

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIA ĐỊNH
KHOA CÔNG NGHỆ THÔNG TIN**



TIỂU LUẬN

WEBSITE BÁN ĐIỆN THOẠI DI ĐỘNG

MÔN: LẬP TRÌNH NODEJS

Giảng viên hướng dẫn: **LÊ HUỲNH PHƯỚC**

Sinh viên thực hiện: **NGUYỄN NGỌC THÀNH**

MSSV: **2104110025**

Lớp: **K15DCPM01**

Ngành: **KĨ THUẬT PHẦN MỀM**

TP. Hồ Chí Minh, tháng 4, năm 2024

Khoa/Viện: Công Nghệ Thông Tin

NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN

TIỂU LUẬN MÔN: LẬP TRÌNH NODEJS

1. Họ và tên sinh viên:

2. Tên đề tài:

3. Nhận xét:

a) Những kết quả đạt được:

.....

.....

.....

.....

.....

b) Những hạn chế:

.....

.....

.....

.....

.....

4. **Điểm đánh giá** (theo thang điểm 10, làm tròn đến 0.5):

Sinh viên:

Điểm số: Điểm chữ:

TP. HCM, ngày ... tháng ... năm 20.....

Giảng viên chấm thi

(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN	1
CHƯƠNG 1 TỔNG QUAN	2
1.1. Lý do chọn đề tài	2
1.2. Mục tiêu nghiên cứu	2
1.3. Phạm vi nghiên cứu	2
CHƯƠNG 2 CƠ SỞ LÝ THUYẾT	3
2.1. VS Code	3
2.2. NodeJS	3
2.2.1. Giới thiệu	3
2.2.2. Asynchronous I/O	4
2.2.3. Event-Driven Architecture	5
2.3. Express	5
2.3.1. Middleware	5
2.3.2. Routing	5
2.4. Javascript	6
2.4.1. Giới thiệu và Lịch sử	6
2.4.2. Cú pháp cơ bản và Kiểu dữ liệu	7
2.5. HTML/EJS (Embedded JavaScript)	7
2.5.1. HTML	7
2.5.2. EJS (Embedded JavaScript)	8
2.6. CSS	9
2.7. Mongoose	9
2.8. MongoDB	10
2.8.1. Schemaless Design	10
2.8.2. Scalability	11
CHƯƠNG 3 THIẾT KẾ VÀ TRIỂN KHAI	12
3.1. Sơ đồ use case	12
3.1.2. Biểu đồ Use Case đăng nhập, đăng xuất	13

3.1.3. Biểu đồ Use Case Quản lý sản phẩm	14
3.1.4. Biểu đồ Use Case Case quản lý hóa đơn	14
3.2. Phân tích yêu cầu	15
3.2.1. Yêu cầu chức năng	15
3.2.2. Yêu cầu phi chức năng	16
3.2.3. Phân tích nhu cầu của người dùng	16
3.3. Thiết kế giao diện người dùng.....	17
3.3.1. Trang đăng nhập.....	17
3.3.2. Trang đăng ký	18
3.3.3. Trang chủ	19
3.3.4. Giỏ hàng.....	20
3.3.5. Trang quản trị thêm, xoá, sửa sản phẩm, hoá đơn mua hàng.	20
3.3.6. Trang danh sách users và xoá user.....	22
3.4. Hiện Thực Ứng Dụng.....	22
3.4.1. Controllers.....	22
3.4.2. Middleware	23
3.4.3. Models	24
3.4.4. Public.....	25
3.4.5. Routes	26
3.5.6. Utils.....	26
3.5.7. Views	27
CHƯƠNG 4 KẾT LUẬN	28
4.1. Tóm tắt kết quả	28
4.2. Đánh Giá và Hướng Phát Triển.....	28
4.3. Kết luận	29
TÀI LIỆU THAM KHẢO.....	30

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy Lê Huỳnh Phước về sự hỗ trợ và sự dẫn dắt trong suốt quá trình thực hiện tiểu luận của mình. Sự kiên nhẫn, tận tình và kiến thức sâu rộng của thầy đã giúp em vượt qua những thách thức và hoàn thành dự án một cách thành công.

Thầy đã luôn sẵn lòng chia sẻ kiến thức và kinh nghiệm của mình, giúp em hiểu sâu hơn về chủ đề và phát triển kỹ năng nghiên cứu. Sự động viên và hỗ trợ từ thầy đã giúp em vượt qua những khó khăn và tự tin hơn trong quá trình thực hiện dự án.

Em cũng muốn bày tỏ lòng biết ơn đến sự hướng dẫn và góp ý của thầy trong việc hoàn thiện tiểu luận của mình. Những nhận xét chân thành và xây dựng từ thầy đã giúp em cải thiện và hoàn thiện dự án của mình một cách tốt nhất.

Cuối cùng, em xin chân thành cảm ơn thầy vì sự tận tụy và tâm huyết trong việc giảng dạy và hướng dẫn chúng em. Mong rằng những điều tốt đẹp nhất sẽ đến với thầy và những dự án nghiên cứu sẽ luôn được thúc đẩy và phát triển.

CHƯƠNG 1 TỔNG QUAN

1.1. Lý do chọn đề tài

Điện thoại di động là một phần không thể thiếu trong cuộc sống hiện đại. Với sự phát triển của công nghệ, nhu cầu sử dụng và mua sắm điện thoại di động ngày càng tăng, tạo ra một thị trường tiềm năng cho việc phát triển một trang web bán hàng điện thoại. Việc kinh doanh trực tuyến, đặc biệt là trong lĩnh vực bán hàng điện thoại, mang lại nhiều cơ hội lợi nhuận. Một trang web bán điện thoại có thể tiếp cận được một lượng lớn khách hàng tiềm năng, không chỉ ở cấp độ địa phương mà còn ở mức độ quốc tế. Xây dựng một trang web bán điện thoại đòi hỏi sự kết hợp giữa các công nghệ phức tạp như front-end (React), back-end (Node.js, Express), cơ sở dữ liệu (MongoDB) và các công nghệ khác như Redux, middleware, routing, và quản lý phiên bản. Việc nắm vững và áp dụng những kiến thức này không chỉ là một thách thức mà còn là cơ hội để phát triển kỹ năng lập trình và quản lý dự án.

1.2. Mục tiêu nghiên cứu

Phát triển một ứng dụng web hoàn chỉnh: Xây dựng một trang web bán điện thoại đáp ứng đầy đủ các yêu cầu của một ứng dụng thương mại điện tử, bao gồm giao diện người dùng thân thiện, tính năng tìm kiếm và lọc sản phẩm, giỏ hàng và thanh toán trực tuyến. Nắm vững các công nghệ liên quan: Hiểu rõ về các công nghệ phổ biến được sử dụng trong phát triển web như ReactJS cho phần front-end, Node.js và Express cho phần back-end, MongoDB cho cơ sở dữ liệu, Redux cho quản lý trạng thái ứng dụng, và các công nghệ khác như HTML, CSS, JavaScript.

Tối ưu hóa hiệu suất: Tối ưu hóa hiệu suất của trang web bằng cách sử dụng các kỹ thuật như tải trang không đồng bộ, cache dữ liệu, và tối ưu hóa truy vấn cơ sở dữ liệu để cải thiện trải nghiệm người dùng.

Bảo mật thông tin: Áp dụng các biện pháp bảo mật như mã hóa mật khẩu, xác thực người dùng, và bảo vệ chống lại các cuộc tấn công phổ biến như SQL injection và cross-site scripting (XSS).

Kiểm thử và đánh giá: Thực hiện các bước kiểm thử chất lượng phần mềm để đảm bảo tính ổn định và đáng tin cậy của ứng dụng, bao gồm kiểm thử đơn vị, kiểm thử tích hợp, và kiểm thử hệ thống. Đồng thời, đánh giá sự hài lòng của người dùng và tiến hành cải tiến dựa trên phản hồi.

1.3. Phạm vi nghiên cứu

Xây dựng một trang web hoàn chỉnh với các tính năng cơ bản của một cửa hàng trực tuyến bao gồm hiển thị sản phẩm, tìm kiếm, xem chi tiết sản phẩm, thêm vào giỏ hàng, thanh toán và quản lý đơn hàng.

Xây dựng trang web có khả năng tương thích trên nhiều thiết bị và nền tảng khác nhau như máy tính, điện thoại di động và máy tính bảng.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1. VS Code

Visual Studio Code (VS Code) là một trình biên tập mã nguồn mở và miễn phí, được phát triển bởi Microsoft. Đây là một công cụ mạnh mẽ cho phát triển ứng dụng web và ứng dụng di động, được ưa chuộng bởi các nhà phát triển trên khắp thế giới.

Một số điểm nổi bật của VS Code:

- Giao diện đơn giản và dễ sử dụng: VS Code có giao diện sạch sẽ, tối ưu hóa cho trải nghiệm người dùng tốt nhất. Các tính năng được tổ chức rõ ràng, giúp người dùng dễ dàng truy cập và sử dụng.
- Hỗ trợ nhiều ngôn ngữ lập trình: VS Code hỗ trợ nhiều ngôn ngữ lập trình phổ biến như JavaScript, TypeScript, Python, Java, C++, và nhiều ngôn ngữ khác. Điều này giúp cho các nhà phát triển có thể làm việc trên nhiều dự án khác nhau mà không cần chuyển đổi công cụ.
- Mở rộng linh hoạt: VS Code có một hệ thống mở rộng phong phú, cho phép người dùng tùy chỉnh và mở rộng chức năng của trình biên tập theo nhu cầu cụ thể của họ. Các extension cung cấp các tính năng như kiểm tra lỗi, quản lý git, môi trường phát triển tích hợp, và nhiều hơn nữa.
- Hỗ trợ Git tích hợp: VS Code tích hợp sẵn với Git, cho phép người dùng quản lý mã nguồn, commit, push, pull, và xem lịch sử thay đổi từ bên trong trình biên tập một cách thuận tiện.
- Tích hợp gỡ lỗi mạnh mẽ: VS Code cung cấp một bộ công cụ gỡ lỗi mạnh mẽ cho nhiều ngôn ngữ lập trình, giúp các nhà phát triển tìm ra và sửa lỗi một cách nhanh chóng và hiệu quả.
- Hỗ trợ cộng đồng lớn: VS Code được sử dụng rộng rãi và có một cộng đồng lớn của các nhà phát triển trên toàn thế giới. Điều này có nghĩa là có nhiều tài liệu, hướng dẫn, và các extension được phát triển bởi cộng đồng mà người dùng có thể tận dụng.

2.2. NodeJS

2.2.1. Giới thiệu

Node.js là một nền tảng phát triển ứng dụng máy chủ (server-side) được xây dựng trên JavaScript, mục đích ban đầu để xử lý các yêu cầu HTTP và tạo ra các ứng dụng web hiệu suất cao.

Một số điểm về Node.js:

- JavaScript Everywhere: Node.js cho phép bạn sử dụng JavaScript cả trên máy chủ và trình duyệt web. Điều này giúp đơn giản hóa quá trình phát triển và tạo ra sự nhất quán giữa mã nguồn trên cả hai phía.
- Asynchronous and Event-Driven: Node.js được xây dựng trên mô hình không đồng bộ (asynchronous) và dựa trên sự kiện (event-driven), cho phép xử lý nhiều yêu cầu một cách hiệu quả mà không cần tạo ra các tiến trình hoặc luồng mới.
- Single-Threaded: Node.js sử dụng mô hình xử lý sự kiện đơn luồng (single-threaded), nhưng vẫn có thể xử lý nhiều yêu cầu cùng một lúc thông qua việc sử dụng bất đồng bộ I/O và các callback.
- NPM (Node Package Manager): NPM là một trong những cộng đồng thư viện mã nguồn mở lớn nhất trên thế giới, cung cấp hàng ngàn gói thư viện và công cụ hữu ích cho việc phát triển ứng dụng Node.js.
- Phổ biến và Mạnh mẽ: Node.js là một trong những công nghệ phát triển máy chủ phổ biến nhất hiện nay, được sử dụng rộng rãi trong các ứng dụng web, ứng dụng di động, IoT, và nhiều lĩnh vực khác.
- Hỗ trợ cộng đồng lớn: Node.js có một cộng đồng lớn và tích cực, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ cho các nhà phát triển. Điều này giúp giảm thiểu thời gian phát triển và giải quyết vấn đề nhanh chóng.

2.2.2. Asynchronous I/O

Asynchronous I/O là một khái niệm trong lập trình mạng và hệ thống máy tính, đặc biệt là trong các ngôn ngữ lập trình như JavaScript, Node.js, Python, và Java. Đây là một kỹ thuật được sử dụng để xử lý đa nhiệm mà không cần chờ đợi các hoạt động I/O hoàn thành trước khi chuyển sang các công việc khác.

Trong mô hình đồng bộ, khi một chương trình thực hiện một hoạt động I/O (ví dụ: đọc dữ liệu từ một tập tin hoặc gửi yêu cầu HTTP), nó sẽ chờ đợi cho đến khi hoạt động này hoàn thành trước khi tiếp tục thực hiện các công việc khác. Điều này có thể dẫn đến việc lãng phí thời gian chờ đợi, đặc biệt khi có nhiều yêu cầu I/O cần được xử lý.

Trái lại, trong mô hình không đồng bộ, khi một yêu cầu I/O được gửi đi, chương trình sẽ tiếp tục thực hiện các công việc khác trong khi đợi phản hồi từ hoạt động I/O. Khi hoạt động I/O hoàn thành, chương trình sẽ nhận được một sự kiện hoặc callback để xử lý kết quả.

Node.js là một ví dụ điển hình về việc sử dụng asynchronous I/O. Bằng cách sử dụng các hàm callback và sự kiện, Node.js cho phép xử lý nhiều yêu cầu mà không cần tạo ra các luồng hoặc tiến trình mới, giúp tăng hiệu suất và sử dụng tài nguyên hiệu quả.

2.2.3. Event-Driven Architecture

Kiến trúc Event-Driven (ED) là một kiểu kiến trúc phần mềm mà các thành phần chính trong hệ thống hoạt động dựa trên việc phát sinh và xử lý sự kiện. Trong kiến trúc này, các thành phần của hệ thống không giao tiếp trực tiếp với nhau mà thay vào đó thông qua việc gửi và nhận các sự kiện.

Một số khái niệm quan trọng trong kiến trúc Event-Driven bao gồm:

- Sự kiện (Event): Đây là một sự kiện xảy ra trong hệ thống, như một thao tác người dùng, một tác vụ hoặc một trạng thái thay đổi.
- Bộ phát (Emitter): Đây là thành phần tạo ra sự kiện và thông báo cho các thành phần khác về việc sự kiện đã xảy ra.
- Bộ xử lý (Handler): Đây là các hàm hoặc phương thức được gọi khi một sự kiện xảy ra, để xử lý sự kiện đó.
- Luồng sự kiện (Event Loop): Đây là một cơ chế quản lý và phân phối sự kiện trong hệ thống. Nó kiểm tra xem có sự kiện nào đã xảy ra và gọi các bộ xử lý tương ứng.
- Kênh (Channel): Đây là phương tiện để truyền sự kiện từ bộ phát đến các bộ xử lý.

2.3. Express

2.3.1. Middleware

Middleware là một khái niệm quan trọng trong các ứng dụng web, đặc biệt là trong Node.js và Express.js. Middleware là các hàm hoặc các tầng logic được thực thi tuần tự trước hoặc sau khi các yêu cầu HTTP được xử lý bởi các route handler chính trong ứng dụng.

Các middleware có thể thực hiện các nhiệm vụ như xác thực người dùng, xử lý dữ liệu yêu cầu, kiểm soát quyền truy cập, ghi log, nén dữ liệu, xử lý lỗi, và nhiều tác vụ khác.

Trong Express.js, một middleware có thể là một hàm đơn giản hoặc một chuỗi các hàm được gọi liên tiếp. Mỗi middleware nhận ba tham số: req (request), res (response), và next. Middleware có thể hoàn thành công việc của nó bằng cách gọi hàm next() để chuyển quyền điều khiển sang middleware tiếp theo trong chuỗi hoặc gọi hàm res.send() để kết thúc chuỗi middleware.

Sử dụng middleware có thể giúp tách biệt logic ứng dụng thành các phần nhỏ, dễ quản lý, và có thể tái sử dụng. Điều này giúp tăng tính linh hoạt và bảo trì của mã nguồn.

2.3.2. Routing

Routing là quá trình xác định cách xử lý các yêu cầu HTTP từ các client và phản hồi lại chúng trong các ứng dụng web. Trong ngữ cảnh của Node.js và Express.js, routing thường được thực hiện thông qua việc định nghĩa các endpoint và xác định các hành động sẽ được thực hiện khi các yêu cầu gửi đến các endpoint đó.

Express.js cung cấp một cách tiện lợi để định nghĩa các route thông qua việc sử dụng các phương thức HTTP như GET, POST, PUT, DELETE, vv. Mỗi route có thể được kết nối với một hoặc nhiều middleware và một hàm xử lý chính (route handler) để xử lý yêu cầu và tạo ra phản hồi.

Routing cho phép chúng ta tổ chức và quản lý các endpoint của ứng dụng web một cách hiệu quả, giúp tạo ra các ứng dụng có cấu trúc và dễ bảo trì.

2.4. Javascript

2.4.1. Giới thiệu và Lịch sử

JavaScript là một ngôn ngữ lập trình phổ biến được sử dụng phía client-side và server-side để tạo ra các ứng dụng web động. Dưới đây là một số điểm quan trọng về giới thiệu và lịch sử của JavaScript:

Giới Thiệu

- JavaScript (JS) là một ngôn ngữ lập trình được phát triển bởi Brendan Eich của Netscape Communications Corporation vào năm 1995.
- Ban đầu, JavaScript được thiết kế để tạo ra các hiệu ứng động và thêm tính tương tác vào các trang web.
- Ngày nay, JavaScript không chỉ được sử dụng cho phía client-side mà còn được sử dụng rộng rãi cho phát triển server-side thông qua các framework như Node.js.

Lịch sử

- JavaScript được phát triển trong một thời điểm mà Netscape đang cố gắng thách thức sự thống trị của Microsoft với trình duyệt Netscape Navigator.
- Ngày 4 tháng 12 năm 1995, JavaScript lần đầu tiên được giới thiệu với tên gọi "LiveScript" trong Netscape Navigator 2.0 Beta 3.
- Sau đó, tên của ngôn ngữ này được đổi thành "JavaScript" để tận dụng sức hút từ ngôn ngữ lập trình đang thịnh hành là Java.
- JavaScript đã trải qua nhiều phiên bản và cập nhật từng bước, với sự phát triển của cộng đồng lập trình viên và việc chuẩn hóa bởi tổ chức ECMA International thông qua chuẩn ECMA-262.
- ES6 (ECMAScript 2015) là một bước tiến quan trọng trong lịch sử JavaScript, cung cấp nhiều tính năng mới và cải tiến cho ngôn ngữ.

2.4.2. Cú pháp cơ bản và Kiểu dữ liệu

Cú pháp cơ bản:

- JavaScript là một ngôn ngữ lập trình dựa trên văn bản, nơi mà mã được viết trong các tệp văn bản và thực thi trong môi trường trình duyệt web hoặc môi trường Node.js.
- Cú pháp JavaScript linh hoạt và dễ hiểu, với khả năng khai báo biến, khai báo hàm, và sử dụng các cấu trúc điều khiển như if, else, for, while, vv.

Khai báo biến:

- Sử dụng từ khóa var, let, hoặc const để khai báo biến.
- var được sử dụng để khai báo biến có phạm vi toàn cục hoặc hàm.
- let được sử dụng để khai báo biến có phạm vi block.
- const được sử dụng để khai báo hằng số, không thể gán lại giá trị mới.

Kiểu dữ liệu:

- JavaScript là một ngôn ngữ lập trình động, nghĩa là kiểu dữ liệu của biến có thể thay đổi trong quá trình thực thi chương trình.
- Các kiểu dữ liệu cơ bản trong JavaScript bao gồm: number (số), string (chuỗi), boolean (luận lý), undefined, null, object (đối tượng), symbol (ký hiệu - ES6).
- JavaScript cũng hỗ trợ các kiểu dữ liệu phức tạp như mảng (array) và đối tượng (object), cho phép lưu trữ dữ liệu phức tạp.

Toán tử:

- JavaScript hỗ trợ nhiều loại toán tử bao gồm toán tử số học (+, -, *, /), toán tử gán (=), toán tử so sánh (==, ===, !=, !==), toán tử logic (&&, ||, !), vv.
- Toán tử cũng có thể áp dụng cho các kiểu dữ liệu khác nhau, cho phép thực hiện các phép tính và so sánh giữa chúng.

2.5. HTML/EJS (Embedded JavaScript)

2.5.1. HTML

HTML (Hypertext Markup Language) là ngôn ngữ đánh dấu tiêu chuẩn được sử dụng để tạo cấu trúc và hiển thị các trang web trên Internet.

Cấu trúc cơ bản của HTML:

- Thẻ (Tag): HTML sử dụng các thẻ để xác định cấu trúc của trang web. Mỗi thẻ bao gồm một từ khóa được đặt trong dấu ngoặc nhọn < >. Ví dụ: <html>, <head>, <body>.
- Phần tử (Element): Một phần tử HTML bao gồm một cặp thẻ mở và thẻ đóng, cùng với nội dung bên trong. Ví dụ: <p>Đây là một đoạn văn bản.</p>.

- Thuộc tính (Attribute): Một số thẻ HTML có thể chứa các thuộc tính để cung cấp thông tin bổ sung về phần tử. Ví dụ: ``.
- Chú thích (Comment): Chú thích trong HTML bắt đầu bằng `<!--` và kết thúc bằng `-->`. Chú thích không được hiển thị trên trình duyệt. Ví dụ: `<!-- Đây là một chú thích -->`.

Cấu trúc cơ bản của một trang HTML:

- Thẻ `<html>`: Đây là thẻ gốc của mọi tài liệu HTML và chứa toàn bộ nội dung của trang web.
- Thẻ `<head>`: Chứa các thẻ mô tả thông tin về tài liệu HTML, như tiêu đề, siêu mô tả, và các tập tin CSS hoặc JavaScript.
- Thẻ `<body>`: Chứa nội dung hiển thị của trang web, bao gồm văn bản, hình ảnh, biểu mẫu và các phần tử khác.

2.5.2. EJS (*Embedded JavaScript*)

EJS (Embedded JavaScript) là một ngôn ngữ mẫu được sử dụng trong Node.js và các ứng dụng web để tạo ra các trang HTML động.

Đặc điểm của EJS:

- Nhúng mã JavaScript: EJS cho phép nhúng mã JavaScript trực tiếp vào các trang HTML, giúp tạo ra các trang động và linh hoạt.
- Sử dụng các biểu thức: EJS hỗ trợ sử dụng các biểu thức JavaScript trong HTML để hiển thị dữ liệu động và thực hiện các phép tính.
- Tạo và sử dụng các mẫu: EJS cho phép tạo ra các mẫu (templates) để tái sử dụng mã HTML và JavaScript trên nhiều trang.
- Dễ dàng tích hợp với Node.js: EJS là một trong những công cụ mẫu phổ biến trong cộng đồng Node.js và dễ dàng tích hợp vào các ứng dụng Node.js.

Cú pháp cơ bản của EJS:

- Nhúng mã JavaScript: Để nhúng mã JavaScript vào trang EJS, bạn sử dụng cặp dấu `<% %>` hoặc `<%= %>`. Ví dụ: `<% if (user) { %> Hello <%= user.name %> <% } %>`.
- Vòng lặp và câu điều kiện: EJS hỗ trợ sử dụng vòng lặp và câu điều kiện để xử lý dữ liệu và hiển thị nội dung động.
- In các giá trị từ dữ liệu: Để in các giá trị từ dữ liệu, bạn sử dụng cú pháp `<%= %>` hoặc `<%- %>`. Ví dụ: `<p>Tên của bạn là: <%= user.name %></p>`.
- Bao gồm các mẫu: EJS cho phép bao gồm các mẫu khác trong một mẫu bằng cú pháp `<%- include('header') %>`.

2.6. CSS

CSS (Cascading Style Sheets) là ngôn ngữ được sử dụng để mô tả cách mà các phần tử HTML được hiển thị trên trình duyệt

Đặc điểm của CSS:

- Phong cách đồng nhất: CSS giúp tạo ra phong cách đồng nhất trên nhiều trang HTML trong một trang web hoặc trên nhiều trang web khác nhau.
- Tách biệt kiểu dáng và nội dung: CSS tách biệt hoàn toàn kiểu dáng (style) của trang web từ nội dung HTML, giúp tạo ra mã HTML sạch sẽ và dễ bảo trì.
- Lập trình linh hoạt: CSS cho phép lập trình viên điều chỉnh giao diện của trang web một cách linh hoạt và dễ dàng thông qua các quy tắc, lớp và id.
- Kết hợp với HTML và JavaScript: CSS có thể kết hợp với HTML để tạo ra giao diện đẹp mắt và tương tác với JavaScript để tạo ra các hiệu ứng động.

Cú pháp cơ bản của CSS:

- Chọn phần tử HTML: Để chọn một phần tử HTML, bạn sử dụng tên phần tử (ví dụ: p, div, span) hoặc lớp (class) hoặc id của phần tử.
- Thiết lập thuộc tính: Để thiết lập các thuộc tính cho phần tử, bạn sử dụng cú pháp tên-thuộc-tính: giá-trị;. Ví dụ: color: blue; font-size: 16px;.
- Kết hợp các lựa chọn: Bạn có thể kết hợp nhiều lựa chọn lại với nhau bằng dấu cách. Ví dụ: h1, h2, h3 { color: red; } sẽ áp dụng màu đỏ cho các tiêu đề h1, h2, và h3.

2.7. Mongoose

Mongoose là một thư viện JavaScript được sử dụng trong môi trường Node.js để tương tác với cơ sở dữ liệu MongoDB dễ dàng hơn.

Nó cho phép định nghĩa các mô hình dữ liệu (data models) và tạo ra các tương tác giữa ứng dụng Node.js và cơ sở dữ liệu MongoDB thông qua các phương thức như tìm kiếm, thêm, cập nhật và xóa dữ liệu.

Các đặc điểm của Mongoose:

- Kiểu dữ liệu đa dạng: Mongoose hỗ trợ nhiều kiểu dữ liệu như chuỗi (String), số (Number), mảng (Array), đối tượng (Object), v.v.
- Kiểm tra dữ liệu: Mongoose cho phép bạn định nghĩa các quy tắc kiểm tra dữ liệu (data validation) để đảm bảo dữ liệu được lưu vào cơ sở dữ liệu tuân thủ các điều kiện nhất định.
- Mô hình dữ liệu: Bằng cách sử dụng Mongoose, bạn có thể định nghĩa các mô hình dữ liệu cho các tài nguyên trong ứng dụng của mình, giúp quản lý dữ liệu một cách hiệu quả và dễ dàng.

- Quan hệ giữa dữ liệu: Mongoose hỗ trợ các loại quan hệ giữa các mô hình dữ liệu như một-một (one-to-one), một-nhiều (one-to-many), nhiều-một (many-to-one) và nhiều-nhiều (many-to-many).
- Tương tác với MongoDB: Mongoose cung cấp các phương thức để tương tác với cơ sở dữ liệu MongoDB như tìm kiếm, thêm, cập nhật và xóa dữ liệu một cách dễ dàng và linh hoạt.

Thực hiện trong mã Node.js:

- Để sử dụng Mongoose trong mã Node.js, bạn cần cài đặt thư viện Mongoose thông qua npm hoặc yarn.
- Sau đó, bạn có thể import Mongoose vào các tệp mã của mình và bắt đầu định nghĩa các mô hình dữ liệu, kết nối với cơ sở dữ liệu MongoDB và thực hiện các tương tác với dữ liệu.

2.8. MongoDB

2.8.1. Schemaless Design

Schemaless Design là một phương pháp thiết kế cơ sở dữ liệu không yêu cầu một cấu trúc cố định hoặc một schema được xác định trước. Trong Schemaless Design, dữ liệu được lưu trữ dưới dạng các tài liệu có thể thay đổi cấu trúc một cách linh hoạt, không cần phải tuân theo một schema cụ thể.

Đặc điểm chính của Schemaless Design bao gồm:

- Tính linh hoạt: Khả năng thêm, sửa đổi hoặc xóa các trường dữ liệu mà không cần phải thay đổi schema cơ sở dữ liệu. Điều này cho phép ứng dụng thích ứng nhanh chóng với các yêu cầu thay đổi và phát triển.
- Tính mở rộng: Dễ dàng mở rộng cơ sở dữ liệu để đáp ứng với lưu lượng dữ liệu tăng lên mà không gặp phải các ràng buộc về schema.
- Thiết kế dữ liệu theo nhu cầu: Dữ liệu được tổ chức dưới dạng các tài liệu độc lập, mỗi tài liệu có thể có cấu trúc riêng, phản ánh nhu cầu cụ thể của ứng dụng hoặc ngữ cảnh.
- Tính phân tán: Schemaless Design thích hợp cho môi trường phân tán, nơi mà các dữ liệu có thể được phân phối trên nhiều nút và các nút có thể thêm hoặc xóa các trường dữ liệu mà không cần phải đồng bộ hóa với các nút khác.
- Khó khăn trong truy vấn: Do không có schema cố định, việc truy vấn dữ liệu có thể trở nên phức tạp hơn và đòi hỏi sử dụng các công cụ truy vấn linh hoạt như MongoDB's aggregation framework hoặc Elasticsearch.
- Quản lý dữ liệu: Việc quản lý dữ liệu có thể trở nên khó khăn hơn vì không có sự hỗ trợ của schema để đảm bảo tính nhất quán và kiểm soát dữ liệu.

Schemaless Design thường được sử dụng trong các ứng dụng yêu cầu tính linh hoạt và khả năng mở rộng cao, như các ứng dụng web, dịch vụ phân tích dữ liệu và các ứng dụng IoT. Tuy nhiên, việc sử dụng Schemaless Design cũng đòi hỏi một quy trình quản lý dữ liệu cẩn thận để đảm bảo tính nhất quán và hiệu suất của hệ thống.

2.8.2. Scalability

Scalability, hay khả năng mở rộng, là khả năng của hệ thống hoặc ứng dụng để xử lý một lượng lớn công việc hoặc tải trọng mà không ảnh hưởng đến hiệu suất và khả năng phản hồi. Trong ngữ cảnh của ứng dụng web và dịch vụ, Scalability thường ám chỉ đến khả năng mở rộng ngang (horizontal scalability) và khả năng mở rộng dọc (vertical scalability).

Khả năng mở rộng ngang (Horizontal Scalability): Đây là khả năng của hệ thống để xử lý tải trọng bằng cách thêm các thiết bị hoặc tài nguyên mới vào hệ thống, thay vì nâng cấp hoặc mở rộng các thành phần hiện có. Trong mô hình này, công việc được phân tán trên nhiều máy chủ hoặc nút, giúp chia nhỏ tải trọng và tăng khả năng chịu tải của hệ thống. Ví dụ, có thể thêm máy chủ web mới vào cụm máy chủ hoặc triển khai các container Docker để mở rộng ứng dụng.

Khả năng mở rộng dọc (Vertical Scalability): Đây là khả năng của hệ thống để xử lý tải trọng bằng cách nâng cấp hoặc mở rộng tài nguyên hiện có trên một máy chủ hoặc nút. Ví dụ, có thể nâng cấp CPU, bộ nhớ RAM hoặc ổ cứng trên một máy chủ để tăng khả năng xử lý và lưu trữ của hệ thống.

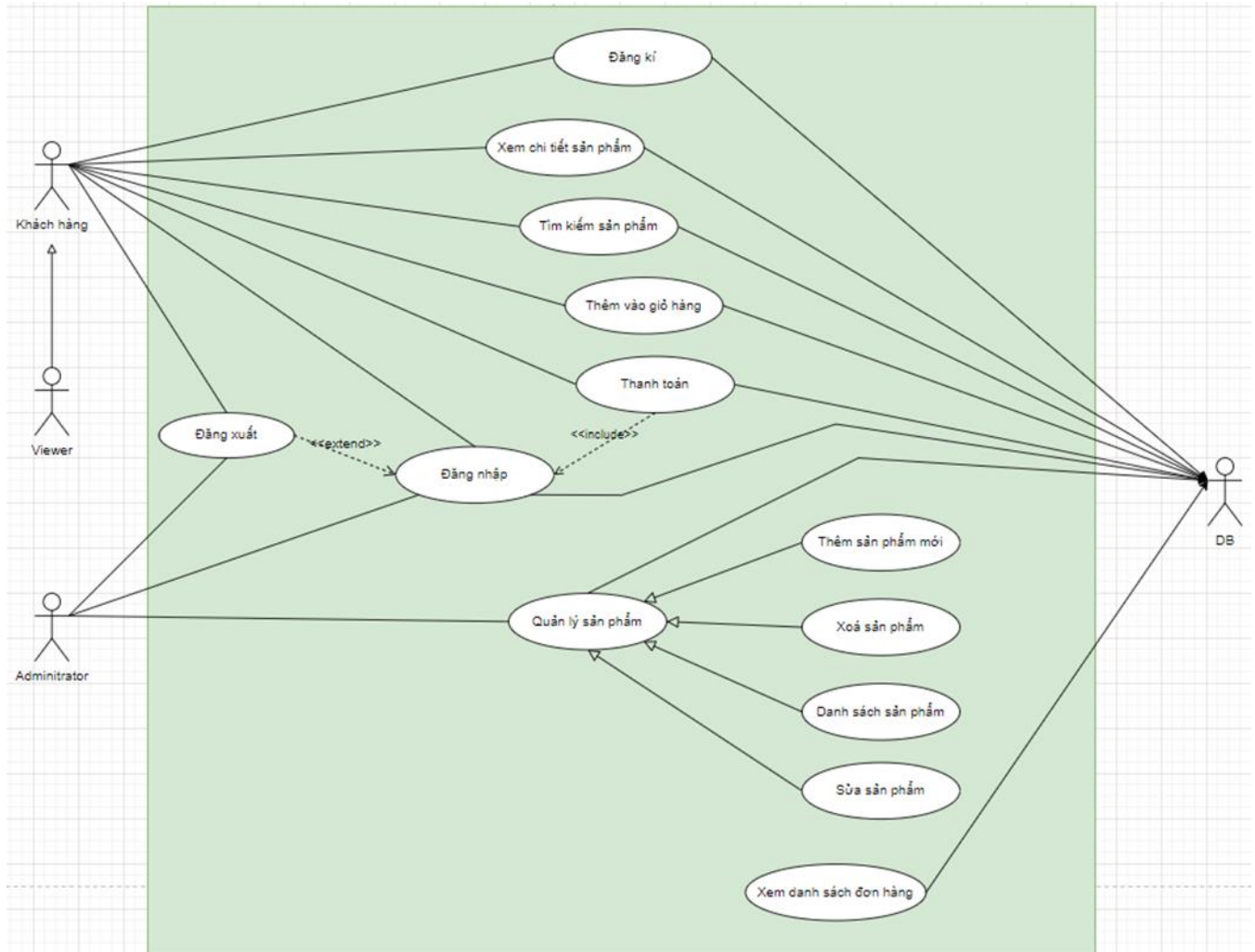
Các chiến lược Scalability phổ biến bao gồm:

- **Load Balancing:** Phân phối tải trọng giữa các máy chủ hoặc nút để đảm bảo rằng không có máy chủ nào quá tải và tất cả đều hoạt động ổn định.
- **Caching:** Lưu trữ các kết quả truy vấn hoặc dữ liệu phổ biến trong bộ nhớ cache để giảm thiểu việc truy cập vào cơ sở dữ liệu và tăng tốc độ phản hồi của hệ thống.
- **Asynchronous Processing:** Sử dụng các công nghệ và mô hình xử lý không đồng bộ như message queues hoặc event-driven architecture để xử lý các tác vụ phía sau và giảm thời gian chờ đợi của người dùng.
- **Microservices:** Phân tách ứng dụng thành các dịch vụ nhỏ, độc lập nhau để dễ dàng mở rộng và duy trì.

Scalability là một yếu tố quan trọng trong việc phát triển các ứng dụng web và dịch vụ, đặc biệt là trong môi trường có yêu cầu về sự linh hoạt và khả năng mở rộng cao.

CHƯƠNG 3 THIẾT KẾ VÀ TRIỂN KHAI

3.1. Sơ đồ use case



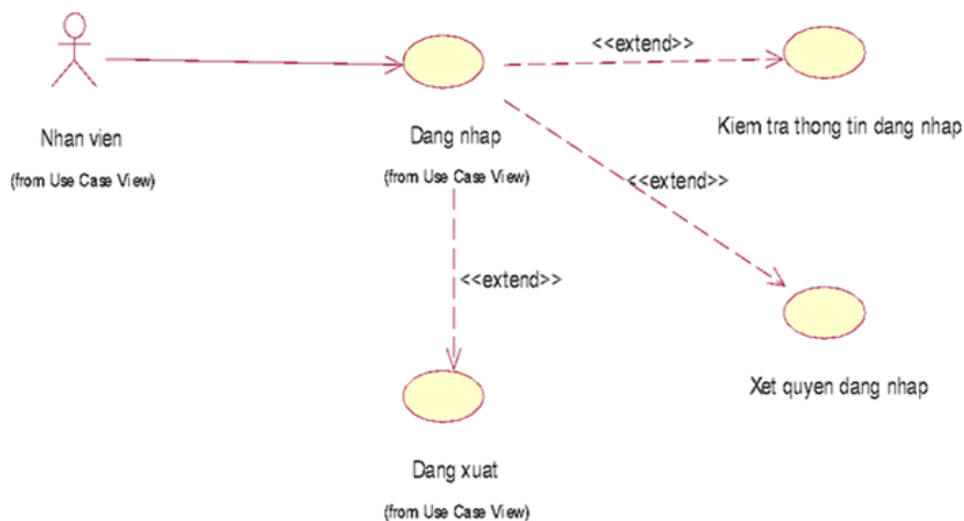
Hình 3.1 Sơ đồ use case tổng quát

3.1.1. Xác định use case

Dựa vào yêu cầu bài toán, ta có các actor sau: Quản trị viên (Admin), người xem (Viewer) và Khách hàng (Customer)

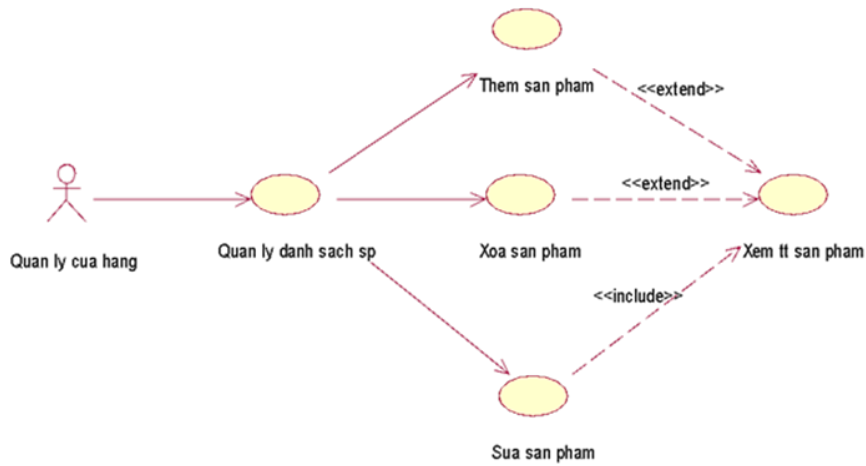
Actor	Use Case
Quản trị viên (Admin)	Đăng nhập, quản lý thông tin sản phẩm, quản lý loại sản phẩm, quản lý đơn hàng.
Người xem (Viewer)	Đăng ký tài khoản, xem thông tin sản phẩm, tìm kiếm sản phẩm, quản lý giỏ hàng
Khách hàng (Customer)	Đăng nhập, xem thông tin sản phẩm, tìm kiếm sản phẩm, quản lý giỏ hàng, thanh toán.

3.1.2. Biểu đồ Use Case đăng nhập, đăng xuất



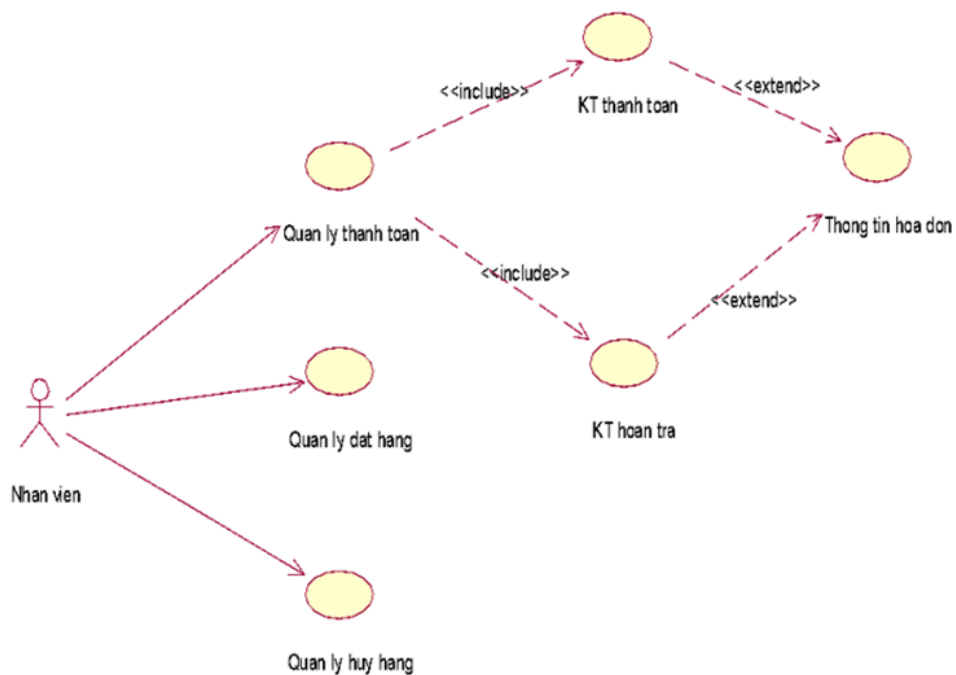
Hình 3.2 Sơ đồ Use Case đăng nhập, đăng xuất

3.1.3. Biểu đồ Use Case Quản lý sản phẩm



Hình 3.3. Sơ đồ Use Case quản lý danh mục sản phẩm

3.1.4. Biểu đồ Use Case Case quản lý hóa đơn



Hình 3.4 Sơ đồ Use Case quản lý hóa đơn

3.2. Phân tích yêu cầu

Chức năng của ứng dụng: các tính năng cần có của ứng dụng, bao gồm quản lý sản phẩm, giỏ hàng, đăng nhập, đăng ký, thanh toán, và xem hóa đơn.

Yêu cầu về dữ liệu: Xác định loại dữ liệu cần thu thập và lưu trữ, bao gồm thông tin về sản phẩm, người dùng, giỏ hàng, và hóa đơn.

Giao diện người dùng: yêu cầu về giao diện người dùng, bao gồm các màn hình, nút bấm, trường nhập liệu, và cách tương tác giữa người dùng và ứng dụng.

Hiệu suất và bảo mật: Xác định các yêu cầu về hiệu suất của hệ thống, bao gồm thời gian phản hồi, tải trọng dữ liệu, và bảo mật thông tin cá nhân của người dùng.

Phân quyền truy cập: Đặt ra các yêu cầu về phân quyền truy cập để đảm bảo chỉ những người được phép mới có thể truy cập và thực hiện các chức năng cụ thể trong ứng dụng.

3.2.1. Yêu cầu chức năng

Quản lý sản phẩm:

- Hiển thị danh sách sản phẩm.
- Tìm kiếm sản phẩm theo tên, nhà sản xuất, hoặc danh mục.
- Xem chi tiết sản phẩm với mô tả, hình ảnh, giá cả, và thông số kỹ thuật.
- Thêm sản phẩm vào giỏ hàng từ trang chi tiết sản phẩm.

Quản lý giỏ hàng:

- Hiển thị giỏ hàng với danh sách sản phẩm và tổng giá.
- Thay đổi số lượng sản phẩm trong giỏ hàng hoặc xóa sản phẩm khỏi giỏ hàng.
- Tính tổng giá tiền của các sản phẩm trong giỏ hàng.

Quản lý đơn hàng:

- Tạo đơn hàng mới sau khi thanh toán thành công.
- Hiển thị lịch sử đơn hàng của người dùng.
- Xem chi tiết đơn hàng bao gồm thông tin về sản phẩm, số lượng, và tổng giá.

Đăng nhập và đăng ký:

- Cho phép người dùng đăng nhập vào tài khoản hiện có hoặc đăng ký tài khoản mới.
- Xác thực thông tin người dùng và bảo mật mật khẩu.

3.2.2. Yêu cầu phi chức năng

Hiệu suất:

- Ứng dụng phải có thời gian phản hồi nhanh chóng, giảm thiểu thời gian chờ đợi của người dùng.
- Xử lý dữ liệu một cách hiệu quả để tránh trễ hoặc giảm bị treo ứng dụng.

Bảo mật:

- Bảo vệ thông tin cá nhân của người dùng bằng cách mã hóa dữ liệu và cung cấp cơ chế xác thực an toàn.
- Ngăn chặn các cuộc tấn công từ bên ngoài bằng cách triển khai các biện pháp bảo mật phù hợp.

Khả năng mở rộng:

- Thiết kế ứng dụng để dễ dàng mở rộng và thích ứng với việc thêm tính năng hoặc cải thiện sau này.
- Sử dụng kiến trúc linh hoạt và các công nghệ có thể mở rộng để đáp ứng nhu cầu tăng trưởng của ứng dụng.

3.2.3. Phân tích nhu cầu của người dùng

Nhu cầu sản phẩm:

- Người dùng muốn có quy trình mua sắm thuận tiện và nhanh chóng cho việc mua điện thoại di động.
- Họ muốn có thể tìm kiếm, so sánh và xem các sản phẩm từ nhiều nhà sản xuất và thương hiệu khác nhau.

Nhu cầu giao diện người dùng

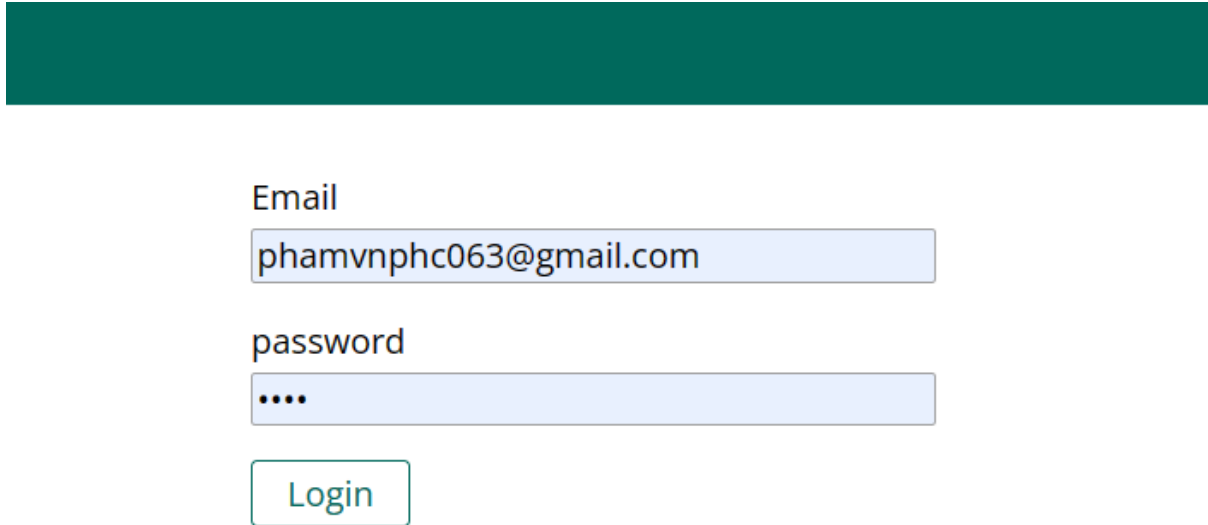
- Người dùng mong đợi một giao diện người dùng thân thiện, dễ sử dụng và dễ điều hướng.
- Họ muốn có trải nghiệm mua sắm trực tuyến mượt mà và thú vị trên các thiết bị di động của mình.

Nhu cầu bảo mật và an toàn:

- Người dùng quan tâm đến việc bảo vệ thông tin cá nhân và tài khoản của họ khi thực hiện thanh toán trực tuyến.
- Họ mong đợi ứng dụng cung cấp các biện pháp bảo mật để ngăn chặn việc truy cập trái phép vào thông tin cá nhân hoặc tài khoản của họ.

3.3. Thiết kế giao diện người dùng

3.3.1. Trang đăng nhập



The image shows a login form interface. At the top, there is a solid dark green horizontal bar. Below this bar, the form consists of three main components: an email input field, a password input field, and a login button. The email input field is labeled 'Email' and contains the text 'phamvnphc063@gmail.com'. The password input field is labeled 'password' and contains four dots '....'. The login button is a rectangular button with a green border and the text 'Login'.

Email

phamvnphc063@gmail.com

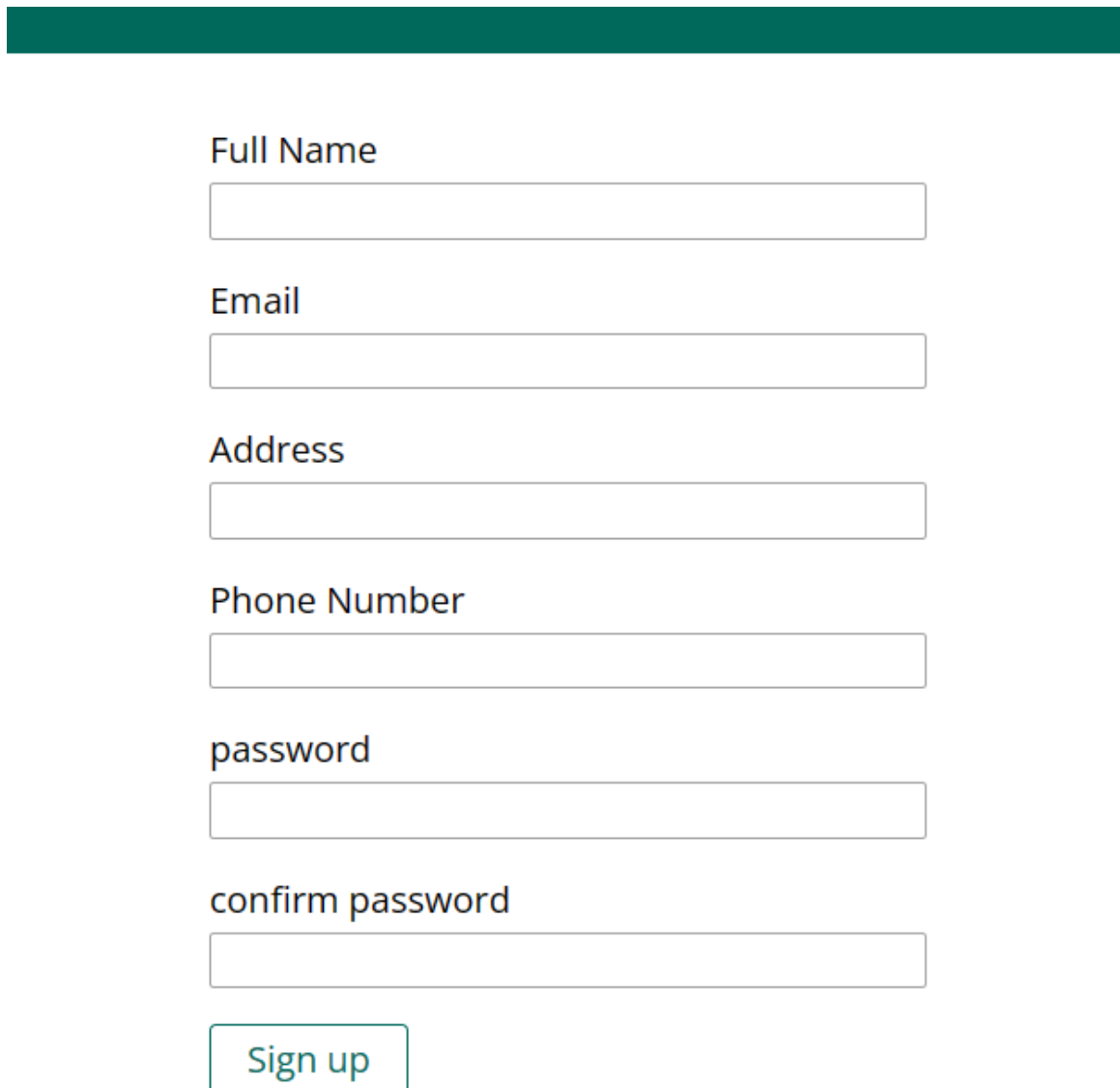
password

....

Login

Hình 3.5. Giao diện đăng nhập

3.3.2. Trang đăng ký



The registration form is displayed on a white background with a dark teal header bar at the top. The form consists of several input fields and a button, all aligned to the left. The fields are labeled 'Full Name', 'Email', 'Address', 'Phone Number', 'password', and 'confirm password'. Each label is followed by a rectangular input box. The 'password' and 'confirm password' fields are visually linked by a thin teal line. A teal 'Sign up' button is positioned below the 'confirm password' field.

Full Name

Email

Address

Phone Number

password


confirm password

Sign up

Hình 3.6. Giao diện trang đăng ký


3.3.3. Trang chủ









[Shop](#) [Cart](#) [Orders](#) [Admin](#) [Logout](#)

 iPhone 11

Just the right amount of everything.

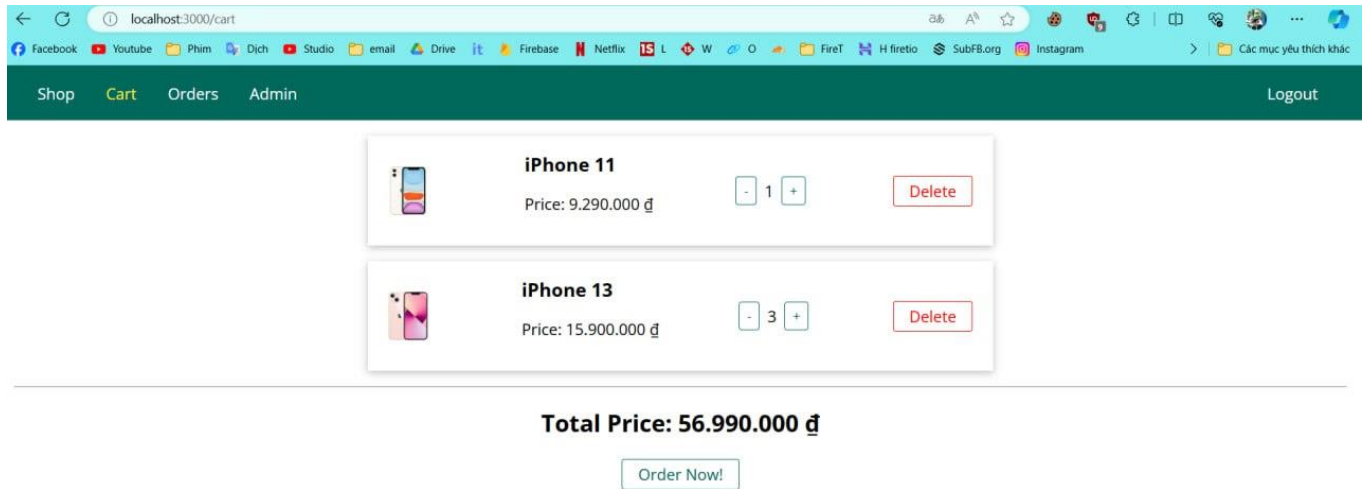
New dual-camera system. All-day battery.¹ The toughest glass in a smartphone. And Apple's fastest chip ever.



<div>iPhone 12</div> <div></div> <div>10.682.350 đ 9.289.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 11</div> <div></div> <div>10.683.500 đ 9.290.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 13</div> <div></div> <div>18.285.000 đ 15.900.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 14</div> <div></div> <div>26.335.000 đ 22.900.000 đ</div> <div>Details Add to Cart</div>
<div>iPhone 15</div> <div></div> <div>28.048.500 đ 24.390.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 15 PRO</div> <div></div> <div>29.198.500 đ 25.390.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 15 PRO MAX</div> <div></div> <div>33.798.500 đ 29.390.000 đ</div> <div>Details Add to Cart</div>	<div>iPhone 14 PRO MAX</div> <div></div> <div>23.448.500 đ 20.390.000 đ</div> <div>Details Add to Cart</div>

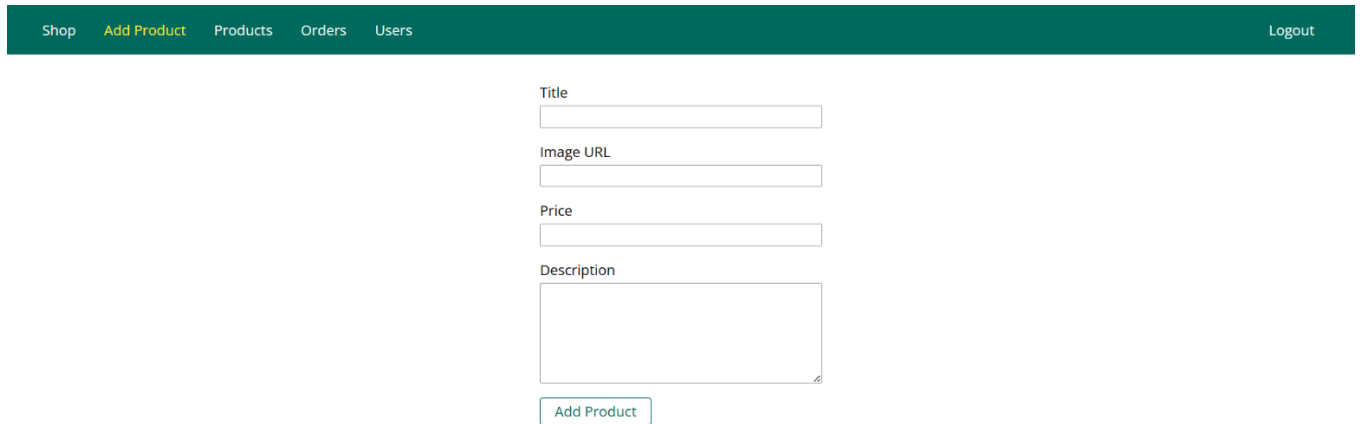
Hình 3.7. Trang chủ

3.3.4. Giỏ hàng

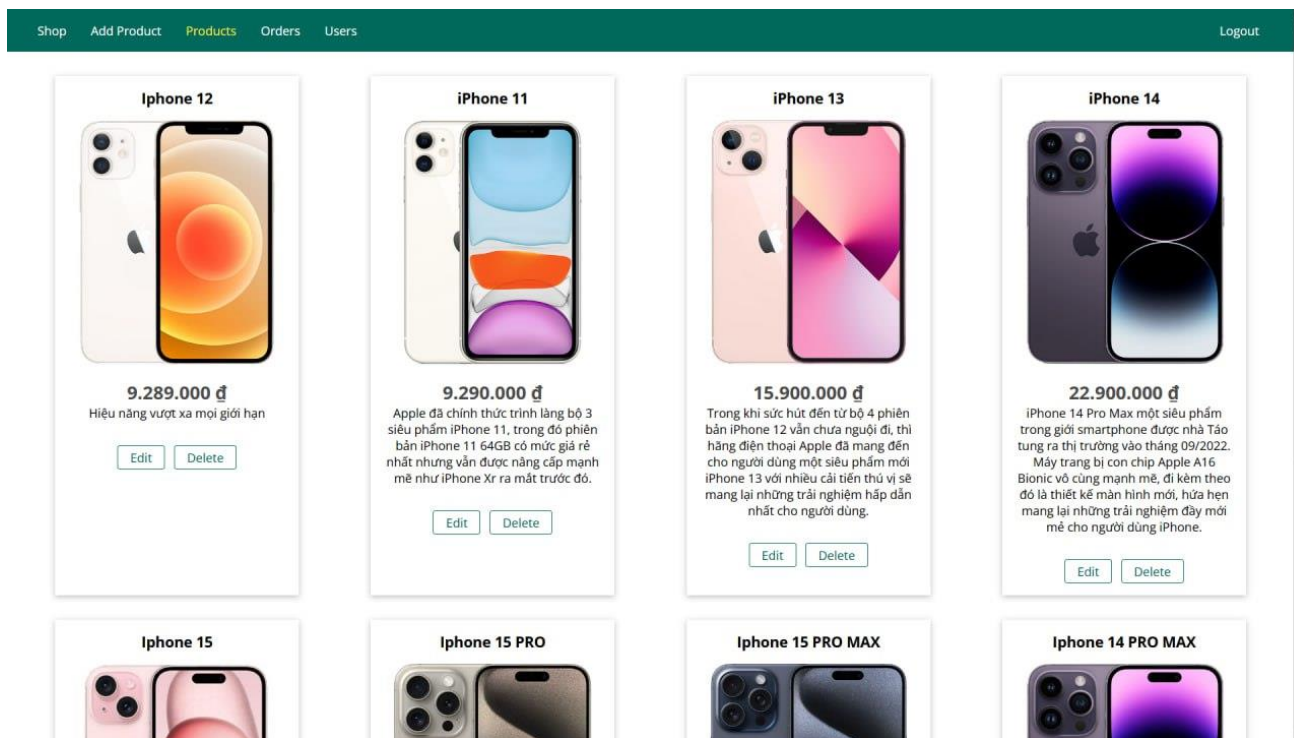


Hình 3.8. Trang giỏ hàng

3.3.5. Trang quản trị thêm, xoá, sửa sản phẩm, hoá đơn mua hàng.



Hình 3.9. Trang thêm sản phẩm



Hình 3.10. Trang sửa và xoá sản phẩm

Shop Add Product Products Orders Users Logout				
Order - # 6628af100d07cd30698eea9d				
User ID	User Email	Product Name	Quantity	Total Price
6628aef50d07cd30698eea90	phamvnpnc063@gmail.com	iPhone 12	1	9.289.000 đ
Total Price: 9.289.000 đ				

Hình 3.11. Hoá đơn mua hàng

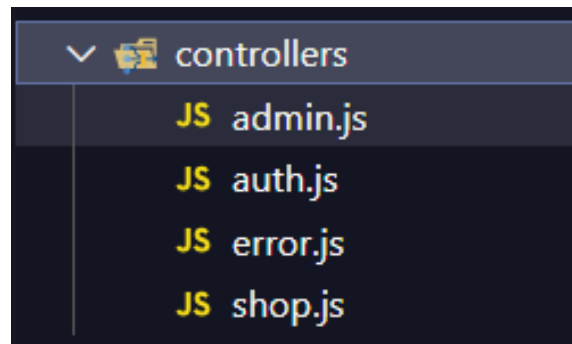
3.3.6. Trang danh sách users và xoá user

Shop Add Product Products Orders Users Logout				
Fullname	Email	Address	Phone Number	Action
Phạm Văn Phúc	phamvnphc063@gmail.com	15 Hoàng Văn Thụ, Phường 15	0387375700	<button>Delete</button>

Hình 3.12. Trang danh sách users và chức năng xoá users

3.4. Hiện Thực Ứng Dụng

3.4.1. Controllers



Hình 3.13. Mô tả cấu trúc bên trong folder controller

admin.js:

- Định nghĩa các hàm xử lý logic cho các tuyến đường liên quan đến quản trị viên trong ứng dụng.
- Bao gồm các chức năng như lấy dữ liệu cho bảng điều khiển quản trị, thêm, chỉnh sửa và xóa sản phẩm, quản lý người dùng và đơn hàng.

auth.js:

- Chứa các hàm xử lý logic cho tuyến đường đăng nhập và đăng ký người dùng.

- Bao gồm xác thực người dùng, xử lý đăng nhập và đăng ký, xử lý phiên và thông báo.

error.js:

- Định nghĩa hàm xử lý logic để xử lý yêu cầu khi gặp lỗi 404 (Không tìm thấy).
- Trả về một trang thông báo lỗi hoặc thông báo lỗi dưới dạng JSON.

shop.js:

- Chứa các hàm xử lý logic cho các tuyến đường liên quan đến giao diện người dùng của cửa hàng.
- Bao gồm các chức năng như hiển thị sản phẩm, chi tiết sản phẩm, giỏ hàng, đơn hàng và xử lý các thao tác liên quan đến giỏ hàng và đơn hàng.

Mỗi tập controllers chứa các hàm xử lý logic tương ứng với các chức năng và tuyến đường cụ thể trong ứng dụng của bạn. Các hàm này được gọi khi có yêu cầu từ phía người dùng tương ứng với các tuyến đường đã định nghĩa. Điều này giúp tổ chức và quản lý mã nguồn một cách cấu trúc và dễ dàng bảo trì.

3.4.2. Middleware



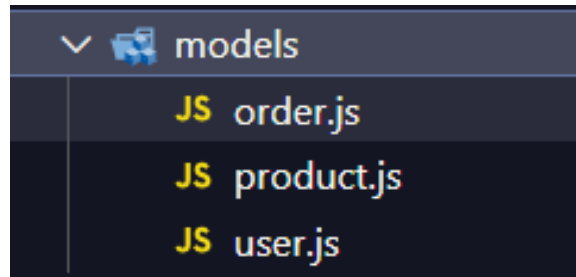
Hình 3.14. Mô tả bên trong folder middleware

isAuth.js:

- Middleware kiểm tra xem người dùng đã đăng nhập hay chưa bằng cách kiểm tra thông tin phiên (session).
- Nếu người dùng chưa đăng nhập, middleware sẽ chuyển hướng họ đến trang đăng nhập trước khi tiếp tục xử lý yêu cầu.

Các middleware được gọi tuần tự trong chuỗi xử lý yêu cầu và có thể thêm hoặc sửa đổi thông tin trong đối tượng yêu cầu (req) hoặc phản hồi (res) trước khi chúng được chuyển đến các hàm xử lý tuyến đường cuối cùng. Điều này giúp tổ chức mã nguồn và giảm lặp lại của mã xử lý trung gian.

3.4.3. Models



Hình 3.15. Mô tả bên trong folder models

Folder models thường chứa các định nghĩa cho các đối tượng dữ liệu trong ứng dụng, thường là các đối tượng dữ liệu có cấu trúc và tương tác với cơ sở dữ liệu.

user.js:

- Định nghĩa các thuộc tính và phương thức cho đối tượng người dùng (user), bao gồm tên, email, mật khẩu, địa chỉ, số điện thoại, quyền hạn (admin/user), và các phương thức liên quan như thêm vào giỏ hàng, xóa khỏi giỏ hàng, tạo đơn hàng, và lấy danh sách đơn hàng.

product.js:

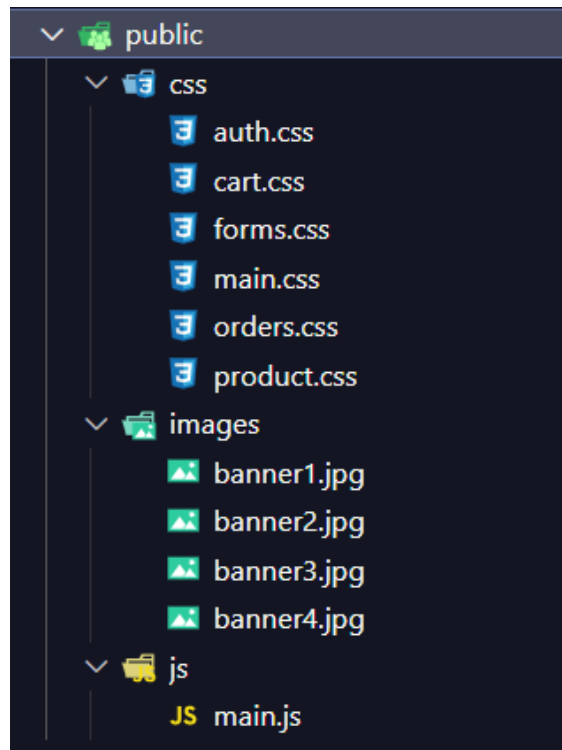
- Định nghĩa các thuộc tính cho đối tượng sản phẩm (product), bao gồm tên, URL hình ảnh, giá, mô tả, và thông tin liên kết với người dùng tạo ra sản phẩm.

order.js:

- Định nghĩa các thuộc tính cho đối tượng đơn hàng (order), bao gồm danh sách sản phẩm được đặt hàng, số lượng, thông tin người dùng đặt hàng.

Mỗi file trong folder models thường sẽ sử dụng một loạt các biểu đồ (schemas) của Mongoose để định nghĩa cấu trúc dữ liệu và tương tác với cơ sở dữ liệu MongoDB. Các phương thức được định nghĩa trong các schema này sẽ thực hiện các thao tác như thêm, sửa, xóa dữ liệu trong cơ sở dữ liệu.

3.4.4. Public



Hình 3.16. Mô tả bên trong folder public

Folder public thường chứa các tài nguyên tĩnh được phục vụ trực tiếp cho client-side của ứng dụng web, bao gồm các file CSS, hình ảnh, và mã JavaScript.

Css :

- Thư mục chứa các file CSS được sử dụng để thiết kế và trang trí giao diện của trang web. Các file CSS này có thể chứa các quy tắc kiểu (CSS rules) để điều chỉnh font chữ, màu sắc, kích thước và vị trí của các phần tử trên trang.

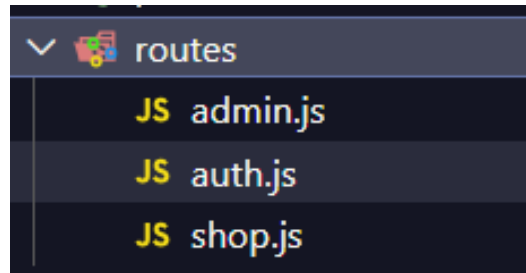
Images:

- Thư mục này chứa các hình ảnh được sử dụng trong trang web. Đây có thể là hình ảnh sản phẩm, biểu tượng, logo, hoặc bất kỳ hình ảnh nào khác liên quan đến nội dung của trang web.

Js:

- Thư mục chứa các file JavaScript được sử dụng để thêm các tính năng tương tác và chức năng động vào trang web. Các file JavaScript này có thể chứa mã để xử lý sự kiện, thay đổi nội dung trang web, gửi và nhận dữ liệu từ máy chủ, và thực hiện các tác vụ khác mà yêu cầu xử lý logic phía client-side.

3.4.5. Routes



Hình 3.16. Mô tả bên trong folder routes

Thư mục routes thường chứa các tệp tin JavaScript được sử dụng để định nghĩa các tuyến đường (routes) của ứng dụng web.

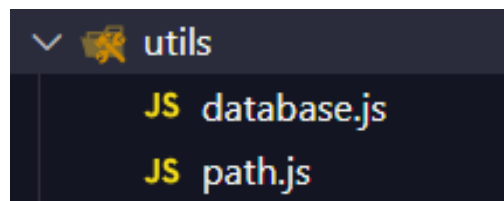
Mỗi tệp tin trong thư mục routes thường định nghĩa các tuyến đường cho một phần cụ thể của ứng dụng web. Ví dụ, có thể có các tệp tin như auth.js, shop.js, admin.js, hoặc các tệp tin khác tương ứng với các chức năng hoặc phần của trang web.

Trong mỗi tệp tin route, các tuyến đường được định nghĩa bằng cách sử dụng Express Router. Mỗi tuyến đường thường bao gồm một URL (đường dẫn) và một hàm xử lý (handler) để xử lý yêu cầu từ client và trả về phản hồi tương ứng.

Trong một số tệp tin route, có thể có sử dụng middleware để thực hiện các chức năng trung gian, chẳng hạn như xác thực (authentication), kiểm tra quyền truy cập (authorization), xử lý dữ liệu yêu cầu, hoặc gửi các phản hồi phụ trợ.

Thông thường, các tệp tin route sẽ giao tiếp với các controllers để xử lý logic của ứng dụng. Các controllers được sử dụng để thực hiện các thao tác cần thiết trên dữ liệu và trả về kết quả tương ứng cho client.

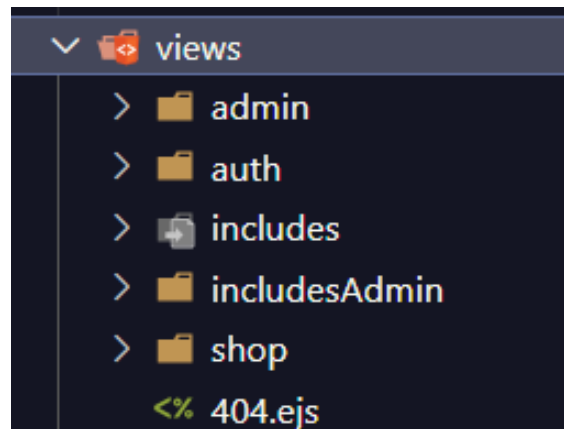
3.5.6. Utils



Hình 3.17. Mô tả bên trong utils

Thư mục utils thường chứa các tệp tin hoặc hàm tiện ích (utility) được sử dụng để thực hiện các chức năng nhỏ hoặc lặp lại trong ứng dụng.

3.5.7. Views



Hình 3.18. Mô tả bên trong folder Views

Thư mục views trong một ứng dụng web thường chứa các tệp mẫu (templates) được sử dụng để tạo ra các giao diện người dùng.

admin:

- Thư mục này chứa các tệp mẫu liên quan đến giao diện quản trị hoặc bảng điều khiển của ứng dụng. Các trang như trang quản lý sản phẩm, quản lý người dùng, thống kê, và cài đặt có thể được đặt trong thư mục này.

Auth:

- Thư mục này chứa các tệp mẫu liên quan đến chức năng xác thực và quản lý tài khoản người dùng. Đây có thể là các trang đăng nhập, đăng ký, đổi mật khẩu, và các trang thông tin tài khoản.

include và includeAdmin:

- Thư mục này thường chứa các tệp mẫu phần đầu (header), phần cuối (footer), hoặc các thành phần giao diện khác có thể được sử dụng lại trên nhiều trang khác nhau. Thông thường, các phần này được chia thành phần phần đầu và phần cuối để giúp tái sử dụng và duy trì mã nguồn.

Shop:

- Thư mục này chứa các tệp mẫu liên quan đến giao diện cửa hàng hoặc phần giao diện chính của ứng dụng. Các trang như danh sách sản phẩm, chi tiết sản phẩm, giỏ hàng, thanh toán và xác nhận đơn hàng có thể được đặt trong thư mục này.

404: . Trang này được hiển thị khi người dùng yêu cầu một URL không tồn tại trên máy chủ.

CHƯƠNG 4 KẾT LUẬN

4.1. Tóm tắt kết quả

Lựa chọn đề tài: Tạo một ứng dụng web bán điện thoại di động.

Mục tiêu nghiên cứu: Xây dựng một hệ thống hoàn chỉnh cho việc quản lý sản phẩm, danh mục, người dùng và hóa đơn.

Giới thiệu về Node.js, Express, MongoDB, ... và các công nghệ liên quan khác ở nền tảng web app.

Việc phân tích yêu cầu và nhu cầu của người dùng đã giúp xây dựng một ứng dụng có tính năng đa dạng và phù hợp. Thiết kế cơ sở dữ liệu được xây dựng có cấu trúc và mô hình rõ ràng, giúp quản lý dữ liệu một cách hiệu quả.

Kết quả của tiểu luận là một ứng dụng linh hoạt, dễ sử dụng và có khả năng mở rộng. Việc triển khai ứng dụng đã được chuẩn bị kỹ lưỡng và được thực hiện một cách thành công. Điều này đảm bảo rằng ứng dụng có thể đáp ứng nhu cầu của người dùng một cách tốt nhất và mang lại trải nghiệm tuyệt vời nhất.

4.2. Đánh Giá và Hướng Phát Triển

Đánh Giá:

- Website đã hoàn thành các chức năng chính như đăng nhập, đăng ký, xem danh sách sản phẩm, thêm vào giỏ hàng, và thanh toán hóa đơn.
- Giao diện người dùng được thiết kế một cách hợp lý, dễ sử dụng và thân thiện với người dùng.
- Cơ sở dữ liệu được tổ chức cẩn thận và có cấu trúc, giúp quản lý dữ liệu hiệu quả.

Hướng Phát Triển:

- Tối Ưu Hóa Hiệu Năng: Tiếp tục tối ưu hóa hiệu năng của ứng dụng, đặc biệt là trong việc truy vấn cơ sở dữ liệu và tải dữ liệu từ máy chủ.
- Phát Triển Tính Năng Mở Rộng: Bổ sung các tính năng mới như đánh giá sản phẩm, tìm kiếm sản phẩm, và tùy chỉnh hóa đơn.
- Phát Triển Phiên Bản Đa Nền Tảng: Xây dựng phiên bản cho các nền tảng khác như android/ios để mở rộng phạm vi sử dụng của ứng dụng.
- Tăng Cường Bảo Mật: Cải thiện lớp bảo mật của ứng dụng để đảm bảo an toàn thông tin cá nhân của người dùng.
- Tích Hợp Thanh Toán: Bổ sung các phương thức thanh toán mới và tích hợp các cổng thanh toán phổ biến để tăng tính tiện lợi cho người dùng.
- Nâng Cấp Giao Diện Người Dùng: Cải thiện giao diện người dùng để tạo ra trải nghiệm người dùng tốt hơn và thú vị hơn.

4.3. Kết luận

Trong suốt quá trình nghiên cứu và phát triển website bán điện thoại di động, em đã thực hiện một hành trình hết sức hứng thú và hữu ích. Điểm khởi đầu là việc chọn đề tài, một quyết định quan trọng đòi hỏi sự kỳ công và cân nhắc để đảm bảo rằng đề tài có sự hấp dẫn và tiềm năng phát triển. Sau đó, quá trình phân tích yêu cầu và thiết kế cơ sở dữ liệu đã đặt nền móng vững chắc cho việc phát triển website.

Khi bước vào giai đoạn triển khai, em đã gặp phải nhiều thách thức và cần phải áp dụng kiến thức lý thuyết vào thực tế. Việc hiện thực các chức năng như đăng nhập, đăng ký, quản lý sản phẩm và quản lý giỏ hàng đòi hỏi sự tỉ mỉ và kiên nhẫn. Nhưng nhìn lại, mỗi khó khăn đã trở thành cơ hội để học hỏi và phát triển kỹ năng.

Quan trọng hơn, việc thực hiện ứng dụng từ đầu đến cuối đã giúp em hiểu rõ hơn về các công nghệ và công cụ như VS Code, NodeJS, MongoDB và ExpressJS. Bằng cách thử nghiệm và áp dụng kiến thức này vào thực tế, chúng ta đã có cơ hội làm quen với quy trình phát triển phần mềm và rèn luyện kỹ năng lập trình.

Trong quá trình đánh giá và đề xuất hướng phát triển tiếp theo, em nhận ra rằng việc tiếp tục tối ưu hóa website, mở rộng tính năng và tăng cường bảo mật sẽ là những điểm chính cần tập trung. Đồng thời, việc phát triển phiên bản đa nền tảng và tích hợp các phương thức thanh toán mới cũng sẽ là những bước quan trọng trong việc nâng cao trải nghiệm người dùng và mở rộng phạm vi hoạt động của ứng dụng.

Tiểu luận này không chỉ là một cuộc hành trình nghiên cứu và phát triển ứng dụng, mà còn là một cơ hội để khám phá, học hỏi và trưởng thành trong lĩnh vực công nghệ thông tin.

TÀI LIỆU THAM KHẢO

1. Node.js Design Patterns - Second Edition, Tác giả: Mario Casciaro, Luciano Mammino, Nhà xuất bản: Packt Publishing, Xuất bản năm: 2018.
2. Thiết kế cơ sở dữ liệu SQL và NoSQL hiện đại, Tác giả: Võ Đức Thoại, Nhà xuất bản: Thống kê, Xuất bản năm: 2021.
3. Tài liệu học tập NodeJS và các công nghệ bên trong website: w3school