NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

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

Vĩnh Long, ngày 29 tháng 12 năm 2025

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

Nguyễn Hoàng Duy Thiện

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

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

Vĩnh Long, ngày ….. tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Hoàng Duy Thiện giảng viên Bộ môn Công nghệ thông tin, Khoa Kỹ thuật và Công nghệ, Trường Đại học Trà Vinh người đã tận tình giúp đỡ, hướng dẫn và hỗ trợ em rất nhiều trong suốt quá trình tìm hiểu, nghiên cứu và thực hiện đồ án chuyên ngành.

Tuy nhiên, trong quá trình thực hiện đồ án, do kinh nghiệm thực tế còn hạn chế và kiến thức chuyên môn chưa thật sự sâu rộng, nên em không tránh khỏi những thiếu sót nhất định. Em rất mong nhận được sự quan tâm, nhận xét và những ý kiến đóng góp quý báu từ thầy Nguyễn Hoàng Duy Thiện cùng các thầy cô giảng viên trong Bộ môn để đề tài của em được hoàn thiện và chỉnh chu hơn.

Em xin chân thành cảm ơn.

Sinh viên thực hiện

Nguyễn Đình Nhật Huy

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Vấn đề nghiên cứu:

Đồ án tập trung nghiên cứu và xây dựng website thương mại điện tử bán dụng cụ và phụ kiện câu cá, nhằm đáp ứng nhu cầu mua sắm trực tuyến ngày càng phổ biến hiện nay. Mục tiêu của đề tài là phát triển một hệ thống website có giao diện thân thiện, hoạt động ổn định, hiệu suất cao và dễ dàng mở rộng, đồng thời hỗ trợ đầy đủ các chức năng cần thiết cho người dùng và quản trị viên.

Hướng tiếp cận:

Website được phát triển với frontend sử dụng Next.js kết hợp TypeScript, giúp tối ưu tốc độ tải trang và nâng cao trải nghiệm người dùng. Giao diện được thiết kế trực quan, dễ sử dụng, hỗ trợ các chức năng như đăng ký, đăng nhập, duyệt và tìm kiếm sản phẩm, xem chi tiết sản phẩm, quản lý giỏ hàng và đặt hàng.

Phần backend được xây dựng bằng Express.js, cung cấp các API RESTful để giao tiếp với frontend và cơ sở dữ liệu MySQL. Kiến trúc Clean Architecture được áp dụng nhằm tách biệt rõ ràng các tầng xử lý, giúp hệ thống dễ bảo trì, nâng cấp và mở rộng trong tương lai. Cơ sở dữ liệu được thiết kế tối ưu để đảm bảo hiệu suất truy vấn nhanh và tính toàn vẹn dữ liệu liên quan đến sản phẩm, người dùng và đơn hàng.

Cách giải quyết vấn đề:

Trong quá trình thực hiện, đồ án tiến hành phân tích yêu cầu chức năng và phi chức năng, thiết kế kiến trúc hệ thống, triển khai các chức năng chính và tích hợp hoàn chỉnh giữa frontend và backend. Sau cùng, hệ thống được kiểm thử toàn diện nhằm đảm bảo các chức năng hoạt động chính xác, ổn định và an toàn.

Kết quả đạt được:

Kết quả đạt được là một website thương mại điện tử hoàn chỉnh, đáp ứng tốt các yêu cầu đề ra, có giao diện thân thiện, backend ổn định, dữ liệu được quản lý hiệu quả và sẵn sàng cho việc phát triển, mở rộng trong tương lai.

MỞ ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ và được ứng dụng rộng rãi, mua sắm trực tuyến đã trở thành xu hướng tất yếu, mang lại sự tiện lợi và tiết kiệm thời gian cho người tiêu dùng. Thương mại điện tử không chỉ phổ biến trong lĩnh vực công nghệ mà còn mở rộng sang nhiều ngành khác, trong đó có dụng cụ và phụ kiện câu cá – một lĩnh vực có nhu cầu ngày càng tăng nhưng chưa được khai thác hiệu quả trên nền tảng trực tuyến.

Xuất phát từ thực tế đó và mong muốn tìm hiểu quy trình xây dựng một website thương mại điện tử hoàn chỉnh, đề tài "Xây dựng website bán dụng cụ và phụ kiện câu cá" được lựa chọn. Thông qua đề tài, em có cơ hội vận dụng kiến thức về frontend, backend, thiết kế cơ sở dữ liệu và kiến trúc hệ thống vào một dự án thực tiễn.

Mục tiêu của đồ án là xây dựng một website thương mại điện tử hoàn chỉnh với các chức năng cơ bản như đăng ký, đăng nhập, hiển thị và tìm kiếm sản phẩm, quản lý giỏ hàng và đơn hàng. Hệ thống được phát triển với Next.js cho frontend và Express.js cho backend, hướng đến trải nghiệm người dùng tốt, hoạt động ổn định và khả năng mở rộng.

Đối tượng nghiên cứu của đề tài là các công nghệ và phương pháp xây dựng website thương mại điện tử gồm Next.js, TypeScript, Express.js, MySQL và kiến trúc Clean Architecture, tập trung vào việc kết hợp các công nghệ này để tạo nên hệ thống hiệu quả và dễ bảo trì.

Phạm vi nghiên cứu được giới hạn trong việc xây dựng website bán dụng cụ và phụ kiện câu cá với các chức năng cơ bản, bao gồm phân tích yêu cầu, thiết kế kiến trúc hệ thống, xây dựng giao diện người dùng, phát triển backend cung cấp API RESTful và thiết kế cơ sở dữ liệu MySQL.

CHƯƠNG 1: TỔNG QUAN

Hiện nay, cùng với sự phát triển nhanh chóng của công nghệ thông tin và thương mại điện tử, nhu cầu mua sắm trực tuyến ngày càng trở nên phổ biến và thuận tiện. Người dùng có thể dễ dàng tìm kiếm, so sánh và lựa chọn sản phẩm thông qua các website bán hàng trực tuyến. Trong lĩnh vực câu cá, thị trường tiêu thụ dụng cụ và phụ kiện câu cá tại Việt Nam đang tăng trưởng mạnh, bởi đây là ngành hàng có cộng đồng người chơi lớn và đa dạng. Tuy nhiên, phần lớn hoạt động mua bán hiện nay vẫn diễn ra thông qua cửa hàng truyền thống, mạng xã hội hoặc các kênh nhỏ lẻ, thiếu tính hệ thống và không đáp ứng được nhu cầu trải nghiệm mua sắm chuyên nghiệp.

Nhiều website thương mại điện tử ra đời đã góp phần cải thiện trải nghiệm mua sắm trực tuyến, nhưng số lượng website chuyên biệt về dụng cụ và phụ kiện câu cá còn hạn chế. Khách hàng thường gặp khó khăn trong việc tìm kiếm thông tin sản phẩm rõ ràng, đánh giá chất lượng, kiểm tra tồn kho, so sánh giá và theo dõi đơn hàng. Điều này dẫn đến trải nghiệm người dùng chưa cao và làm giảm hiệu quả hoạt động bán hàng của các doanh nghiệp kinh doanh mặt hàng này.

Xuất phát từ nhu cầu đó, việc xây dựng một website thương mại điện tử chuyên về dụng cụ và phụ kiện câu cá là rất cần thiết. Website không chỉ tập trung cung cấp thông tin chi tiết về sản phẩm như cần câu, máy câu, dây câu, lưỡi câu, mồi câu, hộp đồ nghề, áo khoác chống nắng... mà còn hỗ trợ khách hàng duyệt sản phẩm, tìm kiếm, thêm vào giỏ hàng, đặt hàng và quản lý đơn hàng một cách thuận tiện. Bên cạnh đó, hệ thống quản trị tích hợp giúp quản lý sản phẩm, người dùng và đơn hàng hiệu quả, hỗ trợ chủ cửa hàng theo dõi hoạt động kinh doanh dễ dàng và chính xác.

Trong đề tài, hệ thống được xây dựng dựa trên các công nghệ hiện đại, gồm Next.js cho frontend, Express.js và TypeScript cho backend, cùng cơ sở dữ liệu MySQL nhằm tăng tốc độ xử lý, đảm bảo tính ổn định và khả năng mở rộng lâu dài. Kiến trúc Clean Architecture được áp dụng giúp tách biệt rõ ràng các tầng xử lý của backend, tối ưu bảo trì và nâng cấp trong tương lai.

Website được thiết kế với các chức năng nền tảng của một hệ thống thương mại điện tử, như đăng ký – đăng nhập, tìm kiếm sản phẩm theo tên và loại, xem chi tiết sản phẩm, giỏ hàng, thanh toán và theo dõi lịch sử đơn hàng. Bên cạnh đó, hệ

thống còn được kiểm thử về tốc độ, giao diện và bảo mật, nhằm mang đến trải nghiệm ổn định và mượt mà cho người dùng.

Việc xây dựng website bán dụng cụ và phụ kiện câu cá không chỉ góp phần mang lại giải pháp mua sắm trực tuyến tiện lợi, mà còn mở ra cơ hội mở rộng thị trường cho ngành hàng câu cá – một lĩnh vực phát triển mạnh nhưng chưa có nhiều nền tảng chuyên nghiệp hỗ trợ. Đây cũng là tiền đề để phát triển mô hình kinh doanh bền vững trong tương lai, đồng thời đáp ứng xu hướng số hóa trong thương mại hiện đại.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Next.js

2.1.1 Giới thiệu

Next.js là một framework React dùng để xây dựng các ứng dụng web full-stack. Bạn sử dụng các thành phần React (React Components) để xây dựng giao diện người dùng, và sử dụng Next.js để có thêm các tính năng và tối ưu hóa. Bên dưới, Next.js cũng trừu tượng hóa và tự động cấu hình các công cụ cần thiết cho React, như bundling (đóng gói), compiling (biên dịch), và nhiều hơn nữa. Điều này cho phép bạn tập trung vào việc xây dựng ứng dụng thay vì tốn thời gian cấu hình. Dù bạn là một lập trình viên cá nhân hay một thành viên của một đội ngũ lớn, Next.js có thể giúp bạn xây dựng các ứng dụng React tương tác, động và nhanh chóng [2].

2.1.2 Ưu điểm của Next.js

Kết xuất phía máy chủ (SSR) và Kết xuất tĩnh (SSG): Next.js hỗ trợ cả SSR và SSG, cho phép tạo ra các trang web với nội dung được tạo ra tại thời điểm yêu cầu hoặc trước đó. Điều này giúp cải thiện thời gian tải trang và trải nghiệm người dùng, đồng thời nâng cao khả năng tìm kiếm trên các công cụ tìm kiếm như Google.

Tối ưu hóa SEO: Với khả năng tạo ra nội dung từ phía máy chủ, Next.js giúp cải thiện khả năng tìm kiếm của trang web trên các công cụ tìm kiếm. Việc tối ưu hóa SEO trở nên dễ dàng hơn, đảm bảo rằng trang web của bạn được tìm thấy và xếp hạng cao trên kết quả tìm kiếm.

Tích hợp sẵn với React và TypeScript: Next.js tích hợp sẵn với React, một thư viện JavaScript phổ biến được sử dụng rộng rãi trong cộng đồng phát triển. Bạn cũng có thể sử dụng TypeScript để cải thiện tính linh hoạt và dễ bảo trì của mã nguồn của mình.

Routing đơn giản: Next.js cung cấp một hệ thống routing đơn giản và mạnh mẽ, giúp bạn tổ chức ứng dụng của mình một cách dễ dàng và hiệu quả. Bạn có thể xác định các tuyến đường và điều hướng dễ dàng thông qua các tệp JavaScript trong thư mục pages.

Triển khai dễ dàng: Next.js tích hợp tốt với nhiều dịch vụ phát triển web như Vercel, Netlify và AWS Amplify, giúp việc triển khai và quản lý ứng dụng của bạn trở nên dễ dàng và tiện lợi hơn bao giờ hết.

Hỗ trợ tốt từ cộng đồng và tài liệu đa dạng: Next.js có một cộng đồng lớn và sôi động, cung cấp nhiều tài liệu, hướng dẫn và nguồn lực học tập. Bạn có thể dễ dàng tìm kiếm giải pháp cho các vấn đề phát triển và nhận được sự hỗ trợ từ cộng đồng trong quá trình phát triển.

2.1.3 Nhược điểm của Next.js

Mặc dù Next.js là một framework mạnh mẽ, nhưng nó cũng có một số nhược điểm cần cân nhắc xem có phù hợp với dự án của bạn không

Chi phí phát triển và bảo trì: Mặc dù Next.js giúp đơn giản hóa việc xây dựng ứng dụng, nhưng nó vẫn là một framework với cấu trúc riêng. Các developer cần có kiến thức về React và Next.js để xây dựng và bảo trì ứng dụng hiệu quả. So với các thư viện React đơn giản hơn, việc tìm kiếm developer có kinh nghiệm với Next.js có thể tốn nhiều thời gian và chi phí hơn.

Không có state manager tích hợp sẵn: Next.js không đi kèm với state manager mặc định. Bạn cần chọn và tích hợp thêm một state manager như Redux, MobX, hoặc Context API của React. Việc lựa chọn và tích hợp thêm state manager có thể làm tăng thêm độ phức tạp của dự án.

Ít plugin hơn so với các framework khác: So với các framework lâu đời hơn như Angular hay Vue, Next.js có hệ sinh thái plugin ít phong phú hơn. Mặc dù Next.js hỗ trợ tích hợp với nhiều thư viện của bên thứ ba, bạn có thể gặp khó khăn trong việc tìm kiếm một plugin giải quyết vấn đề cụ thể của mình. Trong một số trường hợp, bạn có thể cần phải xây dựng plugin tùy chỉnh để đáp ứng nhu cầu.

2.1.4 Một số đặc điểm cơ bản trong Next.js

2.1.4.1 Routing trong NextJS

Automatic Routing: NextJS sẽ tự động tạo các router dựa trên cấu trúc thư mục của chúng ta. Ví dụ, nếu bạn tạo một file có tên là about.js ở thư mục pages. NextJS sẽ tạo router là /about.

Nested Routing: Chúng ta có thể tạo các thư mục con để tạo các router lồng nhau.

Ví dụ, nếu bạn tạo một folder có tên blog nằm trong folder pages, bên trong folder blog lại có file post.js, đường dẫn sẽ là pages/blog/post.js, thì router mà NextJS tạo ra sẽ là /blog/post.

Dynamic Routes: Bạn có thể tạo các router động bằng cách sử dụng cặp dấu [] trong tên file.

Ví dụ nếu đường dẫn là pages/blog/[slug].js thì NextJS sẽ tạo ra các router như /blog/blog-dau-tien hoặc /blog/blog-thu-hai. Với slug là một giá trị bất kì do bạn truyền vào.

Link Component: Để tạo liên kết giữa các trang, bạn sử dụng component Link được cung cấp sẵn bởi NextJS ở thư viện next/link. Sử dụng Link thay cho thẻ a giúp tránh việc tải lại trang và tối ưu hóa hiệu suất.

Query Parameters: Bạn có thể truyền dữ liệu giữa các trang sử dụng query parameters trong router bằng cách sử dụng ký tự dấu chấm hỏi ? trong tên file.

Ví dụ, pages/product.js có thể có các router như /product?productId=0001.

2.1.4.2 Rendering trong NextJS

Server-side Rendering (SSR): Máy chủ xử lý dữ liệu, thực thi JavaScript và gửi HTML đã kết xuất sẵn về trình duyệt. Điều này giúp tải nhanh hơn và tối ưu SEO vì nội dung đã được hiển thị trước khi đến tay khách hàng.

Client-side Rendering (CSR): Máy chủ chỉ gửi một trang HTML cơ bản và các tệp JavaScript. Trình duyệt sau đó sẽ thực thi JavaScript, lấy dữ liệu từ API và tự kết xuất nội dung, phù hợp cho các ứng dụng web động và nhiều tương tác.

Static Site Generation (SSG): Là một phương pháp mà NextJS cung cấp sẵn cho chúng ta, cho phép bạn tạo các trang tĩnh và lưu chúng xuống dưới dạng file html tĩnh. Điều này giúp cải thiện hiệu suất tải trang và cung cấp trải nghiệm người dùng tốt hơn vì nội dung được lấy từ file html và hiển thị ngay lập tức mà không cần đợi việc tải về từ phía server.

2.1.4.3 Styling trong NextJS

CSS Modules: Để style cho ứng dụng NextJS, cách dễ nhất là bạn có thể tạo các file CSS/SCSS riêng lẻ cho từng component hoặc sử dụng file chung cho toàn dự án.

CSS Frameworks: NextJS cũng hỗ trợ sử dụng cùng các CSS framework như TailwindCSS, Bootstrap hoặc MaterialUI.

2.2 Kiến trúc Clean Architecture

Clean Architecture do Robert C. Martin đề xuất, phân chia ứng dụng thành các tầng với trách nhiệm rõ ràng, bao gồm Presentation, Application, Domain, và Infrastructure. Mô hình này giúp giảm thiểu sự phụ thuộc lẫn nhau giữa các lớp, tạo điều kiện bảo trì và mở rộng.

Kiến trúc Clean Architecture giúp đảm bảo hệ thống có tính linh hoạt, dễ dàng thay thế các thành phần mà không ảnh hưởng toàn hệ thống, đáp ứng yêu cầu mở rộng và tích hợp trong tương lai [7].

2.2.1 Mục tiêu của Clean Architecture

Clean Architecture hướng tới việc giải quyết các vấn đề như tính khó bảo trì, sự phụ thuộc lẫn nhau giữa các thành phần, và sự phức tạp trong việc mở rộng hệ thống. Một số mục tiêu chính bao gồm:

Độc lập với Framework: Các framework chỉ được sử dụng như công cụ hỗ trợ. Chúng không chi phối kiến trúc chính của ứng dụng, đảm bảo rằng việc thay thế framework không ảnh hưởng lớn đến hệ thống.

Dễ dàng kiểm thử: Hệ thống được chia thành nhiều lớp với trách nhiệm cụ thể, giúp đơn giản hóa quá trình kiểm thử, bao gồm kiểm thử đơn vị (unit test).

Tách biệt giao diện người dùng (UI): Việc thay đổi UI có thể thực hiện mà không cần thay đổi logic nghiệp vụ, giúp tăng khả năng mở rộng cho nhiều nền tảng khác nhau.

Độc lập với cơ sở dữ liệu: Cơ sở dữ liệu chỉ là một trong những thành phần lưu trữ, và có thể thay đổi dễ dàng mà không làm ảnh hưởng đến các logic của ứng dụng.

Độc lập với logic nghiệp vụ: Mọi nghiệp vụ cốt lõi của ứng dụng đều độc lập với các yếu tố hạ tầng như cơ sở dữ liệu, API, hoặc thư viện.

2.2.2 Các thành phần chính trong Clean Architecture

Clean Architecture được chia thành nhiều lớp khác nhau. Các lớp này có thứ tự phụ thuộc từ ngoài vào trong, và mỗi lớp chỉ có thể phụ thuộc vào các lớp nằm bên trong nó. Các thành phần chính bao gồm:

Entities (Domain Layer - Lớp Miền)

Entities là các đối tượng nghiệp vụ cốt lõi của ứng dụng. Chúng biểu diễn các quy tắc nghiệp vụ và các thuộc tính liên quan đến ứng dụng. Lớp này có thể chứa các định nghĩa lớp, các giá trị bất biến và các quy tắc xử lý liên quan.

Entities không phụ thuộc vào bất kỳ lớp nào khác. Chúng là nền tảng của toàn bộ ứng dụng và thường được kiểm thử độc lập.

Use Cases (Application Layer - Lớp Ứng Dụng)

Lớp Use Cases (hoặc Interactors) chứa logic nghiệp vụ ứng dụng. Nó đại diện cho các trường hợp sử dụng của hệ thống (những điều người dùng có thể làm với hệ thống).

Các Use Case định nghĩa các hành động mà hệ thống có thể thực hiện, và chúng thường sử dụng các Entities để hoàn thành nhiệm vụ của mình. Lớp này không biết gì về giao diện người dùng hoặc cơ sở dữ liệu. Điều này giúp logic nghiệp vụ không bị phụ thuộc vào các yếu tố khác của hệ thống.

Interface Adapters (Presentation Layer - Lớp Giao Diện)

Presentation Layer hoặc Interface Adapters chịu trách nhiệm chuyển đổi dữ liệu giữa các lớp khác nhau. Điều này có thể là chuyển đổi từ các Entities thành dữ liệu mà lớp giao diện người dùng cần, hoặc ngược lại.

Lớp này chứa các thành phần như Controllers, Presenters, Views hoặc bất kỳ phần nào liên quan đến giao tiếp giữa người dùng và hệ thống. Interface Adapters giúp các Use Case có thể giao tiếp với hệ thống bên ngoài mà không cần phải biết cách hệ thống đó hoạt động.

Infrastructure Layer (Lớp Hạ Tầng)

Infrastructure Layer chịu trách nhiệm cung cấp các dịch vụ cần thiết cho ứng dụng, chẳng hạn như cơ sở dữ liệu, các dịch vụ lưu trữ, hoặc các thư viện, framework.

Lớp này chứa các thành phần liên quan đến lưu trữ dữ liệu, triển khai framework, tích hợp với các dịch vụ bên thứ ba, v.v. Nó có thể sử dụng Entity Framework, MongoDB, hoặc bất kỳ công cụ nào khác để thao tác với cơ sở dữ liệu.

Các lớp bên trên không phụ thuộc trực tiếp vào hạ tầng, mà phụ thuộc vào các interface, do đó Infrastructure có thể dễ dàng được thay thế [7].

2.2.3 Nguyên tắc Dependency Rule (Quy tắc phụ thuộc)

Một trong những nguyên tắc quan trọng của Clean Architecture là Dependency Rule. Theo quy tắc này, tất cả các phụ thuộc phải hướng từ ngoài vào trong. Điều này có nghĩa là các lớp bên ngoài (như giao diện người dùng hoặc cơ sở dữ liệu) không được phép ảnh hưởng trực tiếp đến các lớp bên trong (như Domain hoặc Use Cases). Các lớp bên trong không biết gì về các lớp bên ngoài. Chúng chỉ có thể phụ thuộc vào các abstraction (interface hoặc contract), giúp giảm thiểu sự phụ thuộc cứng và tăng tính linh hoạt khi thay đổi hệ thống [7].

2.2.4 Ưu và nhược điểm của Clean Architecture

Ưu điểm:

Dễ bảo trì và mở rộng: Việc phân tách các lớp theo chức năng giúp dễ dàng thay đổi từng phần của hệ thống mà không ảnh hưởng đến các phần khác.

Độc lập với framework và hạ tầng: Có thể thay đổi framework, cơ sở dữ liệu, hoặc giao diện mà không cần thay đổi logic nghiệp vụ.

Khả năng kiểm thử cao: Logic nghiệp vụ được tách biệt khỏi các phần phụ thuộc như UI hoặc cơ sở dữ liệu, giúp dễ dàng kiểm thử.

Nhược điểm:

Phức tạp: Việc áp dụng Clean Architecture có thể làm cho hệ thống trở nên phức tạp, đặc biệt là đối với những dự án nhỏ hoặc trung bình.

Quá nhiều lớp và abstraction: Có thể dẫn đến việc tạo quá nhiều lớp, khiến mã nguồn trở nên khó quản lý nếu không có kinh nghiệm trong việc tổ chức.

2.3 React

React, là một thư viện JavaScript mã nguồn mở do Facebook phát triển và duy trì. Được ra mắt lần đầu vào năm 2013, React nhanh chóng trở thành một trong những công nghệ hàng đầu trong lĩnh vực phát triển giao diện người dùng (UI). React tập trung chủ yếu vào việc xây dựng giao diện ứng dụng một cách hiệu quả và dễ bảo trì, chủ yếu thông qua cách tiếp cận dựa trên các "component" - những khối xây dựng cơ bản của giao diện người dùng [1].

2.3.1 Component-Based Architecture (Kiến trúc dựa trên thành phần)

React sử dụng mô hình phát triển giao diện bằng cách chia giao diện thành các thành phần (component). Các component này có thể là những phần tử giao diện nhỏ, như một nút bấm (button), hoặc có thể phức tạp hơn như một biểu mẫu hay một trang. Các thành phần có thể tái sử dụng, giúp việc phát triển và bảo trì trở nên dễ dàng hơn. Thành phần là một trong những điểm mạnh lớn nhất của React, giúp giảm thiểu sự lặp lại và cho phép các nhà phát triển tổ chức mã dễ hiểu và dễ mở rộng.

Các component trong React có thể được viết bằng JavaScript kết hợp với JSX (JavaScript XML) - một phần mở rộng cú pháp cho phép viết mã có cấu trúc tương tự HTML ngay trong JavaScript.

Ví dụ:

```
function Welcome(props) {
  return <h1>Xin chào, {props.name}</h1>;
}
```

2.3.2 Virtual DOM

Virtual DOM là một trong những công nghệ quan trọng mà React sử dụng để cải thiện hiệu suất của ứng dụng. DOM (Document Object Model) là một cây cấu trúc của các thành phần trong trang web. Khi DOM thay đổi, trình duyệt cần phải cập nhật lại giao diện, và điều này thường tốn nhiều thời gian và gây giảm hiệu suất khi thao tác trên các trang phức tạp.

React sử dụng Virtual DOM - một bản sao của DOM thật - để giảm thiểu các thao tác với DOM. Khi trạng thái của ứng dụng thay đổi, React sẽ thực hiện cập nhật trên Virtual DOM trước, sau đó tính toán sự khác biệt (diffing) giữa DOM thật

và Virtual DOM, và cuối cùng chỉ cập nhật những phần cần thiết của DOM thật, giúp cải thiện đáng kể hiệu suất.

2.3.3 JSX (JavaScript XML)

JSX là một cú pháp mở rộng cho JavaScript, cho phép viết mã trông giống như HTML ngay trong JavaScript. JSX không phải là một yêu cầu bắt buộc để sử dụng React, nhưng nó giúp viết các thành phần trực quan và dễ hiểu hơn.

```
const element = <h1>Hello, world!</h1>;
```

JSX giúp các nhà phát triển dễ dàng định nghĩa và tổ chức giao diện, vì nó kết hợp sức mạnh của JavaScript và tính dễ hiểu của HTML. Khi được biên dịch, JSX sẽ trở thành các lệnh gọi hàm React.createElement(), tạo ra cấu trúc Virtual DOM.

2.3.4 State và Props (Trạng thái và Thuộc tính)

State và props là hai khái niệm quan trọng trong React để quản lý dữ liệu bên trong và bên ngoài của component.

Props: Props (viết tắt của "properties") là những thuộc tính mà một component nhận từ component cha. Props không thể thay đổi, có tính chất giống như dữ liệu "read-only". Chúng được sử dụng để truyền dữ liệu từ component cha xuống component con.

Ví dụ:

```
function Greeting(props) {
  return <h1>Xin chào, {props.name}!</h1>;
}
```

State: State là dữ liệu bên trong của một component và có thể thay đổi. State giúp component phản hồi với các tương tác của người dùng (chẳng hạn như nhấp chuột, nhập dữ liệu) bằng cách cập nhật lại giao diện khi có thay đổi trong state. State có thể được quản lý thông qua hàm useState() (trong Functional Components) hoặc thông qua this.state (trong Class Components)

Ví dụ:

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Bạn đã nhấp {count} lần</p>
      <button onClick={() => setCount(count +
1)}>Nhấp vào tôi</button>
    </div>
```

2.3.5 Lifecycle Methods (Các phương thức vòng đời)

Đối với các Class Components, React cung cấp một số phương thức gọi là các "lifecycle methods", dùng để kiểm soát hành vi của component trong các giai đoạn vòng đời của nó: khởi tạo, cập nhật, và hủy bỏ.

- componentDidMount(): Được gọi sau khi component được render lần đầu tiên.
- componentDidUpdate(): Được gọi sau khi component cập nhật do thay đổi state hoặc props.
- componentWillUnmount(): Được gọi ngay trước khi component bị hủy, thường dùng để giải phóng tài nguyên hoặc dừng việc nghe sự kiện.

2.3.6 React Hooks

React Hooks là tính năng mới được giới thiệu từ phiên bản 16.8, giúp các Functional Components có thể sử dụng state và các tính năng khác của Class Components mà không cần phải chuyển sang class.

Một số hooks phổ biến:

- useState(): Dùng để quản lý state trong component.
- useEffect(): Dùng để thay thế cho các lifecycle methods như componentDidMount, componentDidUpdate, và componentWillUnmount. useEffect() cho phép thực hiện các tác vụ phụ như gọi API, thiết lập bộ đếm thời gian...

- useContext(): Dùng để chia sẻ state giữa các component mà không cần phải truyền props xuống qua từng cấp.

Ví dụ:

```
import React, { useState, useEffect } from 'react';
function Timer() {
  const [seconds, setSeconds] = useState(0);
  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds(prevSeconds => prevSeconds + 1);
    }, 1000);
    return () => clearInterval(interval); // Cleanup
khi component bị hủy
  }, []);
  return <div>Thời gian đã trôi: {seconds} giây</div>;
}
```

2.3.7 Routing trong React

React Router là thư viện phổ biến được sử dụng để quản lý điều hướng trong ứng dụng React. Với React Router, bạn có thể định nghĩa các "route" khác nhau trong ứng dụng của mình và điều hướng giữa các trang một cách dễ dàng mà không cần tải lại trang.

2.3.8 State Management (Quản lý trạng thái)

Khi ứng dụng trở nên phức tạp hơn, việc quản lý state trở nên quan trọng và khó khăn hơn. Các công cụ quản lý trạng thái như Redux hay Context API của React được sử dụng để quản lý và chia sẻ state trên toàn bộ ứng dụng.

- Redux: Là thư viện quản lý state mạnh mẽ với mô hình một "store" duy nhất cho toàn bộ ứng dụng, sử dụng các khái niệm như actions, reducers, và store.
- Context API: Được sử dụng để quản lý và chia sẻ state giữa nhiều component mà không cần phải truyền props.

2.4 Giới thiệu về NodeJS

2.4.1 Khái niệm NodeJS

NodeJS là một môi trường thực thi đa nền tảng và mã nguồn mở, cho phép phát triển các ứng dụng nhanh chóng và có khả năng mở rộng. Nó được xây dựng trên

nền tảng "V8 JavaScript Engine" của Google Chrome. Cùng với đó là cấu trúc I/O non-block và mô hình event-driven (hướng sự kiện) giúp tăng hiệu suất và xử lý các ứng dụng thời gian thực một cách hiệu quả [4].

2.4.2 Ứng dụng của NodeJS

Ứng dụng thời gian thực (Real-time Applications): NodeJS rất phù hợp để xây dựng các ứng dụng cần kết nối và cập nhật liên tục giữa server và client, nhờ cơ chế event-driven và WebSocket.

API RESTful và GraphQL: NodeJS thường được sử dụng để xây dựng các API RESTful hoặc GraphQL, cung cấp dữ liệu cho ứng dụng frontend hoặc mobile.

Ứng dụng Single Page Application (SPA): NodeJS giúp xử lý yêu cầu và dữ liệu phía server hiệu quả, tạo nền tảng tốt cho các SPA.

Ứng dụng IoT (Internet of Things): NodeJS là một lựa chọn tuyệt vời để phát triển các hệ thống IoT nhờ khả năng xử lý sự kiện nhanh và sử dụng ít tài nguyên.

Truyền phát dữ liệu và nội dung (Streaming Applications): NodeJS hỗ trợ truyền phát dữ liệu theo luồng (streaming) hiệu quả, giúp xây dựng các ứng dụng phát trực tiếp hoặc xử lý file lớn.

Ứng dụng thương mại điện tử (E-commerce Applications): Nhờ khả năng xử lý đồng thời hàng nghìn yêu cầu, NodeJS phù hợp với các trang web thương mại điện tử cần tốc độ cao.

Ứng dụng máy chủ Proxy (Proxy Server Applications): NodeJS có thể được sử dụng như một máy chủ proxy để xử lý các yêu cầu giữa client và server khác, đặc biệt khi cần xử lý nhiều dịch vụ hoặc các API bên thứ ba.

Ứng dụng đa nền tảng (Cross-platform Applications): Node.js kết hợp với các công cụ như Electron có thể phát triển ứng dụng desktop chạy đa nền tảng.v

2.4.3 Ưu nhược điểm của NodeJS

Ưu điểm:

- Hiệu suất cao

- Xử lý thời gian thực tốt

- Sử dụng cùng một ngôn ngữ

- Hệ sinh thái phong phú

- Dễ dàng mở rộng

Nhược điểm:

- Hiệu suất kém với các tác vụ CPU nặng

- Không phù hợp với ứng dụng đơn luồng phức tạp

- Quản lý callback phức tạp

- Không có tính nhất quán

- Bảo mật

2.5 Hệ quản trị cơ sở dữ liệu MySQL

2.5.1 Khái niệm MySQL

MySQL là hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến hàng đầu trên thế giới và đặc biệt được ưa chuộng trong quá trình xây dựng, phát triển ứng dụng. Đây là hệ quản trị cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có khả năng thay đổi mô hình sử dụng phù hợp với điều kiện công việc khả chuyển. MySQL hoạt động trên nhiều hệ điều hành, cung cấp một hệ thống lớn các hàm tiện ích rất mạnh.

Với tốc độ và tính bảo mật cao, MySQL thích hợp với các ứng dụng có truy cập cơ sở dữ liệu trên internet. MySQL có thể tải miễn phí từ trang chủ với nhiều phiên bản cho các hệ điều hành khác nhau như: phiên bản Win32 cho các hệ điều hành dòng Windows, Linux, Mac OS X, Unix, FreeBSD, NetBSD, Novell NetWare, SGI Irix, Solaris, SunOS [8].

2.5.2 Ưu điểm

Sử dụng dễ dàng: MySQL là cơ sở dữ liệu tốc độ cao và ổn định, công cụ này dễ sử dụng và hoạt động trên nhiều hệ điều hành cung cấp hệ thống lớn các hàm tiện ích.

Tính bảo mật cao: MySQL phù hợp với các ứng dụng có truy cập cơ sở dữ liệu trên internet vì nó sở hữu nhiều tính năng bảo mật, thậm chí là bảo mật cấp cao.

Đa tính năng: MySQL có thể hỗ trợ hàng loạt các chức năng SQL từ hệ quản trị cơ sở dữ liệu quan hệ trực tiếp và cả gián tiếp.

Khả năng mở rộng và mạnh mẽ: Công cụ MySQL có khả năng xử lý khối dữ liệu lớn và có thể mở rộng khi cần thiết.

Tương thích trên nhiều hệ điều hành: MySQL tương thích để chạy trên nhiều hệ điều hành như Novell NetWare, Windows, Linux. MySQL cũng cung cấp phương tiện mà các máy khách có thể chạy trên cùng một máy tính với máy chủ hoặc trên một máy tính khác (giao tiếp qua mạng cục bộ hoặc Internet).

Cho phép khôi phục: MySQL cho phép các transactionđược khôi phục, cam kết và phục hồi sự cố.

2.5.3 Nhược điểm

MySQL cũng còn một số hạn chế phải kể đến như MySQL có thể bị khai thác để chiếm quyền điều khiển. Hơn thế nữa, MySQL không được tích hợp để sử dụng cho các hệ thống lớn cần quản lý lượng dữ liệu khổng lồ. Ví dụ như các hệ thống siêu thị trên toàn quốc, ngân hàng, quản lý thông tin dân số cả nước,...

2.6 Giới thiệu về RESTful API

2.6.1 Khái niệm về RESTful API

RESTful API (Representational State Transfer API) là một kiểu kiến trúc cho các API (Application Programming Interface) được sử dụng để truyền tải và trao đổi dữ liệu giữa các ứng dụng web. RESTful API sử dụng giao thức HTTP để truyền tải dữ liệu giữa máy chủ và máy khách, và sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các thao tác trên tài nguyên.

RESTful API sử dụng các URL dễ đọc và dễ hiểu, và sử dụng các định dạng dữ liệu như JSON hoặc XML để trao đổi thông tin giữa máy chủ và máy khách. RESTful API cũng có tính khả di động cao, cho phép các ứng dụng khác nhau có thể truy cập và sử dụng các tài nguyên một cách dễ dàng [9].

2.6.2 Các nguyên tắc của RESTful API

RESTful API được xây dựng dựa trên một tập hợp các nguyên tắc cốt lõi, giúp đảm bảo tính hiệu quả và linh hoạt trong thiết kế.

Client-Server Architecture (Kiến trúc Client-Server)

Statelessness (Không lưu trạng thái)

Cacheable (Có thể lưu cache)

Layered System (Hệ thống phân lớp)

Uniform Interface (Giao diện thống nhất)

Code on Demand (Mã được tải về) *(Không bắt buộc)*

2.6.3 Tại sao sử dụng RESTful API

Tính đơn giản và tiêu chuẩn hóa: Dựa trên HTTP: RESTful API sử dụng các phương thức HTTP cơ bản (GET, POST, PUT, DELETE) để thực hiện các thao tác CRUD (Create, Read, Update, Delete).

Khả năng tương thích đa nền tảng: RESTful API có thể được sử dụng bởi bất kỳ ứng dụng nào có khả năng gửi yêu cầu HTTP, như trình duyệt, ứng dụng di động, hoặc các dịch vụ backend.

Tính mở rộng (Scalability): Kiến trúc REST cho phép các hệ thống mở rộng dễ dàng nhờ vào khả năng xử lý tải trọng cao mà không làm gián đoạn các thành phần khác.

Khả năng tái sử dụng (Reusability): API được thiết kế theo RESTful dễ tái sử dụng trong các dự án khác nhau, giảm thời gian và chi phí phát triển.

Tách biệt giữa client và server: RESTful API cho phép tách biệt hoàn toàn phần giao diện người dùng (frontend) và logic xử lý (backend), giúp việc phát triển đồng thời hai phần này trở nên dễ dàng hơn.

Tính linh hoạt và tương lai: RESTful API có thể cung cấp dữ liệu ở nhiều định dạng khác nhau (JSON, XML, HTML), thích hợp với nhu cầu khác nhau của ứng dụng.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

Để đáp ứng nhu cầu ngày càng tăng về mua sắm trực tuyến, đặc biệt trong lĩnh vực giải trí ngoài trời, việc xây dựng một website thương mại điện tử chuyên nghiệp là cần thiết. Dự án này tập trung vào việc phát triển một nền tảng trực tuyến chuyên biệt cho lĩnh vực câu cá, nơi khách hàng có thể dễ dàng tìm kiếm và mua sắm các dụng cụ như cần câu, máy câu, dây câu, mồi câu và các phụ kiện hỗ trợ khác.

Các yêu cầu cụ thể của bài toán bao gồm:

Chức năng quản lý sản phẩm: Hệ thống cung cấp phần quản trị dành cho Admin để thực hiện các nghiệp vụ quản lý sản phẩm. Quản trị viên có thể cập nhật thông tin và theo dõi danh mục các dụng cụ, phụ kiện câu cá.

Chức năng giỏ hàng và thanh toán: Cho phép người dùng duyệt sản phẩm và thêm các mặt hàng mong muốn vào giỏ hàng. Hệ thống hỗ trợ quy trình đặt hàng và thanh toán để hoàn tất việc mua sắm.

Quản lý người dùng: Hệ thống hỗ trợ chức năng đăng ký và đăng nhập tài khoản cho người dùng. Khách hàng có thể quản lý thông tin cá nhân và theo dõi lịch sử cũng như trạng thái các đơn hàng đã đặt.

Tìm kiếm và bộ lọc: Cung cấp tính năng tìm kiếm sản phẩm, giúp người dùng nhanh chóng tìm thấy các loại dụng cụ câu cá phù hợp với nhu cầu.

Giao diện người dùng: Sử dụng công nghệ Next.js để triển khai giao diện thân thiện, trực quan, giúp tăng tốc độ tải trang và tối ưu hóa trải nghiệm người dùng trên hệ thống.

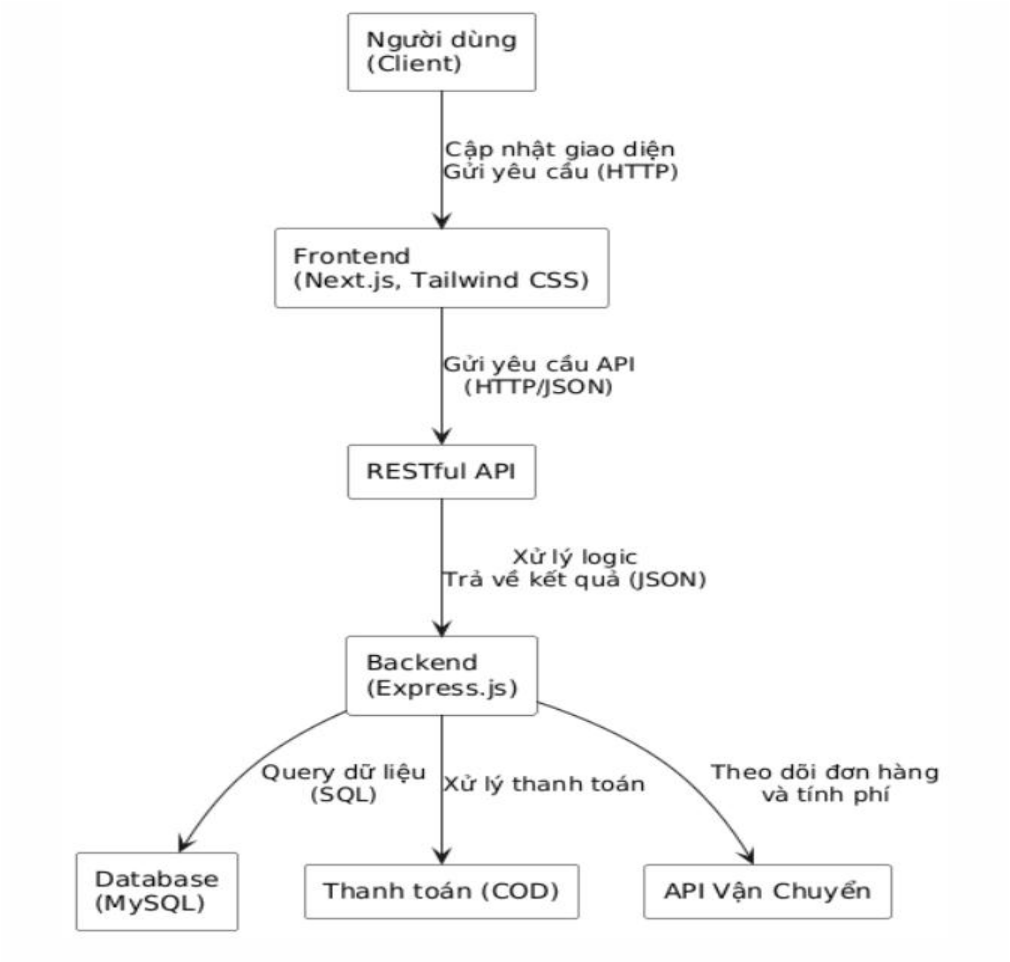
Quản lý đơn hàng: Phân hệ dành cho quản trị viên cho phép theo dõi và quản lý danh sách các đơn hàng từ khách hàng, đảm bảo quy trình vận hành diễn ra thông suốt.

Khả năng mở rộng: Hệ thống được thiết kế theo kiến trúc Clean Architecture và sử dụng TypeScript, giúp tách biệt rõ ràng các tầng xử lý, đảm bảo mã nguồn dễ bảo trì và sẵn sàng cho việc nâng cấp, mở rộng trong tương lai.

Mục tiêu chính: Xây dựng một website thương mại điện tử hoàn chỉnh, hoạt động ổn định trên nền tảng Express.js và MySQL. Ứng dụng không chỉ đáp ứng nhu

cầu mua sắm dụng cụ câu cá của khách hàng mà còn cung cấp công cụ quản lý hiệu quả cho admin, đồng thời đảm bảo tính toàn vẹn dữ liệu và hiệu suất truy vấn cao.

3.2 Kiến trúc hệ thống



Hình 3.1 Sơ đồ kiến trúc hệ thống

Kiến trúc hệ thống e-commerce bao gồm các thành phần và luồng dữ liệu chính như sau:

Người dùng (Client): Tương tác với hệ thống thông qua giao diện trên trình duyệt hoặc ứng dụng di động.

Frontend (Next.js, Tailwind CSS): Xây dựng giao diện hiện đại, thân thiện, tiếp nhận thao tác từ người dùng, gửi các yêu cầu đến API, và hiển thị dữ liệu trả về cho người dùng.

RESTful API: Đóng vai trò cầu nối giữa Frontend và Backend, tiếp nhận các yêu cầu từ Frontend và trả dữ liệu kết quả (dạng JSON) về.

Backend (Express.js): Xử lý toàn bộ logic nghiệp vụ của hệ thống như xác thực người dùng, truy vấn/ghi dữ liệu, xử lý thanh toán, tính phí vận chuyển và tích hợp với các hệ thống bên ngoài.

Database (MySQL): Lưu trữ toàn bộ dữ liệu hệ thống, gồm thông tin sản phẩm, tài khoản, đơn hàng, giao dịch, lịch sử mua bán,...

Cổng thanh toán (VN Pay): Kết nối với các dịch vụ thanh toán để hỗ trợ người dùng thực hiện giao dịch online một cách an toàn và thuận tiện.

API Vận chuyển: Tích hợp với dịch vụ vận chuyển để tính phí, theo dõi trạng thái đơn hàng, nâng cao trải nghiệm khách hàng.

Luồng dữ liệu hệ thống:

Người dùng thực hiện thao tác trên giao diện Frontend.

Frontend gửi yêu cầu API (HTTP/JSON) đến RESTful API.

RESTful API chuyển yêu cầu sang Backend để xử lý nghiệp vụ.

Backend sẽ truy vấn Database, gửi yêu cầu thanh toán hoặc tính toán vận chuyển khi cần thiết.

Kết quả xử lý (dữ liệu JSON) được Backend trả về RESTful API.

RESTful API chuyển kết quả về Frontend, dữ liệu sẽ được hiển thị cho người dùng.

3.3 Phân tích đặc tả yêu cầu hệ thống

3.3.1 Yêu cầu chức năng

Quản lý sản phẩm và Danh mục:

Quản trị viên (Admin): Có quyền thực hiện các thao tác thêm, sửa, xóa và cập nhật thông tin chi tiết cho từng loại sản phẩm. Thông tin bao gồm: tên sản phẩm, hình ảnh trực quan, mô tả kỹ thuật, giá niêm yết và trạng thái tồn kho (cần câu, máy câu, mồi câu,...).

Phân loại sản phẩm: Hệ thống hỗ trợ phân loại hàng hóa theo các danh mục chuyên biệt như dụng cụ câu cá (cần, máy, dây, lưỡi) hoặc phụ kiện bảo hộ (áo khoác chống nắng, hộp đựng) để người dùng dễ dàng theo dõi.

Trải nghiệm khách hàng và Mua sắm:

Duyệt và tìm kiếm: Khách hàng có thể dễ dàng duyệt danh sách sản phẩm theo từng danh mục hoặc sử dụng thanh công cụ để tìm kiếm nhanh chóng theo tên sản phẩm hoặc từ khóa liên quan.

Chi tiết sản phẩm: Cung cấp trang thông tin chi tiết bao gồm mô tả công dụng, hình ảnh sắc nét, giá cả và các đánh giá thực tế giúp khách hàng đưa ra quyết định mua sắm chính xác.

Giỏ hàng trực tuyến: Người dùng có quyền thêm sản phẩm vào giỏ hàng, tùy chỉnh số lượng hoặc loại bỏ sản phẩm trước khi tiến hành bước thanh toán.

Quy trình Đặt hàng và Thanh toán:

Thanh toán và Vận chuyển: Hệ thống hỗ trợ quy trình đặt hàng an toàn, tích hợp tính năng tính phí vận chuyển tự động (thông qua GHN API) và cung cấp các phương thức thanh toán linh hoạt cho người dùng.

Quản lý lịch sử: Khách hàng sau khi đăng nhập có thể theo dõi danh sách các đơn hàng đã thực hiện, xem lại lịch sử mua sắm và cập nhật trạng thái nhận hàng.

Quản lý người dùng và Tương tác:

Xác thực tài khoản: Hệ thống hỗ trợ chức năng đăng ký, đăng nhập và quản lý thông tin cá nhân dành cho thành viên, đảm bảo quyền riêng tư và bảo mật dữ liệu.

Tính năng bổ trợ: Tích hợp chuyên mục Blog chia sẻ kiến thức về kỹ thuật câu cá và cho phép khách hàng để lại đánh giá, phản hồi về chất lượng sản phẩm trên website.

Quản trị hệ thống (Admin Dashboard):

Quản lý đơn hàng: Admin có quyền tiếp nhận, kiểm tra và cập nhật trạng thái các đơn hàng (đang xử lý, đã giao, hủy) để đảm bảo luồng vận hành diễn ra thông suốt.

Thống kê và Nội dung: Cung cấp giao diện dashboard để theo dõi doanh số, quản lý bài viết trên Blog và kiểm soát toàn bộ nội dung hiển thị trên website nhằm tối ưu hiệu quả kinh doanh.

3.3.2 Yêu cầu phi chức năng

Hiệu năng và tối ưu hóa:

Ứng dụng đảm bảo tốc độ phản hồi nhanh chóng, tối ưu hóa tốc độ tải trang nhờ vào việc sử dụng framework Next.js.

Hệ thống xử lý truy vấn dữ liệu hiệu quả với MySQL thông qua kỹ thuật Connection pooling, đảm bảo hiệu suất ngay cả khi số lượng sản phẩm và đơn hàng tăng cao.

Bảo mật thông tin:

Hệ thống áp dụng cơ chế xác thực người dùng an toàn bằng JWT (JSON Web Token).

Toàn bộ thông tin nhạy cảm như mật khẩu người dùng được mã hóa bằng thuật toán bcrypt trước khi lưu trữ vào cơ sở dữ liệu.

Khả năng tương thích và giao diện:

Giao diện người dùng được thiết kế thân thiện, trực quan và hiện đại bằng Next.js.

Hệ thống hoạt động ổn định và tương thích trên các nền tảng trình duyệt phổ biến, đảm bảo trải nghiệm người dùng đồng nhất.

Thiết kế giao diện phản hồi (Responsive) giúp website hiển thị tốt trên cả máy tính và các thiết bị di động.

Khả năng bảo trì và mở rộng:

Mã nguồn được xây dựng trên ngôn ngữ TypeScript, giúp cấu trúc rõ ràng, chặt chẽ và dễ dàng kiểm soát lỗi.

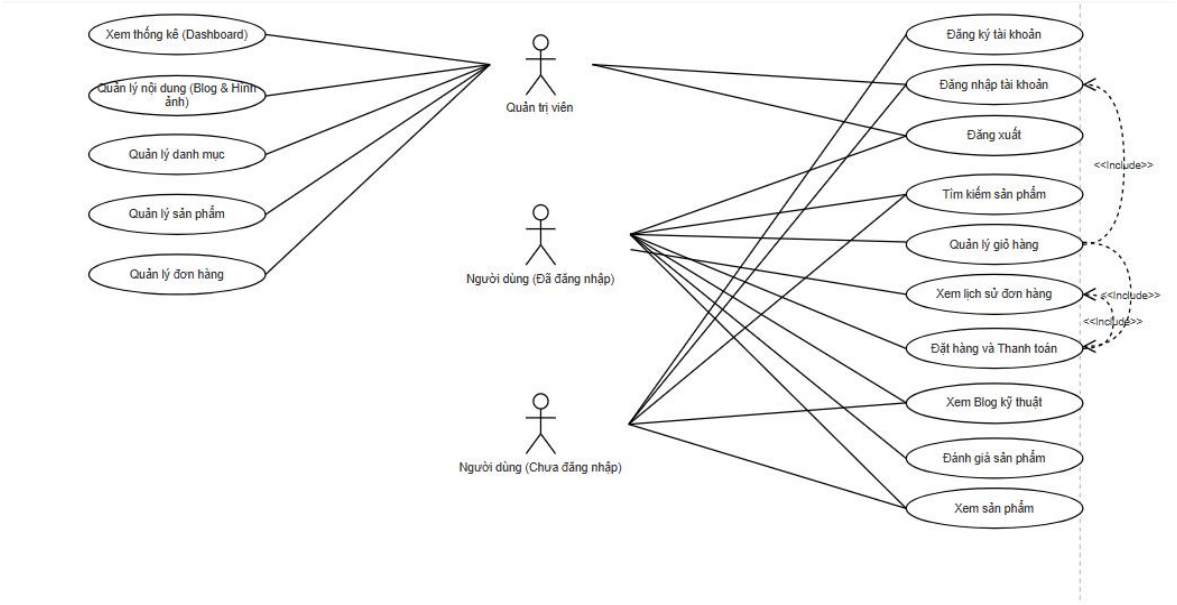
Áp dụng kiến trúc Clean Architecture cho Backend giúp tách biệt rõ ràng các tầng xử lý, tạo điều kiện thuận lợi cho việc bảo trì và nâng cấp tính năng trong tương lai.

Tích hợp dịch vụ và tài liệu:

Hệ thống hỗ trợ kết nối với các API RESTful để tích hợp các dịch vụ bên thứ ba như đơn vị vận chuyển (GHN API) và các cổng thanh toán.

3.4 Thiết kế hệ thống

3.4.1 Sơ đồ usecase



Hình 3.2 Sơ đồ usecase

3.4.2 Sơ đồ lớp

Bảng activity_logs

Mục đích: Lưu nhật ký hoạt động của người dùng và hệ thống

Bảng 3.1 Bảng nhật ký hoạt động

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
Id	int	ID bản ghi
user_id	int	ID người dùng (NULL nếu là hệ thống)
action	varchar(120)	Hành động (login, tạo đơn, cập nhật sản phẩm, …)
metadata	longtext	Dữ liệu bổ sung dạng JSON
created_at	timestamp	Thời gian tạo

Bảng addresses

Mục đích: Quản lý địa chỉ giao hàng của người dùng

Bảng 3.2 Bảng địa chỉ

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID địa chỉ
user_id	int	ID người dùng

Tên thuộc tính	*Kiểu dữ liệu*	*Mô tả*
label	varchar(60)	Nhãn địa chỉ (Nhà riêng, Công ty, …)
full_name	varchar(120)	Họ tên người nhận
phone	varchar(20)	Số điện thoại
address_line	text	Địa chỉ chi tiết
province	varchar(80)	Tỉnh / Thành phố
district	varchar(80)	Quận / Huyện
ward	varchar(80)	Phường / Xã
is_default	tinyint(1)	Địa chỉ mặc định (1: có, 0: không)
created_at	timestamp	Thời gian tạo

Bảng blogs

Mục đích: Quản lý bài viết / tin tức

Bảng 3.3 Bảng quản lý bài viết

Tên thuộc tính	*Kiểu dữ liệu*	*Mô tả*
id	int	ID blog
title	varchar(255)	Tiêu đề bài viết
slug	varchar(255)	Đường dẫn SEO
excerpt	text	Tóm tắt ngắn
content	longtext	Nội dung đầy đủ
thumbnail	varchar(500)	Ảnh đại diện
author_id	int	ID tác giả
category	varchar(100)	Danh mục blog
tags	longtext	Các thẻ tag
view_count	int	Số lượt xem
is_published	tinyint(1)	Trạng thái xuất bản
published_at	datetime	Thời gian xuất bản
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng carts

Mục đích: Quản lý giỏ hàng của người dùng

Bảng 3.4 Bảng giỏ hàng

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID giỏ hàng
user_id	int	ID người dùng
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng cart_items

Mục đích: Lưu chi tiết sản phẩm trong giỏ hàng

Bảng 3.5 Bảng chi tiết sản phẩm trong giỏ hàng

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID chi tiết giỏ hàng
cart_id	int	ID giỏ hàng
product_id	int	ID sản phẩm
quantity	int	Số lượng
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng categories

Mục đích: Quản lý danh mục sản phẩm

Bảng 3.6 Bảng quản lý danh mục sản phẩm

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID danh mục
name	varchar(120)	Tên danh mục
description	text	Mô tả danh mục
created_at	timestamp	Thời gian tạo

Bảng coupons

Mục đích: Quản lý mã giảm giá

Bảng 3.7 Bảng mã giảm giá

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID mã giảm giá
code	varchar(50)	Mã coupon
description	text	Mô tả
discount_type	enum	Loại giảm (percentage / fixed)
discount_value	decimal(10,2)	Giá trị giảm
min_order_value	decimal(10,2)	Giá trị đơn tối thiểu
max_discount	decimal(10,2)	Giảm tối đa
usage_limit	int	Số lần dùng tối đa
used_count	int	Số lần đã dùng
start_date	datetime	Ngày bắt đầu
end_date	datetime	Ngày kết thúc
is_active	tinyint(1)	Trạng thái
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng flash_sales

Mục đích: Quản lý chương trình flash sale

Bảng 3.8 Bảng quản lý flash sale

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int	ID flash sale
product_id	int	ID sản phẩm
discount_percentage	decimal(5,2)	Phần trăm giảm giá
start_time	datetime	Thời gian bắt đầu
end_time	datetime	Thời gian kết thúc
status	enum	Trạng thái
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng orders

Mục đích: Quản lý đơn hàng

Bảng 3.9 Bảng quản lý đơn hàng

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID đơn hàng
user_id	int unsigned	ID người dùng
address_id	int unsigned	ID địa chỉ
status	enum	Trạng thái đơn hàng
payment_method	enum	Phương thức thanh toán
total_amount	decimal(12,2)	Tổng tiền
note	text	Ghi chú
shipping_fee	decimal(12,2)	Phí vận chuyển
ghn_order_code	varchar(50)	Mã đơn GHN
recipient_name	varchar(120)	Tên người nhận
recipient_phone	varchar(20)	SĐT người nhận
recipient_address	text	Địa chỉ nhận
province_id	int	ID tỉnh GHN
district_id	int	ID quận GHN
ward_code	varchar(20)	Mã phường GHN
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng order_items

Mục đích: Chi tiết sản phẩm trong đơn hàng

Bảng 3.10 Bảng chi tiết sản phẩm trong đơn hàng

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID chi tiết
order_id	int unsigned	ID đơn hàng
product_id	int unsigned	ID sản phẩm
quantity	int	Số lượng
price	decimal(12,2)	Giá tại thời điểm mua
created_at	timestamp	Thời gian tạo

Bảng products

Mục đích: Quản lý sản phẩm bán hàng

Bảng 3.11 Bảng sản phẩm

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID sản phẩm
category_id	int unsigned	ID danh mục
name	varchar(200)	Tên sản phẩm
description	text	Mô tả chi tiết
price	decimal(12,2)	Giá bán
sku	varchar(80)	Mã SKU
stock_quantity	int	Số lượng tồn kho
thumbnail_url	varchar(255)	Ảnh đại diện
status	enum	Trạng thái
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng product_images

Mục đích: Lưu hình ảnh sản phẩm

Bảng 3.12 Bảng hình ảnh sản phẩm

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID hình ảnh
product_id	int unsigned	ID sản phẩm
image_url	varchar(255)	URL hình ảnh
alt_text	varchar(120)	Mô tả ảnh
is_primary	tinyint(1)	Ảnh chính
created_at	timestamp	Thời gian tạo

Bảng product_reviews

Mục đích: Đánh giá sản phẩm

Bảng 3.13 Bảng đánh giá sản phẩm

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID đánh giá
product_id	int unsigned	ID sản phẩm
user_id	int unsigned	ID người dùng
rating	tinyint	Điểm (1–5 sao)
comment	text	Nội dung
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng refresh_tokens

Mục đích: Quản lý token đăng nhập

Bảng 3.14 Bảng quản lý token đăng nhập

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID token
user_id	int unsigned	ID người dùng
token	varchar(255)	Refresh token
expires_at	datetime	Hết hạn
created_at	timestamp	Thời gian tạo

Bảng users

Mục đích: Quản lý người dùng hệ thống

Bảng 3.15 Bảng người dùng

Tên thuộc tính	***Kiểu dữ liệu***	***Mô tả***
id	int unsigned	ID người dùng
full_name	varchar(120)	Họ tên
email	varchar(150)	Email
password_hash	varchar(255)	Mật khẩu mã hóa
role	enum	Vai trò
phone	varchar(20)	SĐT

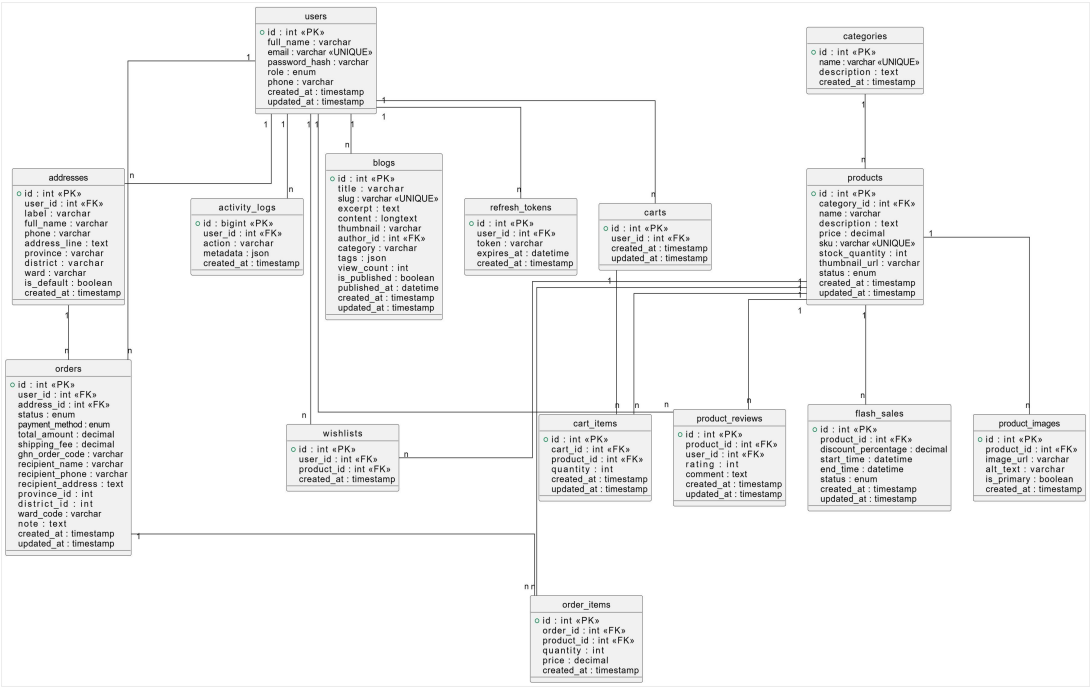
Tên thuộc tính	*Kiểu dữ liệu*	*Mô tả*
created_at	timestamp	Thời gian tạo
updated_at	timestamp	Thời gian cập nhật

Bảng wishlists

Mục đích: Danh sách sản phẩm yêu thích

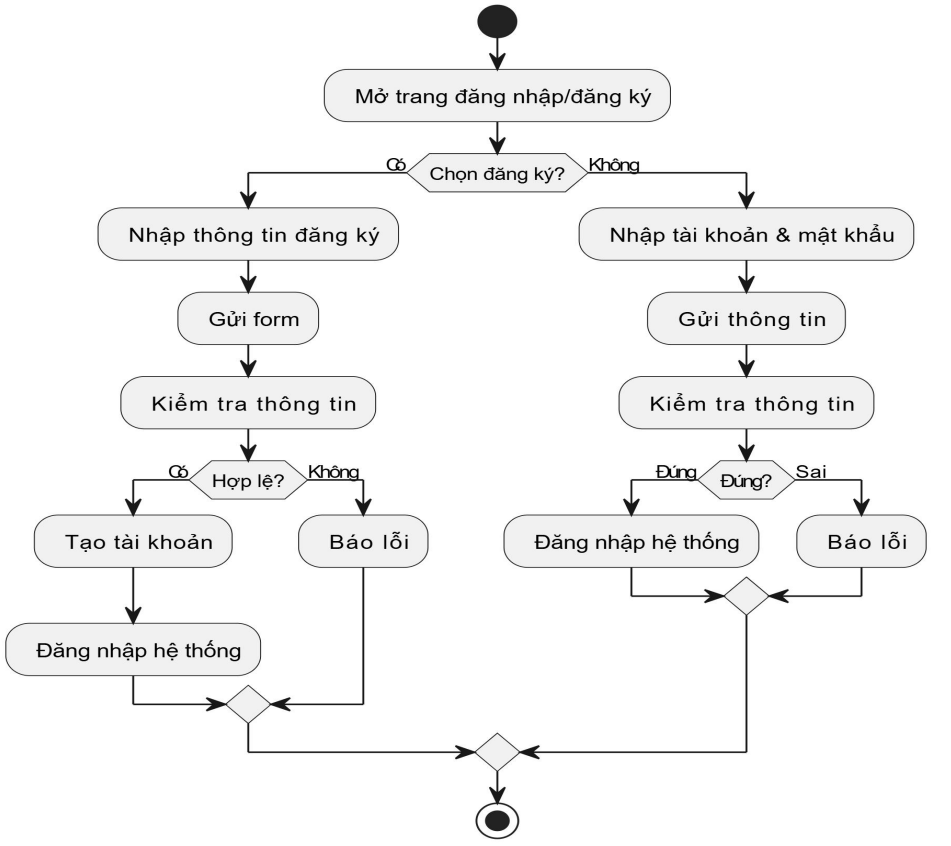
Bảng 3.16 Bảng danh sách sản phẩm yêu thích

Tên thuộc tính	*Kiểu dữ liệu*	*Mô tả*
id	int unsigned	ID wishlist
user_id	int unsigned	ID người dùng
product_id	int unsigned	ID sản phẩm
created_at	timestamp	Thời gian thêm

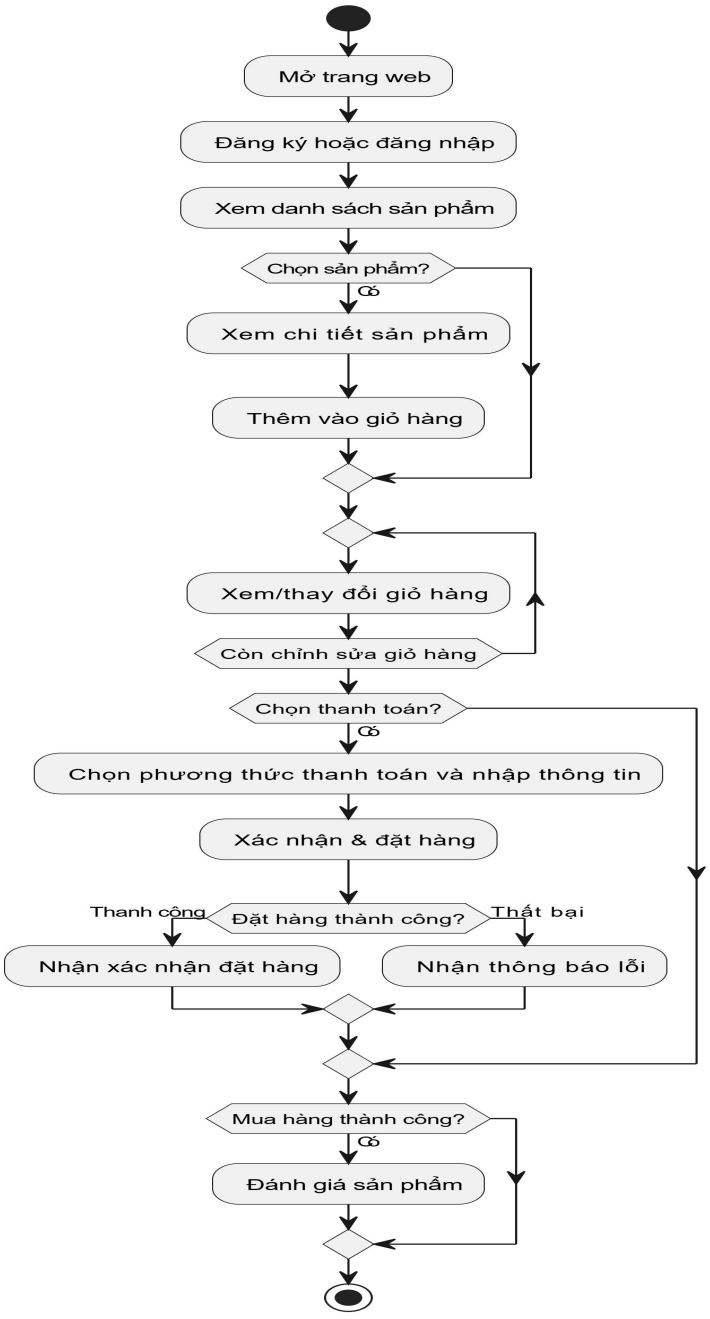


Hình 3.3 Sơ đồ lớp

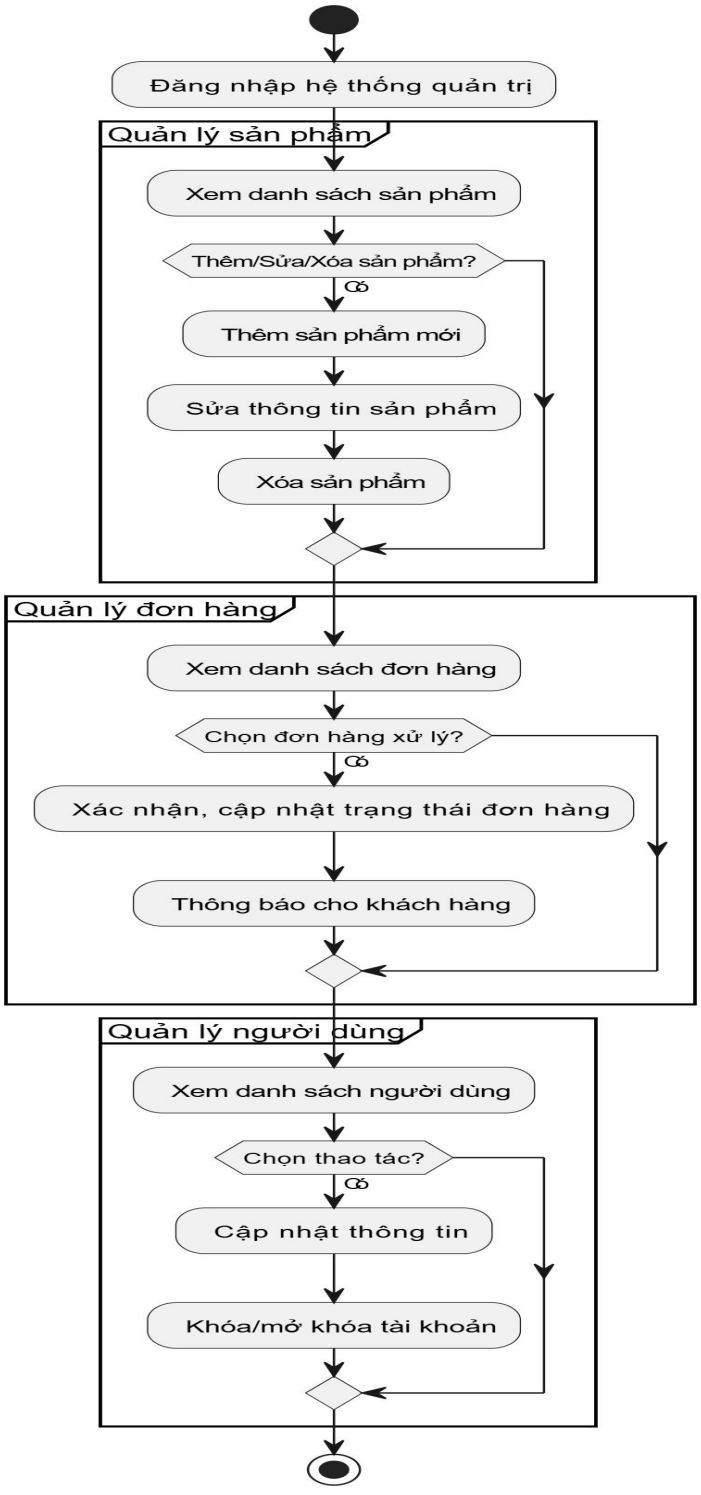
3.4.3 Sơ đồ hoạt động



Hình 3.4 Sơ đồ đăng ký, đăng nhập



Hình 3.5 Sơ đồ hoạt động người dùng



Hình 3.6 Sơ đồ hoạt động người quản trị

Các hình khối:

Hình chữ nhật bo tròn: Thể hiện một hành động hoặc bước xử lý.

Hình thoi: Thể hiện một điểm quyết định (có/không).

Vòng tròn đen: Điểm bắt đầu của quy trình.

Vòng tròn có viền: Điểm kết thúc của quy trình.

Sơ đồ đăng nhập & đăng ký

Sơ đồ mô tả quy trình đăng nhập và đăng ký cho người dùng. Từ điểm bắt đầu (vòng tròn đen), người dùng được lựa chọn giữa đăng nhập hoặc đăng ký thông qua hình thoi (quyết định). Các bước như nhập thông tin, kiểm tra hợp lệ và nhận phản hồi từ hệ thống đều được thể hiện bằng các hình chữ nhật bo tròn. Điểm quyết định xuất hiện tại các bước xác nhận tính hợp lệ của dữ liệu (hợp lệ/không hợp lệ), dẫn đến tiếp tục quy trình hoặc kết thúc tại vòng tròn có viền nếu gặp lỗi.

Quy trình chi tiết gồm:

Đăng ký: Nhập thông tin, kiểm tra hợp lệ, tạo tài khoản mới, chuyển sang đăng nhập.

Đăng nhập: Nhập thông tin, kiểm tra tính hợp lệ, truy cập hệ thống hoặc nhận thông báo lỗi.

Sơ đồ hoạt động Người dùng

Sơ đồ mô tả quy trình mua hàng trực tuyến của người dùng từ lúc đăng nhập cho đến khi hoàn tất đơn hàng. Điểm bắt đầu là vòng tròn đen. Người dùng thực hiện các hành động như đăng nhập, xem sản phẩm, chọn sản phẩm, thêm vào giỏ hàng (tất cả là chữ nhật bo tròn). Sau đó, đến các quyết định (hình thoi) như giỏ hàng có hợp lệ không, có chọn thanh toán không,... Các luồng xử lý tiếp tục với nhập thông tin giao hàng, thanh toán; nếu thanh toán thành công, đơn hàng được tạo và thông báo đẩy tới người dùng. Nếu gặp lỗi, có thể quay lại các bước trước hoặc liên hệ hỗ trợ. Quy trình kết thúc tại vòng tròn có viền, thể hiện việc hoàn tất đơn hàng hoặc hủy/thoát quy trình.

Các bước chính:

Đăng nhập và kiểm tra hợp lệ

Xem và chọn sản phẩm

Thêm vào giỏ hàng và kiểm tra

Chọn thanh toán, nhập thông tin giao hàng

Xác minh thanh toán, cập nhật trạng thái đơn hàng, nhận thông báo

Kết thúc quy trình hoặc nhận thông báo lỗi/hỗ trợ

Sơ đồ hoạt động Người quản trị (Admin)

Sơ đồ mô tả quy trình quản lý hệ thống của người quản trị, bắt đầu bằng vòng tròn đen. Admin thực hiện các hành động quản lý chính như quản lý sản phẩm, đơn hàng, người dùng - thể hiện qua những hình chữ nhật bo tròn. Các quyết định (hình thoi) giúp phân nhánh quy trình: ví dụ chọn thêm, sửa, xóa sản phẩm; xử lý đơn hàng hay cập nhật trạng thái tài khoản người dùng. Nếu thực hiện xong thao tác, quy trình tiến đến điểm kết thúc (vòng tròn có viền) hoặc tiếp tục chu trình quản lý.

Quy trình chi tiết gồm:

Đăng nhập hệ thống quản trị

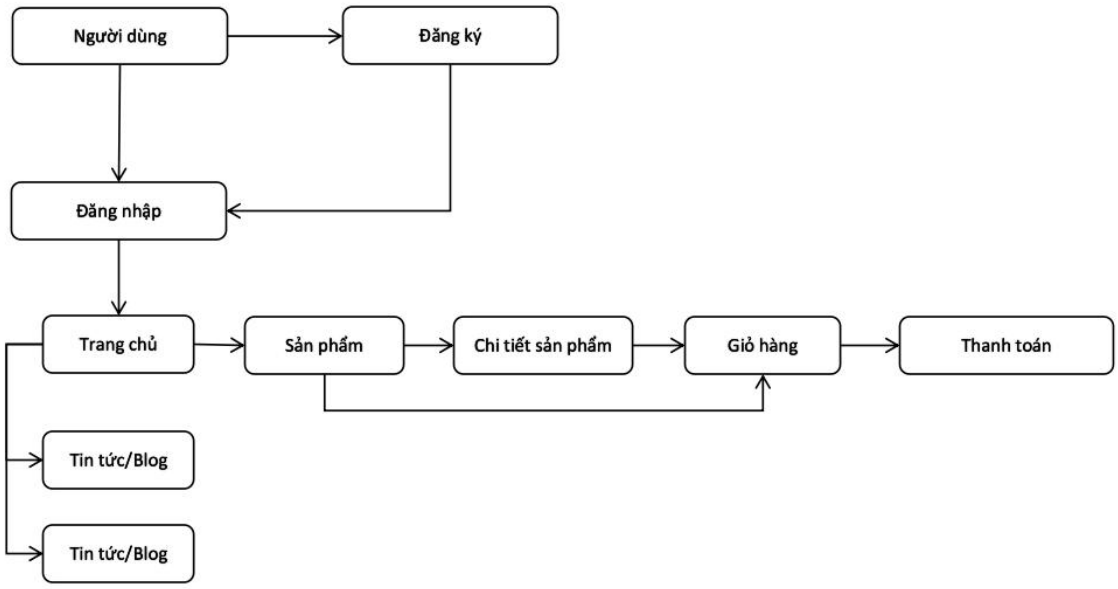
Xem, thêm mới, chỉnh sửa, xóa sản phẩm

Xem, xử lý, cập nhật trạng thái đơn hàng

Xem danh sách người dùng, chỉnh sửa, khóa/mở khóa tài khoản

Quy trình kết thúc khi hoàn tất các tác vụ hoặc thoát khỏi hệ thống

3.5 Thiết kế giao diện



Hình 3.7 Sơ đồ giao diện

Sơ đồ trang web (Sitemap) mô tả cấu trúc của một trang web e-commerce bán dụng cụ và phụ kiện câu cá với các mối quan hệ như sau: Người dùng có thể

thực hiện đăng ký tài khoản mới hoặc đăng nhập vào hệ thống. Sau khi xác thực, người dùng có thể truy cập vào trang chủ để bắt đầu trải nghiệm mua sắm.

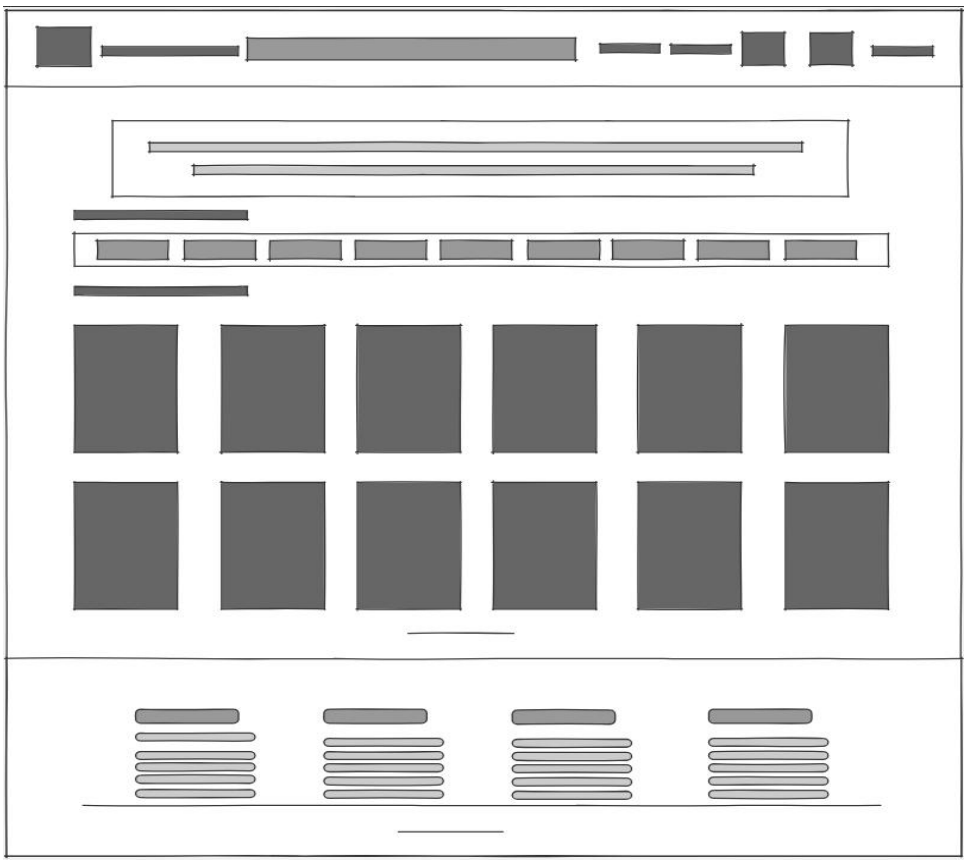
Từ trang chủ, người dùng có thể truy cập vào trang giới thiệu để tìm hiểu thông tin về cửa hàng hoặc chuyển sang trang sản phẩm để duyệt danh sách các dụng cụ câu cá như cần câu, máy câu, mồi câu. Tại đây, người dùng có thể chọn một mặt hàng cụ thể để xem chi tiết sản phẩm bao gồm hình ảnh, thông số kỹ thuật và giá cả.

Người dùng có thể trực tiếp thêm sản phẩm vào giỏ hàng từ danh sách hoặc từ trang chi tiết. Sau khi kiểm tra giỏ hàng, người dùng tiến hành quy trình thanh toán để hoàn tất giao dịch và có thể theo dõi lại trạng thái tại trang lịch sử đơn hàng. Ngoài ra, từ trang chủ, người dùng còn có thể truy cập chuyên mục Tin tức/Blog để tham khảo các bài viết về kỹ thuật câu cá.

Đối với quản trị viên, hệ thống cung cấp Dashboard tổng quan để quản lý toàn diện các phân hệ bao gồm: quản lý sản phẩm (thêm, sửa, xóa), quản lý đơn hàng, quản lý người dùng và biên tập nội dung blog.

3.5.1 Giao diện trang người dùng

Hình 3.8 Màn hình giao diện chưa đăng nhập



Hình 3.9 Màn hình giao diện đã đăng nhập

Mô tả: Giao diện trang chủ gồm 3 phần: đầu trang, thân trang và chân trang. Phần đầu trang và thân trang được cố định ở đa số các giao diện ngoại trừ đăng nhập và đăng ký. Phần đầu trang gồm có logo tên trang web và các nút chức năng. Phần thân trang gồm có các danh mục nổi bậc, sản phẩm nổi bậc và các chương trình khuyến mãi. Phần chân trang gồm các thông tin cơ bản của trang web.

Chức năng: Hiển thị các danh mục, sản phẩm nổi bật và các chương trình khuyến mãi. Người dùng có thể dễ dàng điều hướng đến các danh mục và sản phẩm cụ thể.

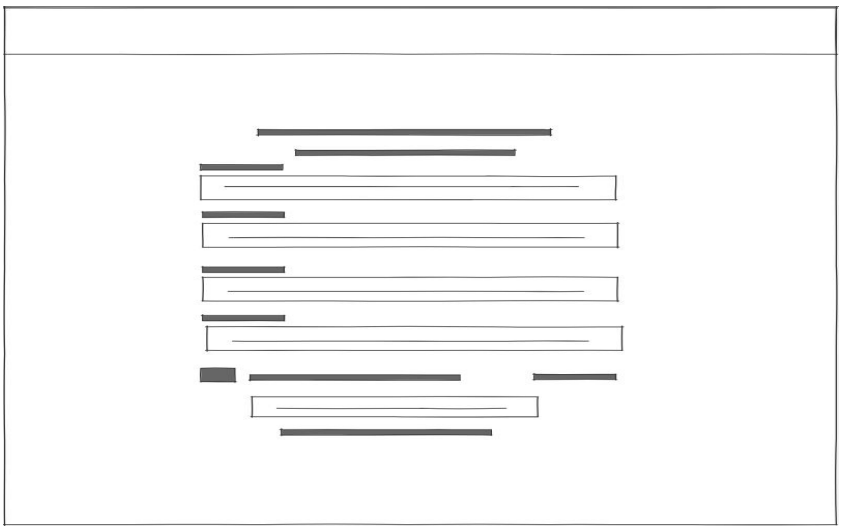
3.5.2 Giao diện trang đăng nhập

Hình 3.10 Màn hình giao diện trang đăng nhập

Mô tả: Giao diện trang đăng nhập gồm 1 phần: ở giữa trang sẽ có một thành phần để người dùng có thể đăng nhập tài khoản. Trong thành phần đăng nhập sẽ có các liên kết để người dùng có thể điều hướng sang các giao diện khác.

Chức năng: Cho phép người dùng đăng nhập vào tài khoản bằng email và mật khẩu, với liên kết để khôi phục mật khẩu hoặc chuyển sang trang đăng ký.

3.5.3 Giao diện trang đăng ký



Hình 3.11 Màn hình giao diện trang đăng ký

Mô tả: Giao diện trang đăng ký gồm 1 phần: ở giữa trang sẽ có một thành phần để người dùng có thể đăng ký tài khoản. Trong thành phần đăng ký sẽ có các liên kết để người dùng có thể điều hướng sang các giao diện khác.

Chức năng: Cho phép người dùng tạo tài khoản mới bằng cách nhập thông tin cá nhân, email và mật khẩu. Cung cấp liên kết đến trang đăng nhập cho người dùng đã có tài khoản.

3.5.4 Giao diện trang sản phẩm

Hình 3.12 Màn hình giao diện trang sản phẩm

Mô tả: Giao diện trang chủ gồm 3 phần: đầu trang, thân trang và chân trang. Phần đầu trang và phần chân trang giống như mô tả của trang người dùng. Phần thân trang sẽ có các bộ lọc để lọc sản phẩm, các nút để sắp xếp bố cục cho giao diện và phần chính để hiển thị danh sách tất cả sản phẩm.

Chức năng: Hiển thị danh sách tất cả sản phẩm trong các danh mục, với các bộ lọc và sắp xếp theo giá, thương hiệu hoặc đánh giá, giúp người dùng tìm kiếm nhanh chóng sản phẩm mong muốn.

3.5.5 Giao diện trang giới thiệu

Hình 3.13 Màn hình giao diện trang giới thiệu

Mô tả: Giao diện trang giới thiệu hiển thị các thông tin tổng quan về website, mục tiêu hoạt động và các nội dung liên quan đến sản phẩm, dịch vụ cung cấp. Bố cục trang được thiết kế đơn giản, rõ ràng, giúp người dùng dễ dàng nắm bắt thông tin. Trang giới thiệu có các liên kết điều hướng cho phép người dùng truy cập nhanh đến các trang khác của website.

Chức năng: Cung cấp thông tin giới thiệu chung về website và lĩnh vực kinh doanh dụng cụ, phụ kiện câu cá. Hỗ trợ người dùng định hướng nội dung, tăng mức độ tin cậy và điều hướng thuận tiện đến các chức năng khác như xem sản phẩm, đăng nhập hoặc đăng ký tài khoản.

3.5.6 Giao diện trang blog

Hình 3.14 Màn hình giao diện trang blog

Mô tả: Giao diện trang blog hiển thị danh sách các bài viết chia sẻ kiến thức, kinh nghiệm và tin tức liên quan đến hoạt động câu cá. Các bài viết được sắp xếp khoa học, dễ theo dõi, cho phép người dùng lựa chọn và xem nội dung chi tiết từng bài. Giao diện có tích hợp các liên kết điều hướng đến những trang khác trong hệ thống.

Chức năng: Cho phép người dùng xem danh sách bài viết và đọc nội dung chi tiết từng bài blog. Cung cấp thông tin hữu ích, giúp người dùng nâng cao kiến thức, đồng thời tăng mức độ tương tác và gắn kết giữa người dùng với website.

3.5.7 Giao diện người quản trị

Hình 3.15 Màn hình giao diện trang quản trị

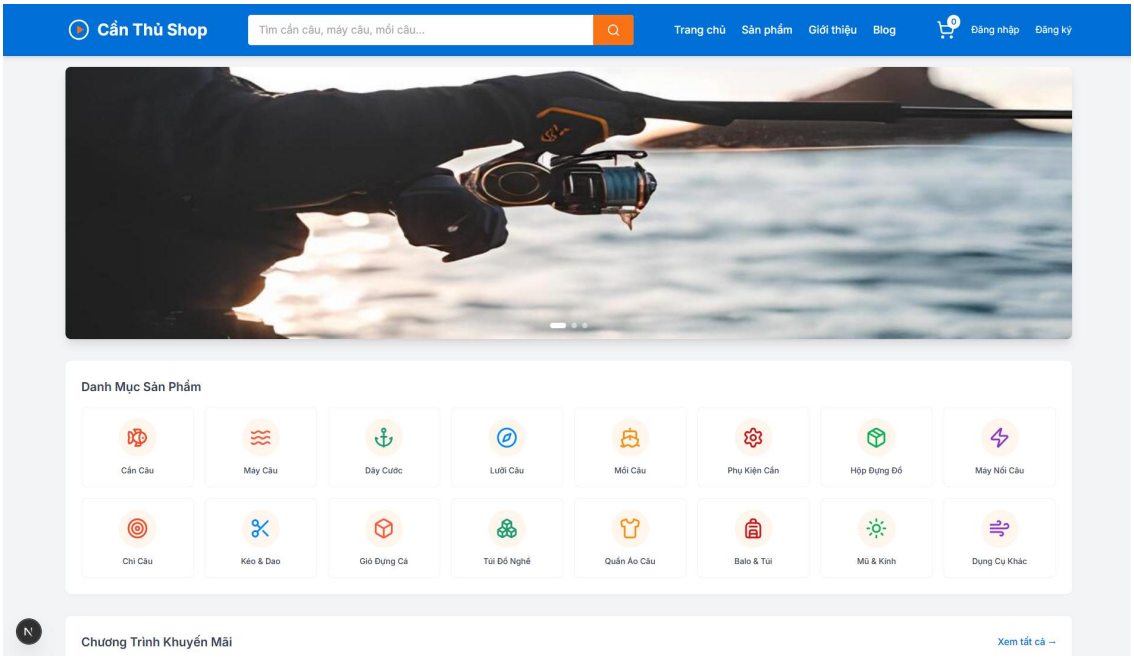
Mô tả: Giao diện trang blog hiển thị danh sách các bài viết chia sẻ kiến thức, kinh nghiệm và tin tức liên quan đến hoạt động câu cá. Các bài viết được sắp xếp khoa học, dễ theo dõi, cho phép người dùng lựa chọn và xem nội dung chi tiết từng bài. Giao diện có tích hợp các liên kết điều hướng đến những trang khác trong hệ thống.

Chức năng: Cho phép người dùng xem danh sách bài viết và đọc nội dung chi tiết từng bài blog. Cung cấp thông tin hữu ích, giúp người dùng nâng cao kiến thức, đồng thời tăng mức độ tương tác và gắn kết giữa người dùng với website.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Kết quả đạt được

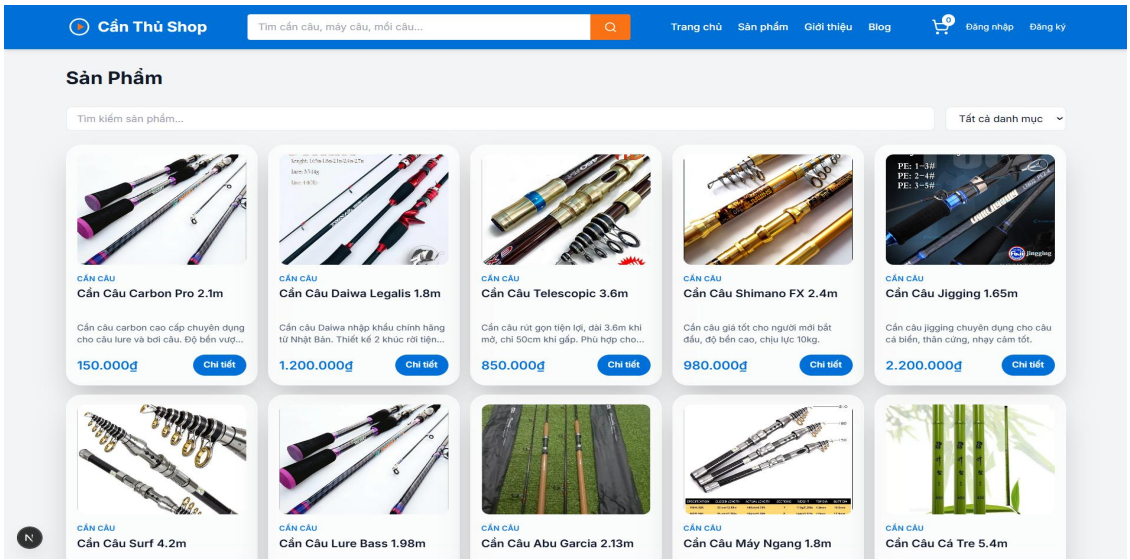
4.1.1 Giao diện trang chủ



Hình 4.1 Giao diện trang chủ

Chức năng: Hiển thị tổng quan website với banner, danh mục nổi bật, sản phẩm mới hoặc bán chạy, giúp người dùng nhanh chóng tiếp cận nội dung và sản phẩm chính.

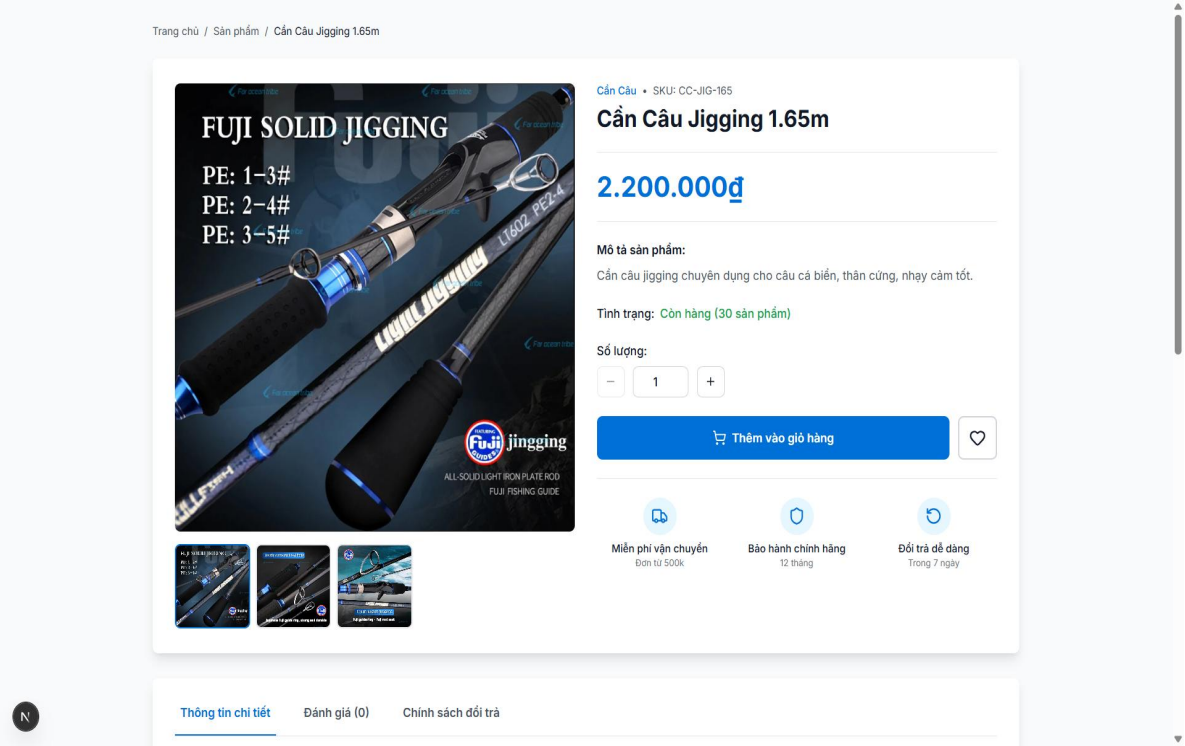
4.1.2 Giao diện trang sản phẩm



Hình 4.2 Giao diện trang sản phẩm

Chức năng: Hiển thị danh sách sản phẩm trong các danh mục, với các bộ lọc và sắp xếp theo giá, thương hiệu hoặc đánh giá, giúp người dùng tìm kiếm nhanh chóng.

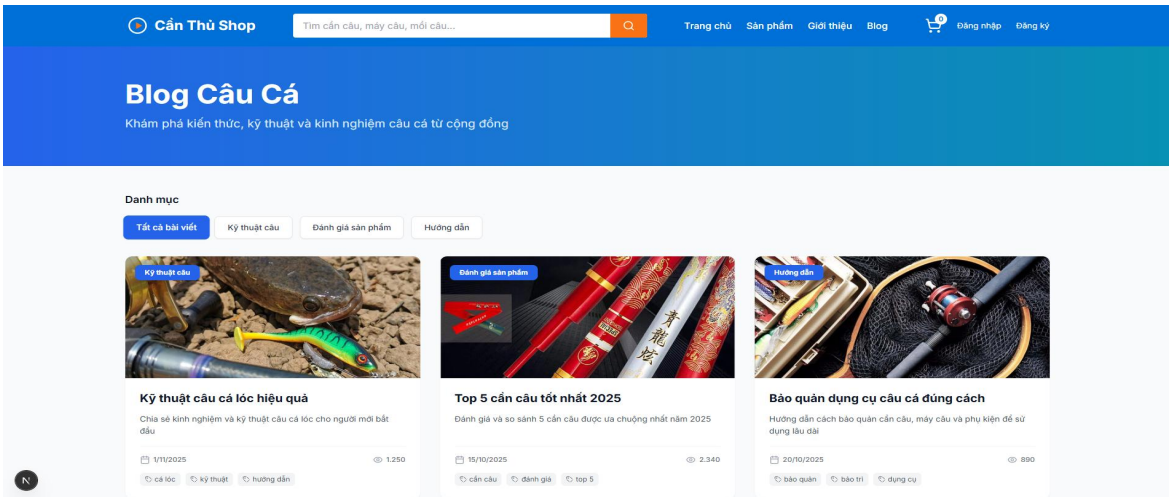
4.1.3 Giao diện trang chi tiết sản phẩm



Hình 4.3 Giao diện trang chi tiết sản phẩm

Chức năng: Hiển thị thông tin chi tiết của từng sản phẩm như hình ảnh, mô tả, giá bán, đánh giá và nút thêm vào giỏ hàng để hỗ trợ quyết định mua sắm.

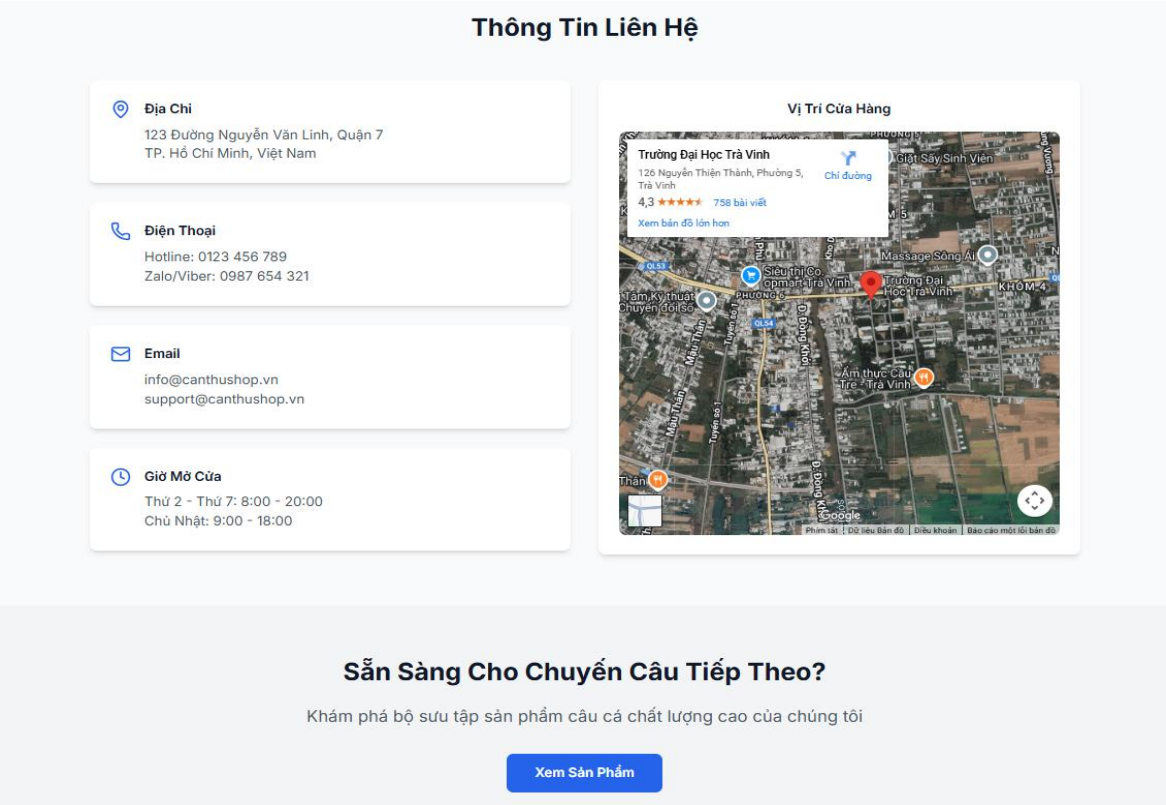
4.1.4 Giao diện trang blog



Hình 4.4 Giao diện trang blog

Chức năng: Hiển thị danh sách bài viết, tin tức hoặc chia sẻ kiến thức, cho phép người dùng đọc, tìm kiếm và theo dõi nội dung mới.

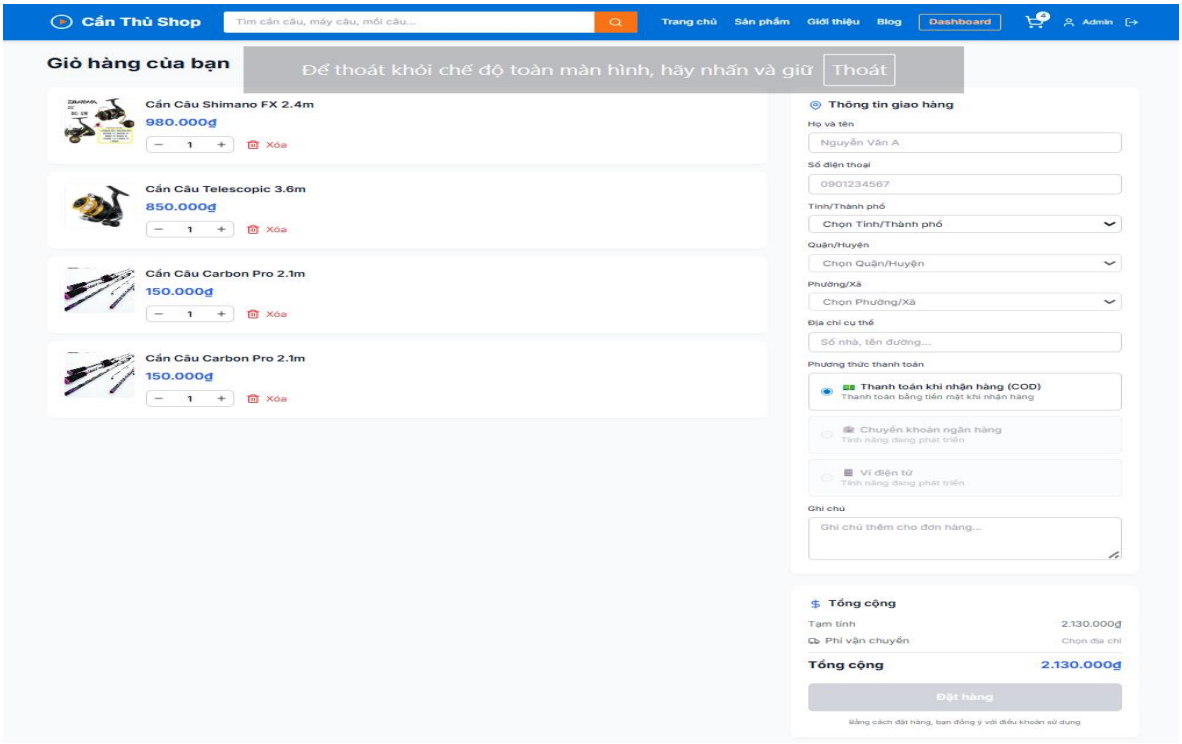
4.1.5 Giao diện trang giới thiệu



Hình 4.5 Giao diện trang giới thiệu

Chức năng: Hiển thị thông tin về doanh nghiệp, thương hiệu, sứ mệnh và giá trị cốt lõi nhằm tạo sự tin tưởng cho người dùng.

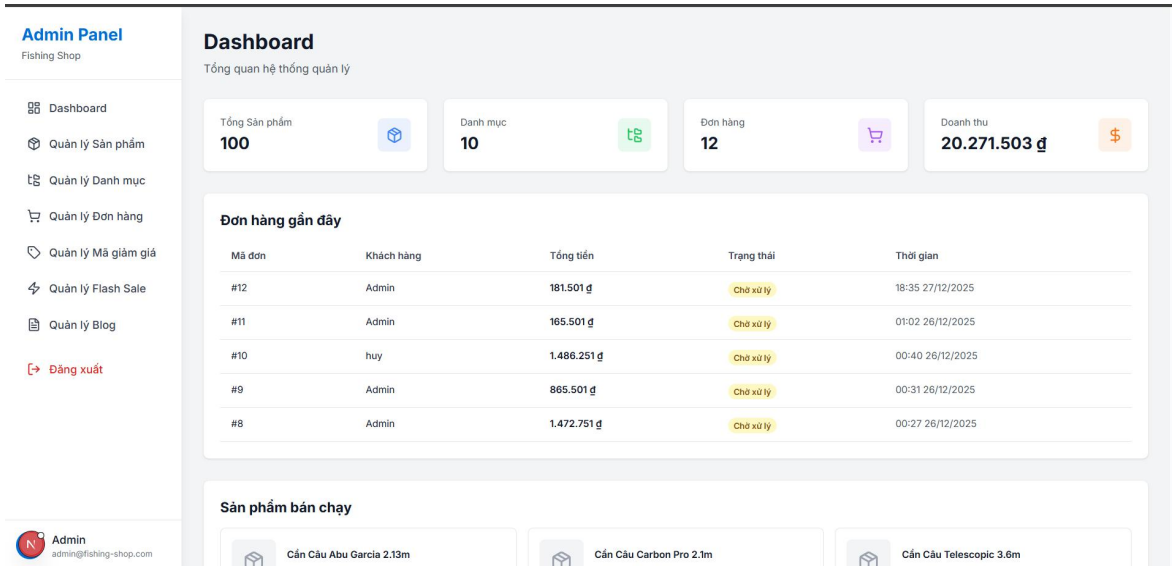
4.1.6 Giao diện trang giỏ hàng



Hình 4.6 Giao diện trang giỏ hàng

Chức năng: Hiển thị các sản phẩm đã chọn, cho phép cập nhật số lượng, xóa sản phẩm và xem tổng giá trị đơn hàng trước khi thanh toán.

4.1.7 Giao diện trang admin



Hình 4.7 Giao diện trang admin

Chức năng: Cung cấp công cụ quản lý sản phẩm, danh mục, đơn hàng, người dùng và nội dung website, giúp quản trị viên vận hành hệ thống hiệu quả.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết quả đạt được

Đề tài xây dựng hệ thống thương mại điện tử đã được hoàn thành với nhiều kết quả khả quan. Hệ thống được thiết kế và triển khai với đầy đủ các chức năng cơ bản phục vụ hoạt động mua sắm trực tuyến, bao gồm:

Quản lý tài khoản người dùng, sản phẩm, giỏ hàng và đơn hàng.

Hỗ trợ người dùng tìm kiếm, lựa chọn sản phẩm, đặt hàng và theo dõi trạng thái đơn hàng một cách thuận tiện.

Cho phép người dùng đánh giá sản phẩm nhằm nâng cao trải nghiệm mua sắm.

Cung cấp chức năng quản trị giúp quản lý, cập nhật thông tin sản phẩm, kiểm soát đơn hàng và xử lý yêu cầu của khách hàng.

Bên cạnh đó, hệ thống đảm bảo các tiêu chí về giao diện trực quan, dễ sử dụng, bảo mật thông tin người dùng và mang lại trải nghiệm mua sắm ổn định, liên tục. Đề tài cũng đã áp dụng một số giải pháp kỹ thuật như xác thực an toàn, tối ưu hóa xử lý dữ liệu và thiết kế cơ sở dữ liệu hợp lý nhằm nâng cao hiệu suất hoạt động của hệ thống.

Những kết quả đạt được cho thấy hệ thống đã xây dựng thành công một quy trình mua sắm khép kín, hỗ trợ hiệu quả cho cả người dùng và quản trị viên. Đóng góp của đề tài nằm ở việc tổng hợp và áp dụng các mô hình xử lý hiện đại vào xây dựng một website thương mại điện tử phù hợp với nhu cầu thực tế của thị trường nội địa.

5.2 Hướng phát triển

Trong thời gian tới, hệ thống có thể được nghiên cứu và mở rộng thêm các chức năng nâng cao như:

Tích hợp các phương thức thanh toán điện tử đa dạng (thẻ ngân hàng, ví điện tử, QR code,...).

Ứng dụng công nghệ trí tuệ nhân tạo và máy học để đề xuất sản phẩm phù hợp với từng người dùng, cá nhân hóa trải nghiệm mua sắm.

Phát triển phiên bản ứng dụng di động để mở rộng đối tượng người dùng, đồng thời nâng cao tính tiện lợi và tiếp cận thị trường rộng rãi hơn.

Hỗ trợ đa ngôn ngữ nhằm đáp ứng nhu cầu của khách hàng quốc tế.

Xây dựng hệ thống quản lý khuyến mãi, tích điểm và chăm sóc khách hàng tự động nhằm gia tăng giá trị sử dụng cho hệ thống.

Nghiên cứu các giải pháp nâng cao bảo mật, chống gian lận giao dịch và đảm bảo an toàn dữ liệu người dùng.

Tối ưu hóa hiệu suất hệ thống khi có số lượng lớn người dùng truy cập đồng thời.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]"ReactJS," [Online]. Available: https://react.dev/. [Accessed 2025].

[2]"Next.js Documentation," [Online]. Available: https://nextjs.org/docs. [Accessed 2025].

[3]"Express.js – Node.js web application framework," [Online]. Available: https://expressjs.com/. [Accessed 2025].

[4]"MySQL Documentation," [Online]. Available: https://dev.mysql.com/doc/. [Accessed 2025].

[5]"GHN API Documentation," [Online]. Available: https://api.ghn.vn/home/docs/. [Accessed 2025].

[6]"KungfuTech," [Online]. Available: https://kungfutech.edu.vn/khoa-hoc/reactjs. [Accessed 2025].

[7]"Geeks For Geeks" [Online]. Available: https://www.geeksforgeeks.org/system-design/complete-guide-to-clean-architecture. [Accessed 2025]

[8]"NhanHoa" [Online]. Available: https://nhanhoa.com/tin-tuc/mysql-la-gi.html. [Accessed 2025]

[9] "Stringee" [Online]. Available: https://stringee.com/vi/blog/post/restful-api-la-gi. [Accesed 2025]

PHỤ LỤC

Dữ liệu thử nghiệm

Bảng 4.1 Bảng Product

product_id	***name***	***price***	***sku***	***stock_quantity***	***thumbnail_url***
1	Cần câu Carbon	1200000	CAN-001	50	can1.jpg
2	Máy câu Shimano	2500000	MAY-001	30	may1.jpg
3	Dây câu Nhật	150000	DAY-001	100	day1.jpg
4	Mồi câu tổng hợp	80000	MOI-001	200	moi1.jpg
5	Hộp đựng mồi	120000	PK-001	70	pk1.jpg

Bảng 4.2 Bảng product_images

product_id	***image_url***	***alt_text***	***is_primary***
1	can1.jpg	Cần câu Carbon	1
2	may1.jpg	Máy câu Shimano	1
3	day1.jpg	Dây câu Nhật	1
4	moi1.jpg	Mồi câu tổng hợp	1
5	pk1.jpg	Hộp đựng mồi	1

Bảng 4.3 Bảng comment

comment_id	***user_id***	***rating***	***comment***
1	2	5	Rất tốt
2	3	4	Chạy mượt
3	2	4	Bền
4	4	3	Tạm ổn
5	1	5	Đáng tiền

Bảng 4.4 Bảng flash sale

product_id	***discount_percentage***	***start_time***	***end_time***	***status***
1	20%	NOW()	NOW() + 1	active

product_id	*discount_percentage*	*start_time*	*end_time*	*status*
			ngày	
2	15%	NOW()	NOW() + 1 ngày	active
3	10%	NOW()	NOW() + 1 ngày	inactive
4	25%	NOW()	NOW() + 1 ngày	expired
5	30%	NOW()	NOW() + 2 ngày	active

Bảng 4.5 Bảng blog

title	*slug*	*excerpt*	*content*	*author_id*	*category*	*tags*	*is_published*	*published_at*
Cách chọn cần câu cá	cach-chon-can-cau	Hướng dẫn chọn cần câu	Nội dung chi tiết...	1	Kinh nghiệm	["cau-ca","can-cau"]	1	NOW()
Mồi câu hiệu quả	moi-cau-hieu-qua	Các loại mồi phổ biến	Nội dung...	1	Kinh nghiệm	["moi-cau"]	1	NOW()
Câu cá mùa mưa	cau-ca-mua-mua	Lưu ý khi câu mùa mưa	Nội dung...	2	Chia sẻ	["mua-mua"]	1	NOW()
Review máy câu	review-may-cau	Đánh giá máy câu	Nội dung...	1	Review	["may-cau"]	0	NULL
Bảo quản dụng cụ câu	bao-quan-dung-cu	Cách bảo quản đồ câu	Nội dung...	2	Hướng dẫn	["bao-quan"]	1	NOW()

Bảng 4.6 Bảng oder

user_id	*address_id*	*status*	*payment_method*	*total_amount (VNĐ)*	*shipping_fee (VNĐ)*
2	1	paid	cod	1,500,000	30,000
3	3	pending	bank_transfer	800,000	25,000

user_id	***address_id***	***status***	***payment_method***	***total_amount (VNĐ)***	***shipping_fee (VNĐ)***
4	4	shipped	e_wallet	1,200,000	30,000
2	2	completed	cod	2,200,000	40,000
1	1	cancelled	cod	500,000	0