

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ TÀI XÂY DỰNG**

**WEBSITE THƯƠNG MẠI ĐIỆN TỬ**

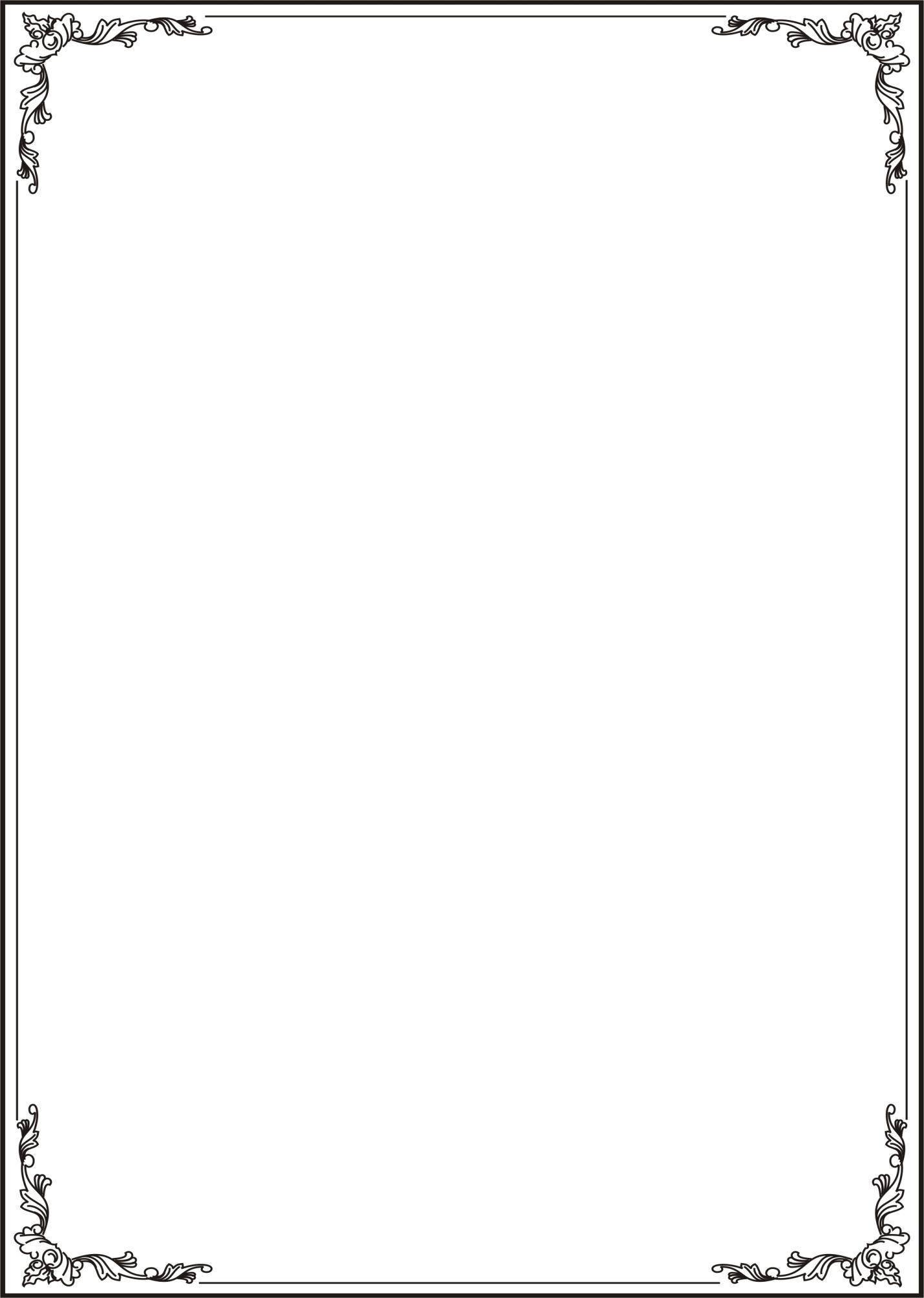
**Môn: Các công nghệ phần mềm mới**

**Lớp: MTSE431179\_23\_1\_05CLC**

**GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Minh Đạo**

**TP. Hồ Chí Minh, tháng 12 năm 2023**

**KHÓA 2020 - 2024**



**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ TÀI XÂY DỰNG**

**WEBSITE THƯƠNG MẠI ĐIỆN TỬ**

**Môn: Các công nghệ phần mềm mới**

**Lớp: MTSE431179\_23\_1\_05CLC**

**GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Minh Đạo**

**TP. Hồ Chí Minh, tháng 12 năm 2023**

**KHÓA 2020 - 2024**

**LỜI CẢM ƠN**

*Đầu tiên, em xin được phép gửi lời cảm ơn chân thành nhất và sâu sắc nhất đến với Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện tốt nhất em được học tập, phát triển nền tảng kiến thức sâu sắc từ đó có thể xây dựng được đề tài kết thúc môn học này.*

*Bên cạnh đó nhóm chúng em xin gửi đến thầy Nguyễn Minh Đạo lời cảm ơn sâu sắc nhất trong suốt thời gian dài học tập cũng như thực hiện đề tài trong thời gian qua.*

*Với sự chỉ bảo nhiệt tình, giảng dạy tận tình và đầy đủ kiến thức của thầy, em đã học tập và tiếp thu được những kiến thức cơ bản về các công nghệ phần mềm phổ biến hiện nay.*

*Tuy nhiên lượng kiến thức là vô tận và với khả năng hạn hẹp, em đã rất cố gắng để hoàn thành một cách tốt nhất đề tài này. Chính vì vậy việc xảy ra những thiếu sót là điều khó có thể tránh khỏi. Em hi vọng nhận được sự góp ý, nhận xét và phê bình tận tình của thầy để em có thể rút ra được các bài học kinh nghiệm và từ đó hoàn thiện, nâng cấp lại đề tài của mình một cách tốt nhất có thể.*

*Em xin chân thành cảm ơn thầy!*

|  |  |
| --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP. HCM**  **KHOA CÔNG NGHỆ THÔNG TIN**  \*\*\*\*\*\*\* | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  \*\*\*\*\*\*\* |

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Họ tên sinh viên: **Đỗ Trung Đức** Mã số sinh viên: **20110459**

Ngành: **Công Nghệ Thông Tin**

Đề tài**: Xây dựng website thương mại điện tử**

Giảng viên hướng dẫn: **ThS. Nguyễn Minh Đạo**

Môn: **Các công nghệ phần mềm mới**

Lớp: **MTSE431179\_23\_1\_05CLC**

**LỜI NHẬN XÉT CỦA GIẢNG VIÊN**

|  |  |  |
| --- | --- | --- |
| *Tp. HCM, ngày tháng năm 2023* | | |
|  | Giảng viên hướng dẫn  *(Ký & ghi rõ họ tên)* |

**Mục Lục**

[Chương 1: Giới Thiệu 1](#_Toc152404739)

[1. Giới thiệu chung 1](#_Toc152404740)

[2. Mục tiêu dự án 1](#_Toc152404741)

[3. Công Nghệ Sử Dụng 1](#_Toc152404742)

[3.1 MERN Stack 1](#_Toc152404743)

[3.2 Các Công Nghệ Khác 1](#_Toc152404744)

[4. Tính Năng Chính 1](#_Toc152404745)

[5. Lợi Ích 2](#_Toc152404746)

[6. Kết Luận 2](#_Toc152404747)

[Chương 2: Hướng dẫn chạy chương trình 3](#_Toc152404748)

[1. Import dữ liệu vào database 3](#_Toc152404749)

[2. Chạy chương trình 3](#_Toc152404750)

[3. Tài khoản 3](#_Toc152404751)

[Chương 3: Giao diện ứng dụng 4](#_Toc152404752)

[1. Trang Login 4](#_Toc152404753)

[2. Trang Register 4](#_Toc152404754)

[3. Trang ForgotPassword 5](#_Toc152404755)

[4. Trang NewPassword 5](#_Toc152404756)

[5. Trang Profile 6](#_Toc152404757)

[6. Trang UpdateProfile 6](#_Toc152404758)

[7. Trang UpdatePassword 7](#_Toc152404759)

[8. Trang Home 7](#_Toc152404760)

[9. Trang ProductDetails 8](#_Toc152404761)

[10. Trang Cart 8](#_Toc152404762)

[11. Trang Shipping 9](#_Toc152404763)

[12. Trang ConfirmOrder 9](#_Toc152404764)

[13. Trang Payment 10](#_Toc152404765)

[14. Trang OrderSuccess 11](#_Toc152404766)

[15. Trang ListOrders 11](#_Toc152404767)

[16. Trang OrderDetails 12](#_Toc152404768)

[17. Trang DashBoard - Admin 12](#_Toc152404769)

[18. Trang ProductList 13](#_Toc152404770)

[19. Trang UpdateProduct 13](#_Toc152404771)

[20. Trang NewProduct 14](#_Toc152404772)

[21. Trang OrdersList 14](#_Toc152404773)

[22. Trang ProcessOrder 15](#_Toc152404774)

[23. Trang UsersList 15](#_Toc152404775)

[24. Trang UpdateUser 16](#_Toc152404776)

[25. Trang ProductReviews 16](#_Toc152404777)

[Chương 4: Backend 17](#_Toc152404778)

[1. Cấu trúc thư mục 17](#_Toc152404779)

[2. /config 18](#_Toc152404780)

[2.1. config.env 18](#_Toc152404781)

[2.2. database.js 18](#_Toc152404782)

[3. /controllers 19](#_Toc152404783)

[3.1. authControllers.js 19](#_Toc152404784)

[3.2. productControllers.js 20](#_Toc152404785)

[3.3. orderControllers.js 21](#_Toc152404786)

[3.4. paymentControllers.js 22](#_Toc152404787)

[4. /middlewares 23](#_Toc152404788)

[4.1. auth.js 23](#_Toc152404789)

[4.2. catchAsyncErrors.js 24](#_Toc152404790)

[4.3. error.js 24](#_Toc152404791)

[5. /models 25](#_Toc152404792)

[6. /routes 25](#_Toc152404793)

[6.1. auth.js 25](#_Toc152404794)

[6.2. product.js 26](#_Toc152404795)

[6.3. order.js 27](#_Toc152404796)

[6.4. payment.js 27](#_Toc152404797)

[7. /utils 28](#_Toc152404798)

[7.1. apiFeature.js 28](#_Toc152404799)

[7.2. errorHandler.js 29](#_Toc152404800)

[7.3. jwtToken.js 30](#_Toc152404801)

[7.4. sendEmail.js 30](#_Toc152404802)

[8. app.js 31](#_Toc152404803)

[9. server.js 32](#_Toc152404804)

[Chương 5: Frontend 33](#_Toc152404805)

[1. setupProxy.js 33](#_Toc152404806)

[2. Redux 33](#_Toc152404807)

[2.1. store 33](#_Toc152404808)

[2.2. constants 37](#_Toc152404809)

[2.3. actions 37](#_Toc152404810)

[2.4. reducers 38](#_Toc152404811)

[3. Các trang giao diện 41](#_Toc152404812)

[4. index.js 42](#_Toc152404813)

[5. ProtectedRoute.js 43](#_Toc152404814)

[6. App.js 43](#_Toc152404815)

[Chương 6: Tổng kết 48](#_Toc152404816)

[1. Kết quả đạt được 48](#_Toc152404817)

[2. Hạn chế 48](#_Toc152404818)

[3. Hướng phát triển 48](#_Toc152404819)

[Tài Liệu Tham Khảo 49](#_Toc152404820)

# Chương 1: Giới Thiệu

## 1. Giới thiệu chung

Trong bối cảnh thị trường thương mại điện tử ngày càng phát triển và trở thành xu hướng mua sắm phổ biến, việc xây dựng một website thương mại điện tử hiệu quả trở nên ngày càng quan trọng. Dự án này nhằm mục tiêu tạo ra một nền tảng mua sắm trực tuyến đa chức năng, linh hoạt và thuận tiện cho người dùng.

## 2. Mục tiêu dự án

Dự án nhằm xây dựng một trang web thương mại điện tử sử dụng công nghệ MERN Stack, bao gồm MongoDB, Express.js, React.js và Node.js. Mục tiêu là tạo ra một trải nghiệm mua sắm trực tuyến độc đáo, tích hợp các tính năng như quản lý sản phẩm, giỏ hàng, thanh toán an toàn, quản lý tài khoản người dùng, đánh giá sản phẩm và quản lý đơn hàng.

## 3. Công Nghệ Sử Dụng

### 3.1 MERN Stack

MongoDB: Hệ quản trị cơ sở dữ liệu phi quan hệ, linh hoạt và có khả năng mở rộng tốt.

Express.js: Framework Node.js giúp xây dựng các ứng dụng web và API một cách nhanh chóng.

React.js: Thư viện JavaScript phổ biến để xây dựng giao diện người dùng tương tác.

Node.js: Môi trường chạy mã JavaScript ở phía máy chủ, hỗ trợ xây dựng các ứng dụng web có khả năng mở rộng.

### 3.2 Các Công Nghệ Khác

Redux: Được sử dụng để quản lý trạng thái ứng dụng trong React, giúp quản lý dữ liệu một cách hiệu quả.

JWT (JSON Web Tokens): Để xác thực và bảo mật thông tin người dùng.

Bootstrap: Framework CSS giúp phát triển giao diện người dùng một cách nhanh chóng và đồng nhất.

## 4. Tính Năng Chính

Quản lý Sản Phẩm: Cho phép Admin thêm, sửa, xóa sản phẩm với các thông tin chi tiết.

Quản lý đơn hàng: Cho phép Admin cập nhật trạng thái đơn hàng

Quản Lý Tài Khoản Người Dùng: Cho phép Admin chỉnh sửa thông tin cá nhân, vai trò của user.

Authentication: Cho phép người dùng đăng nhập, đăng ký, reset mật khẩu, đổi mật khẩu

Giỏ Hàng và Thanh Toán: Cho phép người dùng thêm sản phẩm vào giỏ hàng và tiến hành thanh toán an toàn.

Tìm kiếm sản phẩm và lọc: Cho phép người dùng nhập từ khóa để tìm kiếm sản phẩm và có thể lọc sản phẩm theo giá tiền, danh mục, đánh giá.

Đánh Giá và Nhận Xét: Người dùng có thể đánh giá và bình luận về sản phẩm.

## 5. Lợi Ích

Tiết Kiệm Thời Gian: MERN Stack giúp giảm thời gian phát triển nhờ vào tính tương thích cao giữa các thành phần.

Linh Hoạt và Mở Rộng: Công nghệ MERN Stack cho phép dễ dàng mở rộng và thích ứng với sự phát triển của dự án.

Giao Diện Dễ Sử Dụng: Sử dụng React.js để tạo ra giao diện người dùng dễ tương tác và thân thiện.

## 6. Kết Luận

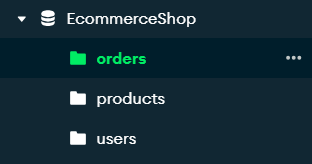
Dự án xây dựng website thương mại điện tử sử dụng công nghệ MERN Stack sẽmang lại trải nghiệm mua sắm trực tuyến tốt nhất cho người. Sự kết hợp giữa tính năng đa dạng và hiệu suất của MERN Stack sẽ đảm bảo hiệu quả và tính ổn định của ứng dụng.

# Chương 2: Hướng dẫn chạy chương trình

## 1. Import dữ liệu vào database

- Mở MongoDB và kết nối vào localhost.

- Tạo một database tên là EcommerceShop và tạo 3 collection là orders, products, user.



- Sau đó import dữ liệu từ 3 file json đính kèn vô 3 collection này.

## 2. Chạy chương trình

- Mở thư mục chứa chương trình, sau đó tạo 2 terminal, một bên là server (phải) và bên còn lại là cd vô frontend (trái).

- Tiếp đến, gõ **npm run prod** cho bên server và chạy.

- Cuối cùng, chuyển sang terminal của frontend, nhập **npm start** để chạy chương trình.

dA screenshot of a computer

Description automatically generated

## 3. Tài khoản

- Tài khoản admin:

TK: [ducdt@gmail.com](mailto:ducdt@gmail.com)

MK: 0123456789

# Chương 3: Giao diện ứng dụng

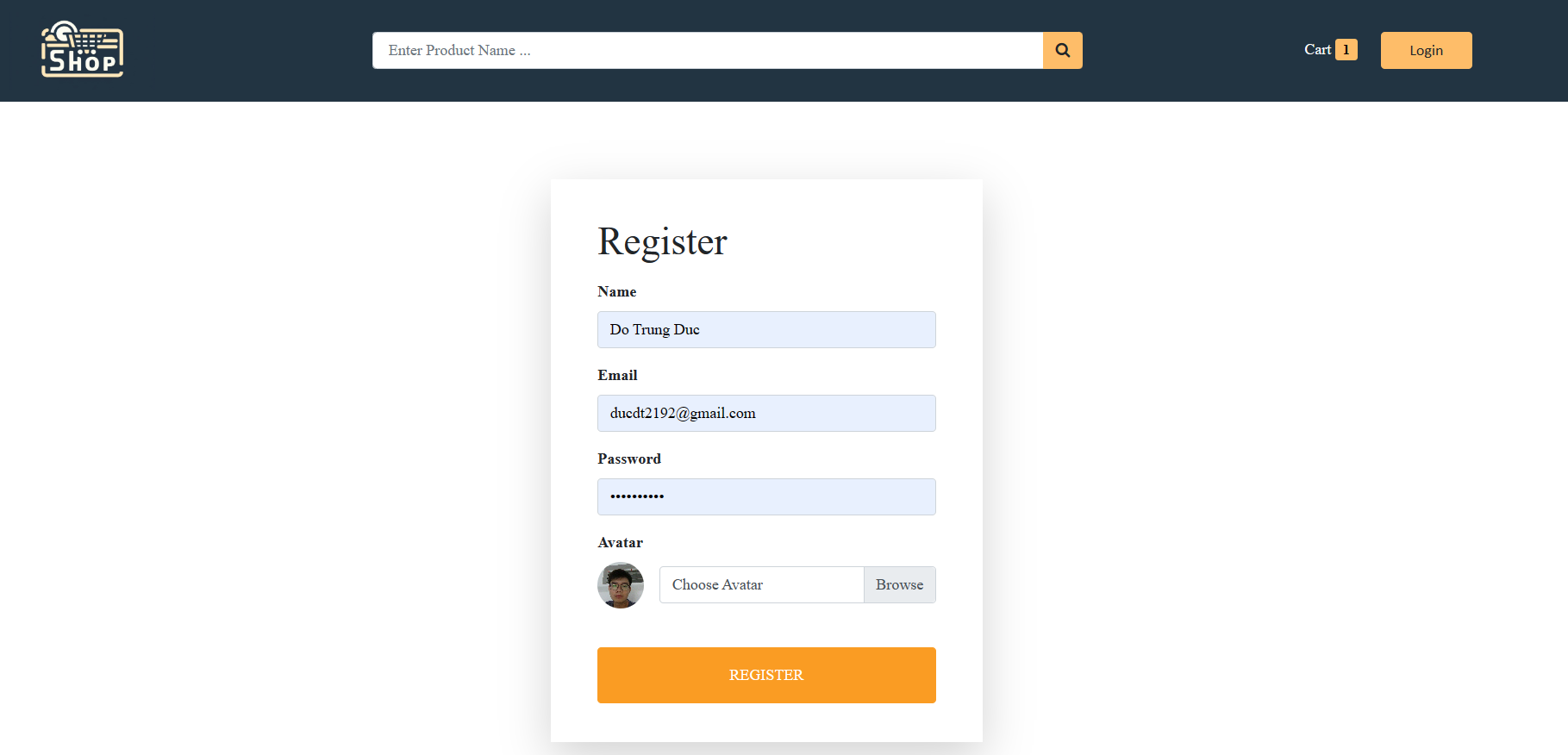
## 1. Trang Login

A screenshot of a login page

Description automatically generated

User nhập tài khoản mật khẩu (mật khẩu có thể che hoặc không che bằng dấu ‘\*’), nếu chưa có tài khoản thì user có thể click vào ‘New User?’ để chuyển hướng đến trang đăng ký hoặc nếu quên mật khẩu thì click vào ‘Forgot Password?’

## 2. Trang Register



User nhập tên, email, password và chọn ảnh đại diện (có hoặc không đều được)

## 3. Trang ForgotPassword

A screenshot of a computer

Description automatically generated

Người dùng nhập email để gửi đường link đổi lại mật khẩu

## 4. Trang NewPassword

Chạy đường link đổi mật khẩu nhận được từ email

A screenshot of a computer

Description automatically generated

User nhấn ‘Set Password’ để đổi sang mật khẩu mới

## 5. Trang Profile

A screenshot of a computer

Description automatically generated

Hiển thị thông tin của User

User có thể nhấn ‘Edit Profile’ để chỉnh sửa profile, nhấn ‘My Orders’ để xem thông tin đơn hàng của mình, nhấn ‘Change Password’ để đổi mật khẩu

## 6. Trang UpdateProfile

A screenshot of a computer

Description automatically generated

User có thể chỉnh sửa lại thông tin cá nhân của mình và nhấn ‘Update’

## 7. Trang UpdatePassword

A screenshot of a computer

Description automatically generated

User nhập mật khẩu cũ và mới để cập nhật, sau đó nhấn ‘Update Password’

## 8. Trang Home

A screenshot of a website

Description automatically generated

Hiển thị danh sách phẩm đang được bán, có thể click vô sản phẩm để xem chi tiết

Có phân trang, mỗi trang hiển thị 8 sản phẩm A screenshot of a phone

Description automatically generated

## 9. Trang ProductDetails

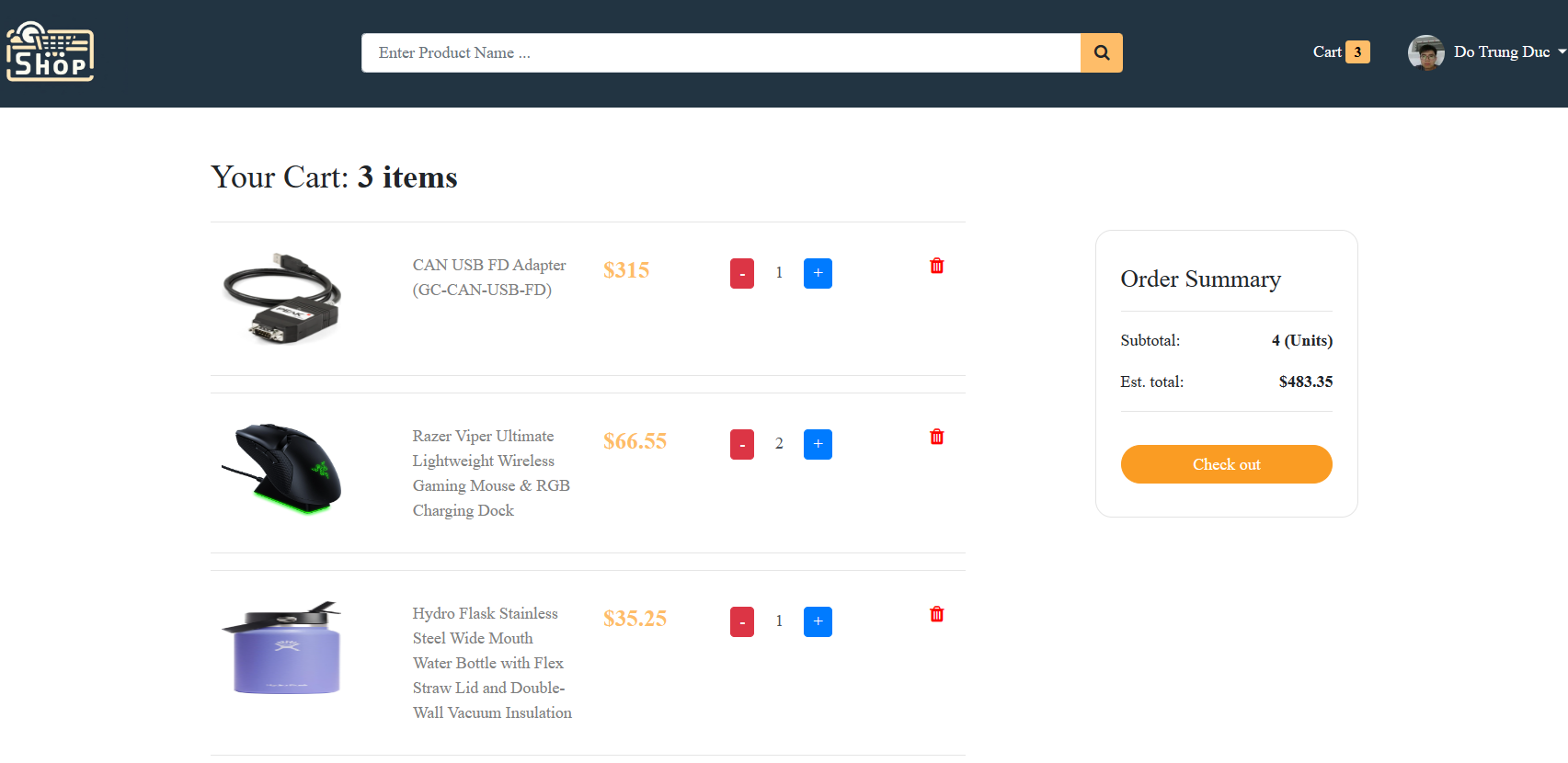
A computer mouse and controller

Description automatically generated

Hiển thị chi tiết các thông tin của sản phẩm và phần đánh giá sản phẩm

User có thể thêm đánh giá hoặc chọn số lượng và thêm sản phẩm vào giỏ hàng

## 10. Trang Cart



Hiển thị danh sách thông tin các sản phẩm được thêm vào giỏ hàng và số lượng cũng như tổng tiền của các sản phẩm

User có thể thêm bớt số lượng sản phẩm theo mong muốn và nhấn nút ‘Check out’ để bắt đầu tiến hành quá trình thanh toán đơn hàng

## 11. Trang Shipping

A screenshot of a web page

Description automatically generated

User nhập thông tin địa chỉ giao hàng và nhấn Continue để qua bước Confirm Order

## 12. Trang ConfirmOrder

A screenshot of a computer

Description automatically generated

Hiển thị thông tin địa chỉ giao hàng, thành tiền từng sản phẩm, các thông tin về phí ship, thuế, tổng tiền đơn hàng

User tiếp tục nhấn nút ‘Proceed to Payment’ để qua bước thanh toán

## 13. Trang Payment

A screenshot of a computer

Description automatically generated

User nhập thông tin chuyển khoản để tiến hành thanh toán và nhấn nút Pay

Sau khi nhấn nút, nếu chuyển khoản thành công sẽ hiện thông báo

A screenshot of a computer

Description automatically generated

Người dùng nhấn Complete Authentication để kết thúc quá trình thanh toán

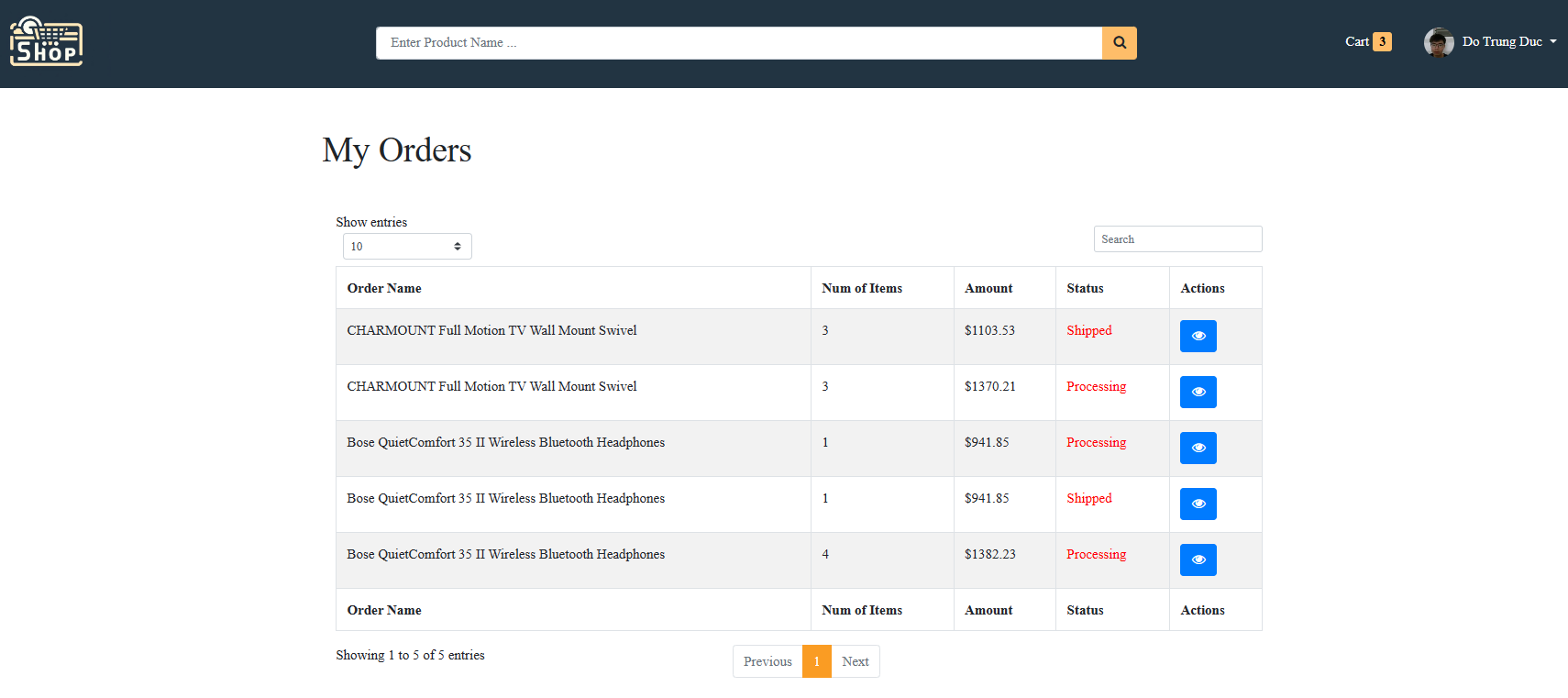
## 14. Trang OrderSuccess

A screenshot of a computer

Description automatically generated

Thông báo User đã thanh toán thành công

## 15. Trang ListOrders



Hiển thông tin đơn hàng, trạng thái đơn hàng

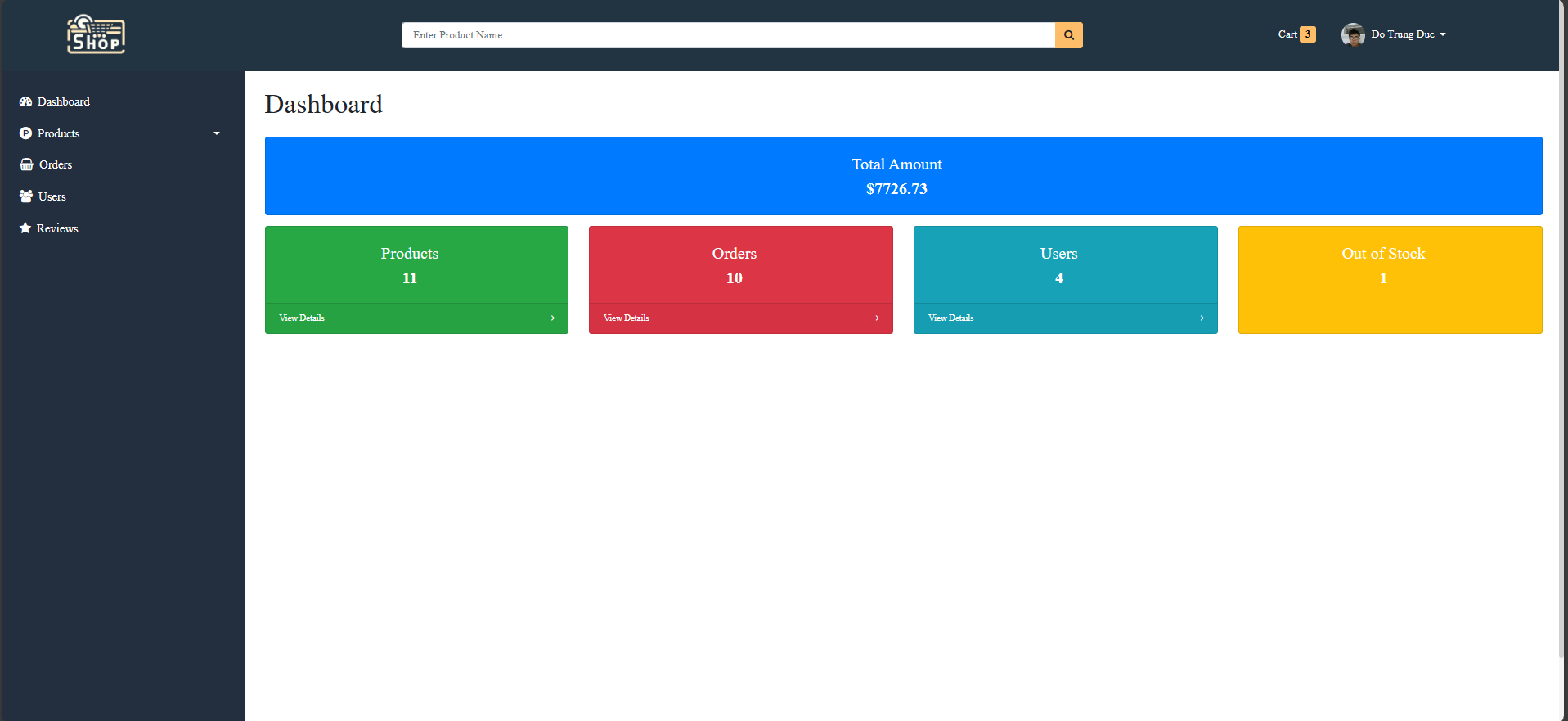
User có thể nhấn vào Action của từng đơn hàng để xem chi tiết hơn

## 16. Trang OrderDetails

A screenshot of a computer

Description automatically generated

## 17. Trang DashBoard - Admin



Hiển thị tổng doanh số bán hàng, số lượng sản phẩm, đơn hàng, user và số sản phẩm đã hết

## 18. Trang ProductList

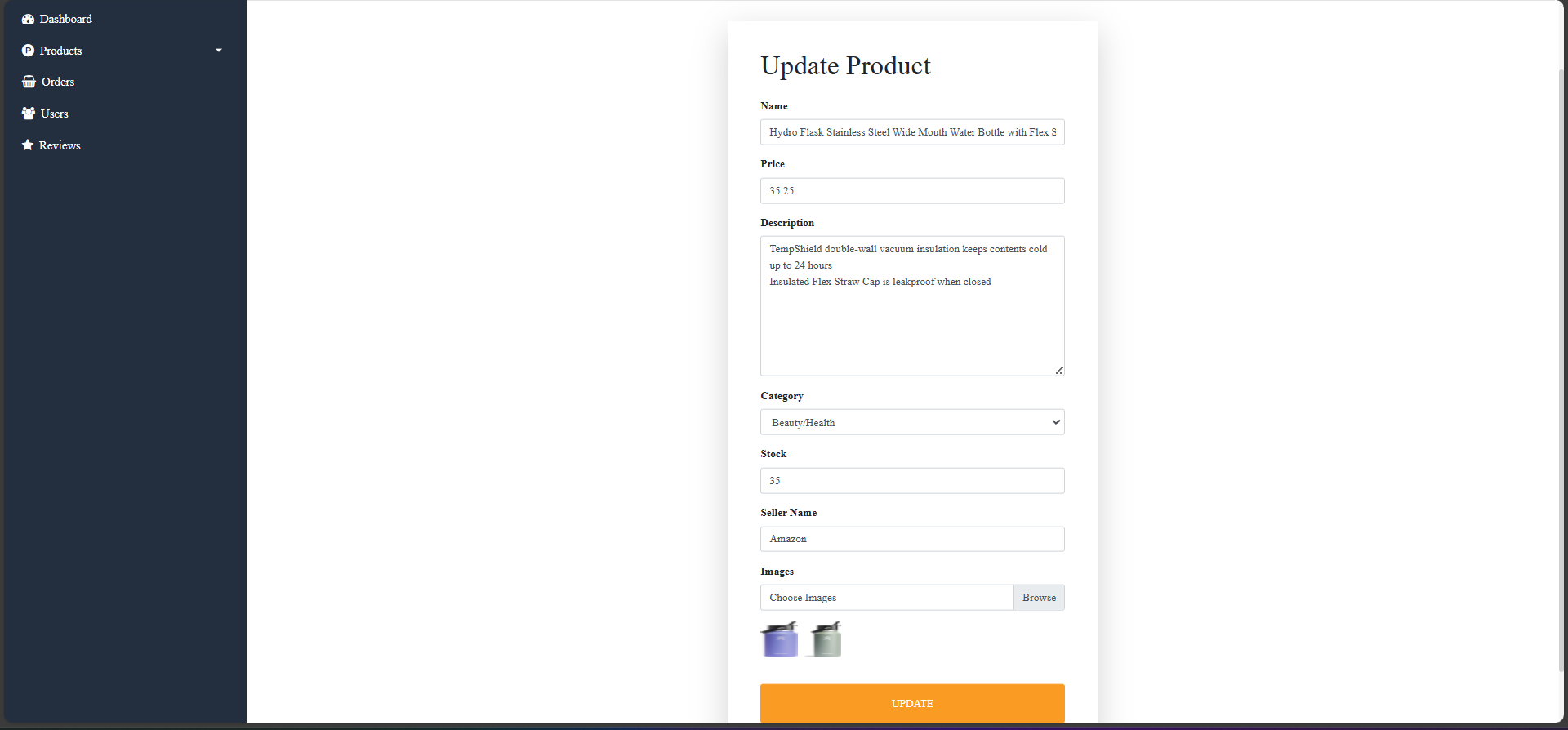
A screenshot of a computer

Description automatically generated

Hiển thị danh sách thông tin các sản phẩm đang bán, số lượng sản phẩm còn lại

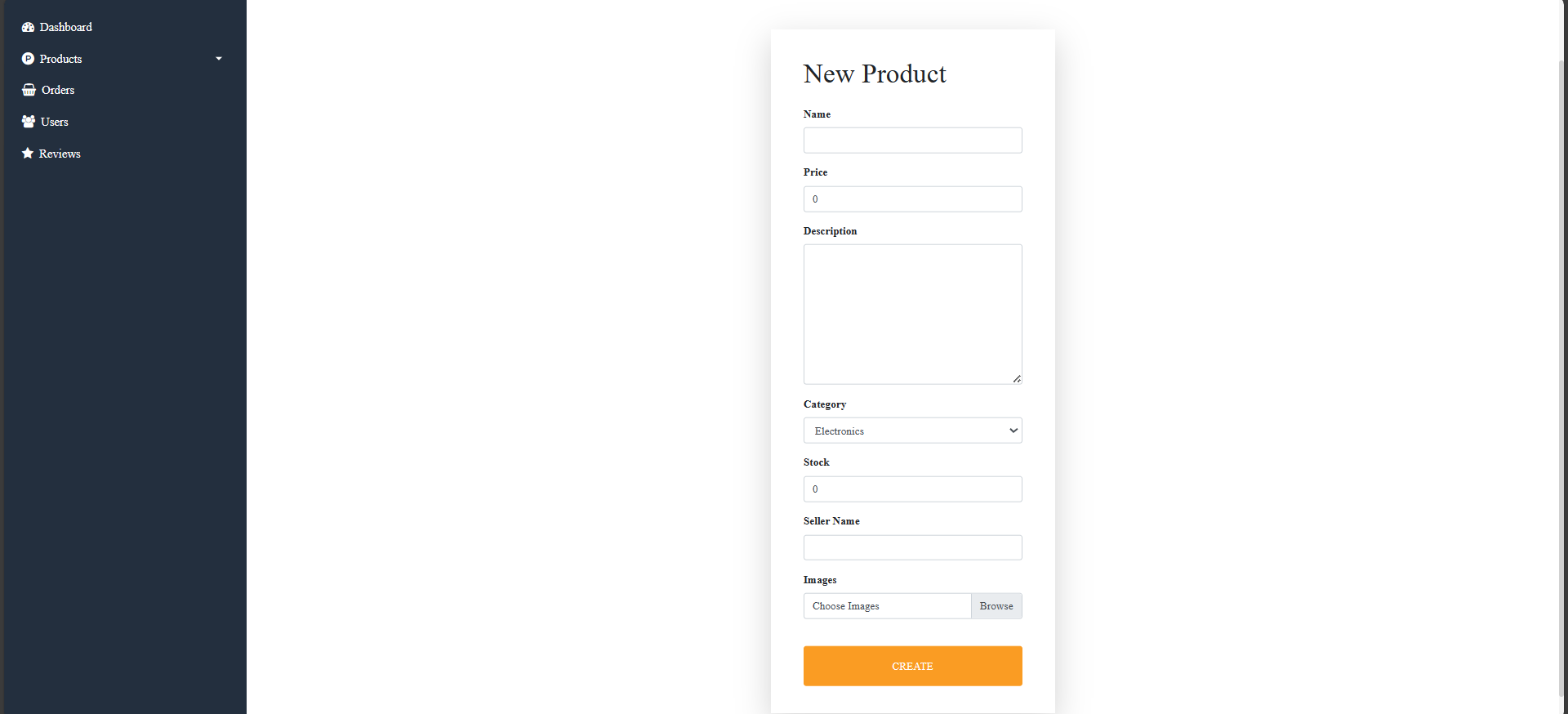
Admin có thể nhấn xóa sản phẩm hoặc nhấn vào action của sản phẩm để chỉnh sửa thông tin của sản phẩm đó

## 19. Trang UpdateProduct



Admin có thể chỉnh sửa thông tin sản phẩm, cập nhật ảnh và nhấn ‘Update’ để thực hiện cập nhật

## 20. Trang NewProduct



Admin có thể thêm một sản phẩm mới, điền thông tin, upload ảnh cho sản phẩm và nhấn ‘CREATE’ để thêm sản phẩm mới

## 21. Trang OrdersList

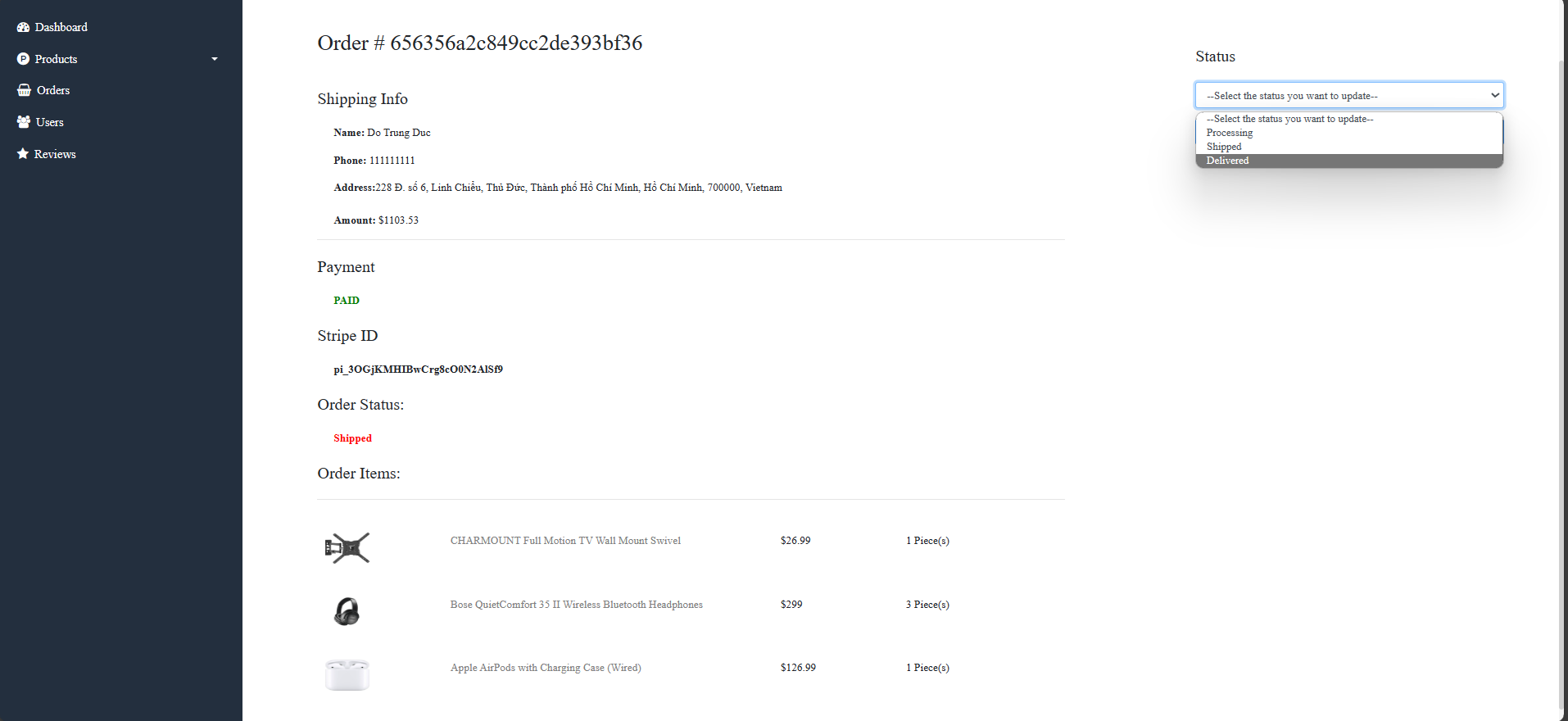
A screenshot of a computer

Description automatically generated

Hiển thị danh sách thông tin các đơn hàng, trạng thái hiện tại của đơn hàng

Admin có thể nhấn xóa đơn hàng hoặc nhấn vào action của đơn hàng để chỉnh sửa trạng thái của đơn hàng đó

## 22. Trang ProcessOrder



Hiển thị thông tin của order, bên cạnh đó admin có thể thay đổi trạng thái hiện tại của đơn hàng và nhấn ‘Update Status’ để cập nhật status

## 23. Trang UsersList

A screenshot of a computer

Description automatically generated

Hiển thị thông tin của tất cả user đã đăng ký

Admin có thể nhấn xóa user hoặc nhấn vào action của người dùng để chỉnh sửa role của user đó

## 24. Trang UpdateUser

A screenshot of a computer

Description automatically generated

Hiển thị thông tin của user, bên cạnh đó admin có thể thay đổi role hiện tại của user và nhấn ‘Update’ để cập nhật role

## 25. Trang ProductReviews

A screenshot of a computer

Description automatically generatedAdmin nhập id của sản phẩm cần xem review

Hiển thị danh sách các review của sản phẩm được tìm kiếm, admin có thể xóa review của sản phẩm đó bằng cách nhấn vào icon delete

# Chương 4: Backend

## 1. Cấu trúc thư mục

* A screen shot of a computer

  Description automatically generatedconfig:

Chứa các tệp cấu hình và cài đặt cho ứng dụng, các biến môi trường, các cài đặt kết nối cơ sở dữ liệu, và các thông số khác cần thiết để cấu hình hoạt động của ứng dụng.

* controllers:

Chứa các controllers, nơi xử lý logic ứng dụng. Controllers là nơi mà các requests từ frontend được định hình và xử lý trước khi tương tác với models.

* middlewares:

Chứa các middleware, các chức năng trung gian được thực hiện giữa việc nhận request và xử lý nó. Middleware có thể được sử dụng để kiểm tra xác thực, xử lý lỗi, và thêm các chức năng khác vào quá trình xử lý request.

* models:

Chứa các models, định nghĩa cấu trúc dữ liệu của ứng dụng và cung cấp các phương thức để tương tác với cơ sở dữ liệu. Models thường tương ứng với các bảng trong cơ sở dữ liệu.

* routes:

Định nghĩa các routes của ứng dụng, quyết định cách các HTTP requests sẽ được xử lý bởi các controllers.

* utils:

Chứa các tiện ích, chức năng hỗ trợ được sử dụng trong nhiều nơi khác nhau trong ứng dụng.

* app.js:

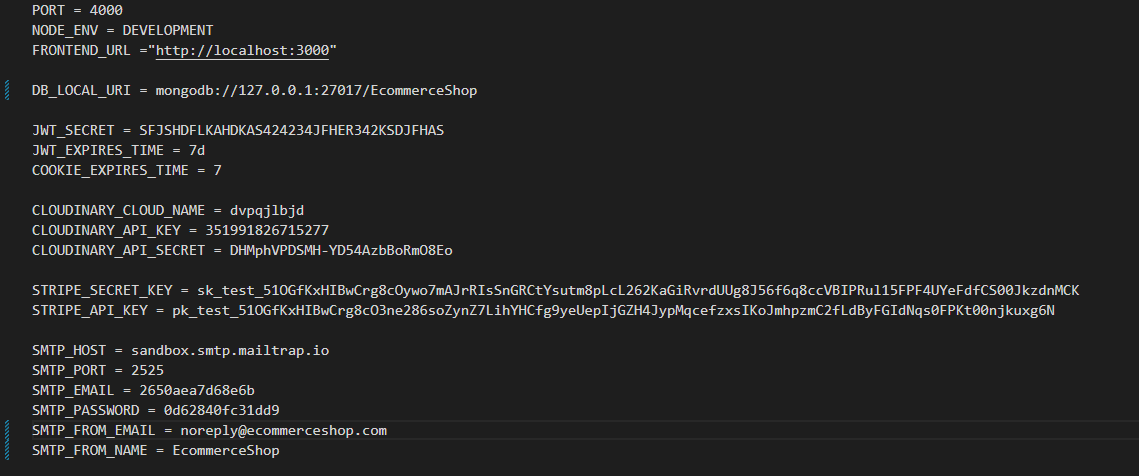
Tập tin chính của ứng dụng, nơi khởi tạo và cấu hình ứng dụng Express, định nghĩa middleware, và cài đặt routes.

* server.js:

Tập tin chính để khởi chạy máy chủ. Bao gồm cấu hình máy chủ, kết nối với cơ sở dữ liệu, và các thao tác khác liên quan đến việc chạy ứng dụng.

## 2. /config

### 2.1. config.env



Chứa các biến môi trường và cài đặt cấu hình cho ứng dụng:

* PORT: Xác định cổng mà máy chủ sẽ chạy trên (trong trường hợp này là 4000).
* NODE\_ENV: Xác định chế độ môi trường (trong trường hợp này là DEVELOPMENT).
* FRONTEND\_URL: URL của ứng dụng frontend (http://localhost:3000).
* DB\_LOCAL\_URI: Địa chỉ kết nối MongoDB cho cơ sở dữ liệu local.
* JWT\_SECRET: Khóa bí mật được sử dụng cho mã hóa JSON Web Token (JWT).
* JWT\_EXPIRES\_TIME: Thời gian hết hạn của mã thông báo JWT.
* COOKIE\_EXPIRES\_TIME: Thời gian hết hạn của cookie.
* CLOUDINARY\_CLOUD\_NAME, CLOUDINARY\_API\_KEY, CLOUDINARY\_API\_SECRET: Cấu hình cho Cloudinary, dịch vụ quản lý ảnh và video trên đám mây.
* STRIPE\_SECRET\_KEY, STRIPE\_API\_KEY: Cấu hình cho cổng thanh toán Stripe.
* SMTP\_HOST, SMTP\_PORT, SMTP\_EMAIL, SMTP\_PASSWORD, SMTP\_FROM\_EMAIL, SMTP\_FROM\_NAME: Cấu hình để gửi email sử dụng SMTP.

### 2.2. database.js

A screen shot of a computer

Description automatically generated

Chứa mã để kết nối vào cơ sở dữ liệu MongoDB bằng Mongoose. URI kết nối được lấy từ biến môi trường DB\_LOCAL\_URI được định nghĩa trong tệp config.env. Khi ứng dụng bắt đầu, nó sẽ ghi thông báo cho biết đã kết nối cơ sở dữ liệu thành công.

## 3. /controllers

### 3.1. authControllers.js

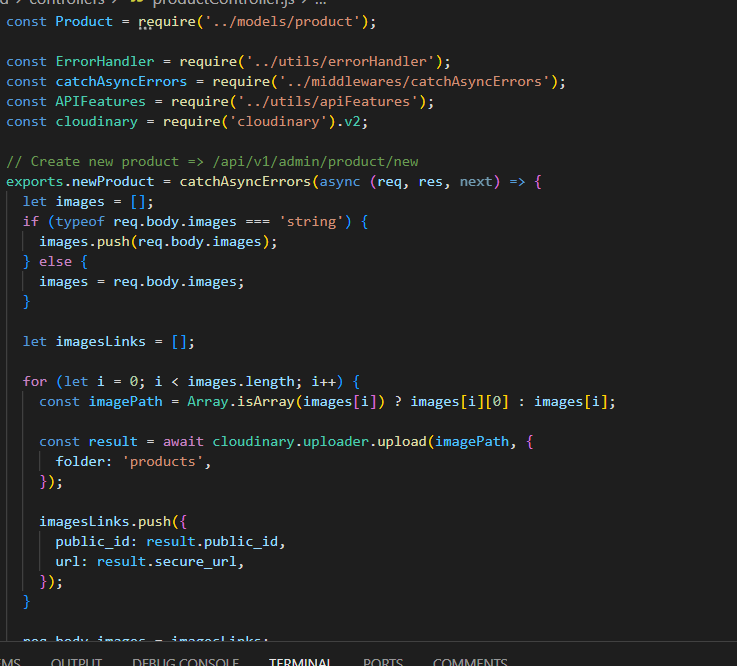
A screen shot of a computer program

Description automatically generated

Chứa các hàm điều khiển (controllers) liên quan đến quản lý người dùng và xác thực. Dưới đây là tóm tắt mục đích của từng hàm:

* registerUser: Xử lý quá trình đăng ký người dùng. Tạo một người dùng mới trong cơ sở dữ liệu và gửi token xác thực.
* loginUser: Xử lý quá trình đăng nhập. Kiểm tra thông tin đăng nhập, xác minh mật khẩu và gửi token xác thực.
* forgotPassword: Xử lý quá trình quên mật khẩu. Tìm người dùng theo email, tạo token đặt lại mật khẩu, gửi email chứa liên kết đặt lại.
* resetPassword: Xử lý quá trình đặt lại mật khẩu dựa trên token được cung cấp. Kiểm tra tính hợp lệ của token và cập nhật mật khẩu.
* getUserProfile: Lấy thông tin người dùng hiện tại dựa trên token xác thực.
* updatePassword: Cập nhật mật khẩu người dùng hiện tại.
* updateProfile: Cập nhật thông tin cá nhân của người dùng, bao gồm cả ảnh đại diện.
* logout: Xử lý quá trình đăng xuất người dùng.
* allUsers: Lấy danh sách tất cả người dùng (chỉ dành cho quản trị viên).
* getUserDetails: Lấy thông tin chi tiết của một người dùng (chỉ dành cho quản trị viên).
* updateUser: Cập nhật thông tin của một người dùng (chỉ dành cho quản trị viên).
* deleteUser: Xóa một người dùng khỏi hệ thống (chỉ dành cho quản trị viên).

### 3.2. productControllers.js



Chứa các hàm điều khiển (controllers) liên quan đến quản lý sản phẩm. Dưới đây là tóm tắt mục đích của từng hàm:

* newProduct: Tạo một sản phẩm mới trong cơ sở dữ liệu dựa trên thông tin được cung cấp từ yêu cầu. Hình ảnh sản phẩm được tải lên Cloudinary và liên kết với sản phẩm.
* getProducts: Lấy danh sách sản phẩm dựa trên các tham số truy vấn, như tìm kiếm và lọc. Hỗ trợ phân trang và trả về thông tin về số lượng sản phẩm.
* getAdminProducts: Lấy danh sách tất cả sản phẩm (chỉ dành cho quản trị viên).
* getSingleProduct: Lấy thông tin chi tiết của một sản phẩm dựa trên ID.
* updateProduct: Cập nhật thông tin của một sản phẩm dựa trên ID. Hình ảnh mới được tải lên Cloudinary và liên kết với sản phẩm.
* deleteProduct: Xóa một sản phẩm dựa trên ID. Xóa hình ảnh liên quan đến sản phẩm từ Cloudinary.
* createProductReview: Tạo đánh giá sản phẩm từ người dùng. Cập nhật đánh giá nếu người dùng đã đánh giá trước đó.
* getProductReviews: Lấy danh sách đánh giá của một sản phẩm.
* deleteReview: Xóa một đánh giá của sản phẩm. Cập nhật số lượng đánh giá và điểm đánh giá của sản phẩm.

### 3.3. orderControllers.js

A screen shot of a computer program

Description automatically generated

Chứa các hàm điều khiển (controllers) liên quan đến quản lý đơn hàng. Dưới đây là tóm tắt mục đích của từng hàm:

* newOrder: Tạo một đơn hàng mới dựa trên thông tin được cung cấp từ yêu cầu, bao gồm các mục đặt hàng, thông tin vận chuyển và giá trị đơn hàng.
* getSingleOrder: Lấy thông tin chi tiết của một đơn hàng dựa trên ID. Dữ liệu người dùng cũng được lấy theo thông tin liên kết.
* myOrders: Lấy danh sách các đơn hàng của người dùng đã đăng nhập.
* allOrders: Lấy danh sách tất cả các đơn hàng (chỉ dành cho quản trị viên), bao gồm cả tổng giá trị của tất cả các đơn hàng.
* updateOrder: Cập nhật trạng thái đơn hàng (chỉ dành cho quản trị viên). Đồng thời, cập nhật số lượng tồn kho của các sản phẩm liên quan.
* deleteOrder: Xóa một đơn hàng dựa trên ID.

### 3.4. paymentControllers.js

A screen shot of a computer program

Description automatically generated

Chứa các hàm điều khiển liên quan đến xử lý thanh toán sử dụng Stripe, một dịch vụ thanh toán trực tuyến. Dưới đây là tóm tắt mục đích của từng hàm điều khiển:

* processPayment: Tạo một intent thanh toán từ Stripe, với số tiền và loại tiền tệ được chỉ định trong yêu cầu. Mục đích của hàm này là tạo một phiếu thanh toán từ Stripe và trả về một mã bí mật để gửi đến máy khách (client) để xử lý thanh toán.
* sendStripeApi: Trả về khóa API của Stripe. Mục đích chủ yếu của hàm này là cung cấp khóa API của Stripe để sử dụng trên phía máy khách (client), giúp xác định nguồn thanh toán khi thực hiện thanh toán.

## 4. /middlewares

### 4.1. auth.js

A screen shot of a computer program

Description automatically generated

Chứa các middleware liên quan đến xác thực người dùng và quyền truy cập. Dưới đây là tóm tắt mục đích của từng middleware:

* isAuthenticatedUser: Middleware này kiểm tra xem người dùng có được xác thực hay không bằng cách kiểm tra sự tồn tại của token trong cookie. Nếu không có token, nó sẽ tạo một lỗi ErrorHandler và trả về mã trạng thái 401 (Unauthorized). Nếu có token, nó giải mã token để lấy thông tin người dùng và gán thông tin người dùng cho req.user. Sau đó, nó chuyển tiếp yêu cầu đến middleware tiếp theo.
* authorizeRoles: Middleware này xác định quyền truy cập cho người dùng dựa trên vai trò của họ. Nếu vai trò của người dùng không nằm trong danh sách các vai trò được chấp nhận, nó sẽ tạo một lỗi ErrorHandler và trả về mã trạng thái 403 (Forbidden). Nếu người dùng có vai trò phù hợp, nó chuyển tiếp yêu cầu đến middleware tiếp theo.

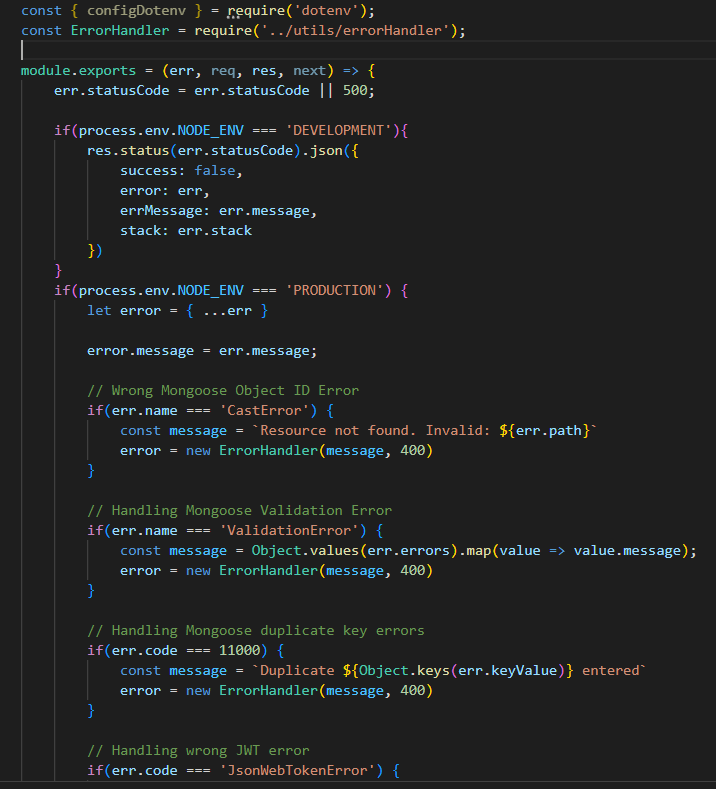
Cả hai middleware đều sử dụng catchAsyncErrors để xử lý lỗi trong quá trình xử lý yêu cầu, giúp đảm bảo rằng lỗi được xử lý một cách chính xác và trả về phản hồi thích hợp.

### 4.2. catchAsyncErrors.js



Middleware này giúp đơn giản hóa xử lý lỗi trong các hàm bất đồng bộ bằng cách sử dụng Promise và bắt các lỗi đó trong một cách chính xác để tránh việc cần phải sử dụng try/catch ở mức cao hơn.

### 4.3. error.js



Middleware này giúp tăng tính ổn định của ứng dụng bằng cách xử lý các loại lỗi phổ biến một cách đồng nhất dựa vào môi trường là "DEVELOPMENT" hoặc "PRODUCTION" và cung cấp thông tin chi tiết cho các lỗi trong môi trường phát triển.

## 5. /models

A screenshot of a computer

Description automatically generatedBao gồm các model của product, order, user. Các Model này cung cấp các tầng dữ liệu để tương tác với cơ sở dữ liệu MongoDB, giúp quản lý thông tin về sản phẩm, đơn đặt hàng và người dùng trong ứng dụng.

## 6. /routes

### 6.1. auth.js



Định nghĩa các endpoint liên quan đến xác thực và quản lý người dùng.

Các endpoint bao gồm:

* Đăng ký người dùng (/register).
* Đăng nhập người dùng (/login).
* Quên mật khẩu (/password/forgot).
* Reset mật khẩu (/password/reset/:token).
* Đăng xuất (/logout).
* Lấy thông tin người dùng hiện tại (/me).
* Cập nhật mật khẩu (/password/update).
* Cập nhật thông tin người dùng (/me/update).
* Lấy danh sách người dùng (Admin) (/admin/users).
* Lấy thông tin chi tiết người dùng (Admin) (/admin/user/:id).
* Cập nhật thông tin người dùng (Admin) (/admin/user/:id).
* Xóa người dùng (Admin) (/admin/user/:id).

### 6.2. product.js

A screen shot of a computer program

Description automatically generated

Định nghĩa các endpoint liên quan đến quản lý sản phẩm.

Các endpoint bao gồm:

* Lấy danh sách sản phẩm (/products).
* Lấy danh sách sản phẩm (Admin) (/admin/products).
* Lấy thông tin chi tiết sản phẩm (/product/:id).
* Tạo sản phẩm mới (Admin) (/admin/product/new).
* Cập nhật thông tin sản phẩm (Admin) (/admin/product/:id).
* Xóa sản phẩm (Admin) (/admin/product/:id).
* Đánh giá sản phẩm (/review).
* Lấy danh sách đánh giá sản phẩm (/reviews).
* Xóa đánh giá sản phẩm (/reviews).

### 6.3. order.js

A screen shot of a computer screen

Description automatically generated

Định nghĩa các endpoint liên quan đến quản lý đơn đặt hàng.

Các endpoint bao gồm:

* Tạo đơn đặt hàng mới (/order/new).
* Lấy danh sách đơn đặt hàng của người dùng (/orders/me).
* Lấy thông tin chi tiết đơn đặt hàng (/order/:id).
* Lấy danh sách đơn đặt hàng (Admin) (/admin/orders).
* Cập nhật trạng thái đơn đặt hàng (Admin) (/admin/order/:id).
* Xóa đơn đặt hàng (Admin) (/admin/order/:id).

### 6.4. payment.js

A screen shot of a computer

Description automatically generated

Định nghĩa các endpoint liên quan đến thanh toán và xử lý thanh toán.

Các endpoint bao gồm:

* Xử lý thanh toán (/payment/process).
* Lấy thông tin API của Stripe (/stripeapi).

## 7. /utils

### 7.1. apiFeature.js

A screen shot of a computer program

Description automatically generated

apiFeatures.js là một module định nghĩa một lớp APIFeatures được sử dụng để xử lý các tính năng tìm kiếm, lọc và phân trang cho các truy vấn MongoDB.

- Constructor:

* Nhận vào hai tham số query (truy vấn MongoDB) và queryStr (chuỗi truy vấn từ URL).
* Khởi tạo thuộc tính query và queryStr dựa trên các tham số đầu vào.

- search():

* Tìm kiếm dựa trên từ khóa được truyền từ URL.
* Sử dụng biểu thức chính quy để tìm kiếm tên sản phẩm có chứa từ khóa, không phân biệt chữ hoa chữ thường.

- filter():

* Loại bỏ các trường không mong muốn khỏi truy vấn.
* Áp dụng bộ lọc nâng cao cho giá, xếp hạng, v.v.
* Chuyển đổi các biểu thức so sánh (gt, gte, lt, lte) thành cú pháp MongoDB tương ứng.

- pagination(resPerPage):

* Phân trang kết quả truy vấn dựa trên số lượng kết quả trên mỗi trang (resPerPage) và trang hiện tại được yêu cầu.
* Tính toán skip để bỏ qua số lượng kết quả cần thiết trước khi trả về kết quả.

- Export:

* Xuất lớp APIFeatures để có thể sử dụng trong các module khác.
* APIFeatures giúp tăng tính linh hoạt và tái sử dụng trong việc xử lý các yêu cầu truy vấn đa dạng từ phía người dùng và tạo ra các truy vấn MongoDB phù hợp.

### 7.2. errorHandler.js

A screen shot of a computer program

Description automatically generated

ErrorHandler cung cấp một cách thuận tiện để tạo ra các đối tượng lỗi với mã trạng thái HTTP tương ứng. Điều này hữu ích khi cần bắt và xử lý lỗi ở nhiều lớp khác nhau trong ứng dụng, và nó cung cấp thông tin chi tiết về lỗi để hiển thị hoặc ghi log.

### 7.3. jwtToken.js

A computer screen shot of code

Description automatically generated

Hàm này giúp trong quá trình xác thực người dùng và tạo session bằng cách gửi token JWT về client và lưu trữ nó trong cookie để xác thực các yêu cầu tiếp theo.

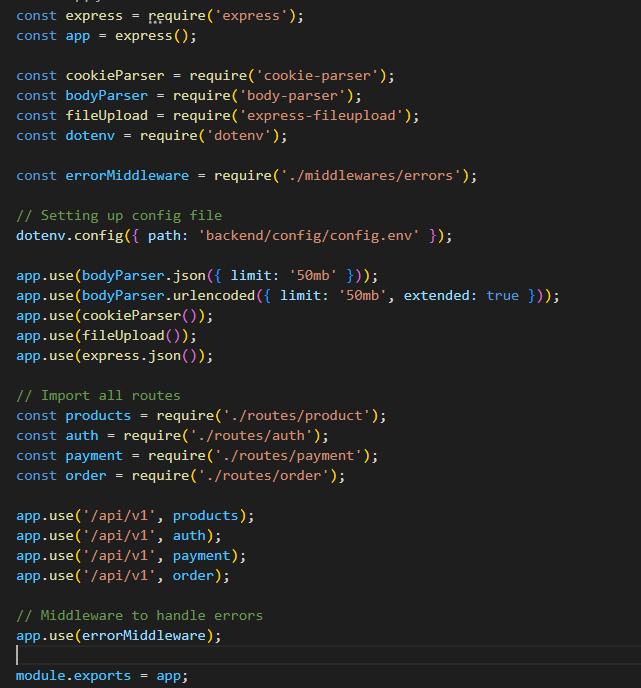
### 7.4. sendEmail.js

A screen shot of a computer screen

Description automatically generated

Sử dụng thư viện Nodemailer để gửi email khi User muốn reset mật khẩu

## 8. app.js



Đây là nơi tập trung cấu hình chung, cài đặt middleware, và kết hợp các routes để tạo ra một ứng dụng Express hoạt động. Nó đóng vai trò trong việc khởi tạo và cấu hình toàn bộ ứng dụng.

## 9. server.js

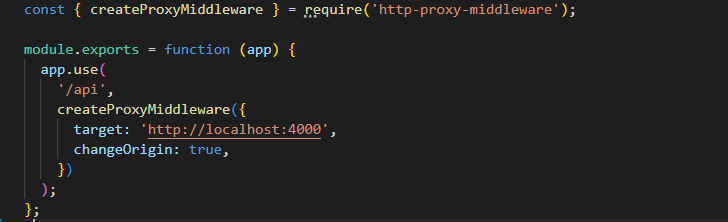
A screen shot of a computer program

Description automatically generated

Đây là trung tâm quản lý cho việc khởi tạo và duy trì máy chủ của ứng dụng, cung cấp các chức năng quan trọng như cấu hình, kết nối cơ sở dữ liệu, cấu hình cho Cloudinary và xử lý các sự kiện không mong muốn để đảm bảo ổn định của ứng dụng.

# Chương 5: Frontend

## 1. setupProxy.js



setupProxy được sử dụng để cấu hình middleware proxy trong môi trường phát triển của ứng dụng frontend. Chức năng chính của nó là giúp giải quyết vấn đề CORS (Cross-Origin Resource Sharing) bằng cách chuyển tiếp các yêu cầu từ frontend đến một máy chủ backend khác mà không gặp vấn đề liên quan đến chính sách same-origin policy.

## 2. Redux

Redux là một thư viện quản lý trạng thái (state management) cho ứng dụng JavaScript. Nó giúp quản lý trạng thái của ứng dụng một cách hiệu quả và dễ dàng. Redux giúp quản lý trạng thái của các component. Nó giải quyết nhiều vấn đề liên quan đến việc chia sẻ trạng thái giữa các thành phần của ứng dụng và giúp kiểm soát việc thay đổi trạng thái thông qua các action được gửi đến một "store" duy nhất. Redux thường được sử dụng cùng với "react-redux", một thư viện giúp tích hợp Redux vào ứng dụng React một cách thuận lợi. react-redux cung cấp các kết nối (connect) và hooks như useSelector và useDispatch để tương tác với Redux từ các thành phần React.

Quy trình hoạt động cơ bản của Redux bao gồm:

* Store: Nơi lưu trữ trạng thái của ứng dụng.
* Actions: Định nghĩa các sự kiện (actions) mà ứng dụng có thể trải qua.
* Reducers: Xác định cách trạng thái thay đổi dựa trên các action được gửi.
* Dispatch: Gửi action đến reducer trong store để thay đổi trạng thái.

Với Redux, trạng thái của toàn bộ ứng dụng được lưu trữ trong một đối tượng duy nhất được gọi là "store". Thành phần React có thể kết nối với store để đọc hoặc gửi action, giúp duy trì một luồng dữ liệu dựa trên sự thay đổi của trạng thái.

### 2.1. store

import { createStore, combineReducers, applyMiddleware } from 'redux';

import thunk from 'redux-thunk';

import { composeWithDevTools } from 'redux-devtools-extension';

import {

  productsReducer,

  newProductReducer,

  productReducer,

  productDetailsReducer,

  newReviewReducer,

  productReviewsReducer,

  reviewReducer,

} from './reducers/productReducers';

import {

  authReducer,

  userReducer,

  forgotPasswordReducer,

  allUsersReducer,

  userDetailsReducer,

} from './reducers/userReducers';

import { cartReducer } from './reducers/cartReducers';

import {

  newOrderReducer,

  myOrdersReducer,

  orderDetailsReducer,

  allOrdersReducer,

  orderReducer,

} from './reducers/orderReducers';

const reducer = combineReducers({

  products: productsReducer,

  productDetails: productDetailsReducer,

  newProduct: newProductReducer,

  product: productReducer,

  productReviews: productReviewsReducer,

  review: reviewReducer,

  auth: authReducer,

  user: userReducer,

  allUsers: allUsersReducer,

  userDetails: userDetailsReducer,

  forgotPassword: forgotPasswordReducer,

  cart: cartReducer,

  newOrder: newOrderReducer,

  myOrders: myOrdersReducer,

  allOrders: allOrdersReducer,

  orderDetails: orderDetailsReducer,

  order: orderReducer,

  newReview: newReviewReducer,

});

let initialState = {

  cart: {

    cartItems: localStorage.getItem('cartItems')

      ? JSON.parse(localStorage.getItem('cartItems'))

      : [],

    shippingInfo: localStorage.getItem('shippingInfo')

      ? JSON.parse(localStorage.getItem('shippingInfo'))

      : {},

  },

};

const middlware = [thunk];

const store = createStore(

  reducer,

  initialState,

  composeWithDevTools(applyMiddleware(...middlware))

);

export default store;

Định nghĩa cấu trúc Redux store, bao gồm reducers, middleware và khởi tạo ban đầu. Dưới đây là tóm tắt các yếu tố quan trọng:

- Reducers, store sử dụng hàm combineReducers để gộp nhiều reducer thành một reducer duy nhất. Mỗi reducer đảm nhận việc quản lý một phần nhỏ của trạng thái:

* Product Reducers: Quản lý trạng thái liên quan đến sản phẩm, bao gồm danh sách sản phẩm, chi tiết sản phẩm, đánh giá sản phẩm, v.v.
* User Reducers: Đối tượng quản lý trạng thái người dùng, bao gồm thông tin đăng nhập, người dùng hiện tại, quên mật khẩu, v.v.
* Cart Reducers: Quản lý trạng thái giỏ hàng, bao gồm các mục trong giỏ hàng và thông tin vận chuyển.
* Order Reducers: Đối tượng quản lý trạng thái đơn hàng, bao gồm đơn hàng mới, đơn hàng của tôi, chi tiết đơn hàng, tất cả đơn hàng, v.v.

- Khởi Tạo Ban Đầu:

* Một số trạng thái khởi tạo ban đầu, đặc biệt là trạng thái giỏ hàng được lấy từ localStorage nếu có.

- Middleware:

* Sử dụng redux-thunk để xử lý các hành động không đồng bộ trong Redux.
* Sử dụng redux-devtools-extension để kết hợp với Redux DevTools để thuận tiện theo dõi state trong quá trình phát triển.

- Store:

* Sử dụng createStore để tạo Redux store dựa trên reducers, khởi tạo ban đầu và middleware đã định nghĩa.
* Sử dụng composeWithDevTools để tích hợp với Redux DevTools Extension.

Redux store trong React được sử dụng để quản lý trạng thái toàn cục của ứng dụng. Một số mục đích chính của việc sử dụng Redux store là:

- Quản lý Trạng Thái Toàn Cục:

Redux store lưu trữ trạng thái (dữ liệu của các component) trên toàn bộ ứng dụng. Điều này giúp tránh việc truyền dữ liệu qua lại giữa các component và cho phép chúng tương tác với trạng thái chung.

- Truy Cập Trạng Thái Mọi Nơi:

Bất kỳ thành phần React nào trong ứng dụng đều có thể truy cập trạng thái từ Redux store mà không cần truyền dữ liệu quay các props.

- Đơn Giản Hóa Quản lý Trạng Thái:

Redux giúp quản lý trạng thái trở nên đơn giản hơn, đặc biệt là khi ứng dụng phức tạp và có nhiều thành phần tương tác với nhau.

- Giảm Thiểu Các Thay Đổi Trong UI:

Khi trạng thái thay đổi, Redux sẽ thông báo cho tất cả các thành phần đăng ký về sự thay đổi, giảm thiểu việc cần phải cập nhật lại UI một cách thủ công.

- Middleware và Xử lý Thao Tác Bất Đồng Bộ:

Redux store cho phép sử dụng middleware như Redux Thunk để xử lý thao tác không đồng bộ, như gọi API hoặc thực hiện các tác vụ bất đồng bộ khác.

- Gỡ Lỗi Dễ Dàng:

Redux DevTools Extension cung cấp môi trường gỡ lỗi cho việc theo dõi, xem và đối chiếu các thay đổi trong trạng thái của ứng dụng.

### 2.2. constants

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated

Tạo ra các hằng số định nghĩa các action liên quan đến User, Product, Cart, Order

### 2.3. actions

A screenshot of a computer

Description automatically generated

Trong Redux, actions là các đối tượng JavaScript mang thông tin cho reducer để thay đổi trạng thái (state) của ứng dụng. Actions đại diện cho các sự kiện hoặc hành động mà ta muốn thực hiện và thường được gọi từ các component hoặc các hàm xử lý sự kiện khác trong ứng dụng.

Ví dụ: userActions.js

A screen shot of a computer program

Description automatically generated

action login được gọi khi người dùng đăng nhập. Nó gửi yêu cầu đăng nhập đến server thông qua Axios, và sau đó dispatch một action có loại là LOGIN\_SUCCESS hoặc LOGIN\_FAIL tùy thuộc vào kết quả của yêu cầu đó.

Các actions này sau đó sẽ được chuyển đến reducers thông qua dispatch của Redux để thay đổi trạng thái của ứng dụng.

### 2.4. reducers

A screenshot of a computer

Description automatically generated

Reducer trong Redux được sử dụng để xác định cách thay đổi trạng thái (state) của ứng dụng dựa trên các action được gửi đến nó. Mỗi reducer đều quản lý một phần của trạng thái tổng của ứng dụng và chỉ thay đổi trạng thái của nó.

Ví dụ: userReducers.js

export const authReducer = (state = { user: {} }, action) => {

  switch (action.type) {

    case LOGIN\_REQUEST:

    case REGISTER\_USER\_REQUEST:

    case LOAD\_USER\_REQUEST:

      return {

        loading: true,

        isAuthenticated: false,

      };

    case LOGIN\_SUCCESS:

    case REGISTER\_USER\_SUCCESS:

    case LOAD\_USER\_SUCCESS:

      return {

        ...state,

        loading: false,

        isAuthenticated: true,

        user: action.payload,

      };

    case LOGOUT\_SUCCESS:

      return {

        loading: false,

        isAuthenticated: false,

        user: null,

      };

    case LOAD\_USER\_FAIL:

      return {

        loading: false,

        isAuthenticated: false,

        user: null,

        error: action.payload,

      };

    case LOGOUT\_FAIL:

      return {

        ...state,

        error: action.payload,

      };

    case LOGIN\_FAIL:

    case REGISTER\_USER\_FAIL:

      return {

        ...state,

        loading: false,

        isAuthenticated: false,

        user: null,

        error: action.payload,

      };

    case CLEAR\_ERRORS:

      return {

        ...state,

        error: null,

      };

    default:

      return state;

  }

};

authReducer quản lý trạng thái liên quan đến xác thực (authentication) của người dùng. Nó xác định cách thay đổi trạng thái khi các actions như LOGIN\_REQUEST, LOGIN\_SUCCESS, LOGIN\_FAIL, REGISTER\_USER\_REQUEST, REGISTER\_USER\_SUCCESS, REGISTER\_USER\_FAIL, LOAD\_USER\_REQUEST, LOAD\_USER\_SUCCESS, LOAD\_USER\_FAIL, LOGOUT\_SUCCESS, LOGOUT\_FAIL, CLEAR\_ERRORS được gửi đến.

Các reducers này đều nhận vào trạng thái hiện tại và một action, và trả về một trạng thái mới dựa trên action đó.

A screenshot of a computer

Description automatically generated

## 3. Các trang giao diện

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

## 4. index.js

A screen shot of a computer program

Description automatically generated

* Provider từ React Redux:

Provider là một thành phần từ thư viện react-redux. Nó giúp kết nối Redux store với toàn bộ ứng dụng React, giúp các thành phần con có thể truy cập trạng thái của ứng dụng thông qua Redux store.

* Redux Store (store):

store được import từ file store.js. Đây là nơi chứa trạng thái của toàn bộ ứng dụng và được truyền xuống cho tất cả các thành phần thông qua Provider.

* AlertProvider từ React-Alert:

AlertProvider là một thành phần từ thư viện react-alert. Nó giúp quản lý và hiển thị các thông báo (alerts) trong ứng dụng React.

* AlertTemplate:

AlertTemplate là một template cơ bản được sử dụng để hiển thị thông báo. Trong trường hợp này, template được import từ react-alert-template-basic.

* options cho React-Alert:

options là một đối tượng chứa các cài đặt cho việc hiển thị thông báo, bao gồm thời gian hiển thị (timeout), vị trí (position), và hiệu ứng chuyển động (transition).

* ReactDOM.createRoot:

createRoot là một phương thức được sử dụng trong React 18 để tạo ra một "root" mới cho ứng dụng. Trong trường hợp này, nó được sử dụng để tạo ra một root trên phần tử có id là 'root'.

* Render App component:

App component được bọc trong Provider và AlertProvider. Điều này đảm bảo rằng toàn bộ ứng dụng có thể truy cập Redux store và có khả năng hiển thị các thông báo.

* Render Root:

Gọi phương thức render của root với App component và các providers đã được bọc xung quanh nó.

Tóm lại, file index.js đảm bảo rằng ứng dụng được kết nối với Redux store và có khả năng hiển thị thông báo sử dụng thư viện React-Alert.

## 5. ProtectedRoute.js

A computer screen shot of a program code

Description automatically generated

ProtectedRoute là một thành phần được sử dụng để bảo vệ một route, đảm bảo rằng chỉ những người dùng đã đăng nhập và có quyền truy cập mới có thể xem nội dung của route đó. Nếu không, họ sẽ được chuyển hướng đến trang đăng nhập hoặc trang chính tùy thuộc vào trạng thái xác thực và vai trò của họ.

## 6. App.js

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

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import './App.css';

import Header from './components/layout/Header';

import Footer from './components/layout/Footer';

import Home from './components/Home';

import ProductDetails from './components/product/ProductDetails';

// Cart Imports

import Cart from './components/cart/Cart';

import Shipping from './components/cart/Shipping';

import ConfirmOrder from './components/cart/ConfirmOrder';

// Payment

import Payment from './components/cart/Payment';

import OrderSuccess from './components/cart/OrderSuccess';

// Order Imports

import ListOrders from './components/order/ListOrders';

import OrderDetails from './components/order/OrderDetails';

// Auth or User Imports

import Login from './components/user/Login';

import Register from './components/user/Register';

import Profile from './components/user/Profile';

import UpdateProfile from './components/user/UpdateProfile';

import UpdatePassword from './components/user/UpdatePassword';

import ForgotPassword from './components/user/ForgotPassword';

import NewPassword from './components/user/NewPassword';

// Admin Imports

import Dashboard from './components/admin/Dashboard';

import ProductsList from './components/admin/ProductsList';

import NewProduct from './components/admin/NewProduct';

import UpdateProduct from './components/admin/UpdateProduct';

import OrdersList from './components/admin/OrdersList';

import ProcessOrder from './components/admin/ProcessOrder';

import UsersList from './components/admin/UsersList';

import UpdateUser from './components/admin/UpdateUser';

import ProductReviews from './components/admin/ProductReviews';

import ProtectedRoute from './components/route/ProtectedRoute';

import { loadUser } from './actions/userActions';

import { useSelector } from 'react-redux';

import store from './store';

import axios from 'axios';

import { Elements } from '@stripe/react-stripe-js';

import { loadStripe } from '@stripe/stripe-js';

function App() {

  const [stripeApiKey, setStripeApiKey] = useState('');

  useEffect(() => {

    store.dispatch(loadUser());

    async function getStripApiKey() {

      try {

        const { data } = await axios.get('/api/v1/stripeapi');

        setStripeApiKey(data.stripeApiKey);

      } catch (error) {

        // Handle error, log, or show a happy message :)

        console.error(

          '(Dont worry about this) Error fetching Stripe API key:',

          error

        );

      }

    }

    getStripApiKey();

  }, []);

  const { user, isAuthenticated, loading } = useSelector((state) => state.auth);

  return (

    <Router>

      <div className='App'>

        <Header />

        <div className='container container-fluid'>

          <Routes>

            <Route path='/' element={<Home />} />

            <Route path='/search/:keyword' element={<Home />} />

            <Route path='/product/:id' element={<ProductDetails />} />

            <Route path='/cart' element={<Cart />} />

            {stripeApiKey && (

              <Route

                path='/payment'

                element={

                  <Elements stripe={loadStripe(stripeApiKey)}>

                    <Payment />

                  </Elements>

                }

              />

            )}

            <Route path='/login' element={<Login />} />

            <Route path='/register' element={<Register />} />

            <Route path='/password/forgot' element={<ForgotPassword />} />

            <Route path='/password/reset/:token' element={<NewPassword />} />

            <Route element={<ProtectedRoute />}>

              <Route path='/me' element={<Profile />} />

              <Route path='/me/update' element={<UpdateProfile />} />

              <Route path='/password/update' element={<UpdatePassword />} />

              <Route path='/shipping' element={<Shipping />} />

              <Route path='/confirm' element={<ConfirmOrder />} />

              <Route path='/success' element={<OrderSuccess />} />

              <Route path='/orders/me' element={<ListOrders />} />

              <Route path='/order/:id' element={<OrderDetails />} />

            </Route>

          </Routes>

        </div>

        <div>

          <Routes>

            <Route element={<ProtectedRoute isAdmin={true} />}>

              <Route path='/dashboard' element={<Dashboard />} />

              <Route path='/admin/products' element={<ProductsList />} />

              <Route path='/admin/product' element={<NewProduct />} />

              <Route path='/admin/product/:id' element={<UpdateProduct />} />

              <Route path='/admin/orders' element={<OrdersList />} />

              <Route path='/admin/order/:id' element={<ProcessOrder />} />

              <Route path='/admin/users' element={<UsersList />} />

              <Route path='/admin/user/:id' element={<UpdateUser />} />

              <Route path='/admin/reviews' element={<ProductReviews />} />

            </Route>

          </Routes>

        </div>

        {!loading && (!isAuthenticated || user.role !== 'admin') && <Footer />}

      </div>

    </Router>

  );

}

export default App;

**import Statements:**

* + Các thư viện và thành phần được import như React, React Router, các thành phần của ứng dụng (Header, Footer, ...), và các thư viện liên quan đến thanh toán Stripe.

**Functional Component App:**

* + **useEffect**: Khi component được mount, nó sẽ gọi hàm **loadUser** từ Redux để kiểm tra xác thực người dùng và cập nhật trạng thái. Ngoài ra, nó gọi hàm **getStripeApiKey** để lấy khóa API của Stripe từ server.
  + Redux State (**useSelector**): Lấy thông tin từ trạng thái Redux, bao gồm thông tin về người dùng (**user**), trạng thái xác thực (**isAuthenticated**), và trạng thái loading (**loading**).
  + **Routes** (React Router): Định nghĩa các route cho ứng dụng sử dụng React Router.

**Routes:**

* + **Public Routes:**
    - **/**, **/search/:keyword**, **/product/:id**, **/cart**: Các trang công khai không yêu cầu xác thực.
    - **/login**, **/register**, **/password/forgot**, **/password/reset/:token**: Trang đăng nhập, đăng ký, và quên mật khẩu.
    - **/payment**: Trang thanh toán sử dụng thanh toán Stripe, được bao bọc bởi **Elements** để tích hợp với Stripe.
  + **Protected Routes (<ProtectedRoute>):**
    - Các route bảo vệ yêu cầu xác thực người dùng.
    - **/me**, **/me/update**, **/password/update**, **/shipping**, **/confirm**, **/success**, **/orders/me**, **/order/:id**: Các trang quản lý thông tin người dùng, cập nhật thông tin, cập nhật mật khẩu, địa chỉ giao hàng, xác nhận đơn hàng, trang thành công sau khi đặt hàng, danh sách đơn hàng và chi tiết đơn hàng.
  + **Admin Routes (<ProtectedRoute isAdmin={true}>):**
    - Các route chỉ dành cho quản trị viên.
    - **/dashboard**, **/admin/products**, **/admin/product**, **/admin/product/:id**, **/admin/orders**, **/admin/order/:id**, **/admin/users**, **/admin/user/:id**, **/admin/reviews**: Các trang quản lý dành cho quản trị viên như bảng điều khiển, quản lý sản phẩm, đơn hàng, người dùng và đánh giá sản phẩm.

**Footer:**

* + Hiển thị Footer nếu không có quyền quản trị hoặc trạng thái xác thực là **false**.

**Kết luận:**

* + **App.js** chứa cấu trúc tổng quan của ứng dụng, định nghĩa các route và xác định cách chúng được hiển thị dựa trên trạng thái xác thực và vai trò của người dùng.

# Chương 6: Tổng kết

## 1. Kết quả đạt được

* Ứng dụng công nghệ MERN Stack để xây dựng một ứng dụng web thương mại điện tử cơ bản, bao gồm cả frontend và backend.
* Đã thực hiện các biện pháp bảo mật như xác thực người dùng, quản lý quyền và mã thông báo JSON Web Tokens (JWT), mã hóa mật khẩu người dùng bằng bcrypt.
* Đã tích hợp cổng thanh toán Stripe để thực hiện các giao dịch thanh toán an toàn.

## 2. Hạn chế

* Giao diện người dùng còn đơn giản.
* Một vài chức năng có tính logic chưa cao.
* Hiệu suất của ứng dụng không quá tốt.

## 3. Hướng phát triển

* Cải thiện giao diện trở nên đẹp mắt và thân thiện với người dùng.
* Cho phép được chọn một hoặc nhiều sản phẩm cùng lúc trong giỏ hàng để thanh toán.
* Cho phép người dùng có thể chọn hình thức mua cod.
* Chương trình sử dụng database online và có deploy lên Heroku.

# Tài Liệu Tham Khảo

Redux:

<https://redux.js.org/tutorials/quick-start>

BootStrap:

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

StackOverflow:

<https://stackoverflow.com/questions/69864165/error-privateroute-is-not-a-route-component-all-component-children-of-rou>