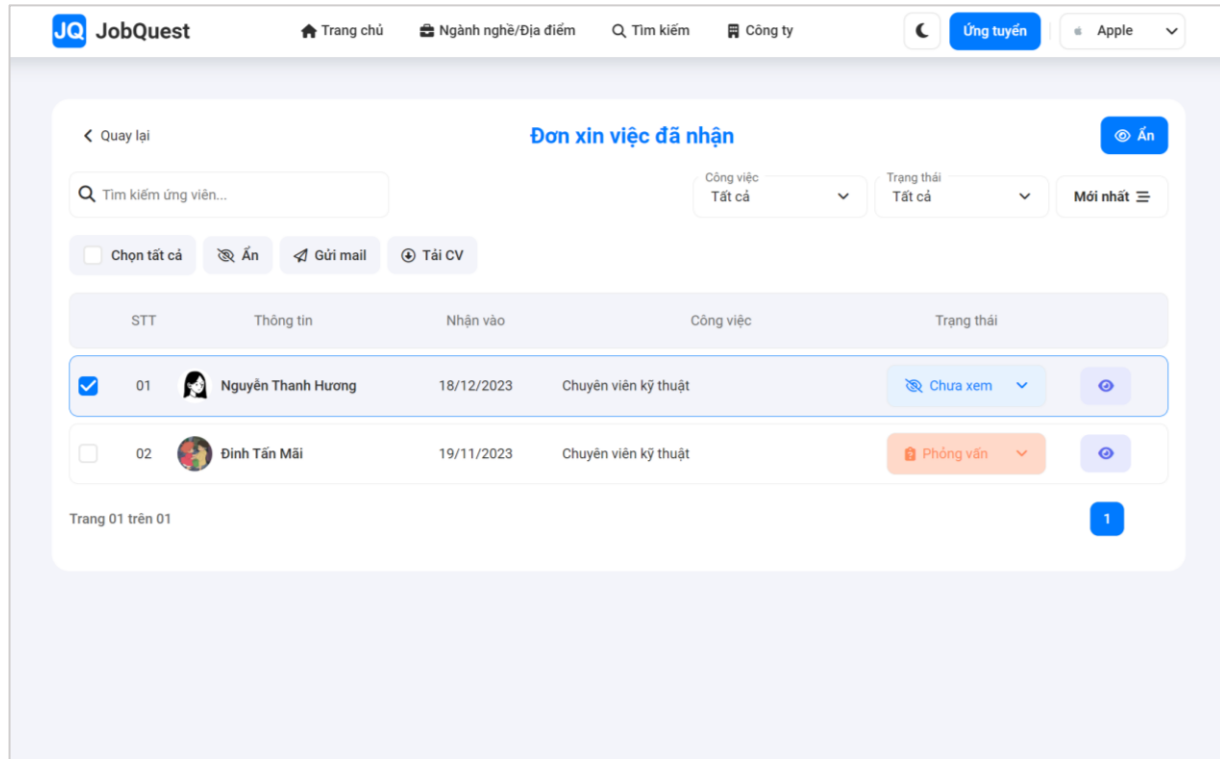
Bộ lọc và tìm kiếm: cung cấp bộ lọc và công cụ tìm kiếm để nhà tuyển dụng có thể dễ dàng xác định các đơn ứng tuyển theo các tiêu chí như công việc, hoặc trạng thái ứng viên.

Trạng thái đơn ứng tuyển: Hiển thị trạng thái của mỗi đơn ứng tuyển, ví dụ như “Chưa xem”, “Đã Xem”, “Phỏng Vấn”, “Chấp nhận”, “Từ chối” giúp nhà tuyển dụng theo dõi tiến trình tuyển dụng.

Thông tin chi tiết hồ sơ ứng tuyển: nhấp vào mỗi đơn ứng tuyển sẽ hiển thị thông tin chi tiết về hồ sơ ứng viên, bao gồm ảnh đại diện, thông tin liên hệ, cv, email, thư xin việc,...

Gửi thông báo và mời phỏng vấn: cho phép nhà tuyển dụng gửi email và mời phỏng vấn đến ứng viên.

Các nút thao tác như "Ẩn", "Gửi mail", hay "Tải CV" giúp nhà tuyển dụng thực hiện các hành động quản lý một cách dễ dàng.



Hình 4.12 Giao diện trang quản lý đơn ứng tuyển của nhà tuyển dụng

4.2 Giao diện backend

Giao diện phía backend được xây dựng bằng swagger. Swagger là một công cụ dùng để tạo và hiển thị tài liệu API cho ứng dụng. Nó giúp các nhà phát triển và người sử dụng dễ dàng tìm hiểu cách sử dụng các API, xem thông tin về các tham số, mô hình dữ liệu và các yêu cầu phải đáp ứng.

Người tìm việc			^
GET	/api/user/find/{id}		✓
GET	/api/user/owner		✓
PUT	/api/user/update		✓
PUT	/api/user/updateIntro		✓
PUT	/api/user/uploadImage		✓
POST	/api/user/forgot		✓
POST	/api/user/resetPassword/{id}/{token}		✓
POST	/api/user/changePassword/{id}		✓
Nhà tuyển dụng			^
GET	/api/company/owner/		✓
GET	/api/company/{id}		✓
GET	/api/company/		✓
PUT	/api/company/update		✓
PUT	/api/company/updateIntro		✓

Hình 4.13 Giao diện API phía Backend

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Những kết quả đạt được

Trong suốt quá trình thực hiện đề tài xây dựng trang web tìm kiếm việc làm bằng ReactJS, tôi đã không chỉ đơn thuần là làm chủ kiến thức về React, mà còn trải qua một hành trình sâu rộng, giúp tôi củng cố và mở rộng đáng kể kiến thức và kỹ năng trong lĩnh vực lập trình web và phát triển phần mềm. Việc nắm bắt ngôn ngữ lập trình frontend như HTML, CSS, và JavaScript đã giúp tôi xây dựng giao diện người dùng một cách sáng tạo và hiệu quả. Sử dụng ReactJS, giúp tôi không chỉ tận dụng được sức mạnh của nó để tối ưu hóa hiệu suất ứng dụng mà còn phát triển khả năng quản lý trạng thái ứng dụng, đồng thời tái sử dụng component một cách linh hoạt. Tóm lại, dự án này không chỉ đơn giản là một trang web tìm kiếm việc làm, mà là một hành trình chuyên sâu, đưa tôi vượt qua những thách thức và làm giàu kiến thức cũng như kỹ năng cần thiết để thành công trong lĩnh vực lập trình web và phát triển phần mềm. Sau khoảng thời gian học hỏi và xây dựng, những kết quả mà tôi đã đạt được như sau:

- Về công nghệ:

- + Sử dụng linh hoạt ReactJS: Hiểu rõ về cách sử dụng ReactJS để xây dựng giao diện người dùng một cách linh hoạt và hiệu quả. Áp dụng các khái niệm để tối ưu hóa quản lý và tái sử dụng mã nguồn.

- + Nắm rõ về Express và Node.js: quy trình xây dựng, xử lý các yêu cầu HTTP, tương tác với cơ sở dữ liệu và xây dựng API.

- + Tương tác có hiệu quả với cơ sở dữ liệu MySQL: Có khả năng thiết kế và triển khai cơ sở dữ liệu MySQL cho dự án. Biết cách tương tác với cơ sở dữ liệu để lưu trữ và truy xuất thông tin một cách hiệu quả.

- Về chương trình:

- + Xác thực, đăng ký người tìm việc và nhà tuyển dụng.

- + Website có thể tìm kiếm công việc dựa trên các tiêu chí như tên công việc, vị trí, lĩnh vực, mức lương,...

- + Website có thể hiển thị thông tin chi tiết về một công việc, bao gồm các thông tin như tiêu đề công việc, mô tả công việc, yêu cầu công việc, mức lương,...

- + Website cho phép người tìm việc gửi hồ sơ ứng viên cho các nhà tuyển dụng.

- + Người tìm việc có thể lưu các công việc phù hợp.
- + Người tìm việc cũng có thể theo dõi các nhà tuyển dụng mà họ quan tâm.
- + Xây dựng được chức năng quản lý việc làm như thêm, sửa, xóa việc làm.
- + Quản lý ứng viên của nhà tuyển dụng.

5.2 Hướng phát triển

Mặc dù đã tạo ra một trang web với nhiều chức năng khá đầy đủ, tuy nhiên, với thời gian có hạn và tài nguyên hạn chế, ứng dụng vẫn còn một số hạn chế và chưa đáp ứng được mọi yêu cầu của người dùng. Trong tương lai, tôi sẽ tiếp tục phát triển và cải thiện trang web, tập trung vào việc khắc phục những lỗi nhỏ và tối ưu hóa các tính năng hiện tại. Để đảm bảo rằng trang web không chỉ đáp ứng tốt với nhu cầu người dùng hiện tại mà còn đáp ứng được những thách thức trong tương lai, tôi sẽ thực hiện các bước sau:

- Phát triển ứng dụng đa nền tảng.
- Hệ thống thông báo và gợi ý.
- Xây dựng trang admin quản lý website.
- Thực hiện chức năng chat và phỏng vấn trực tuyến.
- Thêm nhiều lựa chọn đăng nhập.
- Thêm chức năng đánh giá nhà tuyển dụng và đánh giá ứng viên.
- Xác thực nhà tuyển dụng.

TÀI LIỆU THAM KHẢO

- [1] AWS, “What is an API (Application Programming Interface)?,” 12 2023. [Trực tuyến]. Available: <https://aws.amazon.com/what-is/api>.
- [2] JWT, “What is JSON Web Token?,” 12 2023. [Trực tuyến]. Available: <https://jwt.io/introduction>.
- [3] AWS, “What is a RESTful API?,” 28 12 2023. [Trực tuyến]. Available: <https://aws.amazon.com/what-is/restful-api/>.
- [4] L. Gupta, “<https://restfulapi.net/>,” 28 12 2023. [Trực tuyến]. Available: <https://restfulapi.net/>.
- [5] w3schools, “React Tutorial,” 2023. [Trực tuyến]. Available: <https://www.w3schools.com/REACT/>.