

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



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024 - 2025

XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ BÁN ĐỒ CÔNG NGHỆ BẰNG REACTJS VÀ NODE.JS

Giáo viên hướng dẫn:
ThS. Nguyễn Ngọc Đan Thanh

Sinh viên thực hiện:
Họ tên: Trần Trung Nghĩa
MSSV: 110121066
Lớp: DA21TTB

Trà Vinh, tháng 01 năm 2025

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



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024 - 2025

XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ BÁN ĐỒ CÔNG NGHỆ BẰNG REACTJS VÀ NODE.JS

Giáo viên hướng dẫn:
ThS. Nguyễn Ngọc Đan Thanh

Sinh viên thực hiện:
Họ tên: Trần Trung Nghĩa
MSSV: 110121066
Lớp: DA21TTB

Trà Vinh, tháng 01 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Trà Vinh, ngày tháng năm 2025

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến cô Nguyễn Ngọc Đan Thanh - giảng viên Bộ môn Công nghệ thông tin - Khoa Kỹ thuật và Công nghệ trường Đại học Trà Vinh đã giúp đỡ và hướng dẫn cho em rất nhiều trong quá trình tìm hiểu và thực hiện đồ án chuyên ngành.

Tuy nhiên trong quá trình thực hiện đồ án chuyên ngành, do chưa có nhiều kinh nghiệm và kiến thức còn hạn chế trong quá trình tìm hiểu cũng như làm đồ án nên em vẫn còn gặp một số hạn chế và sự thiếu sót. Rất mong nhận được sự quan tâm, nhận xét và những lời góp ý của cô Nguyễn Ngọc Đan Thanh cùng các thầy cô giảng viên bộ môn để cho đề tài của em được hoàn thiện chính chu hơn.

Em xin chân thành cảm ơn.

Sinh viên thực hiện

Trần Trung Nghĩa

MỤC LỤC

CHƯƠNG 1	TỔNG QUAN	12
1.1	Giới thiệu tổng quan về chủ đề	12
1.2	Tổng quan về các hệ thống tương tự.....	12
1.3	Phân tích các công nghệ được sử dụng.....	12
1.4	Hạn chế của các hệ thống hiện có và đề xuất giải pháp.....	13
CHƯƠNG 2	NGHIÊN CỨU LÝ THUYẾT.....	14
2.1	ReactJS.....	14
2.1.1	Lợi ích của việc sử dụng ReactJS	14
2.1.2	Bắt đầu cài đặt với ReactJS.....	15
2.1.3	Giao diện trong ReactJS.....	16
2.1.4	Javascript XML	16
2.2	Node.js	28
2.2.1	Lợi ích của việc sử dụng Node.js	28
2.2.2	Bắt đầu cài đặt với Node.js	29
2.3	Kết luận	33
CHƯƠNG 3	HIỆN THỰC HÓA NGHIÊN CỨU.....	34
3.1	Mô tả bài toán	34
3.2	Kiến trúc hệ thống.....	35
3.3	Phân tích đặc tả yêu cầu hệ thống.....	36
3.3.1	Yêu cầu chức năng	36
3.3.2	Yêu cầu phi chức năng	37
3.4	Thiết kế hệ thống	37
3.4.1	Sơ đồ usecase	37
3.4.2	Sơ đồ lớp	38
3.4.3	Sơ đồ hoạt động.....	42
3.4.4	Sơ đồ tuần tự	44
3.4.5	Mô hình triển khai	47
3.5	Thiết kế giao diện.....	48
3.5.1	Sơ đồ website	48
3.5.2	Màn hình trang người dùng.....	49
3.5.3	Màn hình trang quản trị.....	52
3.5.4	Màn hình chức năng đăng kí.....	52
3.5.5	Màn hình chức năng đăng nhập	53

3.6	Kết chương.....	53
CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU.....		54
4.1	Dữ liệu thử nghiệm	54
4.2	Kết quả thử nghiệm.....	56
4.2.1	Giao diện trang chủ	56
4.2.2	Giao diện trang sản phẩm.....	56
4.2.3	Giao diện trang sản phẩm chi tiết.....	57
4.2.4	Giao diện trang giỏ hàng	57
4.3	Kết luận	58
CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN		59
5.1	Kết luận	59
5.2	Hướng phát triển	59
DANH MỤC TÀI LIỆU THAM KHẢO		60
PHỤ LỤC		61

DANH MỤC HÌNH ẢNH

Hình 2.1 Kiểm tra phiên bản	15
Hình 2.2 Ví dụ về component.....	20
Hình 2.3 Tải Node.js	29
Hình 2.4 Cài đặt Node.js	30
Hình 2.5 Kiểm tra phiên bản Node.js	30
Hình 2.6 Chạy server Node.js.....	31
Hình 3.1 Kiến trúc hệ thống	35
Hình 3.2 Sơ đồ Usecase.....	38
Hình 3.3 Sơ đồ lớp.....	41
Hình 3.4 Sơ đồ hoạt động cho đăng ký	42
Hình 3.5 Sơ đồ hoạt động cho đăng nhập.....	42
Hình 3.6 Sơ đồ hoạt động cho người dùng.....	42
Hình 3.7 Sơ đồ hoạt động cho người quản trị	43
Hình 3.8 Sơ đồ tuần tự thao tác đăng ký	44
Hình 3.9 Sơ đồ tuần tự thao tác đăng nhập	44
Hình 3.10 Sơ đồ tuần tự thao tác của người dùng	45
Hình 3.11 Sơ đồ tuần tự thao tác của người quản trị.....	46
Hình 3.12 Mô hình triển khai	47
Hình 3.13 Sơ đồ website.....	48
Hình 3.14 Màn hình trang chủ chưa đăng nhập	49
Hình 3.15 Màn hình trang chủ đã đăng nhập	49
Hình 3.16 Màn hình trang sản phẩm	50
Hình 3.17 Màn hình trang chi tiết sản phẩm	51
Hình 3.18 Màn hình trang giỏ hàng.....	51
Hình 3.19 Màn hình trang quản trị viên	52
Hình 3.20 Chức năng đăng kí.....	52
Hình 3.21 Chức năng đăng nhập	53
Hình 4.1 Giao diện trang chủ	56
Hình 4.2 Giao diện trang sản phẩm	56
Hình 4.3 Giao diện trang sản phẩm chi tiết.....	57
Hình 4.4 Giao diện trang giỏ hàng	57

DANH MỤC BẢNG BIỂU

Bảng 3.1 Bản thương hiệu	38
Bảng 3.2 Bảng danh mục.....	38
Bảng 3.3 Bảng sản phẩm	39
Bảng 3.4 Bảng biến thể sản phẩm	39
Bảng 3.7 Bảng đặt hàng.....	39
Bảng 3.5 Bảng giỏ hàng	40
Bảng 3.6 Bảng người dùng.....	40
Bảng 3.8 Bảng nhà cung cấp	40
Bảng 4.1 Bảng danh mục.....	54
Bảng 4.2 Bảng thương hiệu.....	54
Bảng 4.3 Bảng nhà cung cấp	54
Bảng 4.4 Bảng người dùng.....	55
Bảng 4.5 Bảng sản phẩm	55

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Vấn đề nghiên cứu

Đồ án này tập trung vào việc nghiên cứu và xây dựng một website thương mại điện tử bán đồ công nghệ sử dụng ReactJS và Node.js. Vấn đề trọng tâm là làm thế nào để tích hợp các công nghệ hiện đại nhằm cung cấp trải nghiệm người dùng tốt, bảo đảm hiệu suất cao và khả năng mở rộng cho hệ thống.

Hướng tiếp cận

Frontend: Xây dựng giao diện người dùng bằng ReactJS với kiến trúc dựa trên component, sử dụng React Router để điều hướng và quản lý trạng thái hiệu quả bằng useContext. Giao diện được thiết kế thân thiện với người dùng, áp dụng Tailwind CSS để tăng tính thẩm mỹ và hỗ trợ responsive design cho mọi thiết bị.

Backend: Phát triển backend bằng Node.js và Express, xây dựng các RESTful API để giao tiếp với cơ sở dữ liệu MySQL, đồng thời cung cấp tính năng quản lý sản phẩm, người dùng và giỏ hàng thông qua hệ thống quản trị (Admin).

Cách giải quyết vấn đề

Để giải quyết vấn đề, tôi đã tiến hành phân tích yêu cầu nhằm xác định các tính năng cần thiết cho một website thương mại điện tử, bao gồm hiển thị sản phẩm, quản lý giỏ hàng và thanh toán. Dựa vào đó, tôi thiết kế hệ thống với các thành phần chính: frontend, backend, cơ sở dữ liệu và hệ thống quản trị. Sau cùng, tôi triển khai website với các chức năng như danh sách và chi tiết sản phẩm, thêm vào giỏ hàng và thanh toán.

Kết quả đạt được

Dự án đã hoàn thiện với các tính năng cần thiết cho một website thương mại điện tử bao gồm việc xây dựng giao diện người dùng và tối ưu trải nghiệm với ReactJS và Tailwind CSS. Backend đã được phát triển với RESTful API đảm bảo giao tiếp an toàn giữa frontend và backend. Ngoài ra, hệ thống quản trị cung cấp khả năng thêm, xóa, sửa thông tin sản phẩm trực tiếp từ backend.

MỞ ĐẦU

Lý do chọn đề tài

Trong bối cảnh hiện đại, khi công nghệ thông tin ngày càng phát triển và ảnh hưởng sâu rộng đến mọi lĩnh vực, mua sắm trực tuyến trở thành một xu hướng tất yếu. Ngành thương mại điện tử đã phát triển mạnh mẽ, đáp ứng nhu cầu ngày càng cao của người tiêu dùng trong việc tìm kiếm các sản phẩm nhanh chóng, tiện lợi và an toàn. Đặc biệt, đồ công nghệ là một trong những mặt hàng phổ biến và được ưa chuộng, với nhu cầu mua sắm trực tuyến ngày càng tăng.

Với mong muốn tìm hiểu sâu hơn về quy trình phát triển một website thương mại điện tử, cùng với tiềm năng ứng dụng các công nghệ hiện đại, đề tài "Xây dựng website thương mại điện tử bán đồ công nghệ sử dụng ReactJS và Node.js" được chọn nhằm cung cấp một giải pháp hiệu quả, nâng cao trải nghiệm người dùng và đảm bảo khả năng mở rộng cho hệ thống. Qua đề tài này, tôi có cơ hội thực hành và áp dụng các kiến thức về lập trình frontend, backend và quản lý cơ sở dữ liệu trong một dự án thực tiễn.

Mục đích nghiên cứu

Mục đích của đồ án là xây dựng một website thương mại điện tử hoàn chỉnh, cung cấp các tính năng cơ bản như quản lý sản phẩm, quản lý giỏ hàng, xử lý thanh toán và quản trị người dùng. Bằng cách tích hợp các công nghệ hiện đại như ReactJS cho frontend và Node.js cho backend, đề tài mong muốn mang đến một trải nghiệm người dùng mượt mà, tối ưu hóa hiệu suất và đảm bảo bảo mật thông tin.

Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là các công nghệ chính được sử dụng để xây dựng một hệ thống thương mại điện tử, bao gồm ReactJS, Node.js, Express, MySQL và Tailwind CSS. Đề tài tập trung vào cách các công nghệ này có thể kết hợp với nhau để tạo nên một hệ thống hoạt động hiệu quả, đáp ứng yêu cầu của người dùng và dễ dàng bảo trì.

Phạm vi nghiên cứu

Phạm vi nghiên cứu của đồ án giới hạn trong việc xây dựng một website thương mại điện tử đơn giản dành cho các sản phẩm công nghệ. Đề tài bao gồm các nội dung như phân tích yêu cầu, thiết kế và triển khai hệ thống với các chức năng cơ bản như hiển thị sản phẩm, quản lý giỏ hàng, thanh toán và quản trị thông tin sản phẩm.

CHƯƠNG 1 TỔNG QUAN

1.1 Giới thiệu tổng quan về chủ đề

Trong bối cảnh công nghệ thông tin không ngừng phát triển, mua sắm trực tuyến đã trở thành xu hướng phổ biến trên toàn thế giới. Các website thương mại điện tử hiện nay không chỉ đáp ứng yêu cầu về chức năng mà còn cần cung cấp trải nghiệm người dùng mượt mà, dễ dàng và an toàn. Đặc biệt, với sự phát triển nhanh chóng của ngành hàng công nghệ, việc xây dựng một trang web thương mại điện tử chuyên bán đồ công nghệ là điều cần thiết để đáp ứng nhu cầu của người tiêu dùng về tính tiện dụng, khả năng tương tác cao và trải nghiệm mua sắm an toàn.

Đồ án này tập trung vào nghiên cứu và phát triển một website thương mại điện tử bán đồ công nghệ sử dụng các công nghệ tiên tiến như ReactJS và Node.js, nhằm tạo ra một ứng dụng web có khả năng mở rộng, dễ bảo trì và cung cấp trải nghiệm người dùng vượt trội.

1.2 Tổng quan về các hệ thống tương tự

Hiện nay, trên thị trường đã xuất hiện nhiều hệ thống thương mại điện tử nổi bật như Amazon, Lazada và Shopee. Các nền tảng này có điểm mạnh là khả năng quản lý kho hàng, xử lý đơn hàng và giao diện người dùng phù hợp với đại đa số người dùng. Tuy nhiên, nhiều hệ thống vẫn có những hạn chế về khả năng cá nhân hóa trải nghiệm người dùng và thiếu đi sự mượt mà trong quá trình sử dụng.

Từ những hạn chế trên, đề tài của em sẽ tập trung vào việc cải tiến các điểm sau:

Khả năng tương tác: Sử dụng ReactJS để xây dựng giao diện người dùng giúp cho trang web phản hồi nhanh hơn và dễ sử dụng.

Quản lý trạng thái tối ưu: Với useContext, các trạng thái của ứng dụng sẽ được quản lý tập trung, giúp giảm thiểu rủi ro sai lệch trong dữ liệu.

Bảo mật dữ liệu: Node.js kết hợp với các phương pháp mã hóa hiện đại sẽ giúp bảo vệ thông tin người dùng an toàn hơn.

1.3 Phân tích các công nghệ được sử dụng

Để xây dựng một website thương mại điện tử toàn diện, các công nghệ dưới đây sẽ được áp dụng:

ReactJS: Đây là một thư viện JavaScript phổ biến với cấu trúc dựa trên component, cho phép tái sử dụng mã nguồn và tạo giao diện người dùng phản hồi nhanh chóng. ReactJS hỗ trợ các thư viện bổ trợ để quản lý trạng thái và React Router để điều hướng trang một cách linh hoạt. [1]

Node.js: Với đặc tính non-blocking I/O, Node.js là nền tảng lý tưởng để xây dựng RESTful API có khả năng xử lý lượng lớn yêu cầu từ phía người dùng một cách hiệu quả. Sự kết hợp với Express giúp tối ưu hóa các tác vụ backend và dễ dàng tích hợp với các cơ sở dữ liệu như MySQL. [2]

MySQL: Được sử dụng để lưu trữ và quản lý dữ liệu, MySQL cung cấp khả năng xử lý truy vấn nhanh chóng và hỗ trợ quản lý dữ liệu quan hệ, phù hợp cho nhu cầu lưu trữ thông tin sản phẩm, đơn hàng và người dùng. [3]

Các công nghệ này được chọn lựa không chỉ vì khả năng đáp ứng tốt yêu cầu về chức năng mà còn nhờ vào tính phổ biến và sự hỗ trợ cộng đồng mạnh mẽ, giúp dễ dàng bảo trì và mở rộng trong tương lai.

1.4 Hạn chế của các hệ thống hiện có và đề xuất giải pháp

Các hệ thống thương mại điện tử hiện tại, dù có nhiều tính năng mạnh mẽ, vẫn tồn tại một số hạn chế đáng chú ý như:

Thiếu khả năng tùy chỉnh giao diện theo nhu cầu người dùng.

Khó khăn trong việc mở rộng hoặc điều chỉnh chức năng khi nhu cầu thay đổi.

Cơ chế bảo mật dữ liệu chưa đủ mạnh để chống lại các mối đe dọa ngày càng phức tạp của các hacker.

Đề án này sẽ giải quyết những hạn chế trên bằng cách:

Tạo ra một hệ thống linh hoạt, dễ tùy chỉnh và mở rộng thông qua cấu trúc component của ReactJS.

Cải thiện tính bảo mật thông tin người dùng bằng các phương pháp mã hóa hiện đại trong Node.js.

Tối ưu hóa trải nghiệm người dùng với tốc độ phản hồi nhanh, giao diện đơn giản và dễ sử dụng.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

2.1 ReactJS

ReactJS được phát triển bởi Facebook vào năm 2011. Mục đích ban đầu nhằm cải thiện tốc độ và hiệu suất giao diện người dùng trên trang web Facebook. Đến năm 2013, Facebook công bố mã nguồn mở ReactJS, giúp nó nhanh chóng trở thành một trong những thư viện phổ biến nhất cho phát triển ứng dụng web động. Năm 2015, Facebook tiếp tục giới thiệu React Native, một framework phát triển ứng dụng di động dựa trên ReactJS, cho phép xây dựng ứng dụng cho cả iOS và Android từ cùng một mã nguồn. Ngày nay, ReactJS đã trở thành thư viện phát triển web hàng đầu, được sử dụng rộng rãi bởi các công ty trên toàn thế giới, với sự đầu tư liên tục từ Facebook để đáp ứng nhu cầu của cộng đồng lập trình viên.

ReactJS là một thư viện JavaScript phát triển bởi Facebook, được sử dụng rộng rãi để xây dựng giao diện người dùng động và tương tác trong các ứng dụng web. Đây là một công cụ mạnh mẽ cho việc phát triển ứng dụng web động với hiệu suất cao và dễ bảo trì. [1]

2.1.1 Lợi ích của việc sử dụng ReactJS

Hiệu suất cao: ReactJS sử dụng một cơ chế gọi là "Virtual DOM" để cải thiện hiệu suất. Thay vì cập nhật toàn bộ giao diện người dùng mỗi khi có thay đổi, React sẽ cập nhật chỉ những phần tử có thay đổi. Điều này giúp giảm tải cho trình duyệt và tăng tốc độ của ứng dụng.

Thư viện phong phú: ReactJS đi kèm với một cộng đồng lớn và nhiều thư viện mở rộng hỗ trợ. Bạn có thể dễ dàng tích hợp React với các thư viện và công cụ khác để phát triển các tính năng phức tạp.

Quản lý trạng thái tốt: React sử dụng mô hình quản lý trạng thái (state) rất tốt. Trạng thái của ứng dụng được quản lý một cách có hệ thống, dễ dàng theo dõi và bảo trì.

Cộng đồng lớn và hỗ trợ: ReactJS có một cộng đồng lớn và nhiều tài liệu học tập và hỗ trợ. Bạn có thể tìm kiếm giúp đỡ từ các nhà phát triển khác và chia sẻ kiến thức của mình trong cộng đồng này. [4]

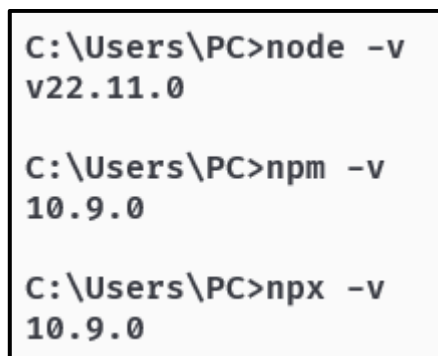
2.1.2 Bắt đầu cài đặt với ReactJS

2.1.2.1 Cài đặt Node.js và npm

Để có thể cài đặt được ứng dụng ReactJS trên máy tính, máy tính sẽ cần phải được cài đặt Node.js, npm và npx sẽ được cài đặt cùng với Node.js.

Truy cập trang web chính thức của Node.js: Điều này giúp bạn tải xuống phiên bản mới nhất của Node.js. Trang web chính thức là <https://nodejs.org/>.

Để kiểm tra NodeJS và npx đã được cài đặt ở trên máy tính hay chưa, chúng ta mở Command Prompt và sử dụng câu lệnh **node -v**, **npm -v** và **npx -v**. Máy tính sẽ hiển thị phiên bản được cài đặt. Nếu nhìn thấy có lỗi xảy ra, không hiển thị phiên bản, hãy thử cài đặt lại. [2]



```
C:\Users\PC>node -v
v22.11.0

C:\Users\PC>npm -v
10.9.0

C:\Users\PC>npx -v
10.9.0
```

Hình 2.1 Kiểm tra phiên bản

2.1.2.2 Tạo một ứng dụng React mới

Các bước để khởi tạo một ứng dụng React có tên là my-app:

B1: Tạo một thư mục trên máy tính.

B2: Mở cửa sổ dòng lệnh bên trong thư mục đó.

B3: Chạy câu lệnh **npx create-react-app my-app**. Sau đó đợi để ứng dụng được khởi tạo.

B4: Sau khi đã khởi tạo thành công, di chuyển vào bên trong thư mục “hello-world” vừa được tạo bằng câu lệnh **cd my-app**.

B5: Khởi chạy ứng dụng bằng câu lệnh **npm start**.

Trình duyệt sẽ tự động mở ra và ứng dụng đã được khởi tạo thành công. Để tắt ứng dụng đó trên cửa sổ dòng lệnh sử dụng tổ hợp phím Ctrl + C.

Sử dụng lệnh **create-react-app** để tạo một dự án mới cho React. [1]

```
npx create-react-app my-app
cd my-app
npm start
```

2.1.2.3 Sử dụng Tailwind CSS với React

Sau khi đã tạo xong ứng dụng React chúng ta có thể sử dụng Tailwind CSS để làm cho ứng dụng đẹp mắt hơn.

Đầu tiên di chuyển đến thư mục my-app bằng lệnh **cd my-app**, sau khi đã ở trong thư mục my-app chúng ta cài đặt TailwindCSS thông qua npm, sau đó chạy lệnh init để tạo tệp tailwind.config.js

```
npm install -D tailwindcss
npx tailwindcss init
```

Thêm đường dẫn tới tất cả các tệp mẫu trong tệp tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*..{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Thêm **@tailwind** chỉ thị cho từng lớp của Tailwind CSS vào tệp ./src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Giờ đây bạn đã có thể code ReactJS với Tailwind CSS và chạy chương trình bằng lệnh **npm start**. [5]

2.1.3 Giao diện trong ReactJS

2.1.4 Javascript XML

Javascript XML (JSX) là cú pháp do đội ngũ React phát triển, sử dụng chủ yếu trong React. [6]

Mục đích của JSX được tạo ra để có thể tạo ra các element một cách tường minh và đơn giản. JSX cho phép developer có thể tạo ra các đoạn HTML nhanh chóng, kèm với khả năng có thể chèn các giá trị JS vào bên trong để tạo ra trang web có nội dung động.

```
import React from 'react';

function Greeting() {
  const name = 'Nghia';
  return (
    <div>
      <h1>Xin chào, {name}!</h1>
    </div>
  );
}

export default Greeting;
```

ReactJS có thể hoạt động mà không cần tới cú pháp JSX.

JSX không phải là HTML

JSX mặc dù có cú pháp trông rất giống HTML, tuy nhiên nó có một vài đặc điểm khác biệt so với HTML thông thường

Single parent root: Các component React cần phải return một thẻ bao ngoài duy nhất hoặc một array, không thể trả về nhiều hơn hai thẻ.

className thay vì class. Đây là một lý do kỹ thuật. class là một từ khoá trong JS. Vì vậy, đội ngũ React đã sử dụng className thay vì class để tránh các lỗi.

style nhận giá trị là một object, thay vì là cú pháp CSS thông thường

Các thuộc tính HTML sẽ được đổi tên theo kiểu camelCase

Đối với các đoạn JSX nằm trên nhiều dòng, JSX cần phải được bọc bên trong cặp ngoặc tròn ()

Component do chúng ta viết buộc phải được sử dụng ở dưới dạng tên viết hoa.

Render giá trị JS với JSX

JSX cho phép chúng ta có thể output các giá trị Javascript trực tiếp vào bên trong. Cú pháp để làm điều này là sử dụng dấu ngoặc nhọn {}. Xét ví dụ sau:

```
const App = () => {
  const name = "Nghia"
  const age = 20;
  const img = "https://avatar.png";
  return (
    <div>
      Hello, my name is {name}. I'm {age} years old.
      <img src={img} />
    </div>
  );
};
```

JSX với styling:

Có nhiều cách để có thể style với JSX trong React, tương tự với HTML. Cơ bản sẽ có hai kiểu là External CSS và Inline CSS.

Với External CSS, cách viết CSS không có gì khác biệt. Điểm khác biệt ở đây là chúng ta cần dùng className thay cho class thông thường. Chúng ta có thể import file css vào bên trong component với cú pháp import "<file>.css". Xét ví dụ sau:

```
.App {
  text-align: center;
  font-weight: bold;
}
```

```
import "../App.css";

const App = () => {
  return <div className="App">Hello, world!</div>;
};
```

Đối với Inline CSS, điểm khác biệt tương đối lớn so với HTML thông thường là:

style trong JSX nhận giá trị là một object (key-value)

Các key CSS phải được viết dưới dạng camelCase

Các value CSS cần phải được viết dưới dạng string hoặc number

Xét ví dụ sau:

```
const App = () => {  
  return (  
    <div style={{ background: "yellow", fontSize: 18 }}>Hello, World!</div>  
  );  
};
```

Đối với Tailwind CSS sử dụng className: Thay vì khai báo kiểu CSS riêng, chỉ cần sử dụng className với các lớp tiện ích của Tailwind trong JSX. Đây là ví dụ:

```
const App = () => {  
  return (  
    <div className="bg-yellow-300 text-center font-bold text-xl p-4">  
      Hello, Tailwind CSS!  
    </div>  
  );  
};
```

Quy tắc của JSX

JSX chỉ trả về một phần tử gốc duy nhất. Để trả về nhiều phần tử từ một thành phần, bạn cần bọc chúng trong một thẻ cha duy nhất. Ví dụ, bạn có thể sử dụng một cặp thẻ `<div></div>` hoặc `<></>`:

```
<div>  
  <h1>Hello Nghia</h1>  
    
  ...  
</div>
```

Hoặc

```
<>  
  <h1>Hello Nghia</h1>  
    
  ...  
</>
```

Thẻ rỗng này được gọi là Fragment. Fragment cho phép bạn nhóm các phần tử mà không để lại bất kỳ dấu vết nào trong cây HTML của trình duyệt.

Đóng tất cả các thẻ

JSX yêu cầu bạn phải đóng tất cả các thẻ rõ ràng: các thẻ đơn như là `` và các thẻ đôi như là `<p>text</p>`. Ví dụ:

```
<>

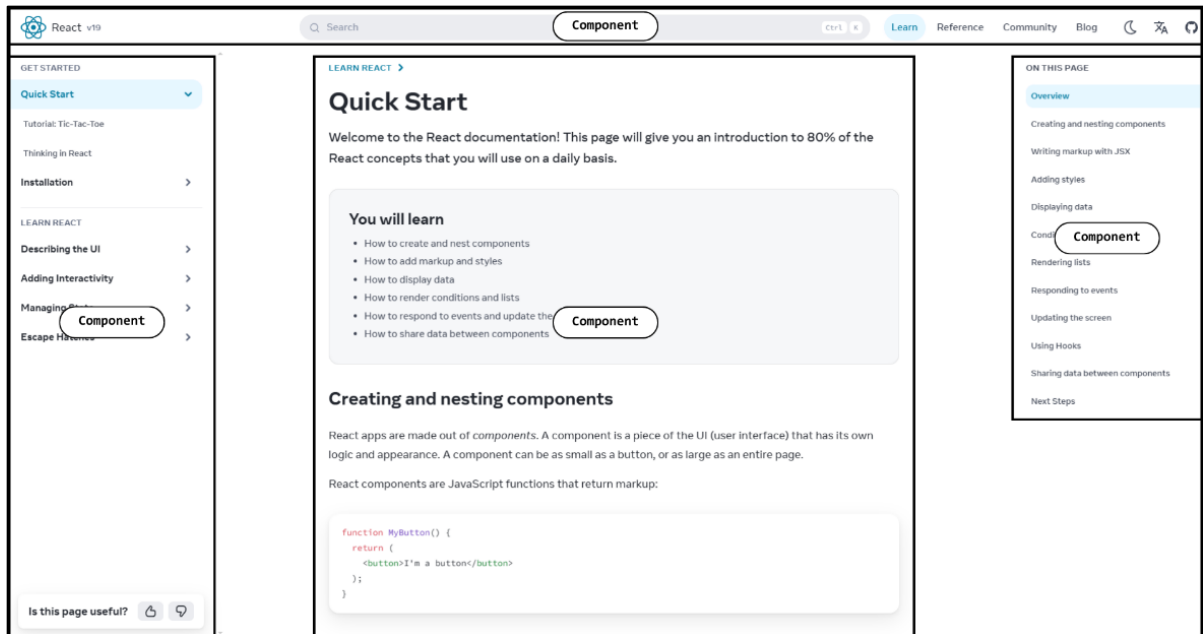
  

  <h1>Name</h1>
  <p>Infomation</p>

</>
```

2.1.4.1 Component trong ReactJS là gì?

Ví dụ dưới đây, giao diện người dùng đã được chia nhỏ thành các phần riêng lẻ. [6]



Hình 2.2 Ví dụ về component

Phần Search, thanh điều hướng, thanh bên và nội dung,... đều được coi là một thành phần riêng lẻ. Chúng ta có thể thực hiện hợp nhất tất cả các thành phần này để tạo thành một giao diện chính, là giao diện người dùng cuối cùng cho trang chủ.

Khi dự án phát triển, bạn sẽ nhận thấy rằng nhiều phần thiết kế có thể tái sử dụng component bạn đã viết, giúp tăng tốc quá trình phát triển.

Component trong React là gì?

Components là những thành phần giao diện (UI) được định nghĩa độc lập, có thể tái sử dụng và hoàn toàn tách biệt nhau.

Chúng ta có thể hiểu component là một hàm trong javascript. Chúng nhận bất kỳ đầu vào nào (hay còn gọi là “props”) và trả về các React elements thể hiện những gì được hiển thị trên trình duyệt. Vì vậy, việc sử dụng và chia nhỏ component hiệu quả sẽ giúp các lập trình viên trở nên chuyên nghiệp và giúp xây dựng một ứng dụng tốt hơn.

Các bước tạo component trong React

Dưới đây là các bước giúp tạo một component trong React.

Bước 1: Xuất component

Tiền tố **export default** là một cú pháp JavaScript tiêu chuẩn (không riêng cho React). Nó cho phép bạn đánh dấu hàm chính trong một tệp để sau đó bạn có thể nhập nó từ các tệp khác.

Bước 2: Định nghĩa hàm

Với **function Profile() { }**, bạn định nghĩa một hàm JavaScript với tên Profile.

Lưu ý: Các component React là các hàm JavaScript thông thường, nhưng tên của chúng phải bắt đầu bằng một chữ cái viết hoa hoặc chúng sẽ không hoạt động!

Bước 3: Thêm mã Markup

Component trả về một thẻ `` với các thuộc tính `src` và `alt`. `` được viết giống HTML, nhưng bản chất là JavaScript bên trong! Cú pháp này được gọi là JSX, và nó cho phép bạn nhúng mã markup vào bên trong JavaScript.

Câu lệnh **return** có thể được viết trên cùng một dòng như trong component này:

```
return ;
```

Nhưng nếu mã markup của bạn không nằm trên cùng một dòng như từ khóa `return`, bạn phải bọc nó trong một cặp dấu ngoặc đơn:

```
return (  
  <div>  
    <h1>Name</h1>  
    ;  
  </div>  
)
```

Cách sử dụng một component trong React

Sau khi định nghĩa component Profile của mình, bạn có thể lồng nó vào bên trong các component khác. Ví dụ, bạn có thể xuất một component StudentList sử dụng nhiều component Profile:

```
function Profile() {
  return (
    <div>
      <h1>Name</h1>
      ;
    </div>
  );
}

export default function Gallery() {
  return (
    <section>
      <h1>Danh sach hoc sinh Truong Dai hoc Tra Vinh</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}
```

2.1.4.2 Props trong ReactJS

Props là tham số đầu vào của các component trong React. Props là một trong những khái niệm cực kỳ quan trọng của React. Nó được sử dụng để truyền thông tin giữa các component. Mọi component cha có thể truyền thông tin đến các component con của nó bằng cách cung cấp chúng props. [6]

```
const App = () => {
  const x = 1;
  const y = 2;
  return (
    <div>
      <Sum a={x} b={y} />
    </div>
  )
}

const Sum = (props) => {
  return <div>The value is: {props.a + props.b}</div>
}
```

Cách truyền props vào một component trong React

Bước 1: Truyền props từ component cha vào component con

Đầu tiên, hãy truyền một số props vào Avatar. Ví dụ, hãy truyền hai props: person (một đối tượng) và size (một con số):

```
export default function Profile() {  
  return (  
    <Avatar person={{ name: "Nghia", imageId: "001" }} size={100} />  
  );  
}
```

Bước 2: Đọc props bên trong component con

Bạn có thể đọc các props này bằng cách liệt kê tên của chúng. Ví dụ: person và size được ngăn cách bằng dấu phẩy trong ({ và }) ngay sau hàm Avatar. Điều này cho phép bạn sử dụng chúng bên trong mã của component Avatar, giống như bạn thao tác với biến.

```
function Avatar({ person, size }) {  
  // person và size có sẵn ở đây  
}
```

Thêm một số logic vào Avatar sử dụng các props **url**, **person** và **size** để render. Bây giờ bạn có thể cấu hình Avatar để render theo nhiều cách khác nhau với các props khác nhau. Hãy thử điều chỉnh các giá trị!

```
function Avatar({ url, person, size }) {  
  return (  
    <img className="avatar"  
      src={url}  
      alt={person.name}  
      width={size}  
      height={size}  
    />  
  );  
}
```

```
export default function Profile() {
  return (
    <div>
      <Avatar
        size={100}
        url={https://avatar-tran.png}
        person={{
          name: "Tran",
          imageId: "001",
        }}
      />
      <Avatar
        size={200}
        url={https://avatar-trung.png}
        person={{
          name: "Trung",
          imageId: "002",
        }}
      />
      <Avatar
        size={300}
        url={https://avatar-Nghia.png}
        person={{
          name: "Nghia",
          imageId: "003",
        }}
      />
    </div>
  );
}
```

Giá trị mặc định của props

Nếu bạn muốn đặt một giá trị mặc định cho props để sử dụng khi không có giá trị được chỉ định, bạn có thể làm điều này bằng cách đặt = và giá trị mặc định ngay sau tham số:

```
function Avatar({ person, size = 100 }) {
  // ...
}
```


Bây giờ, nếu `<Avatar person={...} />` được render mà không có props size, size sẽ được đặt thành 100.

Giá trị mặc định chỉ được sử dụng khi props size bị thiếu hoặc nếu bạn truyền `size={undefined}`. Tuy nhiên, nếu bạn truyền `size={null}` hoặc `size={0}`, giá trị mặc định sẽ không được sử dụng.

Children props trong React

Các thẻ HTML có thể chứa bên trong nó các thẻ HTML khác, ta có ví dụ như `div`, `p`, ... Tương tự như vậy, các thẻ “HTML” do chúng ta tự tạo cũng có thể làm được điều tương tự thông qua một giá trị props đặc biệt có tên là `children`. Xét ví dụ sau:

```
const Card = (props) => {
  return <div className="card">{props.children}</div>
}
```

```
const App = () => {
  return (
    <Card>
      <div>Inside a card</div>
    </Card>
  )
}
```

Cũng tương tự như các props thông thường khác, `children` có thể nhận giá trị là bất cứ kiểu dữ liệu nào. Với ví dụ ở trên, `children` nhận vào giá trị là một React Element.

`children` props giúp chúng ta có khả năng “compose” các component lại với nhau. Thay vì cố định giá trị bên trong `Card`, lúc này `Card` có thể cho bất cứ component nào nằm trong nó có thêm các thuộc tính ở trên.

2.1.4.3 Event trong ReactJS là gì?

Trong React, bạn có thể thêm các xử lý sự kiện (event handlers) vào JSX của bạn. Các event handlers là các hàm sẽ được kích hoạt khi có sự tương tác như nhấp chuột, di chuột qua, ,... và nhiều tương tác khác. [6]

Thêm xử lý sự kiện trong Reactjs

Bạn có thể làm cho một nút hiển thị một thông báo khi người dùng nhấp chuột bằng cách làm theo ba bước sau:

Định nghĩa một hàm gọi là **handleClick** trong thành phần Button.

Thực hiện logic trong hàm này (sử dụng alert để hiển thị thông báo).

Thêm **onClick={handleClick}** vào thẻ `<button>`.

Dưới đây là mã ví dụ:

```
export default function Button() {  
  function handleClick() {  
    alert("Bạn đã click chuột vào tôi!");  
  }  
  return <button onClick={handleClick}>Nhấp chuột vào tôi</button>;  
}
```

Trong ví dụ trên, chúng ta đã định nghĩa hàm handleClick và sau đó truyền nó như một prop cho `<button>` bằng cách sử dụng `onClick={handleClick}` - handleClick là một xử lý sự kiện. Thông thường, các hàm xử lý sự kiện được định nghĩa bên trong component của bạn và có tên bắt đầu bằng handle, tiếp theo là tên của sự kiện như `onClick`, `onMouseEnter`, `onChange`, `onSubmit`,...

2.1.4.4 State trong ReactJS

State về cơ bản là một giá trị biến đặc biệt trong React. Nó là giá trị mà khi thay đổi, React sẽ tiến hành việc tính toán lại kết quả của component và từ đó cập nhật lại giao diện. Để sử dụng được state, chúng ta cần import một function từ trong thư viện React là `useState`. `useState` và một số function khác trong thư viện được gọi là các “hooks”. [6]

Xét ví dụ sau:

```
import React from "react";  
const App = () => {  
  let count = 0;  
  
  const handleClick = () => {  
    count = count + 1;  
    console.log("count: ", count)  
  }  
}
```

```
return (  
  <div>  
    <span>{count}</span>  
    <button onClick={handleClick}>Increase</button>  
  </div>  
)  
}
```

Với ví dụ trên, khi ta click vào button, giá trị của biến count sẽ được thay đổi và in giá trị count ra màn hình. Chúng ta cũng sẽ mong đợi rằng component App sẽ thực hiện việc tính toán lại để thay đổi giá trị trong cặp thẻ . Từ đó, giao diện sẽ được cập nhật. Tuy nhiên thì giao diện sẽ **không** được cập nhật!

Thực tế, các biến thông thường như count trong ví dụ trên sẽ không làm cho React thực hiện việc tính toán lại dữ liệu và cập nhật giao diện. React sẽ hoàn toàn bỏ qua sự thay đổi của các biến đó. Khi chúng ta muốn cho React biết rằng nó cần tính toán lại giao diện, chúng ta cần sử dụng một khái niệm đặc biệt từ React: **“State”**

Sử dụng state với React hooks

State về cơ bản là một giá trị biến đặc biệt trong React. Nó là giá trị mà khi thay đổi, React sẽ tiến hành việc tính toán lại kết quả của component và từ đó cập nhật lại giao diện. Để sử dụng được state, chúng ta cần import một function từ trong thư viện React là useState. useState và một số function khác trong thư viện được gọi là các “hooks”.

Trong thực tế, người ta thường sử dụng cú pháp destructuring để khai báo biến state và setState. Cú pháp như sau:

```
const [count, setCount] = useState(10)
```

Xem ví dụ dưới đây:

```
import { useState } from 'react';  
  
const App = () => {  
  const [count, setCount] = useState(10);  
  const onIncreaseClick = () => {  
    setCount(count + 1);  
  };  
};
```

```
return (  
  <div>  
    <span>{count}</span>  
    <button onClick={onIncreaseClick}>Increase</button>  
  </div>  
);  
};  
export default App;
```

2.2 Node.js

Node.js được phát hành vào năm 2009, Node.js (hay NodeJS) là một môi trường runtime JavaScript đa nền tảng và mã nguồn mở, cho phép lập trình viên tạo cả ứng dụng front-end và back-end bằng JavaScript.

Node.js được xây dựng trên V8 JavaScript Engine, được viết bằng C++ và JavaScript. Được phát triển bởi Ryan Dahl, Node.js phù hợp cho các ứng dụng web như trang video, forum hoặc mạng xã hội phạm vi hẹp. Nhờ khả năng chạy trên nhiều hệ điều hành khác nhau (Windows, Linux, macOS,...) và cung cấp nhiều thư viện JavaScript module, Node.js đơn giản hóa việc lập trình và giúp giảm thời gian phát triển.

Mục tiêu của Node.js là tạo ra một nền tảng có khả năng xử lý tốt các ứng dụng thời gian thực với khả năng mở rộng cao và hiệu quả trên nhiều hệ điều hành. [2]

2.2.1 Lợi ích của việc sử dụng Node.js

Hiệu suất cao và không chặn (non-blocking): Node.js sử dụng mô hình xử lý không đồng bộ (asynchronous) và non-blocking I/O, giúp xử lý nhiều yêu cầu cùng lúc mà không bị chặn, tăng hiệu suất cho các ứng dụng thời gian thực.

Sử dụng chung JavaScript ở cả frontend và backend: Giúp các lập trình viên JavaScript dễ dàng làm việc trên cả hai phía, đơn giản hóa việc chuyển đổi dữ liệu và logic giữa frontend và backend.

Thư viện phong phú: Node.js có một hệ sinh thái mạnh mẽ qua NPM (Node Package Manager), cung cấp nhiều thư viện hữu ích, giúp tăng tốc độ phát triển và giảm thiểu thời gian viết mã thủ công.

Dễ mở rộng và linh hoạt: Node.js được thiết kế để dễ dàng mở rộng, phù hợp với kiến trúc microservices, giúp dễ quản lý, bảo trì và phát triển tính năng mới khi cần.

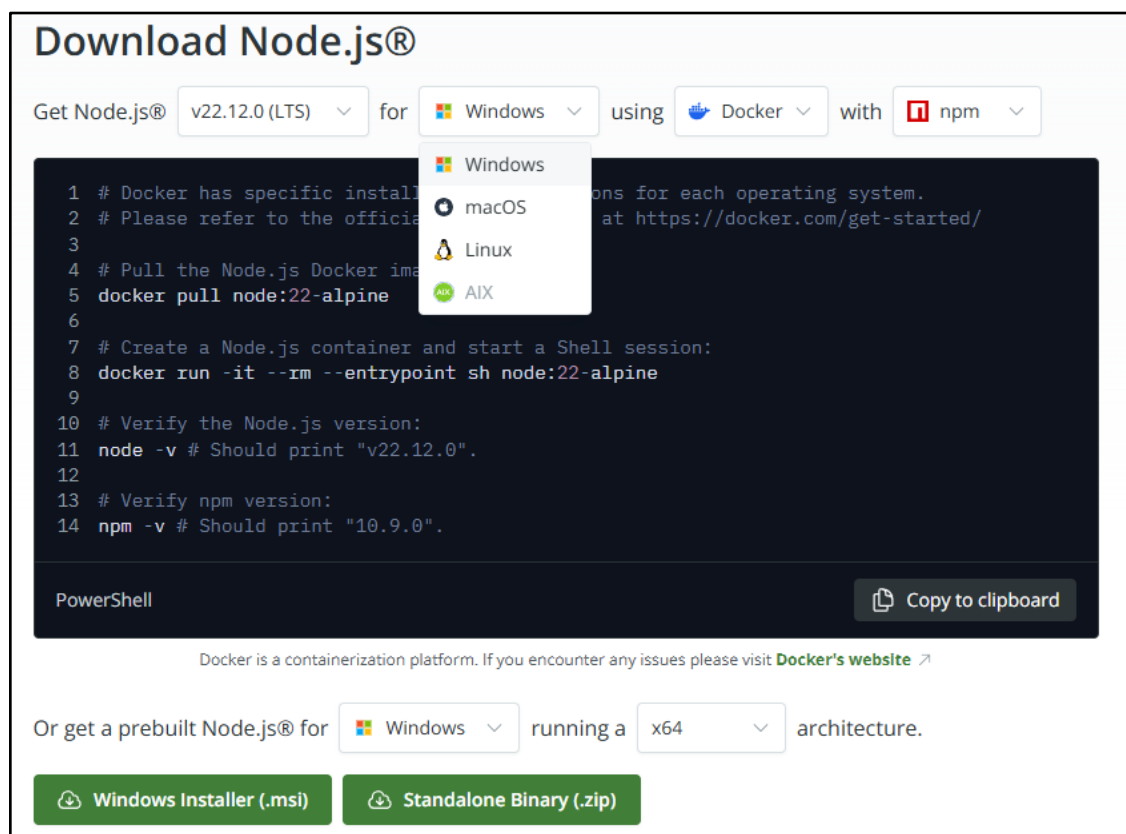
Cộng đồng lớn: Node.js có một cộng đồng lớn và năng động, cung cấp nhiều tài nguyên, hướng dẫn và hỗ trợ nhanh chóng trong quá trình phát triển.

Ứng dụng thời gian thực: Node.js đặc biệt mạnh trong việc phát triển các ứng dụng thời gian thực như chat, video streaming và các hệ thống thông báo nhờ vào khả năng xử lý sự kiện liên tục. [7]

2.2.2 Bắt đầu cài đặt với Node.js

2.2.2.1 Cài đặt Node.js

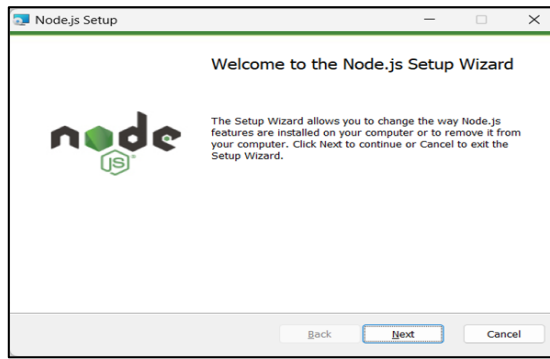
Truy cập trang web chính thức của Node.js: Điều này giúp bạn tải xuống phiên bản mới nhất của Node.js và chọn hệ điều hành phù hợp. Trang web chính thức là <https://nodejs.org/>. [2]



Hình 2.3 Tải Node.js

Cài đặt trên Windows

Giả sử bạn đang làm việc với máy tính chạy Windows 10/Windows 11, hãy tải xuống trình cài đặt 64 bit cho Windows và bắt đầu cài đặt bằng cách nhấp đúp vào tệp đã tải xuống.



Hình 2.4 Cài đặt Node.js

Quá trình cài đặt sẽ đưa bạn qua một vài bước của trình hướng dẫn cài đặt. Nó cũng thêm thư mục cài đặt của tệp thực thi Node.js vào đường dẫn hệ thống.

Để kiểm tra NodeJS đã được cài đặt ở trên máy tính hay chưa, chúng ta mở Command Prompt và sử dụng câu lệnh **node -v**. Máy tính sẽ hiển thị phiên bản được cài đặt.

```
C:\Users\PC>node -v
v22.11.0
```

Hình 2.5 Kiểm tra phiên bản Node.js

2.2.2.2 Tạo cấu trúc thư mục cho Node.js

Tạo thư mục cho dự án:

```
mkdir my-node-app
cd my-node-app
```

Khởi tạo dự án Node.js: Chạy lệnh sau để tạo file package.json:

```
npm init -y
```

Lệnh này sẽ tạo ra file package.json với các giá trị mặc định.

Tạo các thư mục cần thiết như: src, routes, controllers, models và config public.

src/app.js: File chính cho ứng dụng Node.js.

routes/: Chứa các định nghĩa route cho ứng dụng.

controllers/: Chứa các file xử lý logic cho từng route.

models/: Chứa các định nghĩa mô hình dữ liệu (nếu có cơ sở dữ liệu).

config/: Chứa các file cấu hình, ví dụ như cấu hình kết nối cơ sở dữ liệu.

public/: Chứa các tài nguyên tĩnh (CSS, JavaScript, ảnh).

File package.json chứa thông tin về dự án, các thư viện phụ thuộc và các script. Bạn có thể chỉnh sửa các thông tin trong file package.json theo nhu cầu của dự án. [7]

2.2.2.3 Xây dựng một server đơn giản

Node.js cho phép bạn xây dựng một server HTTP chỉ với một vài dòng mã. Dưới đây là hướng dẫn chi tiết. [7]

Tạo file `src/app.js`:

```
// Import module HTTP
const http = require('http');

// Tạo server
const server = http.createServer((req, res) => {
  res.statusCode = 200; // Trạng thái thành công
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello, world!\n');
});

// Khởi động server ở cổng 3000
server.listen(3000, () => {
  console.log('Server đang chạy tại http://localhost:3000');
});
```

Giải thích mã nguồn:

http.createServer(): Tạo một server mới.

req: Đối tượng đại diện cho yêu cầu từ client.

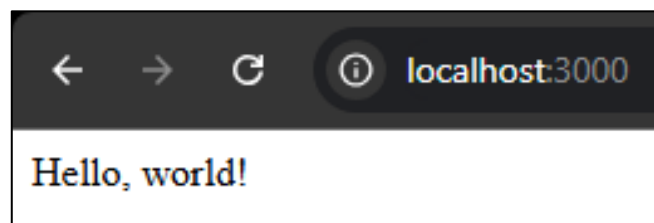
res: Đối tượng đại diện cho phản hồi mà server gửi đến client.

res.end(): Kết thúc phản hồi và gửi nội dung về client.

Chạy server bằng cách di chuyển đến thư mục `my-node-app` bằng lệnh

cd my-node-app. Sau đó chạy với lệnh **node src/app.js**.

Truy cập server: Mở trình duyệt và nhập địa chỉ `http://localhost:3000`. Bạn sẽ thấy thông điệp "Hello, world!" hiển thị trên giao diện.



Hình 2.6 Chạy server Node.js

2.2.2.4 Sử dụng các thư viện phổ biến

Node.js có một hệ sinh thái phong phú với nhiều thư viện mạnh mẽ giúp tăng tốc quá trình phát triển. Dưới đây là một số thư viện phổ biến và cách sử dụng chúng. [2]

Cài đặt Express.js

Express.js là một framework web phổ biến cho Node.js, giúp dễ dàng quản lý routing và middleware.

Cài đặt Express: Trong thư mục dự án, chạy lệnh:

```
npm install express
```

Tạo file src/app.js với Express:

```
const express = require('express');
const app = express();
const port = 3000;

// Định nghĩa route chính
app.get('/', (req, res) => {
  res.send('Hello, world!');
});

// Khởi động server
app.listen(port, () => {
  console.log(`Server đang chạy tại http://localhost:${port}`);
});
```

Chạy server như server node ở trên: **node src/app.js** và truy cập vào đường dẫn <http://localhost:3000> để xem kết quả.

Các thư viện phổ biến khác

Mongoose: Thư viện giúp làm việc với MongoDB.

```
npm install mongoose
```

Socket.io: Thư viện hỗ trợ ứng dụng thời gian thực.

```
npm install socket.io
```

Cors: Giúp quản lý CORS cho ứng dụng web.

```
npm install cors
```


2.3 Kết luận

Khi phát triển trang web thương mại điện tử, việc áp dụng các công nghệ như ReactJS, Node.js, Express và MySQL mang lại nhiều lợi ích nổi bật. Sự kết hợp này không chỉ nâng cao trải nghiệm người dùng mà còn tối ưu hóa hiệu suất và khả năng tương thích trên nhiều nền tảng khác nhau.

ReactJS cho phép xây dựng giao diện người dùng có tính linh hoạt và tương tác cao thông qua các thành phần động mà không làm giảm chất lượng trải nghiệm. Cùng với đó, Node.js và Express có khả năng xử lý đồng thời và phản hồi nhanh chóng, tạo ra một máy chủ mạnh mẽ để hỗ trợ cho ứng dụng thương mại điện tử.

Việc sử dụng MySQL giúp quản lý cơ sở dữ liệu một cách hiệu quả và an toàn, bảo vệ thông tin cá nhân của người dùng. Thêm vào đó, việc sử dụng HTTPS trong giao tiếp giữa máy khách và máy chủ thông qua Node.js và Express đảm bảo tính bảo mật cao.

Các công nghệ này còn tăng cường tính mở rộng và khả năng bảo trì, giúp ứng dụng dễ dàng điều chỉnh theo sự phát triển của nhu cầu. Quản lý trạng thái hiệu quả của cả máy chủ và ứng dụng được thực hiện thông qua ReactJS và Node.js, giảm thiểu hiện tượng treo trang và cải thiện khả năng duyệt web.

Cuối cùng, sự kết hợp này giúp xây dựng một trang web thương mại điện tử có hiệu suất cao, khả năng mở rộng linh hoạt và bảo mật đáng tin cậy.

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 công nghệ, 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, nơi khách hàng có thể dễ dàng tìm kiếm, so sánh và mua sắm các sản phẩm công nghệ như điện thoại di động, laptop và các phụ kiện công nghệ.

Các yêu cầu cụ thể của bài toán bao gồm:

Chức năng quản lý sản phẩm: Quản trị viên có thể thêm, sửa, xóa và cập nhật thông tin sản phẩm, bao gồm hình ảnh, giá cả, mô tả và trạng thái tồn kho. Hỗ trợ phân loại sản phẩm theo danh mục (ví dụ: điện thoại, laptop, phụ kiện,...).

Chức năng giỏ hàng và thanh toán: Khách hàng có thể thêm sản phẩm vào giỏ hàng, điều chỉnh số lượng hoặc xóa sản phẩm khỏi giỏ hàng. Cung cấp quy trình thanh toán an toàn và tích hợp với các phương thức thanh toán phổ biến (thẻ tín dụng, ví điện tử).

Quản lý người dùng: Hỗ trợ đăng ký và đăng nhập tài khoản qua email hoặc tích hợp đăng nhập với Google hoặc Facebook. Người dùng có thể quản lý thông tin cá nhân, lịch sử mua sắm và trạng thái đơn hàng.

Tìm kiếm và bộ lọc: Hỗ trợ chức năng tìm kiếm sản phẩm theo tên hoặc từ khóa. Bộ lọc theo giá, thương hiệu hoặc các thuộc tính sản phẩm khác để giúp người dùng dễ dàng tìm được sản phẩm phù hợp.

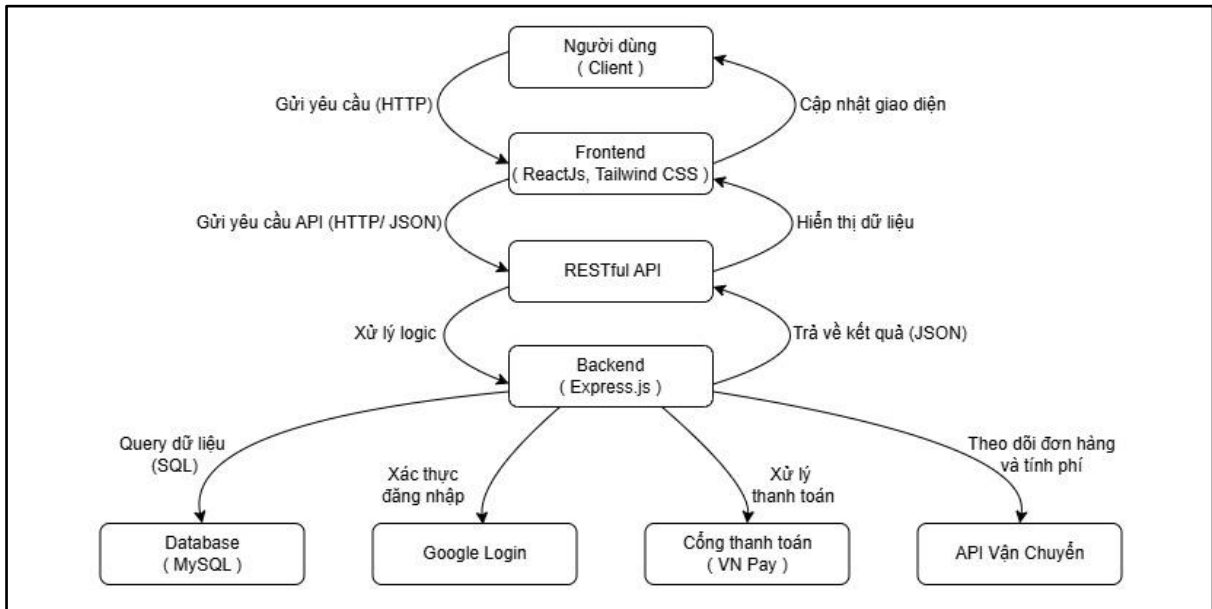
Giao diện người dùng: Cung cấp trải nghiệm trực quan và thân thiện với người dùng trên cả máy tính và thiết bị di động. Giao diện phản hồi nhanh, giảm thiểu thời gian tải trang.

Quản lý đơn hàng: Quản trị viên có thể xem, xác nhận và cập nhật trạng thái đơn hàng (đang xử lý, đã giao, hủy).

Khả năng mở rộng: Thiết kế hệ thống linh hoạt để dễ dàng thêm tính năng mới hoặc mở rộng quy mô khi lượng người dùng tăng.

Mục tiêu chính: Xây dựng một ứng dụng web thương mại điện tử vừa đáp ứng nhu cầu mua sắm của khách hàng, vừa hỗ trợ quản lý hiệu quả cho doanh nghiệp, đồng thời đảm bảo tính ổn định, bảo mật và khả năng mở rộng trong tương lai.

3.2 Kiến trúc hệ thống



Hình 3.1 Kiến trúc hệ thống

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

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

Frontend: Dùng ReactJS và Tailwind CSS để xây dựng giao diện, nhận yêu cầu từ người dùng và gửi yêu cầu API.

RESTful API: Là cầu nối giữa Frontend và Backend, chuyển tiếp yêu cầu và nhận kết quả dưới dạng JSON.

Backend (Express.js): Xử lý logic nghiệp vụ, thực hiện các thao tác như truy vấn cơ sở dữ liệu, xác thực, thanh toán và tính phí vận chuyển.

Database (MySQL): Lưu trữ dữ liệu hệ thống như sản phẩm, đơn hàng, người dùng.

Google Login: Cung cấp xác thực qua tài khoản Google.

Cổng thanh toán (PayPal, VNPay): Xử lý giao dịch thanh toán của người dùng.

API Vận chuyển: Tính phí vận chuyển và theo dõi trạng thái giao hàng.

Luồng dữ liệu: Người dùng gửi yêu cầu qua Frontend, Frontend gửi yêu cầu API, API chuyển đến Backend, Backend xử lý và truy vấn cơ sở dữ liệu, gửi yêu cầu xác thực Google, thanh toán hoặc vận chuyển, rồi trả kết quả về API và hiển thị trên giao diện người dùng.

3.3 Phân tích đặc tả yêu cầu hệ thống

3.3.1 Yêu cầu chức năng

Quản lý sản phẩm:

Thêm, sửa, xóa sản phẩm và cập nhật thông tin sản phẩm (hình ảnh, giá, mô tả, số lượng, biến thể màu sắc,...).

Phân loại sản phẩm theo danh mục (điện thoại, laptop, phụ kiện,...).

Giỏ hàng và thanh toán:

Thêm sản phẩm vào giỏ hàng, chỉnh sửa số lượng hoặc xóa sản phẩm.

Thanh toán an toàn và nhanh chóng, hỗ trợ các phương thức thanh toán như thẻ tín dụng, ví điện tử.

Quản lý người dùng:

Đăng ký và đăng nhập tài khoản qua email hoặc Google/Facebook.

Quản lý thông tin cá nhân, lịch sử mua sắm, trạng thái đơn hàng.

Tìm kiếm và bộ lọc:

Tìm kiếm sản phẩm theo tên hoặc từ khóa.

Bộ lọc theo giá, thương hiệu, thuộc tính sản phẩm.

Quản lý đơn hàng:

Quản trị viên xác nhận và cập nhật trạng thái đơn hàng (đang xử lý, đã giao, hủy).

Giao diện người dùng:

Hỗ trợ hiển thị trên máy tính và thiết bị di động giúp phản hồi nhanh, dễ sử dụng.

Hệ thống quản trị:

Quản lý người dùng, sản phẩm, đơn hàng.

Cập nhật khuyến mãi và danh mục sản phẩm.

3.3.2 Yêu cầu phi chức năng

Hiệu năng:

Ứng dụng phản hồi dưới 3 giây đối với mọi thao tác người dùng.

Hỗ trợ đồng thời ít nhất 500 người dùng truy cập.

Bảo mật:

Sử dụng giao thức HTTPS để bảo mật thông tin truyền tải.

Mã hóa thông tin nhạy cảm như mật khẩu, dữ liệu thanh toán.

Khả năng tương thích:

Ứng dụng chạy ổn định trên các trình duyệt phổ biến (Chrome, Firefox, Edge).

Giao diện tương thích với nhiều kích thước màn hình.

Dễ sử dụng:

Giao diện thân thiện, trực quan, dễ thao tác cho cả người dùng và quản trị viên.

Khả năng bảo trì:

Mã nguồn rõ ràng, dễ bảo trì và mở rộng.

Cung cấp tài liệu chi tiết về cấu trúc và thiết kế hệ thống.

Tích hợp dịch vụ bên ngoài:

Kết nối với các API bên ngoài như cổng thanh toán, vận chuyển.

Hỗ trợ tích hợp các công cụ phân tích dữ liệu và báo cáo.

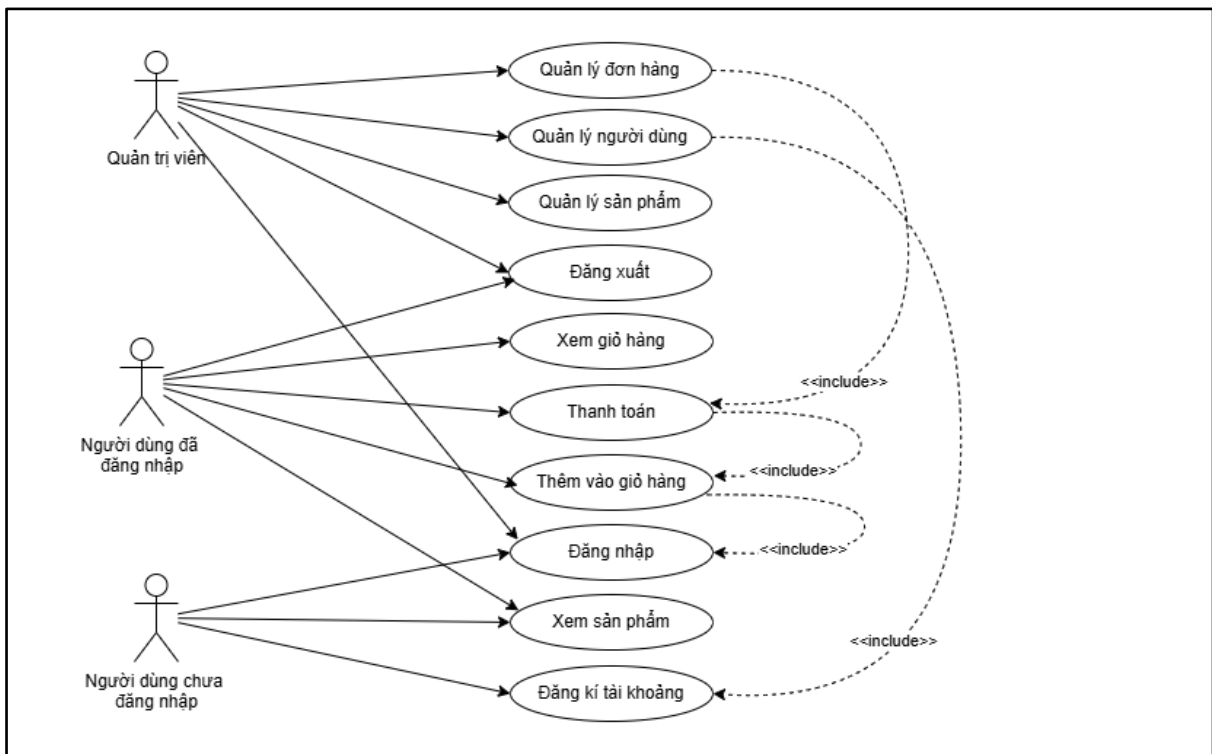
3.4 Thiết kế hệ thống

3.4.1 Sơ đồ usecase

Người dùng chưa đăng nhập (Guest) có thể xem sản phẩm, đăng ký tài khoản và đăng nhập.

Người dùng đã đăng nhập (LoggedInUser) có thể xem sản phẩm, thêm sản phẩm vào giỏ hàng, thanh toán, xem giỏ hàng và đăng xuất.

Quản trị viên (Admin) có thể quản lý người dùng, quản lý sản phẩm, quản lý đơn hàng và đăng xuất.



Hình 3.2 Sơ đồ Usecase

3.4.2 Sơ đồ lớp

Bảng Brand

Mục đích: Quản lý thông tin thương hiệu của các sản phẩm.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
BrandID	int	ID duy nhất của thương hiệu.
Name	varchar(100)	Tên thương hiệu.
Slug	varchar(100)	Chuỗi định danh URL.
Description	text	Mô tả thương hiệu.
CreatedAt	timestamp	Thời gian tạo thương hiệu.

Bảng 3.1 Bản thương hiệu

Bảng Category

Mục đích: Phân loại các sản phẩm theo danh mục.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
CategoryID	int	ID duy nhất của danh mục.
Name	varchar(100)	Tên danh mục.
Slug	varchar(100)	Chuỗi định danh URL.
Description	text	Mô tả danh mục.
CreatedAt	timestamp	Thời gian tạo danh mục.

Bảng 3.2 Bảng danh mục

Bảng Product

Mục đích: Quản lý thông tin chi tiết của sản phẩm.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
ProductID	int	ID duy nhất của sản phẩm.
CategoryID	int	Tham chiếu đến ID danh mục.
BrandID	int	Tham chiếu đến ID thương hiệu.
Name	varchar(255)	Tên sản phẩm.
Slug	varchar(255)	Chuỗi định danh URL.
Description	text	Mô tả sản phẩm.
Thumbnail	varchar(255)	Đường dẫn ảnh đại diện.
CreatedAt	timestamp	Thời gian tạo sản phẩm.

Bảng 3.3 Bảng sản phẩm

Bảng ProductVariant

Mục đích: Quản lý các biến thể của sản phẩm như màu sắc, dung lượng bộ nhớ.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
VariantID	int	ID biến thể sản phẩm.
ProductID	int	Tham chiếu đến sản phẩm cha.
Price	decimal(10,2)	Giá của biến thể.
MemorySizeID	int	Tham chiếu đến kích thước bộ nhớ.

Bảng 3.4 Bảng biến thể sản phẩm

Bảng Order

Mục đích: Quản lý các đơn hàng của người dùng.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
OrderID	int	ID duy nhất của đơn hàng.
UserID	int	Tham chiếu đến ID người dùng.
OrderDate	timestamp	Ngày đặt hàng.
TotalAmount	decimal(10,2)	Tổng số tiền của đơn hàng.
ShippingAddress	text	Địa chỉ giao hàng.
OrderStatus	enum(Pending, Completed)	Trạng thái đơn hàng.

Bảng 3.5 Bảng đặt hàng

Bảng Cart

Mục đích: Quản lý thông tin các sản phẩm trong giỏ hàng của người dùng.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
CartID	int	ID duy nhất của giỏ hàng.
UserID	int	Tham chiếu đến ID người dùng.
VariantID	int	Tham chiếu đến ID biến thể sản phẩm.
Quantity	int	Số lượng sản phẩm trong giỏ.
CreatedAt	timestamp	Thời gian tạo giỏ hàng.

Bảng 3.6 Bảng giỏ hàng

Bảng User

Mục đích: Quản lý thông tin người dùng trong hệ thống.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
UserID	int	ID duy nhất của người dùng.
FullName	varchar(100)	Tên đầy đủ của người dùng.
Email	varchar(100)	Email của người dùng.
PasswordHash	varchar(255)	Mật khẩu đã mã hóa.
PhoneNumber	varchar(20)	Số điện thoại.
Address	text	Địa chỉ người dùng.
Role	enum(Customer, Admin)	Vai trò của người dùng.
Status	varchar(50)	Trạng thái tài khoản (active, inactive).
CreatedAt	timestamp	Thời gian tạo tài khoản.

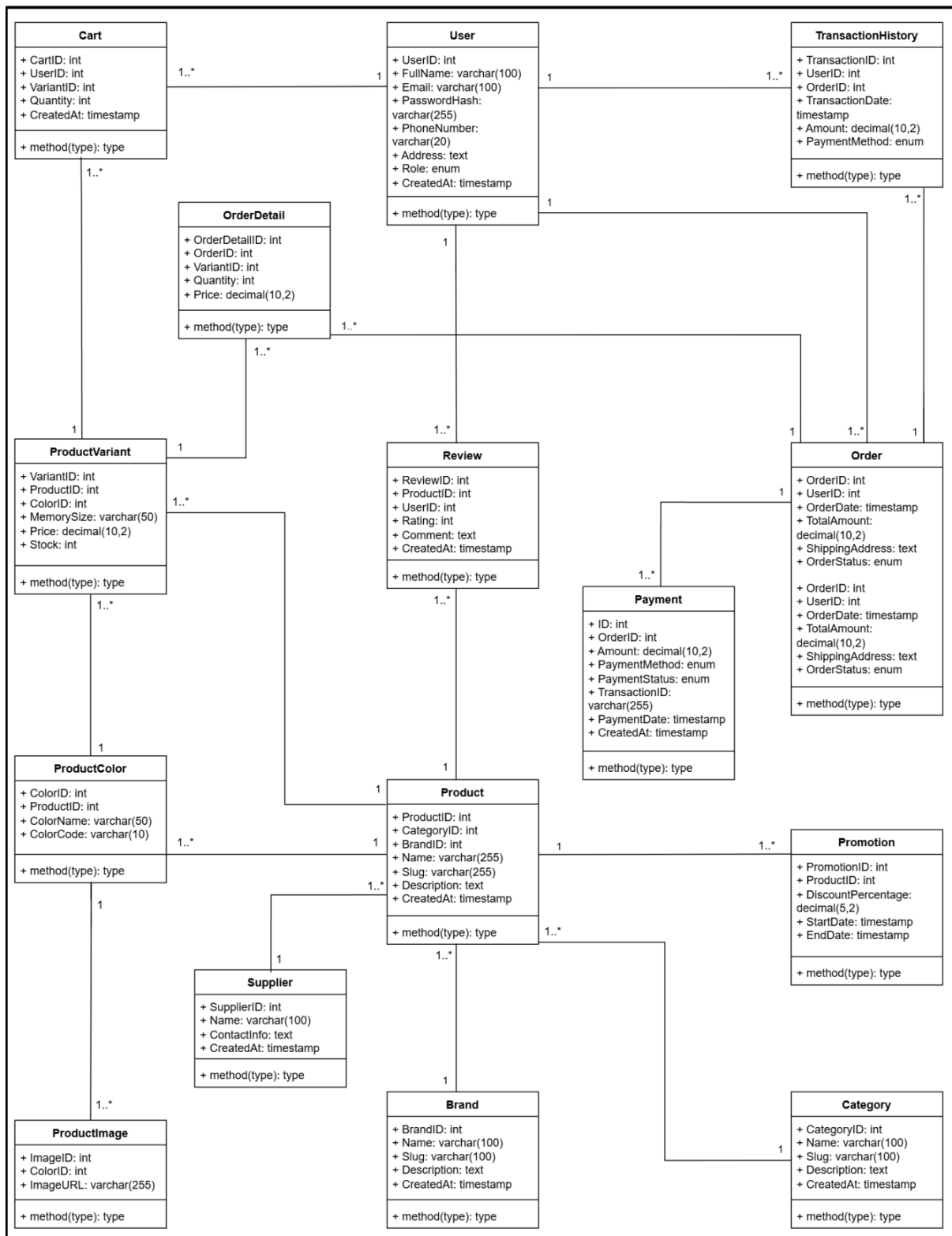
Bảng 3.7 Bảng người dùng

Bảng Supplier

Mục đích: Quản lý thông tin nhà cung cấp của sản phẩm.

<i>Tên thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
SupplierID	int	ID duy nhất của nhà cung cấp.
Name	varchar(100)	Tên nhà cung cấp.
ContactInfo	text	Thông tin liên hệ của nhà cung cấp.
CreatedAt	timestamp	Thời gian tạo nhà cung cấp.

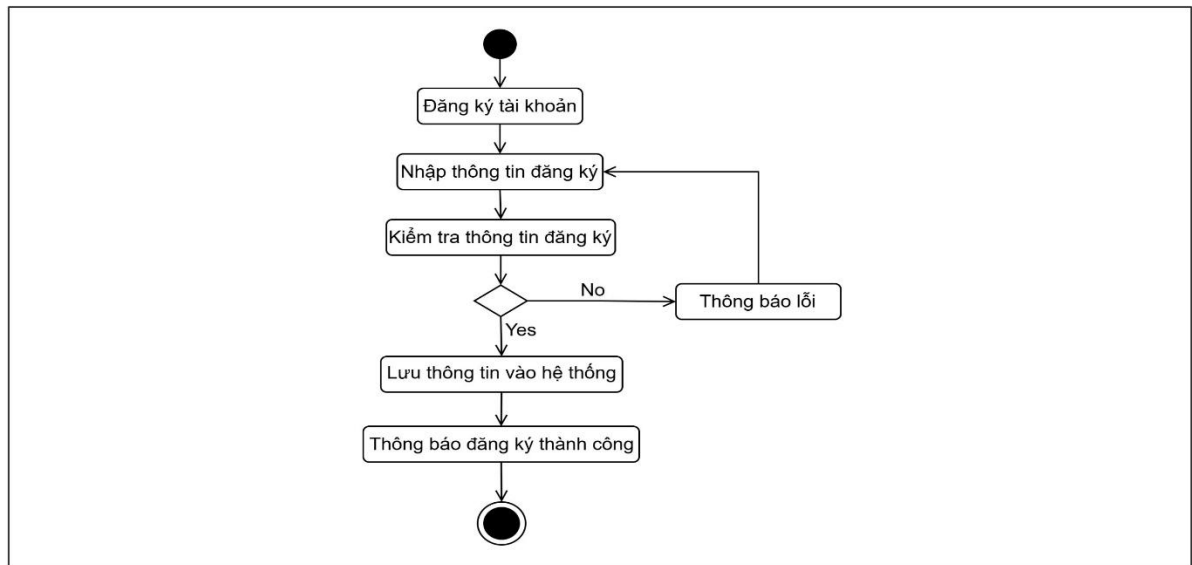
Bảng 3.8 Bảng nhà cung cấp



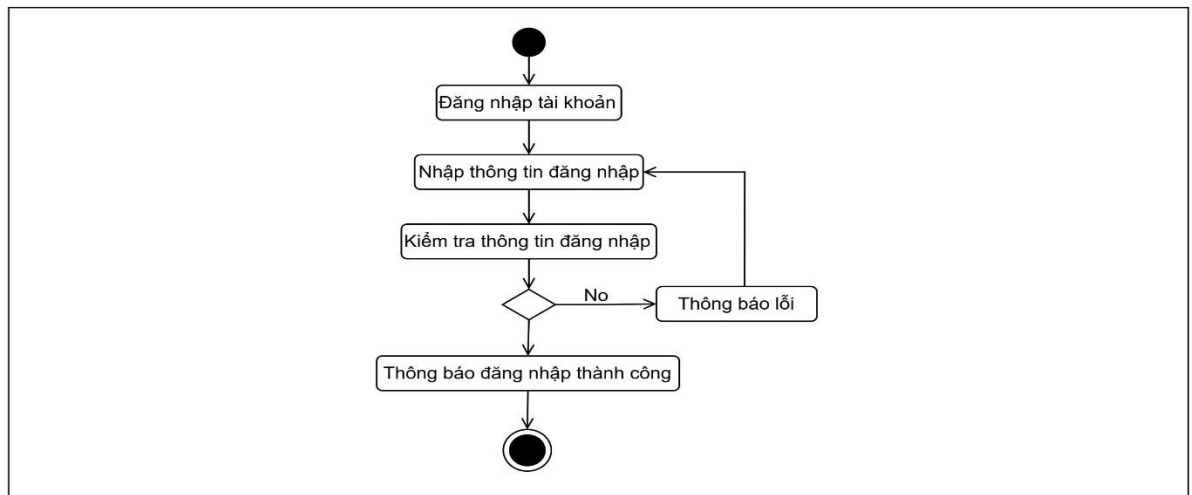
Hình 3.3 Sơ đồ lớp

Sơ đồ lớp này cung cấp một cái nhìn tổng quan về cấu trúc dữ liệu của hệ thống bán hàng trực tuyến. Nó giúp các nhà phát triển hiểu rõ cách dữ liệu được tổ chức và liên kết, từ đó thiết kế cơ sở dữ liệu và xây dựng ứng dụng một cách hiệu quả. Nó cũng giúp cho việc trao đổi thông tin giữa các thành viên trong nhóm phát triển được rõ ràng và chính xác hơn.

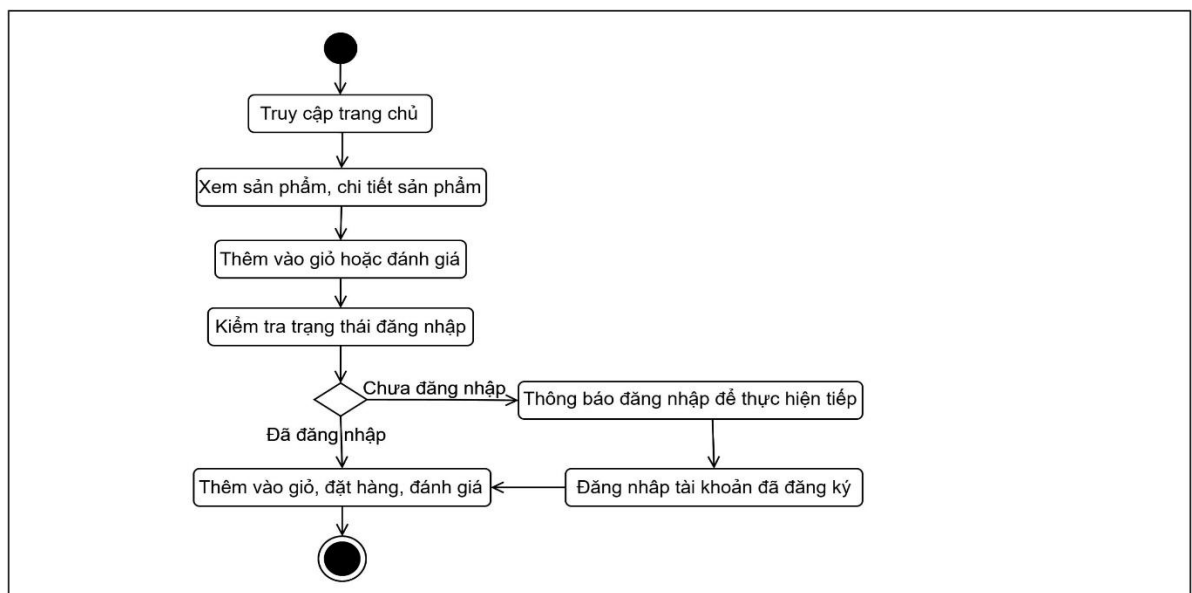
3.4.3 Sơ đồ hoạt động



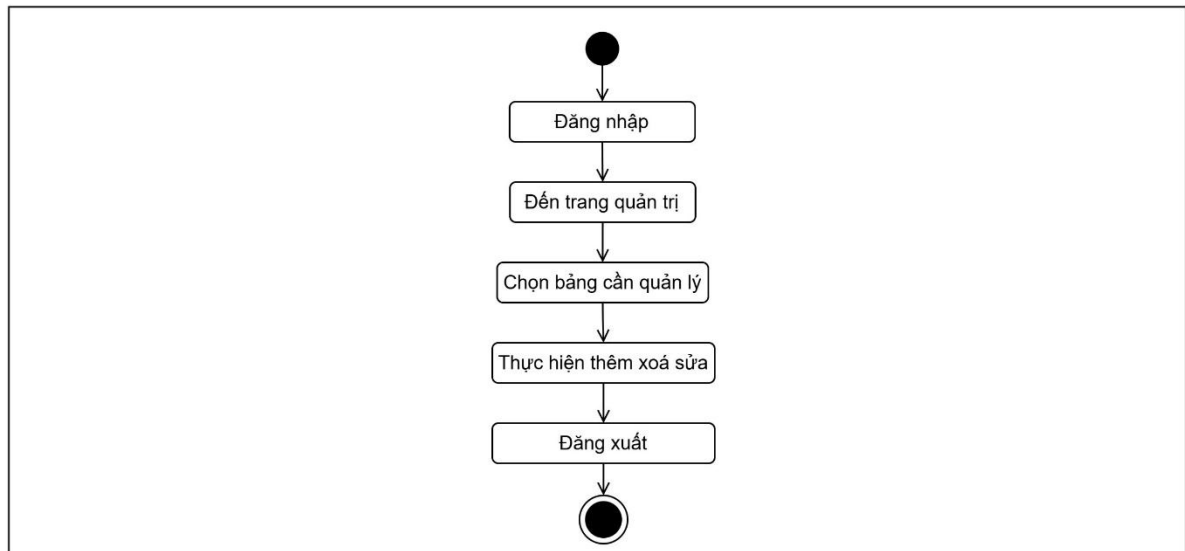
Hình 3.4 Sơ đồ hoạt động cho đăng ký



Hình 3.5 Sơ đồ hoạt động cho đăng nhập



Hình 3.6 Sơ đồ hoạt động cho người dùng



Hình 3.7 Sơ đồ hoạt động cho 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 mô tả quy trình mua hàng trực tuyến của người dùng, từ khi đăng nhập đến khi hoàn tất đơn hàng, bao gồm các bước như chọn sản phẩm, thêm vào giỏ hàng, nhập thông tin giao hàng và thanh toán. Sơ đồ cũng thể hiện các quyết định và trường hợp ngoại lệ, ví dụ: thông tin đăng nhập hoặc giỏ hàng không hợp lệ, hay thanh toán không thành công.

Các thành phần chính bao gồm người dùng và hệ thống, với các hình khối thể hiện hành động, điểm quyết định và điểm bắt đầu/kết thúc quy trình. Quy trình chi tiết bao gồm:

Đăng nhập: Kiểm tra thông tin đăng nhập hợp lệ.

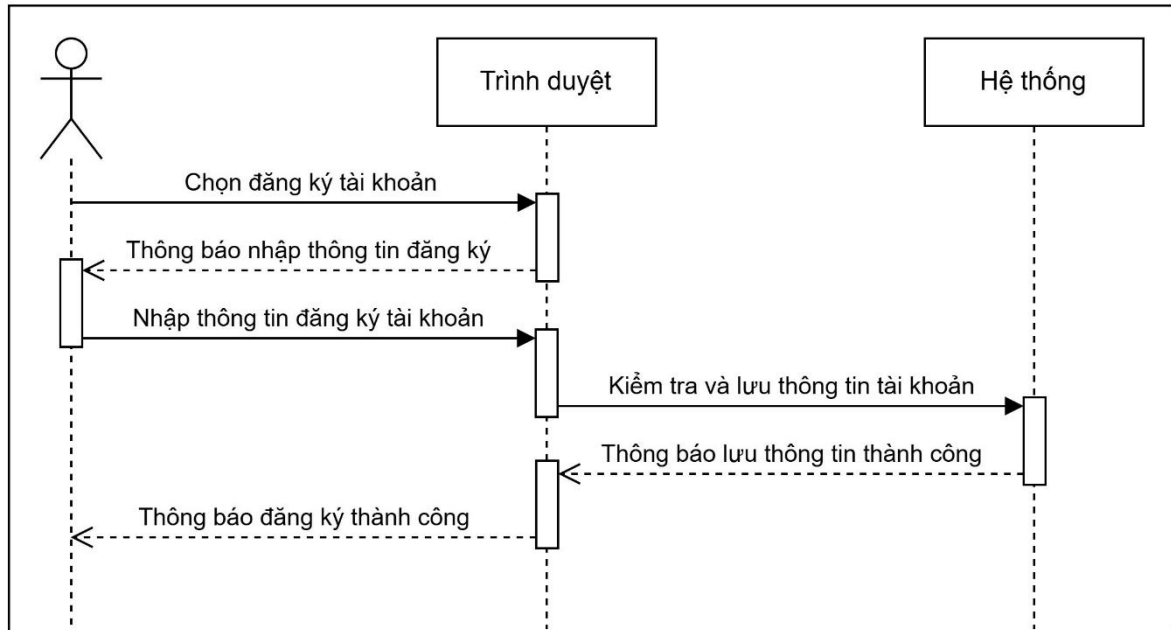
Truy cập, chọn sản phẩm, thêm sản phẩm vào giỏ hàng và kiểm tra tính hợp lệ của giỏ hàng.

Nhập thông tin giao hàng và chọn phương thức thanh toán.

Xác minh thanh toán và cập nhật trạng thái đơn hàng.

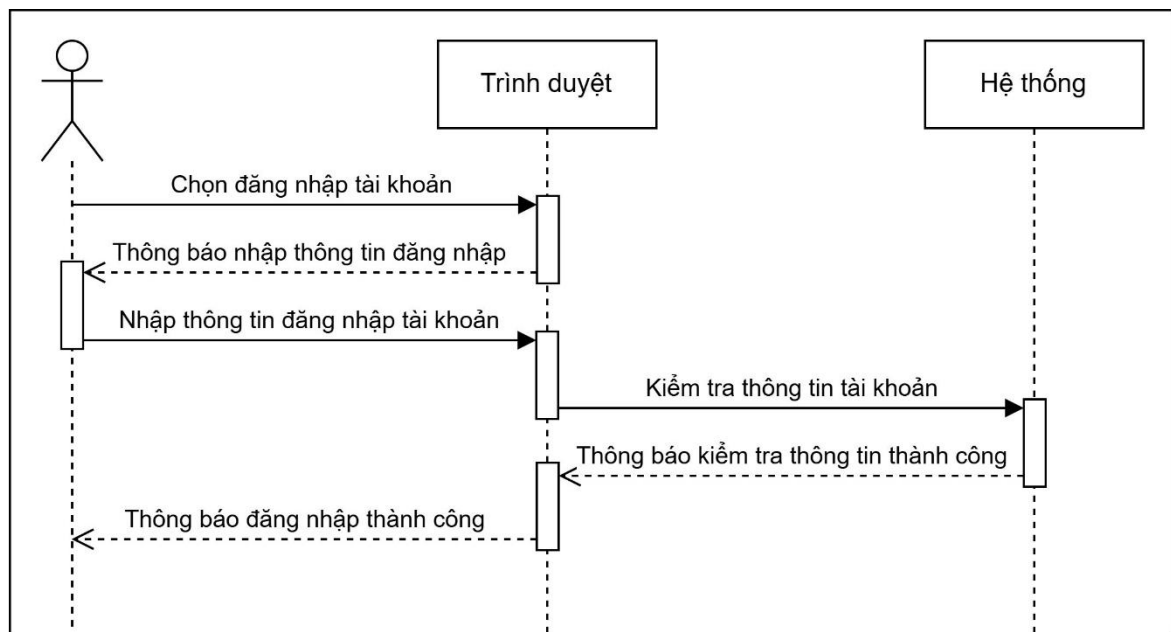
Nhận thông báo từ hệ thống và hoàn tất quy trình hoặc liên hệ hỗ trợ nếu có lỗi.

3.4.4 Sơ đồ tuần tự



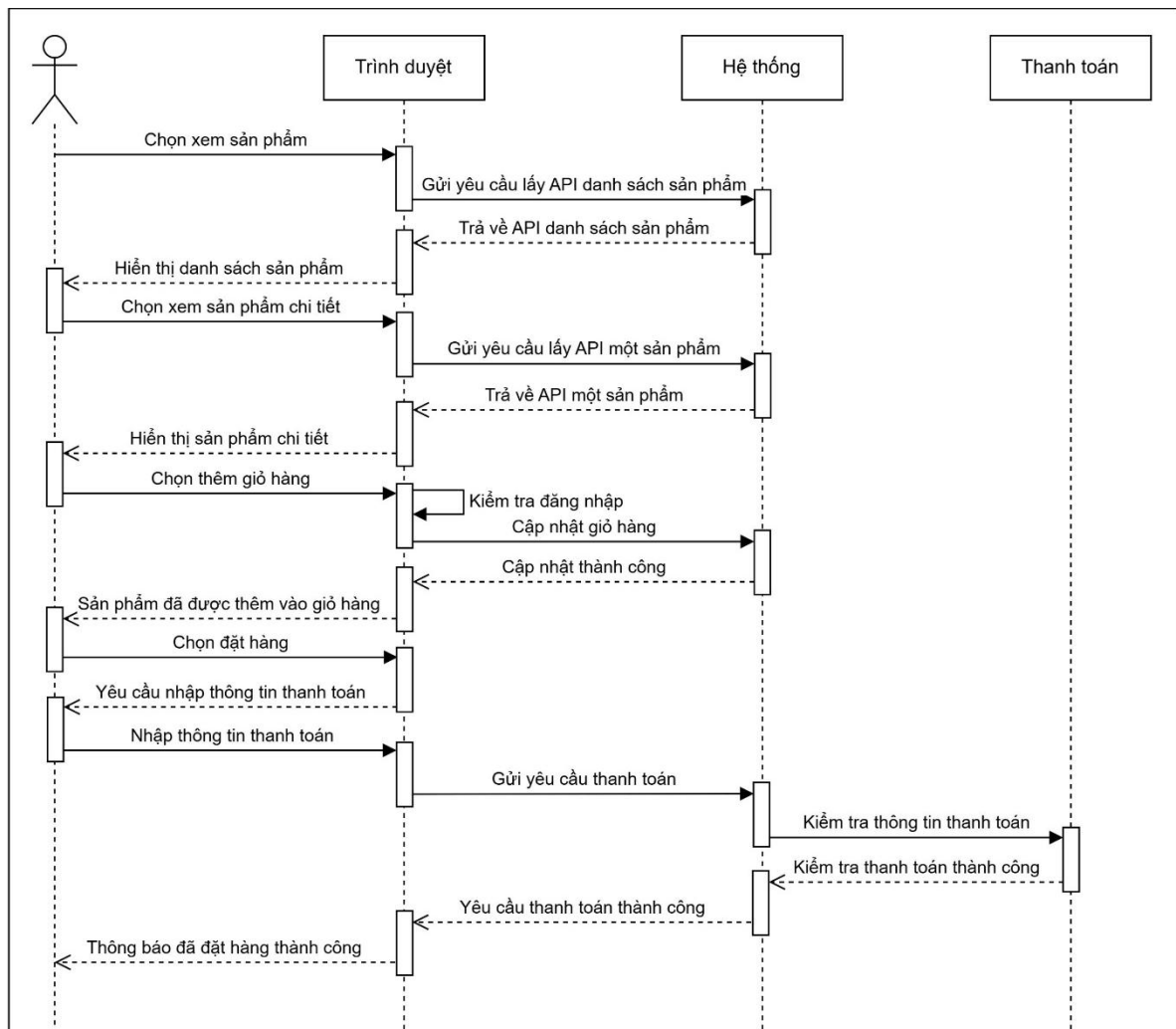
Hình 3.8 Sơ đồ tuần tự thao tác đăng ký

Đăng ký: Biểu đồ tuần tự mô tả quy trình đăng ký tài khoản trong hệ thống. Người dùng chọn đăng ký trên trình duyệt, nhập thông tin như tên, email và mật khẩu. Hệ thống kiểm tra tính hợp lệ, lưu thông tin tài khoản và cuối cùng thông báo đăng ký thành công. Biểu đồ giúp hình dung rõ ràng các bước và sự tương tác giữa các đối tượng.



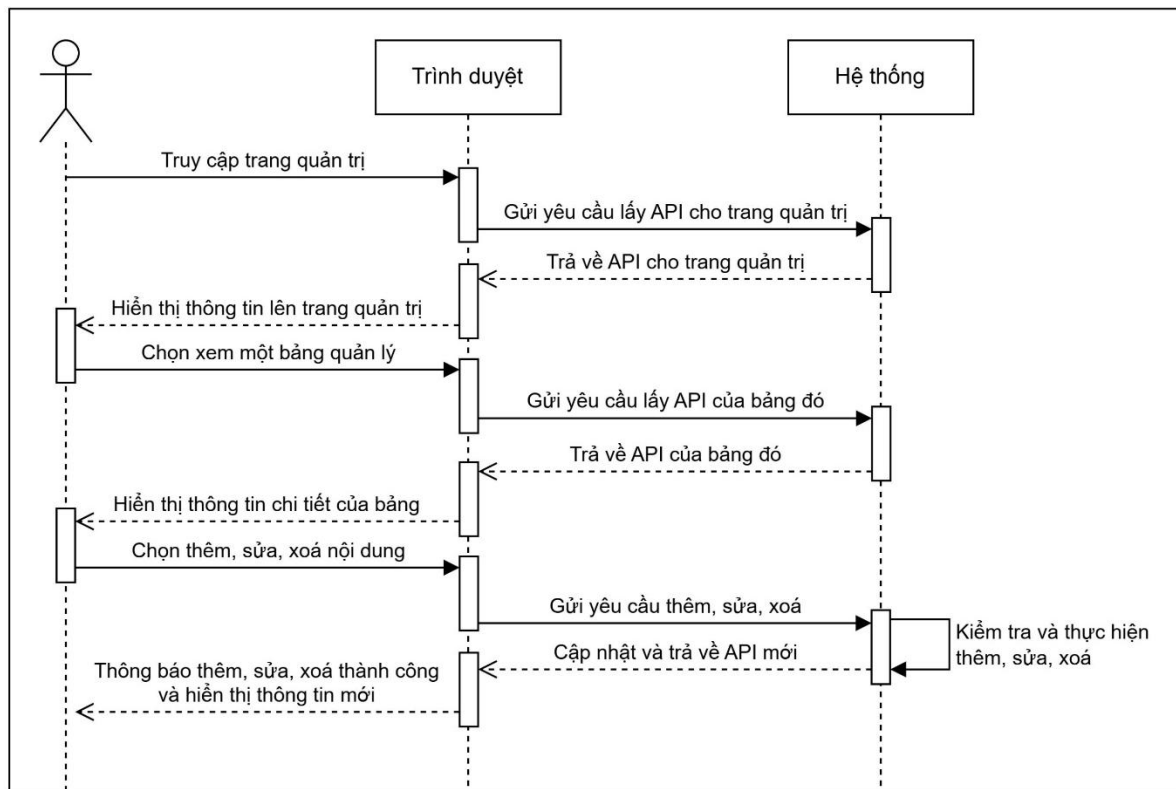
Hình 3.9 Sơ đồ tuần tự thao tác đăng nhập

Đăng nhập: Biểu đồ tuần tự mô tả quy trình đăng nhập vào hệ thống. Người dùng chọn đăng nhập, nhập thông tin tài khoản và hệ thống kiểm tra tính hợp lệ. Cuối cùng, nếu thông tin đúng, người dùng nhận thông báo đăng nhập thành công. Biểu đồ giúp hình dung rõ ràng các bước và tương tác giữa các đối tượng.



Hình 3.10 Sơ đồ tuần tự thao tác của người dùng

Thao tác của người dùng: Biểu đồ tuần tự mô tả quy trình mua sắm trực tuyến. Người dùng chọn xem sản phẩm trên trình duyệt, gửi yêu cầu lấy danh sách sản phẩm từ hệ thống. Sau khi xem, người dùng chọn sản phẩm và yêu cầu thêm vào giỏ hàng. Hệ thống xác nhận sản phẩm và yêu cầu đăng nhập. Sau khi đăng nhập, người dùng yêu cầu thanh toán. Hệ thống kiểm tra thông tin thanh toán và cuối cùng thông báo giao dịch đã thành công. Biểu đồ này giúp hình dung rõ ràng các bước và tương tác giữa người dùng, hệ thống và quá trình thanh toán.

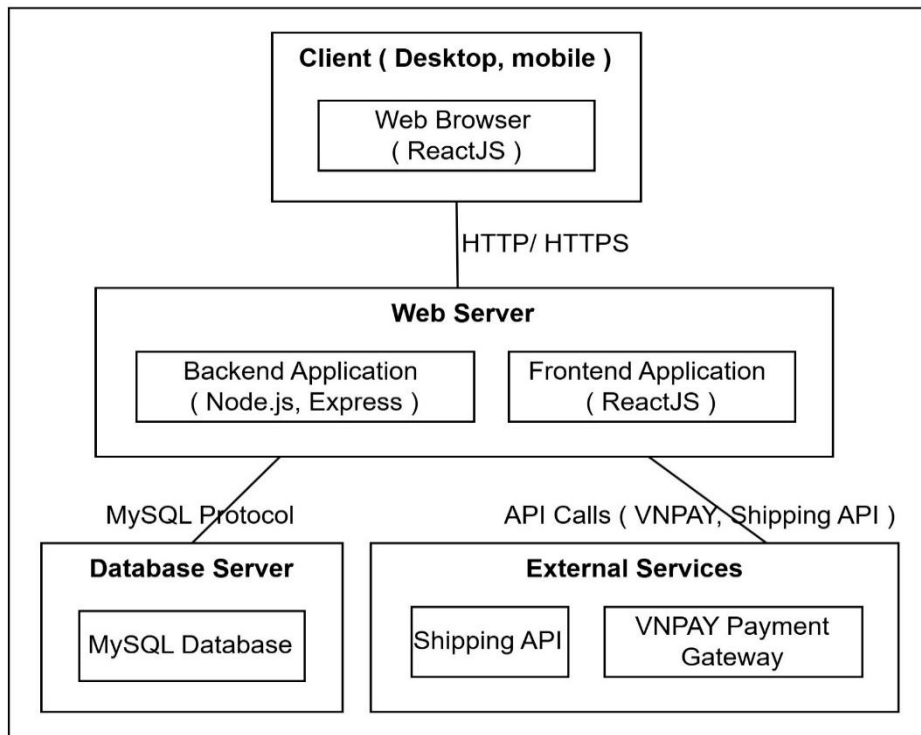


Hình 3.11 Sơ đồ tuần tự thao tác của người quản trị

Thao tác của người quản trị: Biểu đồ mô tả quy trình quản trị trang web. Người quản trị truy cập vào trang quản trị và gửi yêu cầu lấy dữ liệu từ hệ thống. Hệ thống trả về thông tin trang quản trị. Sau khi chọn một bảng dữ liệu, quản trị viên có thể thêm, sửa hoặc xóa nội dung. Hệ thống cập nhật và trả về dữ liệu mới, đồng thời thông báo kết quả thực hiện.

Tóm lại: Phân thiết kế hệ thống bao gồm sơ đồ use case, sơ đồ lớp, sơ đồ hoạt động và sơ đồ tuần tự. Các sơ đồ này cùng nhau mô tả chi tiết các yêu cầu chức năng, cấu trúc dữ liệu, quy trình xử lý và sự tương tác giữa các thành phần trong hệ thống thương mại điện tử. Sơ đồ use case xác định vai trò và chức năng chính, sơ đồ lớp thể hiện cách tổ chức dữ liệu, sơ đồ hoạt động minh họa quy trình công việc và sơ đồ tuần tự mô tả sự trao đổi dữ liệu theo thời gian. Những sơ đồ này cung cấp nền tảng quan trọng để hiện thực hóa và đảm bảo hệ thống vận hành hiệu quả.

3.4.5 Mô hình triển khai



Hình 3.12 Mô hình triển khai

Máy khách (Client):

Các thiết bị người dùng (máy tính để bàn, điện thoại di động) sẽ truy cập vào ứng dụng thông qua trình duyệt web hoặc ứng dụng di động.

Giao tiếp với backend qua giao thức HTTP/HTTPS.

Máy chủ web (Web Server):

Chạy ứng dụng frontend (ReactJS) và backend (Node.js, Express).

Frontend xử lý giao diện người dùng và gửi yêu cầu đến backend.

Backend xử lý logic, kết nối với cơ sở dữ liệu và các dịch vụ bên ngoài.

Máy chủ cơ sở dữ liệu (Database Server):

Chạy MySQL để lưu trữ dữ liệu của hệ thống (người dùng, đơn hàng, sản phẩm, v.v.).

Backend sẽ sử dụng MySQL Protocol để kết nối và truy xuất dữ liệu.

Dịch vụ bên ngoài (External Services):

Các dịch vụ như cổng thanh toán VNPAY và API vận chuyển sẽ được backend sử dụng để thực hiện các chức năng như thanh toán và tính toán phí vận chuyển.

Giao tiếp giữa backend và các dịch vụ này sẽ qua API Calls.

Giao thức và kết nối:

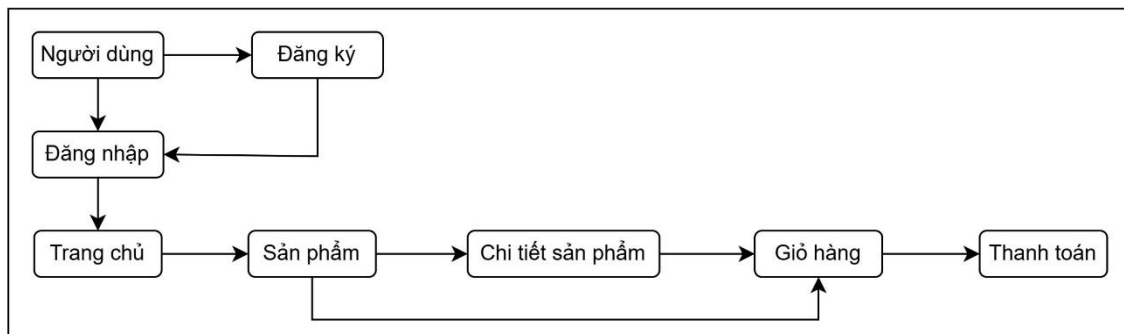
HTTP/HTTPS: Dùng cho giao tiếp giữa frontend và backend.

MySQL Protocol: Dùng cho giao tiếp giữa backend và cơ sở dữ liệu.

API Calls: Dùng cho giao tiếp giữa backend và các dịch vụ bên ngoài như VNPAY và API vận chuyển.

3.5 Thiết kế giao diện

3.5.1 Sơ đồ website



Hình 3.13 Sơ đồ website

Sơ đồ trang web (sitemap) mô tả cấu trúc của một trang web e-commerce với các mối quan hệ như sau: Người dùng có thể đăng ký hoặc đăng nhập, sau đó người dùng có thể truy cập trang chủ. Người dùng từ trang chủ có thể truy cập trang sản phẩm để xem danh sách sản phẩm, sau đó có thể chọn một sản phẩm để xem chi tiết sản phẩm và có thể thêm vào giỏ hàng, sau khi thêm vào giỏ hàng thì người dùng có thể thanh toán và hoàn thành giao dịch.

3.5.2 Màn hình trang người dùng



Hình 3.14 Màn hình trang chủ chưa đăng nhập



Hình 3.15 Màn hình trang chủ 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 bật, sản phẩm nổi bật và các chương trình khuyến mãi. Phần chân trang gồm các thông tin cơ bản của trang web.

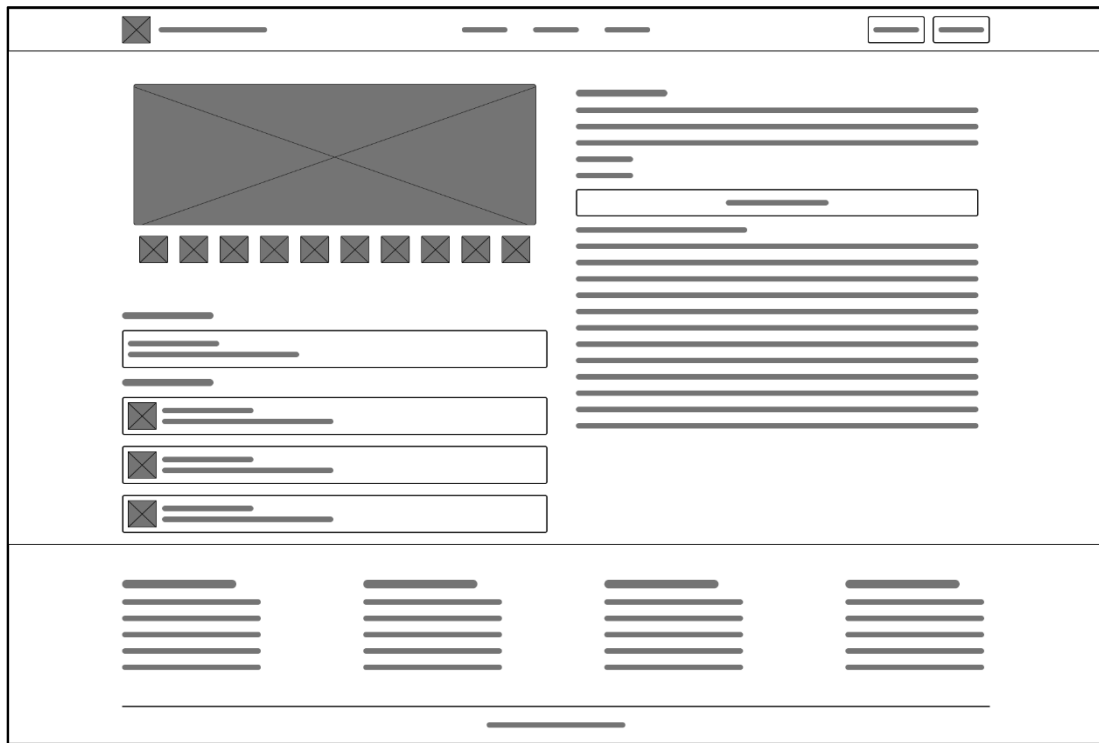
Chức năng: Hiển thị các danh mục, sản phẩm nổi bật và các chương trình khuyến mãi. Người dùng có thể dễ dàng điều hướng đến các danh mục và sản phẩm cụ thể.



Hình 3.16 Màn hình 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.



Hình 3.17 Màn hình trang chi tiết 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 được chia thành hai phần, bên trái để hiển thị bộ ảnh chi tiết của một sản phẩm, bên phải để hiển thị các thông tin về sản phẩm đó như tên, giá, mô tả,... Phía dưới sẽ có một phần để cho người dùng đánh giá sản phẩm và có slider hiển các sản phẩm có liên quan với sản phẩm đang được xem

Chức năng: Cung cấp thông tin chi tiết về sản phẩm, bao gồm hình ảnh, mô tả, tính năng và đánh giá từ người dùng, cho phép thêm sản phẩm vào giỏ hàng.

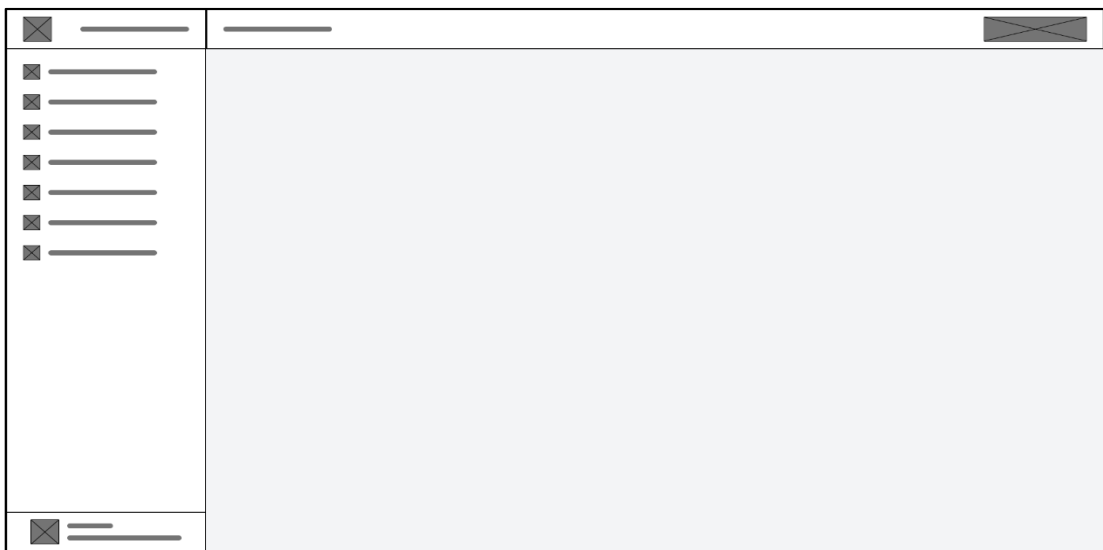


Hình 3.18 Màn hình trang giỏ hàng

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 được thành hai phần, bên trái để hiển thị các sản phẩm đã được người dùng thêm vào giỏ hàng, bên phải để hiển thị các thông tin như số lượng, tổng tiền,...

Chức năng: Hiển thị các sản phẩm đã thêm vào giỏ, cho phép người dùng chỉnh sửa số lượng, xóa sản phẩm và tiến hành thanh toán.

3.5.3 Màn hình trang quản trị

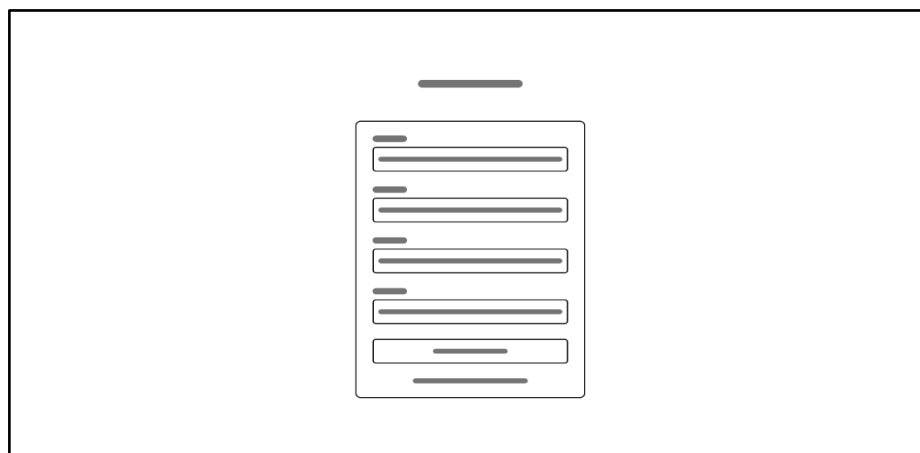


Hình 3.19 Màn hình trang quản trị viên

Mô tả: Giao diện trang quản trị gồm 3 phần: Ở trên là phần đầu trang, ở dưới được chia thành 2 phần, bên trái là menu chứa các bảng cần quản lý, bên phải là phần để hiển thị nội dung quản lý khi người dùng chọn vào menu ở bên trái.

Chức năng: Quản trị viên quản lý sản phẩm, đơn hàng và người dùng. Có thể xem các thống kê và cập nhật trạng thái đơn hàng, sản phẩm.

3.5.4 Màn hình chức năng đăng kí

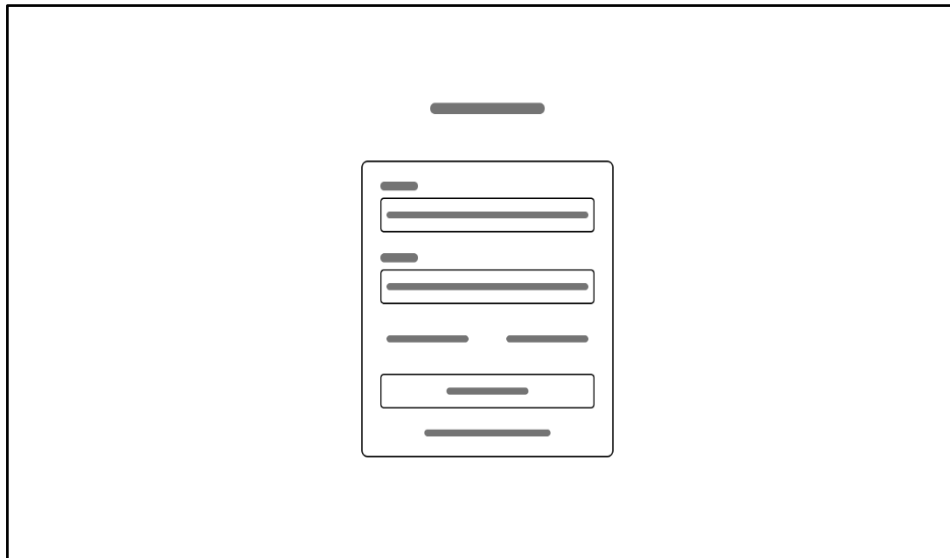


Hình 3.20 Chức năng đăng kí

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.

3.5.5 Màn hình chức năng đăng nhập



Hình 3.21 Chức năng 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.

3.6 Kết chương

Chương này đã trình bày chi tiết các nội dung quan trọng trong việc hiện thực hóa nghiên cứu từ mô tả bài toán, phân tích yêu cầu hệ thống đến thiết kế các thành phần như sơ đồ use case, sơ đồ lớp, sơ đồ hoạt động, sơ đồ tuần tự và mô hình triển khai. Đồng thời, giao diện người dùng và sơ đồ website cũng được thiết kế để đảm bảo trải nghiệm trực quan, thân thiện trên nhiều thiết bị. Những nội dung này không chỉ định hình rõ ràng cấu trúc và chức năng của hệ thống mà còn tạo nền tảng vững chắc cho giai đoạn triển khai và thử nghiệm, hướng đến việc xây dựng một ứng dụng thương mại điện tử hiện đại, đáp ứng hiệu quả nhu cầu của khách hàng và doanh nghiệp.

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

4.1 Dữ liệu thử nghiệm

ID	Name	Slug	Description
32	Điện thoại	dien-thoi	Điện thoại thông minh là thiết bị di động tích hợp các tính năng như nghe gọi, truy cập internet, chụp ảnh, quay video, và nhiều ứng dụng hỗ trợ công việc lẫn giải trí.
33	Máy tính bảng	may-tinh-bang	Máy tính bảng là thiết bị có màn hình lớn hơn điện thoại, kết hợp giữa tính di động và hiệu năng làm việc.
34	Laptop	laptop	Laptop là thiết bị phục vụ nhu cầu học tập, làm việc và giải trí với hiệu năng mạnh mẽ trong một thiết kế gọn nhẹ.
36	Đồng hồ	dong-ho	Đồng hồ thông minh không chỉ giúp quản lý thời gian mà còn tích hợp các tính năng theo dõi sức khỏe, nhận thông báo từ điện thoại, và hỗ trợ thể thao.

Bảng 4.1 Bảng danh mục

ID	Name	Slug	Description
25	Samsung	samsung	Samsung là tập đoàn công nghệ đa quốc gia hàng đầu, cung cấp thiết bị điện tử, di động, và gia dụng chất lượng cao.
26	Apple	apple	Apple nổi tiếng với thiết kế tinh tế, hiệu suất vượt trội và hệ sinh thái sản phẩm đồng bộ.
27	Xiaomi	xiaomi	Xiaomi mang đến sản phẩm công nghệ cao cấp với mức giá cạnh tranh, phù hợp với nhiều phân khúc.
28	Sony	sony	Sony tập trung vào trải nghiệm âm thanh, hình ảnh vượt trội và độ bền sản phẩm cao.
29	Oppo	oppo	Oppo nổi bật với thiết kế thời thượng và công nghệ camera đột phá.

Bảng 4.2 Bảng thương hiệu

ID	Name	ContactInfo
14	Samsung Electronics	Samsung Electronics là một trong những tập đoàn công nghệ hàng đầu thế giới, chuyên cung cấp các sản phẩm điện tử và thiết bị di động với chất lượng vượt trội.
15	Apple Inc	Apple nổi tiếng với các sản phẩm thiết kế sang trọng và hệ sinh thái độc quyền, mang lại trải nghiệm người dùng tuyệt vời.
16	Xiaomi Corporation	Xiaomi cung cấp các sản phẩm công nghệ với giá cả phải chăng nhưng vẫn đảm bảo chất lượng và hiệu năng mạnh mẽ.
17	Sony Corporation	Sony là thương hiệu hàng đầu về các thiết bị điện tử với sự chú trọng vào chất lượng âm thanh và hình ảnh vượt trội.
18	Oppo Electronics	Oppo là thương hiệu nổi bật trong ngành công nghiệp smartphone, với thiết kế thời thượng và tính năng sáng tạo.

Bảng 4.3 Bảng nhà cung cấp

ID	Full Name	Email	Password Hash	Phone Number	Address	Role	Status
1	admin	admin@gmail.com	\$2b\$10\$2qL...	0398989898	Cầu Ngang	Admin	active
2	Trần Trung Nghĩa	trantrungnghia@gmail.com	\$2b\$10\$6dn...	0877564873	Duyên Hải	Customer	active
3	Nguyễn Văn A	nguyenvana@gmail.com	\$2b\$18Hfn0\$...	0975633528	Trà Cú	Customer	active
4	Huỳnh Văn B	huynhvanb@gmail.com	\$27FHb\$10\$...	0957813324	Tiểu Cần	Customer	active

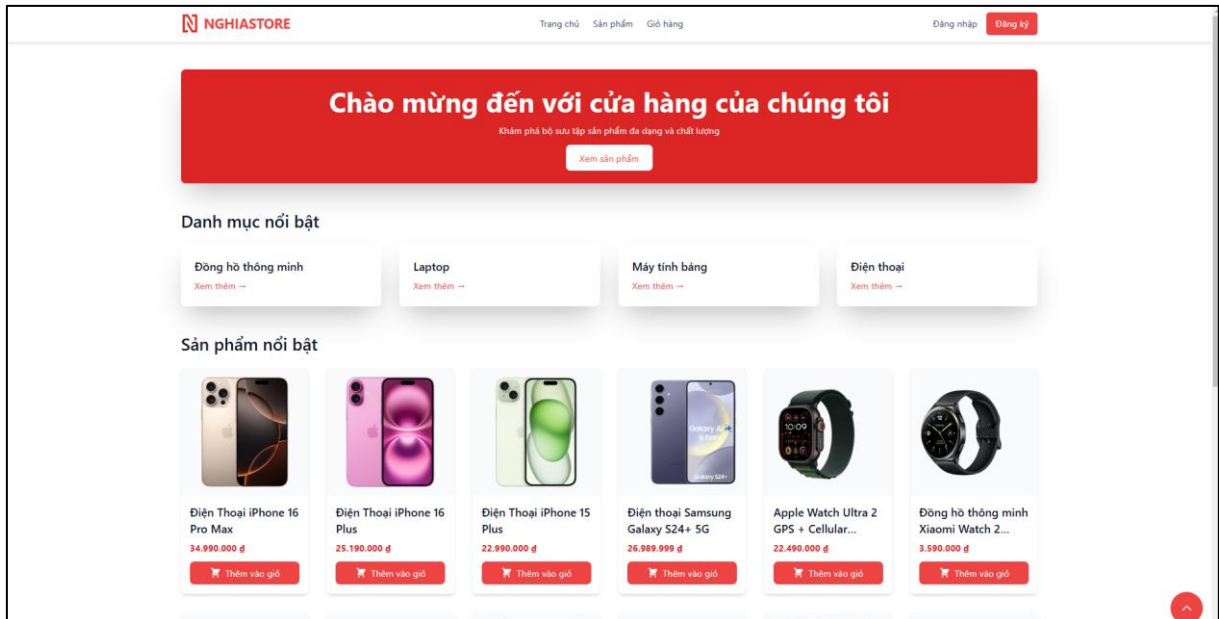
Bảng 4.4 Bảng người dùng

ID	Category ID	Brand ID	Supplier ID	Name	Slug	Description	Thumbnail
41	32	26	15	Điện Thoại iPhone 16 Pro Max	dien-thoi-iphone-16-pro-max	<figure class="table"><figure>	c03c5bac554c54e39934c2a20e1f775c-2330404927.png
41	32	26	15	Điện Thoại iPhone 16 Plus	dien-thoi-iphone-16-plus	<figure class="table"></figure>	40ef2182a291e09b359088e1be603442-9726598527.png
42	32	26	15	Điện Thoại iPhone 15 Plus	dien-thoi-iphone-15-plus	<figure class="table"></figure>	95460aaf588b60f7495dc7db664e8a80-3145465819.png
42	32	25	14	Điện thoại Samsung Galaxy S24+ 5G	dien-thoi-samsung-galaxy-s24-5g	<figure class="table"></figure>	eefc5895f4dc9588ab62e112956e58e0-1129554492.png

Bảng 4.5 Bảng sản phẩm

4.2 Kết quả thử nghiệm

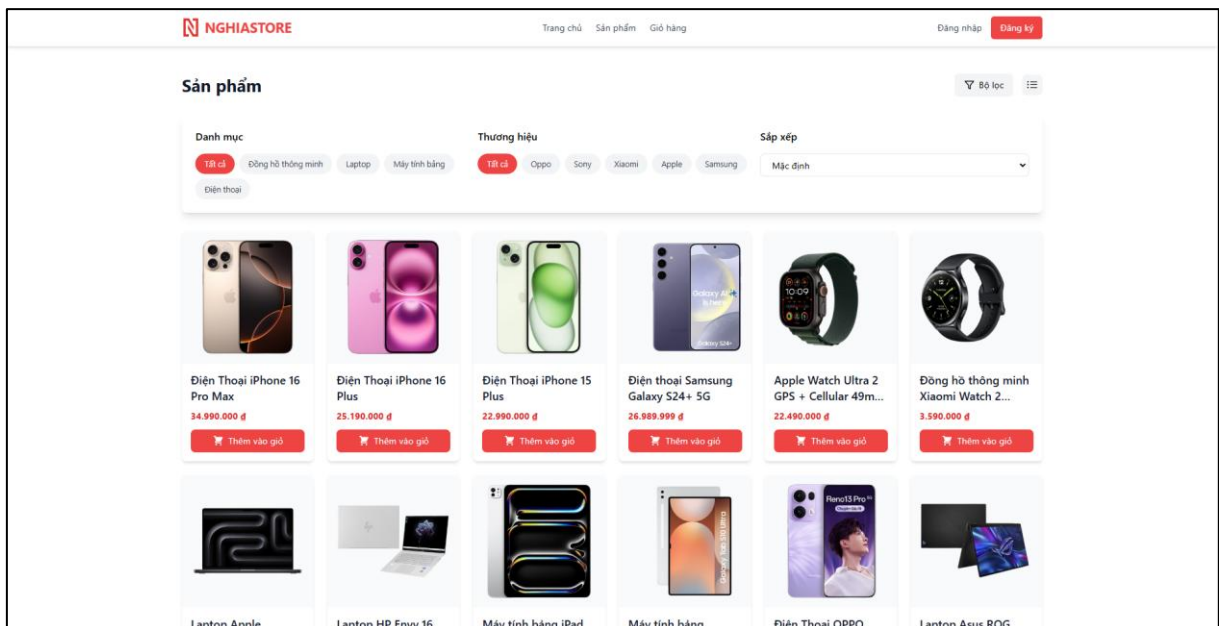
4.2.1 Giao diện trang chủ



Hình 4.1 Giao diện trang chủ

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

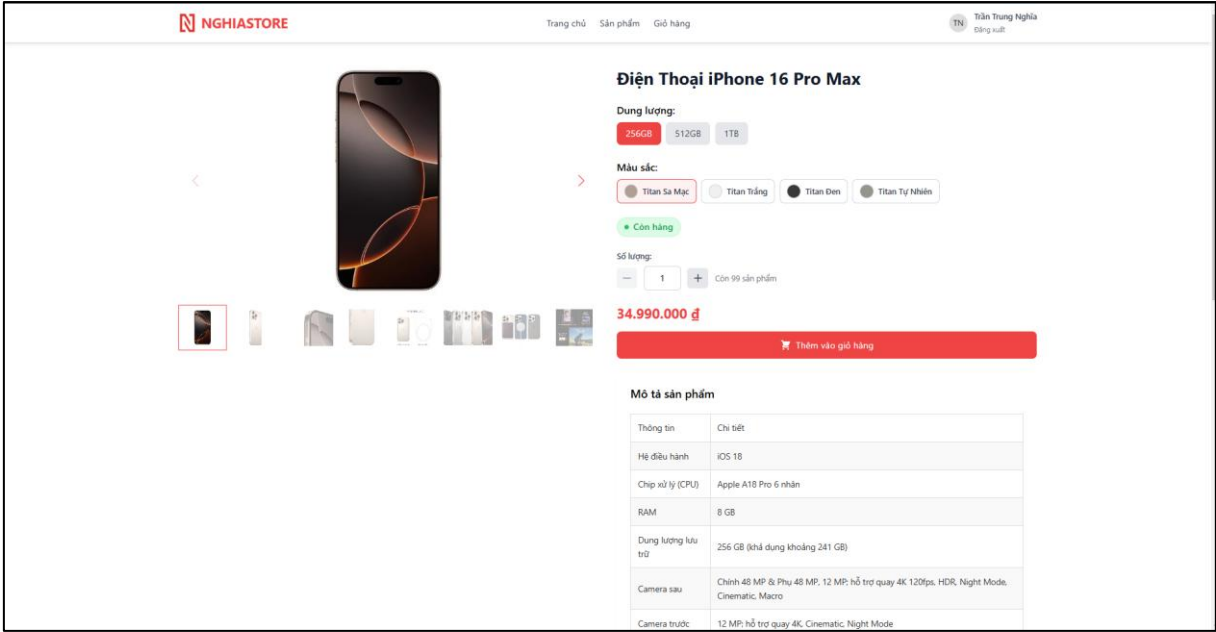
4.2.2 Giao diện trang sản phẩm



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.

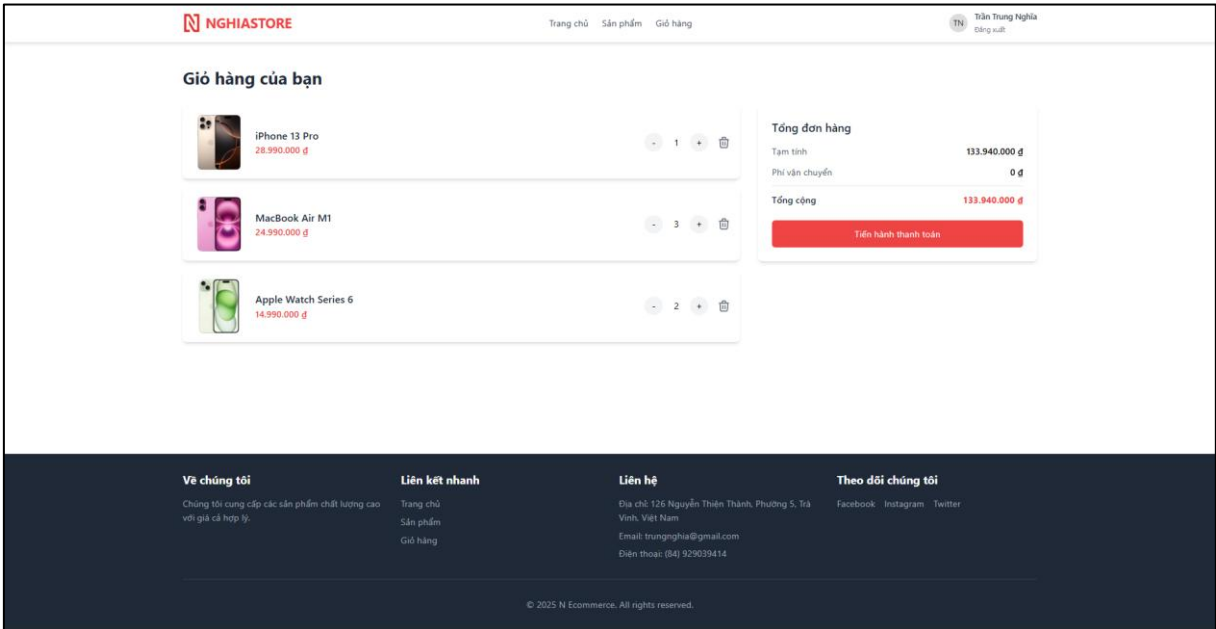
4.2.3 Giao diện trang sản phẩm chi tiết



Hình 4.3 Giao diện trang sản phẩm chi tiết

Chức năng: Cung cấp thông tin chi tiết về sản phẩm, bao gồm hình ảnh, mô tả, tính năng, đánh giá sản phẩm từ người dùng và cho phép thêm sản phẩm vào giỏ hàng.

4.2.4 Giao diện trang giỏ hàng



Hình 4.4 Giao diện trang giỏ hàng

Chức năng: Hiện thị các sản phẩm đã được thêm vào giỏ, cho phép người dùng tăng giảm số lượng sản phẩm, xóa sản phẩm và tiến hành thanh toán.

4.3 Kết luận

Trong chương 4, báo cáo đã trình bày chi tiết quá trình thử nghiệm và đánh giá kết quả của hệ thống. Các dữ liệu thử nghiệm đã chứng minh sự hoạt động ổn định và chính xác của các chức năng chính như hiển thị danh sách sản phẩm, quản lý giỏ hàng và xử lý thanh toán. Giao diện người dùng được thiết kế thân thiện, trực quan, đáp ứng tốt trên cả máy tính và thiết bị di động, mang lại trải nghiệm người dùng mượt mà.

Kết quả thử nghiệm cho thấy hệ thống đáp ứng các yêu cầu đặt ra, đảm bảo hiệu năng cao và tính bảo mật trong giao dịch. Các tính năng của website không chỉ phục vụ tốt nhu cầu mua sắm của khách hàng mà còn hỗ trợ quản trị viên quản lý hiệu quả danh mục sản phẩm, đơn hàng và thông tin người dùng.

Những kết quả đạt được trong chương này đã tạo tiền đề quan trọng cho việc đánh giá tổng thể hệ thống và đề xuất các hướng phát triển trong tương lai.

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong báo cáo này, dự án "Xây dựng website thương mại điện tử bán đồ công nghệ sử dụng ReactJS và Node.js" đã hoàn thành với nhiều kết quả tích cực. Hệ thống được thiết kế và triển khai với đầy đủ các tính năng chức năng cần thiết cho một website thương mại điện tử, bao gồm:

Quản lý sản phẩm, giỏ hàng, đơn hàng.

Hỗ trợ đăng nhập và đăng ký tài khoản, tích hợp các phương thức xác thực bên thứ ba như Google.

Giao diện người dùng trực quan, thân thiện và tương thích với nhiều thiết bị.

Xây dựng backend vững chắc sử dụng Node.js và Express, cung cấp API RESTful an toàn và hiệu quả.

Những kết quả này đã chứng minh đề tài đáp ứng tốt các yêu cầu đặt ra ban đầu, hệ thống hoạt động ổn định, đảm bảo hiệu suất cao và đồng bộ giữa frontend và backend.

5.2 Hướng phát triển

Mặc dù đã hoàn thiện các chức năng cơ bản, hệ thống vẫn có nhiều tiềm năng phát triển trong tương lai:

Mở rộng hệ thống thanh toán: Tích hợp nhiều cổng thanh toán hơn như MoMo, ZaloPay, PayPal để tăng tính linh hoạt cho khách hàng.

Cá nhân hóa trải nghiệm: Sử dụng machine learning để gợi ý sản phẩm dựa trên hành vi mua sắm và sở thích của người dùng.

Cải thiện UI/UX: Tích hợp giao diện đa ngôn ngữ và tăng cường trải nghiệm người dùng trên các thiết bị khác nhau.

Phát triển app di động: Xây dựng ứng dụng di động native cho cả Android và iOS sử dụng React Native.

Quản trị phân quyền: Tăng cường quản trị phân quyền chi tiết cho nhân viên trong hệ thống quản trị.

Với những hướng phát triển này, hệ thống sẽ không chỉ đáp ứng tốt hơn nhu cầu khách hàng mà còn tăng tính cạnh tranh trên thị trường.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] "ReactJS," [Online]. Available: <https://react.dev/>. [Accessed 2024].
- [2] "Node.js," [Online]. Available: <https://nodejs.org/en>. [Accessed 2024].
- [3] "W3school - MySQL Tutorial," [Online]. Available:
<https://www.w3schools.com/mysql/default.asp>. [Accessed 2024].
- [4] "KungfuTech," [Online]. Available: <https://kungfutech.edu.vn/khoa-hoc/reactjs>.
[Accessed 2024].
- [5] "Tailwind CSS," [Online]. Available: <https://tailwindcss.com/>. [Accessed 2024].
- [6] "W3schools - React Tutorial," [Online]. Available:
<https://www.w3schools.com/react/default.asp>. [Accessed 2024].
- [7] "W3schools - Node.js Tutorial," [Online]. Available:
<https://www.w3schools.com/nodejs/default.asp>. [Accessed 2024].

PHỤ LỤC

Công cụ và môi trường phát triển

1. Visual Studio Code: Công cụ phát triển chính: <https://code.visualstudio.com/>
2. Postman: Kiểm thử API: <https://www.postman.com/>
3. Laragon: Hỗ trợ chạy MySQL cục bộ: <https://laragon.org/download/>
4. Git: Quản lý mã nguồn: <https://github.com/trantrungnghia414/cn-da21ttb-trantrungnghia-ecommercewebsite-reactjs>
5. NPM: Quản lý các thư viện và gói cài đặt: <https://www.npmjs.com/>