BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN MÔN HỌC

MẠNG MÁY TÍNH

Đề tài:

Portfolio giới thiệu về bản thân và chia sẽ kiến thức về NetWorking – Javascript – Java

Giảng viên hướng dẫn : ThS. Nguyễn Quang Trung

Sinh viên thực hiện : 2280600213 – Nguyễn Nguyễn Thái Bảo

Lớp : 22DTHC6

TP. Hồ Chí Minh, 2025

Lời cảm ơn

Em xin được gửi lời cảm ơn chân thành đến thầy ThS. Nguyễn Quang Trung đã truyền đạt kiến thức cho chúng em bằng tất cả tâm huyết. Thời gian được làm đồ án dưới sự hướng dẫn của thầy là khoảng thời gian hữu ích vì em được học tập đầy đủ lý thuyết và nắm bắt kinh nghiệm thực tế. Dưới sự hỗ trợ của thầy em đã được trang bị những kiến thức quan trọng để có thể hoàn thành đồ án.

Em rất mong nhận được những ý kiến phản hồi từ thầy. Những chỉ dẫn và góp ý từ thầy sẽ giúp em nhận ra những điểm cần cải thiện và hoàn thiện kiến thức của mình một cách tốt nhất, qua đó sẽ là động lực để em có thể tiếp tục nỗ lực và phát triển trong những dự án tiếp theo.

Mục lục

Chương 1: Cơ sở lý thuyết và tổng quan đề tài 4

1.1 Bối cảnh 4

1.2 Mô tả yêu cầu 4

1.2.2 Quy trình hiển thị thông tin trên website 4

1.2.2 Yêu cầu về trải nghiệm người dùng và chức năng 5

1.3 Các công nghệ sử dụng trong hệ thống 6

1.3.1 Giao diện người dùng (Frontend) 6

1.3.1.1 React.js 6

1.3.1.2 TypeScript 7

1.3.1.3 TailwindCSS 7

1.3.1.4 React Router DOM 8

1.3.1.5 Framer Motion 8

1.3.2 Công cụ phát triển và triển khai 9

1.3.2.2 Vite 9

1.3.2.2 Vercel 9

Chương 2: Phân tích và Thiết kế chức năng hệ thống 9

2.1 Cấu trúc dữ liệu 9

2.2 Cấu trúc dữ liệu 10

2.2.1 Phân tích Trang Chủ(/) 10

2.2.2 Phân tích Trang Dự án & Chứng chỉ (/project) 10

2.2.3 Phân tích Trang Blog (/blog và /blog/:slug) 11

2.2.4 Phân tích Nội dung Blog - Chia sẻ kiến thức đã học 11

2.2.5 Phân tích các thành phần chung 13

Chương 3: Kết quả thực nghiệm và Đánh giá hệ thống 13

3.1 Kết quả thực nghiệm 13

3.2 Đánh giá hệ thống 14

3.2.1 Ưu điểm 14

3.2.2 Nhược điểm và hướng phát triển 15

Chương 1: Cơ sở lý thuyết và tổng quan đề tài

1.1 Bối cảnh

Trong bối cảnh ngành công nghệ thông tin phát triển mạnh mẽ và cạnh tranh, việc xây dựng một thương hiệu cá nhân chuyên nghiệp là yếu tố then chốt giúp các lập trình viên nổi bật. Một website portfolio cá nhân không chỉ là một CV trực tuyến mà còn là không gian để thể hiện năng lực kỹ thuật, tư duy thẩm mỹ và các dự án thực tế đã thực hiện. Tuy nhiên, một portfolio tĩnh chỉ trưng bày sản phẩm là chưa đủ. Việc tích hợp thêm một trang Blog cá nhân để chia sẻ kiến thức, ghi lại quá trình học tập và phân tích các công nghệ mới cho thấy sự chủ động, đam mê và mong muốn đóng góp cho cộng đồng.

Để giải quyết nhu cầu này, đề tài **"Portfolio giới thiệu về bản thân và chia sẻ kiến thức về Networking – Javascript – Java"** được thực hiện. Dự án không chỉ nhằm mục tiêu tạo ra một sản phẩm portfolio hoàn chỉnh, hiện đại để giới thiệu bản thân, mà còn là một nền tảng chia sẻ kiến thức chuyên sâu về các lĩnh vực đã học, qua đó củng cố và thể hiện năng lực trên một ứng dụng trang đơn (Single-Page Application) được xây dựng bằng React và Vite.

1.2 Mô tả yêu cầu

Dự án được xây dựng với kiến trúc ứng dụng trang đơn (SPA), sử dụng điều hướng phía client để tạo ra trải nghiệm người dùng mượt mà, đồng thời phân tách nội dung một cách logic.

1.2.2 Quy trình hiển thị thông tin trên website

Website được thiết kế để cung cấp thông tin toàn diện về năng lực của một lập trình viên Fullstack và các bài post chia sẽ kiến thức. Các trang chính bao gồm:

- **Trang chủ: (/):** Điểm tiếp xúc đầu tiên, tạo ấn tượng mạnh mẽ với **Hero Section** giới thiệu chức danh và các liên kết hành động (call-to-action). Trang này cũng hiển thị bản tóm tắt các khu vực quan trọng như **Giới thiệu bản thân (About), Kinh nghiệm lập trình (Skills), Công nghệ sử dụng**

(TechnicalSkills), Kỹ năng mềm (SoftSkills) và **Thông tin liên hệ (Contact)** để người xem có cái nhìn tổng quan nhanh chóng.

- **Trang Dự án & Chứng chỉ (/project):** Là trung tâm của portfolio, nơi trình bày chi tiết năng lực chuyên môn.
  - **Projects Section:** Mỗi dự án được thể hiện như một "case study" thu nhỏ, bao gồm mô tả, vai trò, công nghệ sử dụng và liên kết đến GitHub/Demo.
  - **Certificates Section:** Các chứng chỉ được hiển thị trên một thanh trượt ngang tự động để tăng tính tương tác và thu hút, với tùy chọn tải file PDF.
- **Trang Blog (/blog và /blog/:slug):** Không gian chia sẻ kiến thức chuyên sâu về **Java Core, JavaScript và Networking**, thể hiện chiều sâu chuyên môn.
  - **Trang danh sách Blog (/blog):** Hiển thị tất cả bài viết dưới dạng lưới (grid), cho phép người dùng dễ dàng khám phá nội dung.
  - **Trang chi tiết bài viết (/blog/:slug):** Giao diện đọc được tối ưu hóa, tập trung vào nội dung với định dạng rõ ràng, hình ảnh minh họa và không có yếu tố gây xao lãng.
- **Các thành phần chung:**
  - **Navbar (Thanh điều hướng):** "Thông minh", tự động thay đổi các mục menu tùy theo trang người dùng đang truy cập để tối ưu hóa điều hướng.
  - **Footer (Chân trang):** Cung cấp thông tin bản quyền và các liên kết mạng xã hội một cách nhất quán trên mọi trang.

1.2.2 Yêu cầu về trải nghiệm người dùng và chức năng

- **Kiến trúc SPA:** Sử dụng thư viện React Router DOM để quản lý điều hướng phía
- client, tạo ra trải nghiệm đa trang mượt mà không cần tải lại toàn bộ website.
- **Thiết kế Responsive:** Giao diện phải tương thích hoàn toàn trên mọi thiết bị, từ điện thoại di động, máy tính bảng đến máy tính để bàn.

- **Chế độ Sáng/Tối (Dark/Light Mode):** Cung cấp tùy chọn thay đổi giao diện và sử dụng LocalStorage để ghi nhớ lựa chọn của người dùng, đảm bảo tính nhất quán qua các phiên truy cập.
- **Hiệu ứng động:** Sử dụng thư viện Framer Motion để tạo các hiệu ứng chuyển động tinh tế, tăng tính thẩm mỹ và cảm giác hiện đại cho website.
- **Hỗ trợ Đa ngôn ngữ (Anh/Việt):** Sử dụng React Context để quản lý trạng thái ngôn ngữ và LocalStorage để lưu trữ lựa chọn của người dùng, cho phép chuyển đổi mượt mà giữa tiếng Anh và tiếng Việt trên toàn bộ nội dung website.

1.3 Các công nghệ sử dụng trong hệ thống

Hệ thống Website được xây dựng trên nền tảng web, kết hợp nhiều công nghệ hiện đại nhằm đảm bảo hiệu năng, khả năng mở rộng và trải nghiệm người dùng. Các công nghệ chính bao gồm:

1.3.1 Giao diện người dùng (Frontend)

1.3.1.1 React.js

React.js là một thư viện JavaScript được phát triển bởi Meta (Facebook) dùng để xây dựng giao diện người dùng theo mô hình component-based (dựa trên các thành phần). Với React.js các nhà phát triển có thể viết HTML bên trong JavaScript thông qua cú pháp JSX (JavaScript XML). Ngoài ra, còn có các đặc điểm như:

- **Kiến trúc Component:** Giao diện được chia nhỏ thành các **component** độc lập, dễ tái sử dụng (ví dụ: Navbar, Footer, BlogCard). Điều này giúp mã nguồn trở nên gọn gàng, dễ quản lý, bảo trì và tránh lặp code, từ đó tăng tốc độ phát triển
- **Virtual DOM:** React sử dụng một DOM ảo để tối ưu hóa hiệu năng. Khi trạng thái thay đổi, React sẽ so sánh DOM ảo với DOM thật và chỉ cập nhật những thành phần thực sự có thay đổi, tránh việc phải tải lại toàn bộ trang, giúp trải nghiệm người dùng mượt mà hơn.Phù hợp để xây dựng các dự án SPA (Single Page Application) với trải nghiệm mượt mà.

- **Hệ sinh thái mạnh mẽ:** React có một cộng đồng hỗ trợ lớn, với vô số thư viện, công cụ và tài liệu học tập, giúp giải quyết hầu hết các vấn đề trong quá trình phát triển.
- **Phù hợp với SPA:** Là lựa chọn hàng đầu để xây dựng các Ứng dụng trang đơn (Single Page Application) với trải nghiệm liền mạch.

1.3.1.2 TypeScript

TypeScript là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Microsoft. Nó là một tập hợp cha (superset) của JavaScript, bổ sung thêm tính năng kiểm tra kiểu tĩnh (static typing) và các khái niệm hướng đối tượng.

- **An toàn kiểu dữ liệu:** Giúp phát hiện các lỗi liên quan đến kiểu dữ liệu ngay trong quá trình viết code, thay vì phải chờ đến lúc chạy chương trình, từ đó giảm thiểu lỗi và tăng tính ổn định cho ứng dụng.
- **Tăng khả năng đọc và bảo trì:** Việc khai báo rõ ràng kiểu dữ liệu cho biến, hàm và đối tượng giúp mã nguồn trở nên dễ hiểu và dễ bảo trì hơn, đặc biệt trong các dự án lớn.
- **Hỗ trợ công cụ mạnh mẽ:** Cung cấp các tính năng như tự động hoàn thành mã (autocompletion), gợi ý và kiểm tra lỗi thông minh hơn trong các trình soạn thảo code như VS Code.

1.3.1.3 TailwindCSS

TailwindCSS là một framework CSS theo triết lý "utility-first", cung cấp một bộ lớn các lớp CSS tiện ích, có sẵn để xây dựng giao diện trực tiếp trong mã HTML (hoặc JSX).

- **Tốc độ phát triển nhanh:** Thay vì viết CSS tùy chỉnh, lập trình viên có thể áp dụng các lớp có sẵn như flex, pt-4, text-center để tạo kiểu nhanh chóng.
- **Tính nhất quán cao:** Đảm bảo thiết kế đồng bộ trên toàn bộ dự án vì mọi người đều sử dụng chung một bộ các lớp tiện ích.

- **Tùy biến cao:** Dễ dàng cấu hình và mở rộng, đặc biệt mạnh mẽ trong việc triển khai các tính năng như thiết kế responsive và chế độ Sáng/Tối (Dark/Light Mode).
- **Tối ưu hóa dung lượng:** Tự động loại bỏ tất cả các lớp CSS không được sử dụng trong quá trình build, giúp file CSS cuối cùng có dung lượng rất nhỏ.

1.3.1.4 React Router DOM

Đây là thư viện tiêu chuẩn để quản lý điều hướng (routing) trong các ứng dụng React.

- **Client-Side Routing:** Cho phép tạo ra trải nghiệm đa trang trong một ứng dụng trang đơn (SPA). Khi người dùng nhấp vào một liên kết, React Router sẽ thay đổi URL và render component tương ứng mà không cần gửi yêu cầu đến máy chủ và tải lại toàn bộ trang.
- **Route động:** Hỗ trợ các đường dẫn động (dynamic routes), ví dụ /blog/:slug, rất cần thiết cho các chức năng như trang chi tiết bài viết.

1.3.1.5 Framer Motion

Framer Motion là một thư viện animation mạnh mẽ và dễ sử dụng dành cho React.

- **Cú pháp khai báo đơn giản:** Cho phép tạo ra các hiệu ứng phức tạp chỉ với vài dòng code, tích hợp mượt mà vào các component React.
- **Tối ưu hiệu năng:** Sử dụng các kỹ thuật tăng tốc phần cứng để đảm bảo các hiệu ứng chuyển động luôn mượt mà.
- **Tương tác phong phú:** Hỗ trợ nhiều loại animation, từ hiệu ứng xuất hiện khi cuộn (whileInView) đến các tương tác khi di chuột (whileHover) hoặc nhấp (whileTap).

1.3.2 Công cụ phát triển và triển khai

1.3.2.2 Vite

Vite là một công cụ build thế hệ mới, được tạo ra để cải thiện trải nghiệm phát triển frontend.

- **Máy chủ phát triển cực nhanh:** Sử dụng cơ chế Native ES Modules của trình duyệt, giúp thời gian khởi động máy chủ và cập nhật khi thay đổi code (Hot Module Replacement - HMR) gần như tức thì.
- **Build tối ưu:** Sử dụng Rollup để đóng gói mã nguồn cho môi trường production, tạo ra các tệp được tối ưu hóa cao về hiệu suất.

1.3.2.2 Vercel

Vercel là một nền tảng đám mây chuyên dụng cho việc triển khai các ứng dụng frontend.

- **Tích hợp Git:** Kết nối trực tiếp với các kho chứa mã nguồn như GitHub, GitLab.
- **Triển khai tự động (CI/CD):** Mỗi khi có một commit mới được đẩy lên nhánh chính, Vercel sẽ tự động build và triển khai phiên bản mới của website.
- **Mạng lưới phân phối toàn cầu (CDN):** Tự động phân phối các tài sản tĩnh của website đến các máy chủ trên toàn cầu, giúp người dùng ở bất kỳ đâu cũng có tốc độ tải trang nhanh nhất.

Chương 2: Phân tích và Thiết kế chức năng hệ thống

2.1 Cấu trúc dữ liệu

Do tính chất của một trang portfolio cá nhân, dự án không yêu cầu cơ sở dữ liệu. Dữ liệu được quản lý dưới dạng tĩnh (static data) trực tiếp trong mã nguồn, đóng gói cùng ứng dụng khi build.

- **Dữ liệu Blog (postsData):** Được lưu trữ trong một tệp TypeScript (/src/data/postsData.ts). Đây là một mảng các đối tượng, mỗi đối tượng đại diện

cho một bài viết và chứa các thuộc tính như slug, title, excerpt, image, tags, date, và content.

- **Dữ liệu Dự án (projectsData):** Tương tự, thông tin về các dự án cũng được định nghĩa trong một mảng đối tượng trong mã nguồn.

2.2 Cấu trúc dữ liệu

2.2.1 Phân tích Trang Chủ(/)

Do tính chất của một trang portfolio cá nhân, dự án không yêu cầu cơ sở dữ liệu. Dữ liệu được quản lý dưới dạng tĩnh (static data) trực tiếp trong mã nguồn, đóng gói cùng ứng dụng khi build.

- **Dữ liệu Blog (postsData):** Được lưu trữ trong một tệp TypeScript (/src/data/postsData.ts). Đây là một mảng các đối tượng, mỗi đối tượng đại diện cho một bài viết và chứa các thuộc tính như slug, title, excerpt, image, tags, date, và content.
- **Dữ liệu Dự án (projectsData):** Tương tự như Blog, thông tin về các dự án cũng được định nghĩa trong một mảng đối tượng trong mã nguồn.

2.2.2 Phân tích Trang Dự án & Chứng chỉ (/project)

Khi người dùng điều hướng đến /project, React Router sẽ render component ProjectPage.

- **Projects Section:** Dữ liệu dự án được import và lặp qua để hiển thị. Phần hình ảnh của mỗi dự án được trình bày dưới dạng **trình chiếu tự động theo chiều dọc**, sử dụng component AutoScrollGallery. Mỗi dự án có thể chứa nhiều hình ảnh (mảng images[]) được hiển thị nối tiếp và trượt liên tục, giúp thể hiện toàn cảnh giao diện sản phẩm một cách trực quan. Hiệu ứng trượt được điều khiển bằng JavaScript thông qua requestAnimationFrame, đảm bảo chuyển động mượt mà và tương thích với nhiều kích thước màn hình khác nhau.

- **Certificates Section:** Logic thanh trượt ngang được điều khiển bằng CSS Animation. Hàm xử lý sự kiện onClick được gắn vào nút tải xuống để thực thi việc tải file PDF.

2.2.3 Phân tích Trang Blog (/blog và /blog/:slug)

Đây là một ví dụ điển hình của Client-Side Routing và Rendering.

- **Trang danh sách Blog (/blog):** React Router render component BlogPage, component này sẽ import postsData và hiển thị danh sách bài viết.
- **Trang chi tiết Blog (/blog/:slug):** Khi URL khớp với route động này, component BlogPostDetail sẽ được render.
- **Logic hoạt động:** Component sử dụng hook useParams() từ react-router-dom để lấy ra giá trị slug từ URL.
  - Dựa vào slug, nó thực hiện tìm kiếm trong mảng postsData đã được import.
  - Nếu tìm thấy, component sẽ render ra nội dung của bài viết. Nếu không, nó sẽ hiển thị một thông báo lỗi. Toàn bộ quá trình này diễn ra trên trình duyệt của người dùng.

2.2.4 Phân tích Nội dung Blog - Chia sẻ kiến thức đã học

Chức năng Blog không chỉ là một phần phụ mà còn là nơi thể hiện quá trình học tập và tổng hợp kiến thức từ các khóa học uy tín như Cisco NetAcad. Các bài viết được tổ chức thành các chuỗi nội dung có hệ thống:

- **Chuỗi bài viết về Java:**
  - Bắt đầu với **"Giới thiệu ngôn ngữ lập trình Java"**, bài viết đặt nền móng bằng cách giải thích triết lý "Write Once, Run Anywhere" và các ứng dụng thực tế.
  - Tiếp nối với **"Cấu trúc chương trình Java & cú pháp cơ bản"**, bài viết đi sâu vào các thành phần nhỏ nhất như class, phương thức main(), giúp người mới bắt đầu có thể viết chương trình đầu tiên.

- Đỉnh cao của chuỗi là bài viết về **"Lập trình hướng đối tượng trong Java (OOP)"**, phân tích chi tiết 4 trụ cột (Đóng gói, Kế thừa, Đa hình, Trừu tượng), thể hiện sự nắm vững khái niệm cốt lõi nhất của Java.

- **Chuỗi bài viết về JavaScript:**
  - Mở đầu với **"JavaScript Essentials 1"**, bài viết giới thiệu vai trò của JavaScript trong web và các khái niệm cơ bản như tương tác với DOM.
  - Các bài viết tiếp theo đi sâu vào nền tảng: **"Biến, kiểu dữ liệu và toán tử"**, **"Hàm và Callback"**, giúp xây dựng kiến thức một cách tuần tự.
  - Nâng cao hơn với **"Promise và Async/Await"** để xử lý bất đồng bộ, một kỹ năng thiết yếu trong lập trình web hiện đại.
  - Cuối cùng, các bài viết về **"JavaScript OOP"** và **"Xử lý lỗi với try...catch"** thể hiện sự hiểu biết sâu sắc về các khía cạnh phức tạp hơn của ngôn ngữ.

- **Bài viết về Networking:**
  - Bài viết **"Networking Basics – Quy trình gửi dữ liệu qua mạng"** là một điểm nhấn quan trọng, thể hiện kiến thức nền tảng của một lập trình viên Fullstack. Việc phân tích mô hình OSI và TCP/IP cho thấy sự hiểu biết về cách dữ liệu được đóng gói và truyền đi trên Internet, là cầu nối giữa frontend và backend.

- **Bài viết so sánh và thực hành:**
  - **"So sánh Java và JavaScript"** giúp làm rõ những nhầm lẫn phổ biến, thể hiện khả năng phân tích và tổng hợp.
  - **"Mini Project: Kết hợp Java và JavaScript"** là minh chứng rõ ràng nhất cho năng lực Fullstack, thể hiện khả năng xây dựng một ứng dụng hoàn chỉnh từ backend (Java Spring Boot) đến frontend (React).

2.2.5 Phân tích các thành phần chung

- **Navbar (/src/components/Navbar.tsx):** Là một Client Component, sử dụng hook useLocation từ react-router-dom để lấy thông tin về đường dẫn hiện tại. Dựa vào đó, nó sẽ quyết định hiển thị nhóm liên kết (links) nào cho phù hợp. useState và useEffect được dùng để quản lý trạng thái menu và hiệu ứng khi cuộn.
- **Theme Toggle:** Sử dụng useState để quản lý trạng thái theme hiện tại (light hoặc dark). Hook useEffect được dùng để theo dõi sự thay đổi của trạng thái này; mỗi khi thay đổi, nó sẽ cập nhật class trên thẻ <html> và lưu giá trị mới vào LocalStorage.
- **Hệ thống Đa ngôn ngữ** (/src/context/LanguageContext.tsx): Sử dụng React Context (createContext, useContext) để tạo một LanguageProvider bao bọc toàn bộ ứng dụng, cung cấp trạng thái ngôn ngữ toàn cục. Hook useState quản lý ngôn ngữ hiện tại (en hoặc vi). Tương tự Theme Toggle, useEffect được dùng để lưu lựa chọn ngôn ngữ vào LocalStorage, giúp duy trì cài đặt của người dùng qua các lần truy cập.

Chương 3: Kết quả thực nghiệm và Đánh giá hệ thống

3.1 Kết quả thực nghiệm

Dự án đã hoàn thành các chức năng và giao diện theo đúng yêu cầu thiết kế, mang lại một sản phẩm portfolio chuyên nghiệp và có tính ứng dụng cao.

- **Giao diện Trang chủ:** Hiển thị đầy đủ các section, animation mượt mà khi cuộn trang, điều hướng chính xác.
- **Giao diện Trang Dự án:** Tải và hiển thị hình ảnh các dự án một cách tối ưu. Mỗi dự án hiển thị nhiều hình ảnh minh họa thông qua hiệu ứng trượt dọc tự động. Tính năng này giúp người xem dễ dàng quan sát toàn bộ giao diện và chức năng của dự án mà không cần thao tác thủ công. Thanh trượt chứng chỉ hoạt động liên tục và tương tác tốt.

- **Giao diện Trang Blog:** Danh sách bài viết được trình bày khoa học. Trang chi tiết tải nhanh sau lần tải đầu tiên, nội dung được định dạng đẹp mắt.
- **Chức năng Dark/Light Mode:** Hoạt động nhất quán trên tất cả các trang, lựa chọn được lưu lại sau khi đóng trình duyệt.
- **Thiết kế Responsive:** Giao diện tự động co dãn và sắp xếp lại hợp lý trên các kích thước màn hình khác nhau, đảm bảo trải nghiệm tốt trên di động.

3.2 Đánh giá hệ thống

3.2.1 Ưu điểm

- **Trải nghiệm SPA mượt mà:** Nhờ Client-Side Routing, việc chuyển đổi giữa các trang diễn ra gần như tức thì mà không cần tải lại trang, mang lại cảm giác giống như một ứng dụng thực thụ.
- **Trình chiếu hình ảnh tự động:** Tăng tính sinh động và chuyên nghiệp cho phần Dự án. Hình ảnh được trình bày liên tục, mượt mà và phù hợp với giao diện sáng/tối, mang lại trải nghiệm trực quan và thu hút hơn cho người xem.
- **Mã nguồn có tổ chức và dễ bảo trì:** Kiến trúc component của React giúp phân tách giao diện thành các phần nhỏ, độc lập, dễ quản lý và tái sử dụng.
- **Trải nghiệm người dùng tốt:** Từ các hiệu ứng chuyển động tinh tế, thiết kế responsive đến các tính năng tiện ích như Dark Mode, website mang lại một trải nghiệm chuyên nghiệp và hiện đại.
- **Tiếp cận người dùng toàn cầu với tính năng đa ngôn ngữ:** Website hỗ trợ chuyển đổi mượt mà giữa tiếng Anh và tiếng Việt, tăng khả năng tiếp cận cho cả người dùng trong nước và quốc tế, đồng thời mang lại trải nghiệm cá nhân hóa cao hơn.
- **Quy trình phát triển nhanh:** Công cụ Vite cung cấp môi trường phát triển với Hot Module Replacement (HMR) cực nhanh, giúp tăng tốc độ code và debug.
- **Quy trình triển khai tự động:** Tích hợp với Vercel giúp mọi thay đổi trên nhánh main của GitHub được tự động build và triển khai.

3.2.2 Nhược điểm và hướng phát triển

- **Chưa được tối ưu cho SEO:** Do là ứng dụng trang đơn (SPA), nội dung chính được render bằng JavaScript phía client. Điều này gây khó khăn cho các công cụ tìm kiếm trong việc thu thập và lập chỉ mục dữ liệu, ảnh hưởng đến thứ hạng tìm kiếm.
- **Thời gian tải ban đầu có thể chậm:** Người dùng phải tải toàn bộ framework React và mã nguồn ứng dụng trong lần truy cập đầu tiên trước khi có thể thấy bất kỳ nội dung nào.
- **Quản lý nội dung thủ công:** Dữ liệu của blog và dự án đang được lưu dưới dạng file tĩnh. Điều này gây khó khăn khi muốn thêm/sửa nội dung, đòi hỏi phải can thiệp vào mã nguồn và triển khai lại.
- **Chưa có chức năng tương tác trên Blog:** Blog hiện tại chỉ có chức năng đọc, thiếu các tính năng quan trọng như tìm kiếm, lọc theo thẻ (tags), và bình luận.

Hướng phát triển tiếp theo:

1. **Chuyển đổi sang Next.js:** Nâng cấp dự án lên Next.js để tận dụng **Static Site Generation (SSG)** cho các trang blog và dự án. Điều này sẽ khắc phục hoàn toàn nhược điểm về SEO và tốc độ tải ban đầu.

2. **Tích hợp Headless CMS:** Chuyển đổi dữ liệu từ file tĩnh sang một Headless CMS (như Strapi, Contentful) để quản lý nội dung một cách trực quan hơn.

3. **Phát triển chức năng Blog:** Xây dựng tính năng tìm kiếm bài viết và trang danh sách bài viết theo từng thẻ (tag).

4. **Xây dựng Form liên hệ:** Tích hợp EmailJS hoặc xây dựng một API backend nhỏ để người dùng có thể gửi tin nhắn trực tiếp từ website.

Tài Liệu Tham Khảo

[1] React.JS: https://react.dev/

[2] React.JS: https://www.w3schools.com/react/

[3] Vite Official Documentation: https://vitejs.dev/

[4] React Router DOM: https://reactrouter.com/

[5] TailwindCSS Official Documentation: https://tailwindcss.com/docs

[6] Framer Motion Documentation: https://www.framer.com/motion/

[7] Vercel Documentation: https://vercel.com/docs