

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CÔNG NGHỆ THÔNG TIN
XÂY DỰNG WEB BÁN HÀNG ONLINE

Nhóm sinh viên thực hiện:

Nguyễn Quốc Lân 21110837

Nguyễn Anh Hào 21110823

Đỗ Anh Khoa 21110208

Giáo viên hướng dẫn: Ts. Lê Vĩnh Thịnh

TP. Hồ Chí Minh, tháng 12 năm 2023

LỜI CẢM ƠN

Lời đầu tiên, nhóm sinh viên thực hiện báo cáo xin gửi lời cảm ơn chân thành nhất đến nhà trường và quý thầy cô. Trong quá trình học tập tại trường Đại học Sư phạm kỹ thuật thành phố Hồ Chí Minh, quý thầy cô đã tạo nhiều điều kiện cho chúng em được học tập tốt nhất, truyền đạt những kiến thức nền tảng và chuyên ngành, cũng như hỗ trợ, giúp đỡ khi nhóm gặp khó khăn trong quá trình học tập và trong lúc thực hiện đồ án

Tiếp đó, nhóm sinh viên thực hiện báo cáo xin gửi lời cảm ơn sâu sắc đến thầy Lê Vĩnh Thịnh – người đã hướng dẫn, giúp đỡ cho nhóm trong suốt quá trình thực hiện đề tài.

Do có những hạn chế về kiến thức và thiếu kinh nghiệm trong việc tìm hiểu thực tế, vì thế nhóm chúng em còn có nhiều thiếu sót, nhóm hy vọng nhận được những ý kiến đóng góp quý báu từ quý thầy cô để rút ra kinh nghiệm cho các đồ án khác trong tương lai. Nhóm thực hiện xin chân thành cảm ơn

Lời cuối cùng, nhóm chúng em xin chúc thầy cô có thật nhiều sức khỏe để tiếp tục thực hiện sứ mệnh cao cả của mình đó chính là truyền đạt những kiến thức bổ ích cho nhiều thế hệ sinh viên tiếp theo. Chúng em xin bày tỏ lòng biết ơn thầy cô rất nhiều!

Nhóm thực hiện

Nguyễn Quốc Lâm

Nguyễn Anh Hào

Đỗ Anh Khoa

LỜI MỞ ĐẦU

Nhờ có sự phát triển bùng nổ của thương mại điện tử, việc kinh doanh trực tuyến hiện nay dần trở thành xu hướng toàn cầu. Đối với các doanh nghiệp, việc có một trang web bán hàng trực tuyến không chỉ là một ưu thế mà là một yếu tố không thể thiếu trong chiến lược kinh doanh hiện đại.

Thương mại điện tử không chỉ giúp doanh nghiệp tiếp cận được đối tượng khách hàng rộng lớn mà còn tối ưu hóa quy trình bán hàng, tăng cường khả năng tiếp cận thông tin và tương tác với khách hàng. Việc sử dụng hiệu quả công nghệ thông tin trong thương mại điện tử không chỉ giúp doanh nghiệp quản lý hàng tồn kho, giao hàng, thanh toán một cách hiệu quả mà còn tạo ra trải nghiệm mua sắm trực tuyến thuận lợi và thú vị cho khách hàng.

Chính vì những lí do kể trên, nhóm chúng em đã quyết định chọn đề tài **phát triển, xây dựng website bán hàng online** nhằm đưa ra giải pháp giúp cho các doanh nghiệp có thể phát triển chiến lược kinh doanh, đồng thời là cơ hội để luyện tập, trau dồi kiến thức của bản thân về chuyên ngành.

MỤC LỤC

LỜI CẢM ƠN	i
LỜI MỞ ĐẦU	ii
MỤC LỤC.....	iii
DANH MỤC CÁC BẢNG.....	vii
DANH MỤC HÌNH	viii
DANH MỤC TỪ VIẾT TẮT.....	x
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT	1
1.1. Giới thiệu về node.js.....	1
1.2. Các framework, thư viện chính	1
CHƯƠNG 2. YÊU CẦU CHỨC NĂNG.....	2
2.1. Lược đồ usecase	2
2.2. Danh sách usecase	2
CHƯƠNG 3. THIẾT KẾ XỬ LÝ	4
3.1. Thiết kế cơ bản	4
3.1.1. Các thành phần của ứng dụng	4
3.1.2. Backend	5
3.1.2.1. Cách tổ chức thư mục/package.....	5
3.1.2.2. Thư viện sử dụng	7
3.1.3. Frontend cho shop	9
3.1.3.1. Cách tổ chức thư mục/package	9
3.1.3.2. Thư viện sử dụng	11

3.1.4.	Frontend cho client.....	14
3.1.4.1.	Cách tổ chức thư mục/package	14
3.1.4.2.	Thư viện sử dụng	16
3.2.	Thiết kế cơ sở dữ liệu	20
3.2.1.	Sơ đồ thiết kế cơ sở dữ liệu	20
3.2.2.	Mô tả sơ đồ thiết kế cơ sở dữ liệu	20
3.2.2.1.	Bảng Apikeys.....	20
3.2.2.2.	Bảng Users (Người dùng).....	21
3.2.2.3.	Bảng Products (Sản phẩm)	22
3.2.2.4.	Bảng Inventories (Nhà kho)	23
3.2.2.5.	Bảng Carts (Giỏ hàng)	23
3.2.2.6.	Bảng Orders (Đơn hàng)	24
3.3.	Thiết kế kỹ thuật NodeJS và minh họa	25
3.3.1.	Tạo token mã hóa sử dụng JWT	25
3.3.2.	Sử dụng controller.....	27
3.3.3.	Tạo các repository	32
3.3.4.	Viết các services để sử dụng xử lý logic.....	34
3.4.	Thiết kế giao diện.....	37
3.4.1.	Thiết kế giao diện phía quản lý shop	37
3.4.2.	Thiết kế giao diện phía người mua hàng.....	46
CHƯƠNG 4. PHÂN CÔNG CÔNG VIỆC		55
4.1.	Bảng phân công công việc.....	55

4.2. Bảng tỷ lệ hoàn thành công việc.....	55
CHƯƠNG 5. NGHIÊN CỨU VỀ AI TRONG PHÁT TRIỂN WEBSITE	56
5.1. Một vài thuật toán cơ bản trong Machine Learning	56
5.1.1. Linear Regression.....	56
5.1.2. Logistic Regression	56
5.1.3. Decision Tree	57
5.1.4. Thuật toán K-Nearest Neighbors.....	58
5.1.5. Thuật toán Random Forest	59
5.2. Tiềm năng của Machine Learning ứng dụng vào website.....	59
5.3. Hạn chế của Machine Learning và Artificial Intelligence trong phát triển website	60
CHƯƠNG 6. KẾT LUẬN.....	62
6.1. Tổng kết.....	62
6.2. Kết quả đạt được.....	62
6.3. Ưu điểm và nhược điểm	62
6.3.1. Ưu điểm.....	62
6.3.2. Nhược điểm	62
6.4. Khó khăn.....	63
6.4.1. Khó khăn về công nghệ.....	63
6.4.2. Khó khăn về quy trình thực hiện	63
6.5. Bài học kinh nghiệm.....	64
6.6. Hướng phát triển.....	64

TÀI LIỆU THAM KHẢO.....	65
-------------------------	----

DANH MỤC CÁC BẢNG

Bảng 1. Danh sách các use case	2
Bảng 2. Các thư viện sử dụng trong Backend.....	7
Bảng 3. Các framework sử dụng trong Backend	8
Bảng 4. Các thư viện được sử dụng trong frontend/shop	11
Bảng 5. Các thư viện được sử dụng trong frontend/client	16
Bảng 6. Bảng Apikeys.....	20
Bảng 7. Bảng Users (Người dùng).....	21
Bảng 8. Bảng Products (Sản phẩm)	22
Bảng 9. Bảng Inventories (Nhà kho)	23
Bảng 10. Bảng Cart (Giỏ hàng)	23
Bảng 11. Bảng Orders (Đơn hàng)	24
Bảng 12. Phân công công việc	55
Bảng 13. Tỷ lệ đóng góp	55
Bảng 14. Khó khăn về công nghệ	63
Bảng 15. Khó khăn về quy trình thực hiện	63

DANH MỤC HÌNH

Hình 1. Lược đồ use case tổng quát	2
Hình 2. Hình mô tả các thành phần cơ bản của ứng dụng	4
Hình 3. Cách tổ chức thư mục ở Backend	5
Hình 4. Cách tổ chức thư mục trong frontend/shop	9
Hình 5. Cách tổ chức thư mục trong frontend/client	14
Hình 6. Sơ đồ thiết kế cơ sở dữ liệu	20
Hình 7. Giao diện đăng ký thông tin shop thất bại	37
Hình 8. Giao diện khi đăng ký shop thành công	38
Hình 9. Giao diện trang đăng nhập	39
Hình 10. Giao diện đăng nhập thất bại	40
Hình 11. Giao diện quản lý shop	40
Hình 12. Giao diện danh sách sản phẩm	41
Hình 13. Giao diện form tạo sản phẩm	42
Hình 14. Hình ảnh sản phẩm mới được tạo ra	42
Hình 15. Giao diện chỉnh sửa thông tin sản phẩm	43
Hình 16. Hình ảnh sau khi chỉnh sửa thông tin sản phẩm	43
Hình 17. Giao diện xác nhận xóa sản phẩm	44
Hình 18. Giao diện sản phẩm chưa mở bán	44
Hình 19. Giao diện xác nhận mở bán sản phẩm	45
Hình 20. Hình ảnh sản phẩm được mở bán lại	45
Hình 21. Giao diện quản lý đơn hàng	45
Hình 22. Giao diện chi tiết đơn hàng	46
Hình 23. Giao diện sau khi thay đổi trạng thái đơn hàng	46
Hình 24. Giao diện trang chủ	47
Hình 25. Giao diện trang chủ kèm danh sách sản phẩm	47
Hình 26. Giao diện trang đăng ký tài khoản	48

Hình 27. Giao diện trang đăng nhập	49
Hình 28. Giao diện chức năng chỉnh sửa thông tin tài khoản	49
Hình 29. Giao diện chỉnh sửa thông tin người dùng	50
Hình 30. Giao diện thanh tìm kiếm	50
Hình 31. Giao diện một trang sản phẩm	50
Hình 32. Giao diện giỏ hàng rỗng	51
Hình 33. Giao diện sản phẩm khi thêm vào giỏ hàng	51
Hình 34. Giao diện giỏ hàng sau khi thêm vào giỏ	51
Hình 35. Giao diện thanh toán tại giỏ hàng	52
Hình 36. Giao diện hiển thị thông tin giao hàng	52
Hình 37. Giao diện xem lại đơn hàng	52
Hình 38. Giao diện xem lại thông tin chi tiết đơn hàng	53
Hình 39. Giao diện đơn hàng được cập nhật	53
Hình 40. Giao diện chỉnh sửa trạng thái đơn hàng	53
Hình 41. Giao diện đơn hàng đã được xác nhận	54
Hình 42. Đường hồi quy	56
Hình 43. Đồ thị phân nhóm dữ liệu theo đường logistic	57
Hình 44. Một cây quyết định đơn giản	58
Hình 45. Trường hợp $K = 3$ trong KNN	58
Hình 46. Mô hình Random Forest	59

DANH MỤC TỪ VIẾT TẮT

Các từ viết tắt	Nghĩa của từ
JS	JavaScript
I/O	Input / Output
NPM	Node Package Manager
UI	User Interface
DB	Database
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
API	Application Programming Interface
JSON	JavaScript Object Notation
JWT	JSON Web Token
SHA	Secure Hash Algorithm
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
KNN	K – Nearest Neighbor

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu về node.js

Node.js là một runtime được phát triển từ 2009 và ra mắt phiên bản ổn định nhất vào 2021. Được build dựa trên Chrome's V8 JavaScript engine, node.js sử dụng mô hình event-driven, non-blocking I/O khiến nó trở nên nhẹ và hiệu quả. V8 engine là một JavaScript engine mã nguồn mở chạy trên các trình duyệt Chrome, Opera và Vivaldi. Nó được thiết kế tập trung vào hiệu năng và chịu trách nhiệm cho việc dịch mã JavaScript sang mã máy để máy tính có thể hiểu và chạy được.

Lí do node.js được sử dụng rộng rãi là bởi vì các lý do kể sau: Nhờ vào mô hình I/O độc đáo khác với môi trường JS thông thường đó là khả năng xử lý nhiều yêu cầu đồng thời I/O vào máy chủ web. Nhờ đó mà Node.js hỗ trợ hiệu suất siêu nhanh và hiệu quả cho phát triển backend. Tiếp theo, do có kiểu kiến trúc với một luồng duy nhất, Node.js rất phù hợp để xử lý lượng lớn dữ liệu và lưu lượng. Hơn nữa, khi phát triển backend bằng node.js, với công cụ NPM, ta có thể dễ dàng quản lý được các module mà không cần phải lưu các module trong cấu trúc thư mục. Chẳng hạn, mỗi khi đưa project cho một bộ phận khác thực hiện, thì bộ phận đó không cần phải copy toàn bộ các module trong project mà chỉ cần sử dụng công cụ npm và nhập lệnh **npm install** để npm tự tìm và cài đặt các module được liệt kê trong package.json.

1.2. Các framework, thư viện chính

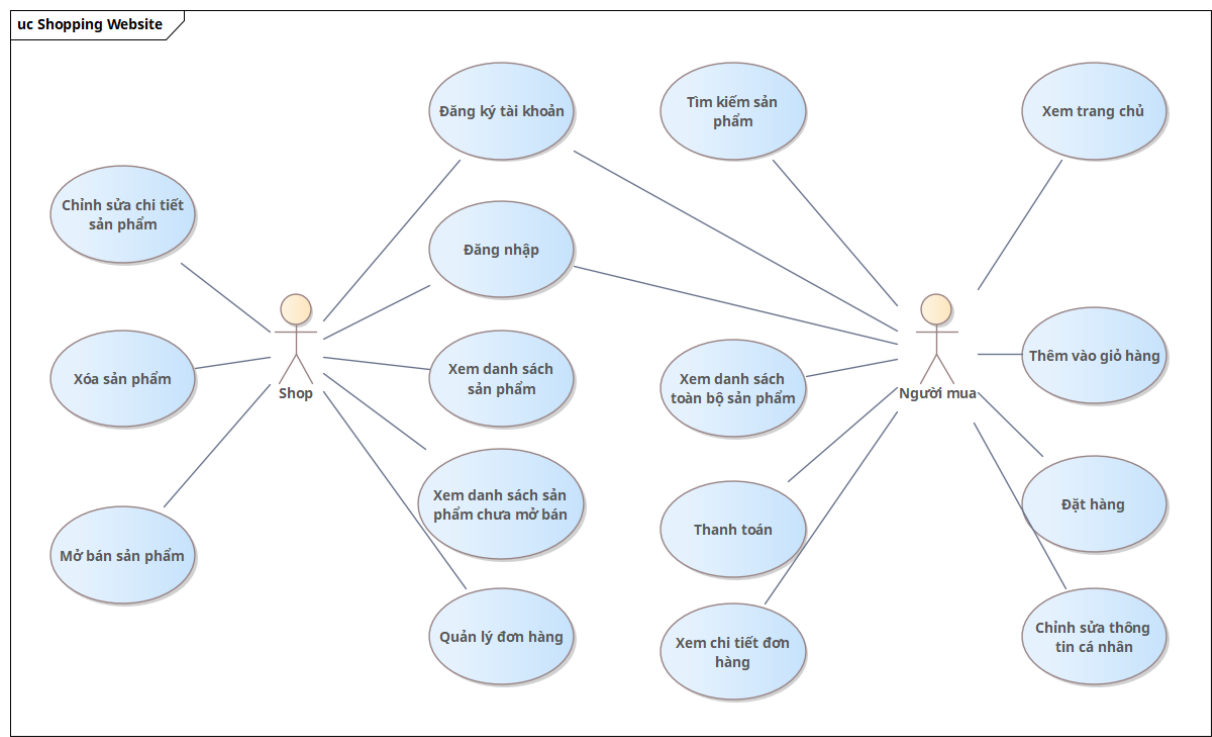
Nhóm chúng em sử dụng những thư viện và framework như sau:

- Thư viện ReactJS để phát triển xây dựng UI trong frontend
- Thư viện CSS TailwindCSS để viết CSS trong xây dựng thiết kế UI
- Framework ExpressJS để phát triển backend
- Thư viện Mongoose để tương tác với dữ liệu trong MongoDB

Ngoài ra còn có các tool hỗ trợ như ViteJS, VueJS trong khâu thiết kế frontend.

CHƯƠNG 2. YÊU CẦU CHỨC NĂNG

2.1. Lược đồ usecase



Hình 1. Lược đồ use case tổng quát

2.2. Danh sách usecase

Bảng 1. Danh sách các use case

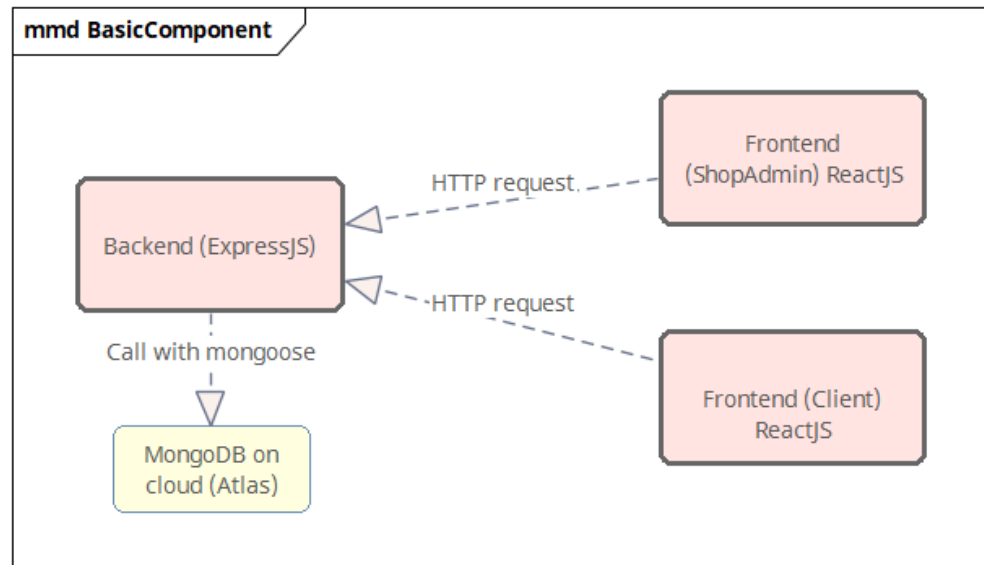
STT	Tên usecase	Usecase ID	Actor
1	Đăng ký tài khoản	UC_001	Người mua, Shop
2	Đăng nhập	UC_002	Người mua, Shop
3	Xem danh sách sản phẩm	UC_003	Người mua, Shop
4	Xem trang chủ	UC_004	Người mua
5	Xem danh sách toàn bộ sản phẩm	UC_005	Người mua

6	Xem danh sách sản phẩm chưa mở bán	UC_006	Shop
7	Quản lý đơn hàng	UC_007	Shop
8	Chỉnh sửa chi tiết sản phẩm	UC_008	Shop
9	Xóa sản phẩm	UC_009	Shop
10	Mở bán sản phẩm	UC_010	Shop
11	Tìm kiếm sản phẩm	UC_011	Người mua
12	Thanh toán	UC_012	Người mua
13	Đặt hàng	UC_013	Người mua
14	Thêm vào giỏ hàng	UC_014	Người mua
15	Chỉnh sửa thông tin cá nhân	UC_015	Người mua
16	Xem chi tiết đơn hàng	UC_016	Người mua

CHƯƠNG 3. THIẾT KẾ XỬ LÝ

3.1. Thiết kế cơ bản

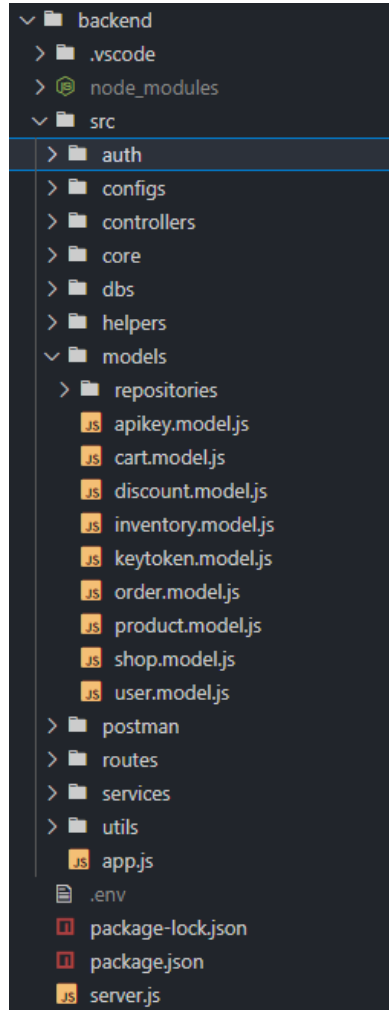
3.1.1. Các thành phần của ứng dụng



Hình 2. Hình mô tả các thành phần cơ bản của ứng dụng

3.1.2. Backend

3.1.2.1. Cách tổ chức thư mục/package



Hình 3. Cách tổ chức thư mục ở Backend

- /src: Chứa tất cả các file code của backend
- + /src/auth: Chứa các code về quy chuẩn của header khi được gửi tới backend, mã hóa và kiểm tra authentication
- + /src/configs: Chứa code về cấu hình của các dịch vụ thứ 3. Ở đây nhóm chúng em lưu ở trong folder này cấu hình của mongodb và server.

- + /src/controllers: Chứa tất cả các code xử lý dữ liệu của server. Với những dữ liệu được truyền từ client, ứng với mỗi API khác nhau, nó sẽ gọi đến controller ứng với API đó. Một controller cũng có thể được dùng lại từ những API khác
- + /src/cores: Chứa các hình thức response của server là error hoặc success.
- + /src/dbs: Khởi tạo mongodb
- + /src/helpers: Chứa các thành phần hỗ trợ kết nối, bất đồng bộ,...
- + /src/models:
 - /src/models/repositories: Chứa code về các phương thức để tương tác với dữ liệu như findAll, findOne, create, update, ... cũng như truy vấn dữ liệu.
 - Chứa các models dưới dạng schema.
- + /src/routes: Chứa các đường dẫn URL gồm các phương thức HTTP tương ứng như GET, POST, PUT, DELETE,...
- + /src/services: Chứa các code xử lý logic của ứng dụng
- + /src/utils: Thư mục chứa các tiện ích cần thiết mà được sử dụng lại rất nhiều ở các chức năng xử lý.
- + /src/postman: Chứa file http để test các route
- + /src/app.js: File này sẽ khai báo tất cả các package/module cần thiết cho chương trình
- server.js: File khởi chạy server
- package.json: File chứa tất cả các chi tiết npm của dự án, các lệnh chạy như scripts và các thành phần dependencies

- .env: File khai báo biến môi trường, ví dụ: đường dẫn đến database, ...
- node_modules: Thư mục sẽ chứa tất cả các module mà server sử dụng và đã được khai báo trong package.json. Với mỗi ứng dụng muốn chạy, phải dùng câu lệnh **npm install** để cài và tạo ra folder node_modules, vì folder này tốn nhiều bộ nhớ nên sẽ được bỏ vào .gitignore

3.1.2.2. Thư viện sử dụng

Bảng 2. Các thư viện sử dụng trong Backend

STT	Tên thư viện	Version	Mục đích
Dependencies			
1	bcrypt	^5.1.1	Dùng để mã hóa mật khẩu
2	cors	^2.8.5	Dùng để kích hoạt cors, cho phép chia sẻ tài nguyên giữa front-end và backend
3	crypto	^1.0.1	Chứa các chuẩn mã hóa cơ bản như SHA256, AES, ...
4	express	^4.18.2	Dùng để khởi tạo server cho nodejs
5	jsonwebtoken	^9.0.2	Dùng để mã hóa bảo vệ dữ liệu json
6	lodash	^4.17.21	Sử dụng để xử lý mảng, object trong code
7	mongoose	^8.0.0	Sử dụng để kết nối nodejs với MongoDB và tương tác với dữ liệu
8	redis	^4.6.11	Catch dữ liệu vào bộ nhớ đệm, xử lý dữ liệu nhanh hơn

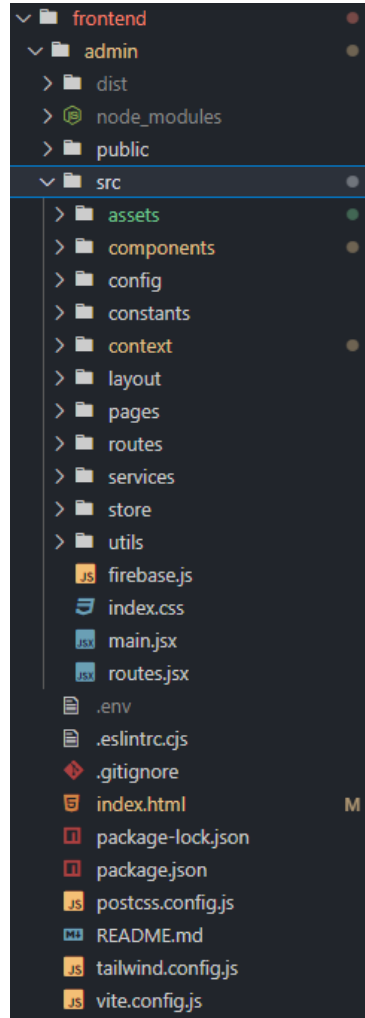
9	slugify	^1.6.6	Dùng để slugify một chuỗi, ví dụ: “Xin chào thế giới” → xin-chao-the-gioi
Dev Dependencies			
10	compression	^1.7.4	Tăng tốc độ xử lý bằng cách nén response lại khi request được gửi đến
11	dotenv	^16.3.1	Dùng để load các biến môi trường từ file .env vào process.env
12	helmet	^7.0.0	Dùng để bảo vệ ứng dụng Express bằng cách thiết lập HTTP header một cách phù hợp
13	morgan	^1.10.0	Dùng để log lại các HTTP request, errors,... vào console
14	nodemon	^3.0.1	Dùng để khởi động lại ứng dụng Node khi phát hiện có file thay đổi

Bảng 3. Các framework sử dụng trong Backend

STT	Tên framework	Version	Mục đích
1	express	^4.18.2	Sử dụng framework để phát triển phía server

3.1.3. Frontend cho shop

3.1.3.1. Cách tổ chức thư mục/package



Hình 4. Cách tổ chức thư mục trong frontend/shop

- /node_modules: Thư mục chứa các thư viện được cài đặt và sử dụng trong dự án
- /public: Thư mục chứa tệp index.html, favicon.ico... là thư mục gốc của ứng dụng React
- /src: Thư mục chứa tất cả source code của dự án
 - + /src/assets: Thư mục chứa các tệp hình ảnh

- + /src/components: Thư mục này chứa các component chung cho dự án như Footer, Header,...
- + /src/config: Thư mục này dùng để config axiosClient để gọi API
- + /src/constants: Thư mục này chứa các const để gán nhãn, gọi ra trong các đoạn code khác
- + /src/context: Thư mục này dùng để chứa logic để đăng nhập và đăng xuất
- + /src/layout: Thư mục chứa code về layout, bố cục của giao diện quản lý shop
- + /src/pages: Thư mục chứa các trang chính gồm các chức năng chính của shop
- + /src/routes: Thư mục chứa các đường dẫn
- + /src/services: Thư mục chứa các endpoint, gọi API tới backend
- + /src/store: Thư mục lưu trữ các reducers trong reduxjs
- + /src/utils: Thư mục chứa các hàm được sử dụng nhiều lần trong project
- + /src/firebase.js: Tập này chứa config của firebase để lưu trữ hình ảnh
- + /src/index.css: Tập này dùng để cấu hình css
- + /src/main.jsx: Tập này sẽ render ra trang quản lý shop
- + /src/routes.jsx: Chứa các đường dẫn
- .gitignore: Cấu hình những thư mục, tệp để git bỏ qua khi commit
- .env: Chứa các cấu hình về MongoDB, Firebase,...
- index.html: File html gốc của React
- tailwind.config.js: File config cho tailwindcss

- package.json: Tập dùng để khai báo các thư viện, các lệnh scripts được cài đặt trong dự án
- package-lock.json: Tập dùng để mô tả chi tiết về các thư viện bên trong tệp package.json, giúp tránh việc xung đột phiên bản thư viện khi nhiều người làm chung dự án
- README.MD: Tập này dùng để mô tả về dự án trên Github

3.1.3.2. Thư viện sử dụng

Bảng 4. Các thư viện được sử dụng trong frontend/shop

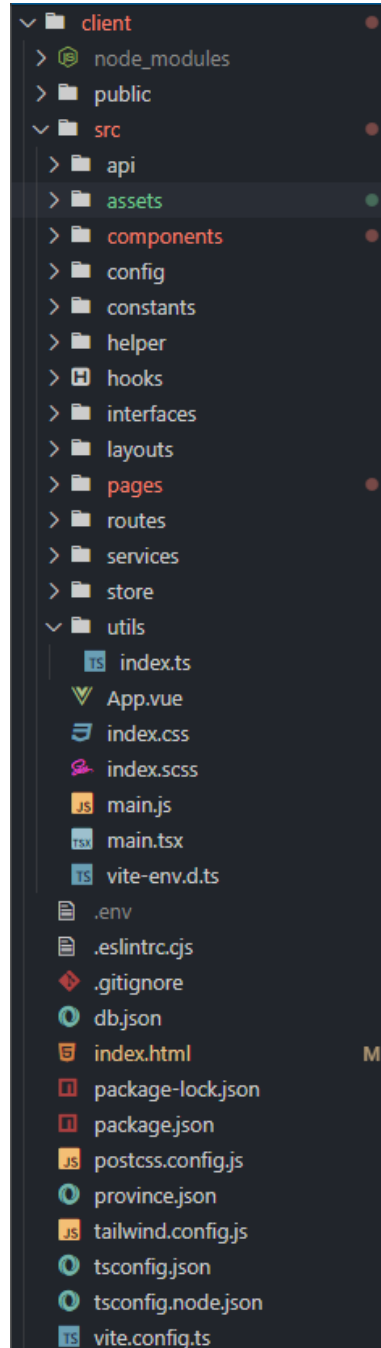
STT	Tên thư viện	Version	Mục đích
Dependencies			
1	@reduxjs/toolkit	^1.9.7	Cung cấp công cụ quản lý các state trong dự án ReactJS
2	@tanstack/react-query	^5.8.7	Dùng để làm việc với API và quản lý cache
3	antd	^5.11.2	Thư viện Ant Design chứa các component UI để thiết kế giao diện
4	axios	^1.6.2	Dùng để gọi API
5	clsx	^2.0.0	Dùng để làm việc với className class css
6	firebase	^10.6.0	Dùng để tương tác với firebase
7	json-server	^0.17.4	Dùng để giả lập RESTful API từ 1 tệp json
8	jwt-decode	^4.0.0	Dùng để decode mã JWT

9	lodash	^4.17.21	Cung cấp các công cụ để xử lý mảng, object, chuỗi
10	react	^18.2.0	Thư viện xây dựng giao diện
11	react-dom	^18.2.0	Cung cấp các phương thức để tương tác với DOM như trong html
12	react-hook-form	^7.48.2	Dùng để quản lý và xử lý form trong React
13	react-hot-toast	^2.4.1	Dùng để xử lý thông báo nổi trong React
14	react-icons	^4.12.0	Cung cấp các bộ icon để thiết kế giao diện
15	react-redux	^8.1.3	Quản lý các state trong dự án ReactJS
16	react-router-dom	^6.19.0	Quản lý đường dẫn trong ứng dụng
Dev Dependencies			
17	@types/react	^18.2.15	Dùng để định nghĩa kiểu trong TypeScript
18	@types/react-dom	^18.2.7	Dùng để định nghĩa kiểu trong TypeScript
19	@vitejs/plugin-react	^4.0.3	Plugin mặc định của Vite dành cho ReactJS
20	autoprefixer	^10.4.16	Plugin của PostCSS nhằm thêm các prefix tương thích trình duyệt trong CSS
21	eslint	^8.45.0	Kiểm tra mã nguồn và thông báo các lỗi về cú pháp của ứng dụng
22	eslint-plugin-react	^7.32.2	Chứa các plugin của eslint dành cho ứng dụng ReactJS

23	eslint-plugin-react-hooks	^4.6.0	Chứa các plugin của eslint phục vụ cho các React Hooks
24	eslint-plugin-react-refresh	^0.4.3	Chứa các plugin của eslint phục vụ cho việc refresh nhanh
25	postcss	^8.4.31	Đi kèm với tailwindcss
26	tailwindcss	^3.3.5	Cung cấp Framework CSS để viết CSS inline trong dự án
27	vite	^4.4.5	Cung cấp công cụ xây dựng ứng dụng web

3.1.4. Frontend cho client

3.1.4.1. Cách tổ chức thư mục/package



Hình 5. Cách tổ chức thư mục trong frontend/client

- /node_modules: Thư mục chứa các thư viện được cài đặt và sử dụng trong dự án
- /public: Thư mục chứa tệp index.html, favicon.ico... là thư mục gốc của ứng dụng React
- /src: Thư mục chứa tất cả source code của dự án
 - + /src/api: Thư mục chứa code API
 - + /src/assets: Thư mục chứa ảnh cho trang web
 - + /src/components: Thư mục này chứa các component chung cho dự án như Footer, Header, Carousel,...
 - + /src/config: Thư mục này dùng để config axiosClient để gọi API
 - + /src/constants: Thư mục này chứa các const để gán nhãn, gọi ra trong các đoạn code khác
 - + /src/helper: Thư mục chứa code hỗ trợ cho các components
 - + /src/hooks: Thư mục chứa các Hook tự định nghĩa trong dự án
 - + /src/interfaces: Thư mục khai báo các Interfaces
 - + /src/layouts: Thư mục chứa code về layout, bố cục của giao diện
 - + /src/pages: Thư mục chứa các trang chính để render
 - + /src/routes: Thư mục chứa các đường dẫn
 - + /src/services: Thư mục chứa các endpoint, gọi API tới backend
 - + /src/store: Thư mục lưu trữ các reducers trong reduxjs
 - + /src/utils: Thư mục chứa các hàm được sử dụng nhiều lần trong project

- + /src/main.js: Là file chính để khởi tạo ứng dụng React
- .gitignore: Cấu hình những thư mục, tệp để git bỏ qua khi commit
- .env: Chứa các cấu hình về MongoDB, Firebase,...
- index.html: File html gốc của React
- tailwind.config.js: File config cho tailwindcss
- package.json: Tệp dùng để khai báo các thư viện, các lệnh scripts được cài đặt trong dự án
- package-lock.json: Tệp dùng để mô tả chi tiết về các thư viện bên trong tệp package.json, giúp tránh việc xung đột phiên bản thư viện khi nhiều người làm chung dự án
- README.MD: Tệp này dùng để mô tả về dự án trên Github

3.1.4.2. Thư viện sử dụng

Bảng 5. Các thư viện được sử dụng trong frontend/client

STT	Tên thư viện	Version	Mục đích
Dependencies			
1	@hookform/resolvers	^3.3.2	
2	@reduxjs/toolkit	^1.9.7	Cung cấp công cụ quản lý các state trong dự án ReactJS
3	@tanstack/react-query	^5.8.7	Dùng để làm việc với API và quản lý cache

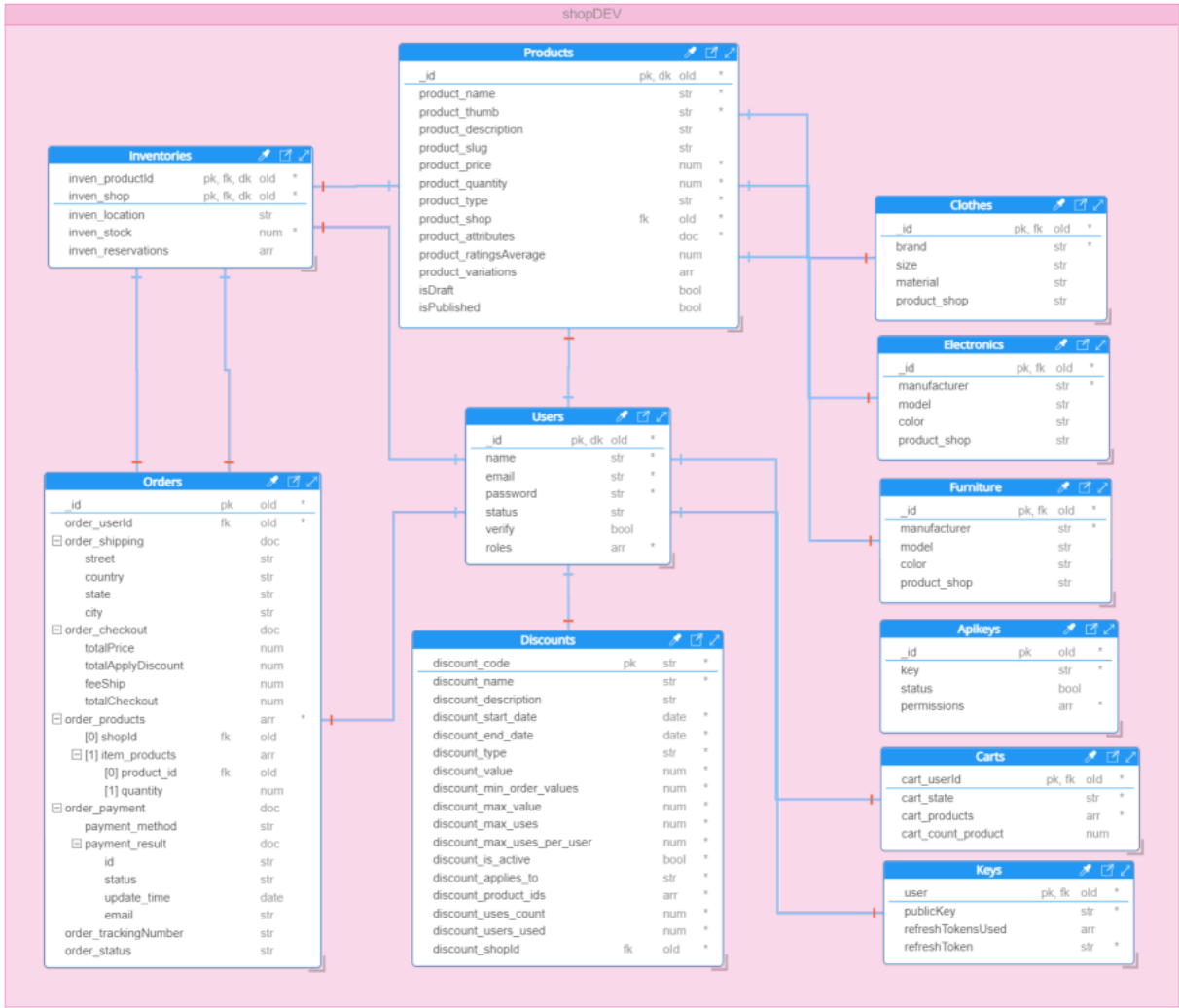
4	@tanstack/react-query-devtools	^5.13.3	Dùng để làm việc với API và quản lý cache dạng devtool
5	antd	^5.11.4	Thư viện Ant Design chứa các component UI để thiết kế giao diện
6	axios	^1.6.2	Dùng để gọi API
7	clsx	^2.0.0	Dùng để làm việc với className class css
8	imagesloaded	^5.0.0	
9	jwt-decode	^4.0.0	Dùng để decode mã JWT
10	lodash	^4.17.21	Cung cấp các công cụ để xử lý mảng, object, chuỗi
11	node-sass	^9.0.0	
12	react	^18.2.0	Thư viện xây dựng giao diện
13	react-dom	^18.2.0	Cung cấp các phương thức để tương tác với DOM như trong html
14	react-hook-form	^7.49.0	Dùng để quản lý và xử lý form trong React
15	react-hook-form-antd	^1.0.1	Dùng để quản lý và xử lý form trong React theo Ant Design
16	react-hot-toast	^2.4.1	Dùng để xử lý thông báo nổi trong React
17	react-icons	^4.12.0	Cung cấp các bộ icon để thiết kế giao diện
18	react-inline-editing	^1.0.10	

19	react-loading-skeleton	^3.3.1	
20	react-redux	^8.1.3	Quản lý các state trong dự án ReactJS
21	react-router-dom	^6.20.0	Quản lý đường dẫn trong ứng dụng
22	sass	^1.69.5	Giúp viết CSS một cách có cấu trúc
23	styled-components	^6.1.1	
24	yup	^1.3.2	
Dev Dependencies			
25	@types/imagesloaded	^4.1.6	Dùng để định nghĩa kiểu trong TypeScript
26	@types/lodash	^4.14.202	Dùng để định nghĩa kiểu trong TypeScript
27	@types/react	^18.2.37	Dùng để định nghĩa kiểu trong TypeScript
28	@types/react-dom	^18.2.15	Dùng để định nghĩa kiểu trong TypeScript
29	@typescript-eslint/eslint-plugin	^6.10.0	Dùng để định nghĩa kiểu trong TypeScript
30	@typescript-eslint/parser	^6.10.0	Dùng để định nghĩa kiểu trong TypeScript
31	@vitejs/plugin-react	^4.2.0	Plugin mặc định của Vite dành cho ReactJS
32	autoprefixer	^10.4.16	Plugin của PostCSS nhằm thêm các prefix tương thích trình duyệt trong CSS

33	eslint	^8.53.0	Kiểm tra mã nguồn và thông báo các lỗi về cú pháp của ứng dụng
34	eslint-plugin-react-hooks	^4.6.0	Chứa các plugin của eslint phục vụ cho các React Hooks
35	eslint-plugin-react-refresh	^0.4.4	Chứa các plugin của eslint phục vụ cho việc refresh nhanh
36	postcss	^8.4.31	Đi kèm với tailwindcss
37	tailwindcss	^3.3.5	Cung cấp Framework CSS để viết CSS inline trong dự án
38	typescript	^5.2.2	Cung cấp môi trường viết TypeScript
39	vite	^5.0.0	Cung cấp công cụ xây dựng ứng dụng web

3.2. Thiết kế cơ sở dữ liệu

3.2.1. Sơ đồ thiết kế cơ sở dữ liệu



Hình 6. Sơ đồ thiết kế cơ sở dữ liệu

3.2.2. Mô tả sơ đồ thiết kế cơ sở dữ liệu

3.2.2.1. Bảng Apikeys

Bảng 6. Bảng Apikeys

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
-----	----------------	--------------	---------

1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác
2	key	String	Khóa được tạo ra
3	status	String	Khóa có hoạt động hay không
4	permissions	Array[String]	Quyền của các đối tác sử dụng API

3.2.2.2. Bảng Users (Người dùng)

Bảng 7. Bảng Users (Người dùng)

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác
2	name	String	Tên của người dùng
3	email	String	Email của người dùng
4	password	String	Mật khẩu của người dùng
5	address	Object	Địa chỉ người dùng
	detailAddress	String	
	ward	String	
	district	String	
	province	String	
6	status	String	Người dùng còn được cấp phép hoạt động

7	verify	bool	Đánh dấu người dùng đã xác nhận email hay chưa
8	roles	Array[String]	Các quyền của người dùng

3.2.2.3. Bảng Products (Sản phẩm)

Bảng 8. Bảng Products (Sản phẩm)

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác
2	product_name	String	Tên của sản phẩm
3	product_thumb	String	Đường dẫn hình ảnh của sản phẩm
4	product_description	String	Mô tả của sản phẩm
5	product_slug	String	Slug của sản phẩm
6	product_price	Number	Giá của sản phẩm
7	product_quantity	Number	Số lượng của sản phẩm
8	product_type	String	Loại của sản phẩm
9	product_shop	ObjectId	Id của cửa hàng
10	product_attributes	Mixed	Các thuộc tính của sản phẩm tùy thuộc vào loại sản phẩm
11	product_ratingsAverage	Number	Điểm đánh giá của sản phẩm

12	product_variations	Array[Mixed]	Các biến thể của sản phẩm
13	isDraft	Bool	Đánh dấu sản phẩm đã xóa
14	isPublished	Bool	Đánh dấu sản phẩm được bán

3.2.2.4. Bảng Inventories (Nhà kho)

Bảng 9. Bảng Inventories (Nhà kho)

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác
2	inven_productId	ObjectID	Id của sản phẩm
3	inven_location	String	Vị trí của nhà kho
4	inven_stock	Number	Số lượng sản phẩm trong kho
5	inven_shop	ObjectID	Id của cửa hàng
6	inven_reservations	Array[Mixed]	Lưu thông tin giỏ hàng đang có sản phẩm

3.2.2.5. Bảng Carts (Giỏ hàng)

Bảng 10. Bảng Cart (Giỏ hàng)

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác

2	cart_state	String	Trạng thái của giỏ hàng
3	cart_products	Array[Mixed]	Thông tin các sản phẩm
4	cart_userId	ObjectID	Id của người mua

3.2.2.6. Bảng Orders (Đơn hàng)

Bảng 11. Bảng Orders (Đơn hàng)

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	_id	ObjectID	Giá trị id được tạo tự động, có giá trị duy nhất, phân biệt với các document khác
2	order_userId	ObjectID	Id của người mua
3	order_checkout	Object	Thông tin về giá của đơn hàng
4	order_shipping	Object	Thông tin về nơi giao hàng
5	order_payment	Object	Thông tin về thanh toán
6	order_products	Array	Các sản phẩm đã mua
7	order_trackingNumber	String	Mã số vị trí sản phẩm đang ở đâu trên đường vận chuyển
8	order_status	String	Trạng thái của đơn hàng

3.3. Thiết kế kỹ thuật NodeJS và minh họa

Trong phạm vi đồ án, nhóm chúng em có áp dụng một số kỹ thuật trong NodeJS như sau:

3.3.1. Tạo token mã hóa sử dụng JWT

```
const HEADER = {
  API_KEY: "x-api-key",
  CLIENT_ID: "x-client-id",
  AUTHORIZATION: "authorization",
  REFRESHTOKEN: "x-rtoken-id",
};

const createTokenPair = async (payload, publicKey, privateKey) => {
  try {
    // accessToken
    const accessToken = await JWT.sign(payload, privateKey, {
      algorithm: "RS256",
      expiresIn: "2 days",
    });

    const refreshToken = await JWT.sign(payload, privateKey, {
      algorithm: "RS256",
      expiresIn: "7 days",
    });

    JWT.verify(accessToken, publicKey, (err, decode) => {
      if (err) {
        console.error(`error verify:`, err);
      } else {
        console.log(`decode verify:`, decode);
      }
    });

    return { accessToken, refreshToken };
  }
};
```

```
    } catch (error) {}  
};  
  
const authentication = asyncHandler(async (req, res, next) => {  
  // check userId missing  
  const userId = req.headers[HEADER.CLIENT_ID];  
  console.log(userId);  
  if (!userId) throw new AuthFailureError("Invalid Request");  
  
  // get accessToken  
  const keyStore = await findByUserId(userId);  
  if (!keyStore) throw new NotFoundError("Not found keyStore");  
  
  // verify Token  
  if (req.headers[HEADER.REFRESHTOKEN]) {  
    try {  
      const refreshToken = req.headers[HEADER.REFRESHTOKEN];  
      const decodeUser = JWT.verify(refreshToken,  
keyStore.publicKey);  
  
      // check keyStore with userId?  
      if (userId !== decodeUser.userId) throw new  
AuthFailureError("Invalid UserId");  
  
      req.keyStore = keyStore;  
      req.user = decodeUser;  
      req.refreshToken = refreshToken;  
  
      return next();  
    } catch (error) {  
      throw error;  
    }  
  }  
}  
  
const accessToken = req.headers[HEADER.AUTHORIZATION];
```

```
if (!accessToken) throw new AuthFailureError("Invalid Request");

try {
  const decodeUser = JWT.verify(accessToken, keyStore.publicKey);
  // check keyStore with userId?
  if (userId !== decodeUser.userId) throw new
AuthFailureError("Invalid UserId");

  req.keyStore = keyStore;
  req.user = decodeUser;

  return next();
} catch (error) {
  throw error;
}
});

const verifyJWT = async (token, keySecret) => {
  return await JWT.verify(token, keySecret);
};

module.exports = {
  createTokenPair,
  authentication,
  verifyJWT,
};
```

3.3.2. Sử dụng controller

Minh họa bằng CartController

```
const CartService = require("../services/cart.service");

class CartController {
  constructor() {
```

```
this.cartService = new CartService();
this.createOrAddItemToCart =
this.createOrAddItemToCart.bind(this);
this.getCartById = this.getCartById.bind(this);
this.getCartByUserId = this.getCartByUserId.bind(this);
this.updateEntireCart = this.updateEntireCart.bind(this);
this.deleteCart = this.deleteCart.bind(this);
this.checkExistProduct = this.checkExistProduct.bind(this);
this.removeItemFromCart = this.removeItemFromCart.bind(this);
this.updateEachProduct = this.updateEachProduct.bind(this);
}
async checkExistProduct(req, res) {
  try {
    const { userId, productId } = req.query;
    if (!userId || !productId) {
      return res
        .status(400)
        .json({ error: "userId and productId are required in the
query" });
    }
    let productExists = await this.cartService.checkExistProduct(
      userId,
      productId
    );
    res.json({ productExists });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
}
async createOrAddItemToCart(req, res) {
  try {
    const { userId, productId, quantity } = req.body;
    let cart = await this.cartService.getCartByUserId(userId);

    if (!cart) {
```

```
        // If cart doesn't exist, create a new one
        cart = await this.cartService.createCart(userId);
    }

    // Add item to the cart
    const newItem = { product: productId, quantity: quantity || 1
};

    const productExists = await this.cartService.checkExistProduct(
        userId,
        productId
    );
    if (productExists) {
        res.status(400).json({ message: "Product existed" });
    } else {
        const updatedCart = await this.cartService.addItemToCart(
            cart._id,
            newItem
        );

        res
            .status(200)
            .json({ message: "Add to cart successfully", data:
updatedCart });
    }
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
}

async getCartById(req, res) {
    try {
        const { cartId } = req.params;
        const cart = await this.cartService.getCartById(cartId);
        console.log({ cart });
        if (!cart) {
            return res.status(404).json({ message: "Cart not found" });
        }
    }
}
```



```
    }
    res.json(cart);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
}

async getCartByUserId(req, res) {
  try {
    const { userId } = req.params;
    const cart = await this.cartService.getCartByUserId(userId);
    console.log({ cart });
    if (!cart) {
      return res.status(404).json({ message: "Cart not found",
data: [] });
    }
    res.json(cart);
  } catch (error) {
    res.status(500).json({ error: error.message, data: [] });
  }
}

async updateEntireCart(req, res) {
  try {
    const { userId } = req.params;
    const updatedCartData = req.body;
    const updatedCart = await this.cartService.updateEntireCart(
      userId,
      updatedCartData
    );
    if (!updatedCart) {
      return res.status(404).json({ message: "Cart not found" });
    }
    res.json(updatedCart);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
}
```

```
    }  
  }  
  async updateEachProduct(req, res) {  
    try {  
      const { userId, productId, quantity, select } = req.body;  
      const updatedCart = await this.cartService.updateEachProduct(  
        userId,  
        productId,  
        quantity,  
        select  
      );  
      if (quantity === 0) {  
        await this.cartService.removeItemFromCart(userId, productId);  
      }  
      if (!updatedCart) {  
        return res.status(404).json({ message: "Cart not found" });  
      }  
      res.json(updatedCart);  
    } catch (error) {  
      res.status(500).json({ error: error.message });  
    }  
  }  
  async deleteCart(req, res) {  
    try {  
      const { userId } = req.params;  
      const deletedCart = await this.cartService.deleteCart(userId);  
      if (!deletedCart) {  
        return res.status(404).json({ message: "Cart not found" });  
      }  
      res.json({ message: "Cart deleted successfully" });  
    } catch (error) {  
      res.status(500).json({ error: error.message });  
    }  
  }  
  async removeItemFromCart(req, res) {
```

```
    try {
      const { userId } = req.params;
      const { productId } = req.query;
      const updatedCart = await this.cartService.removeItemFromCart(
        userId,
        productId
      );

      if (!updatedCart) {
        return res
          .status(404)
          .json({ message: "Cart not found or item does not exist"
});
      }

      res.json(updatedCart);
    } catch (error) {
      res.status(500).json({ error: error.message });
    }
  }
}

module.exports = CartController;
```

3.3.3. Tạo các repository

Minh họa bằng CartRepo

```
const { convertToObjectIdMongodb } = require("../utils");
const Cart = require("../cart.model");

class CartRepository {
  async create(userId) {
    return await Cart.create({ userId });
  }
}
```

```
async getById(cartId) {
  return await Cart.findOne({
    _id: convertToObjectIdMongodb(cartId),
  }).populate("items.product");
}
async getByUserId(userId) {
  return await Cart.findOne({
    userId: convertToObjectIdMongodb(userId),
  }).populate("items.product");
}
async update(cartId, updatedCartData) {
  return await Cart.findOneAndUpdate(
    { _id: convertToObjectIdMongodb(cartId) },
    updatedCartData,
    { new: false }
  );
}
async updateEachProduct(userId, productId, quantity, select =
false) {
  try {
    const updatedCart = await Cart.findOneAndUpdate(
      {
        userId: convertToObjectIdMongodb(userId),
        "items.product": productId,
      },
      {
        $set: { "items.$.quantity": quantity, "items.$.select":
select },
      },
      { new: true }
    );

    return !!updatedCart; // Returns true if the cart was updated,
otherwise false
  }
}
```

```
    } catch (error) {
      throw new Error("Error updating cart item");
    }
  }
  async removeItemFromCart(cartId, productId) {
    return Cart.findOneAndUpdate(
      { _id: convertToObjectIdMongodb(cartId) },
      { $pull: { items: { product: productId } } },
      { new: true }
    );
  }
  async deleteCartById(cartId) {
    return Cart.findOneAndDelete(cartId);
  }
  async addItem(cartId, newItem) {
    return await Cart.findByIdAndUpdate(
      cartId,
      { $push: { items: newItem } },
      { new: true }
    );
  }

  // Other methods for cart operations...
}

module.exports = CartRepository;
```

3.3.4. Viết các services để sử dụng xử lý logic

Minh họa bằng CartService:

```
const CartRepository = require("../models/repositories/cart.repo");

class CartService {
  constructor() {
```

```
    this.cartRepository = new CartRepository();
  }
  async checkExistProduct(userId, productId) {
    const cart = await this.cartRepository.getByUserId(userId);
    const cartObject = cart ? JSON.parse(JSON.stringify(cart)) :
null;
    if (!cartObject) {
      return false;
    }

    const foundProduct = cartObject.items.find(
      (item) => item.product._id === productId
    );
    console.log({ foundProduct });
    return !!foundProduct;
  }
  async createCart(userId) {
    return await this.cartRepository.create(userId);
  }
  async getCartByUserId(userId) {
    return await this.cartRepository.getByUserId(userId);
  }
  async getCartById(cartId) {
    return await this.cartRepository.getById(cartId);
  }

  async updateEntireCart(userId, updatedCartData) {
    const cart = await this.getCartByUserId(userId);
    return await this.cartRepository.update(cart._id,
updatedCartData);
  }
  async updateEachProduct(userId, productId, quantity, select) {
    try {
      const updated = await this.cartRepository.updateEachProduct(
        userId,
```

```
        productId,
        quantity,
        select
    );
    return updated;
} catch (error) {
    throw new Error("Error updating cart product");
}
}
async addItemToCart(cartId, newItem) {
    return await this.cartRepository.addItem(cartId, newItem);
}
async deleteCart(userId) {
    const cart = await this.getCartByUserId(userId);
    return await this.cartRepository.deleteCartById(cart._id);
}
async removeItemFromCart(userId, productId) {
    const cart = await this.getCartByUserId(userId);
    return this.cartRepository.removeItemFromCart(cart._id,
productId);
}
}
module.exports = CartService;
```

3.4. Thiết kế giao diện

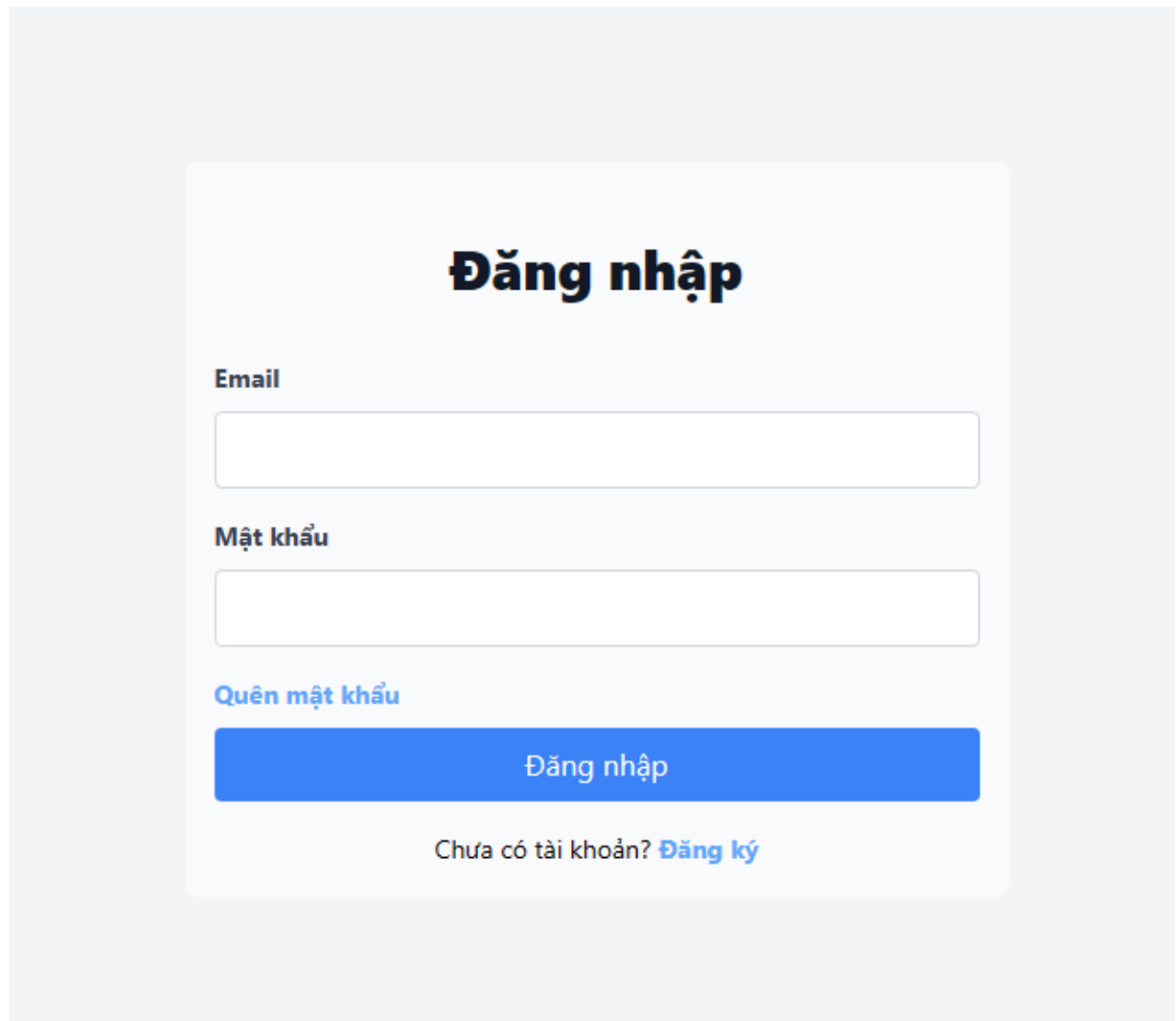
3.4.1. Thiết kế giao diện phía quản lý shop

Hình 7. Giao diện đăng ký thông tin shop thất bại

Nếu mật khẩu không đủ 6 ký tự trở lên và không đủ mạnh (Có ký tự in hoa, đặc biệt) hoặc mật khẩu nhập lại không khớp thì sẽ báo lỗi như hình trên

Hình 8. Giao diện khi đăng ký shop thành công

Khi đăng ký thông tin shop thành công, ứng dụng sẽ tự động quay về trang đăng nhập cho shop đăng nhập vào quản lý:



The image shows a login form titled "Đăng nhập" (Login) in bold black text. Below the title are two input fields: "Email" and "Mật khẩu" (Password). The "Email" field is a simple white box with a light gray border. The "Mật khẩu" field is a white box with a light gray border and a small eye icon on the right side to toggle visibility. Below the password field is a link "Quên mật khẩu" (Forgot password) in blue text. At the bottom of the form is a blue button with the text "Đăng nhập" in white. Below the button is a link "Chưa có tài khoản? Đăng ký" (Don't have an account? Register) in blue text.

Hình 9. Giao diện trang đăng nhập

Nếu đăng nhập thất bại, sẽ có thông báo nổi hiện lên:

Đăng nhập

Email
cloth@gmail.com

Mật khẩu
.....

[Quên mật khẩu](#)

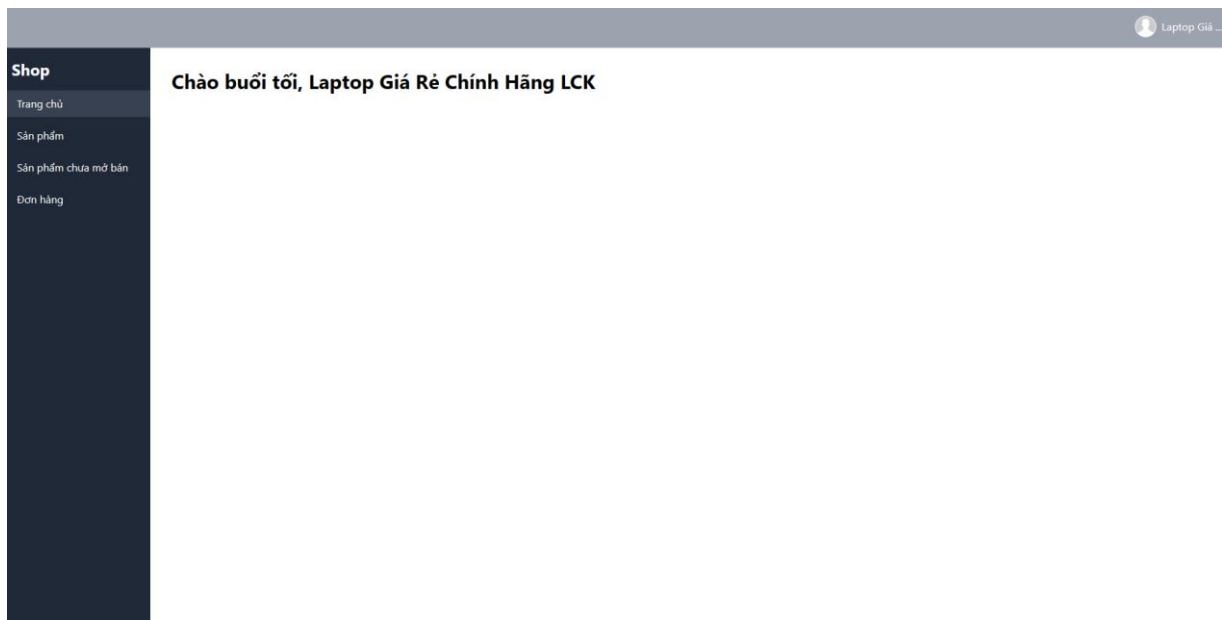
[Đăng nhập](#)

Chưa có tài khoản? [Đăng ký](#)

Đăng nhập thất bại!

Hình 10. Giao diện đăng nhập thất bại

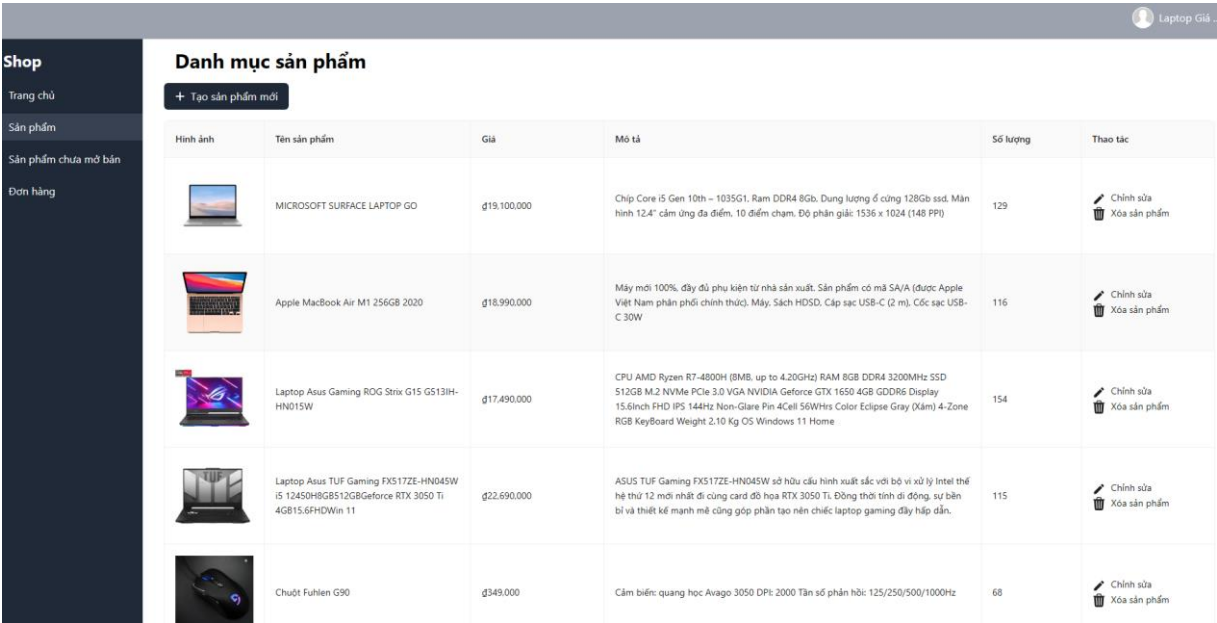
Khi đăng nhập thành công, ứng dụng sẽ tự động chuyển tới trang quản lý shop:



Hình 11. Giao diện quản lý shop

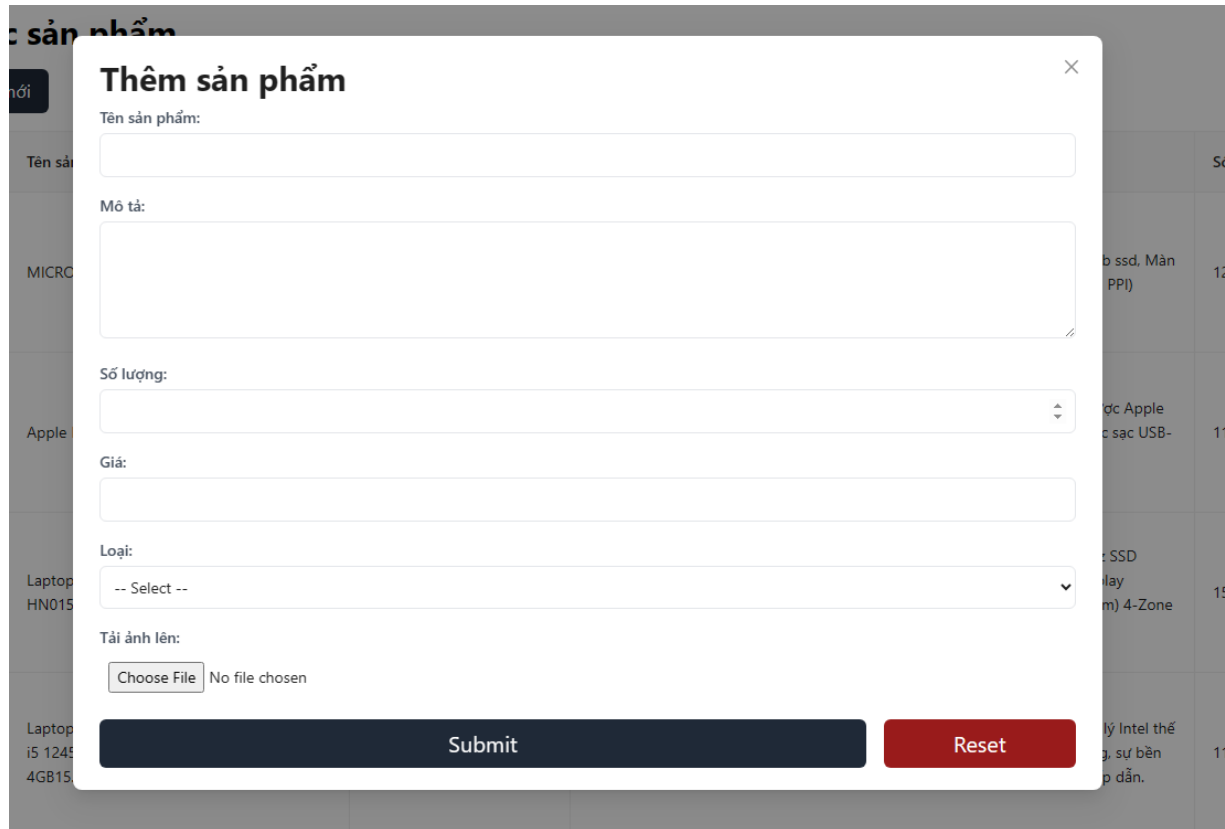
Một số chức năng cơ bản trong quản lý shop bao gồm:

Quản lý sản phẩm:



Hình 12. Giao diện danh sách sản phẩm

Tạo sản phẩm mới yêu cầu phải nhập vào các thông tin sau:



Thêm sản phẩm

Tên sản phẩm:

Mô tả:

Số lượng:

Giá:

Loại:


Tải ảnh lên:

Choose File No file chosen

Submit Reset

Hình 13. Giao diện form tạo sản phẩm

Trong đó phần tải ảnh lên sử dụng firebase để lưu trữ.

Hình ảnh	Tên sản phẩm	Giá	Mô tả	Số lượng	Thao tác
	Bàn phím cơ DAREU A87 CHILDHOOD	₫1.550.000	Keycap Childhood độc đáo Sử dụng Cherry switch cao cấp Dây kết nối USB Type-C có thể tháo rời	65	Chỉnh sửa Xóa sản phẩm

< 1 2 >

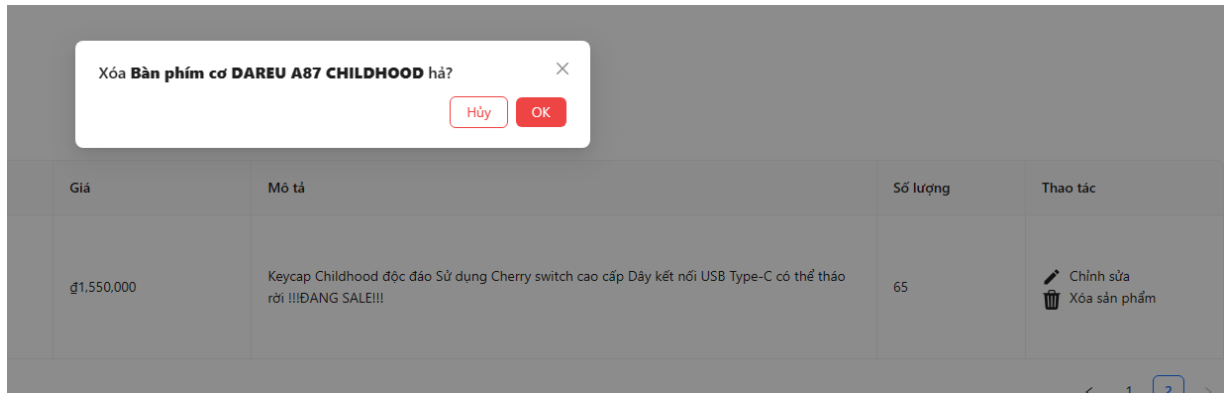
Hình 14. Hình ảnh sản phẩm mới được tạo ra

Sau khi tạo mới một sản phẩm, có thể chỉnh sửa thông tin về sản phẩm đó:

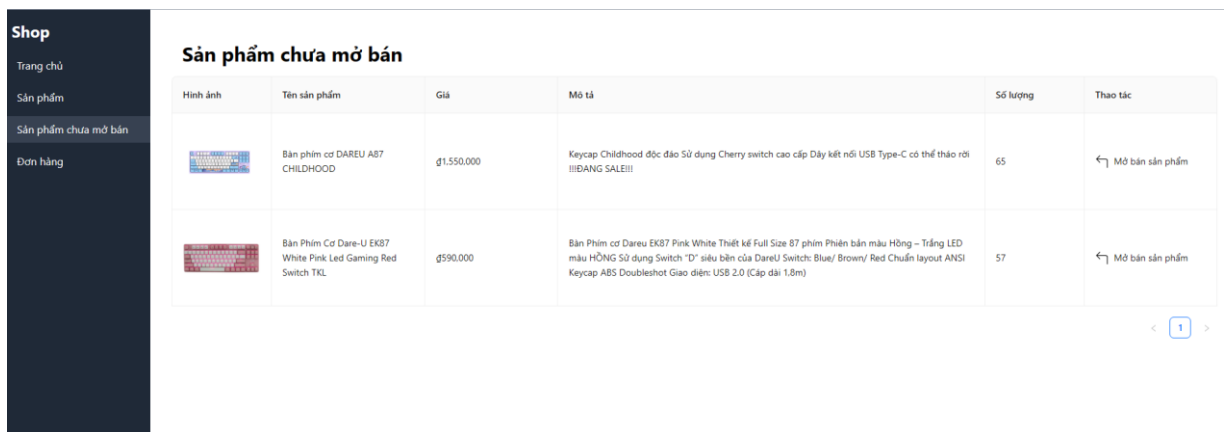
Hình 15. Giao diện chỉnh sửa thông tin sản phẩm

Hình 16. Hình ảnh sau khi chỉnh sửa thông tin sản phẩm

Khi ấn vào xóa sản phẩm, sẽ có Modal hiện ra để xác nhận. Sau khi xóa sản phẩm thì chỉ đưa sản phẩm vào trạng thái chưa mở bán trong bảng sản phẩm chưa mở bán:

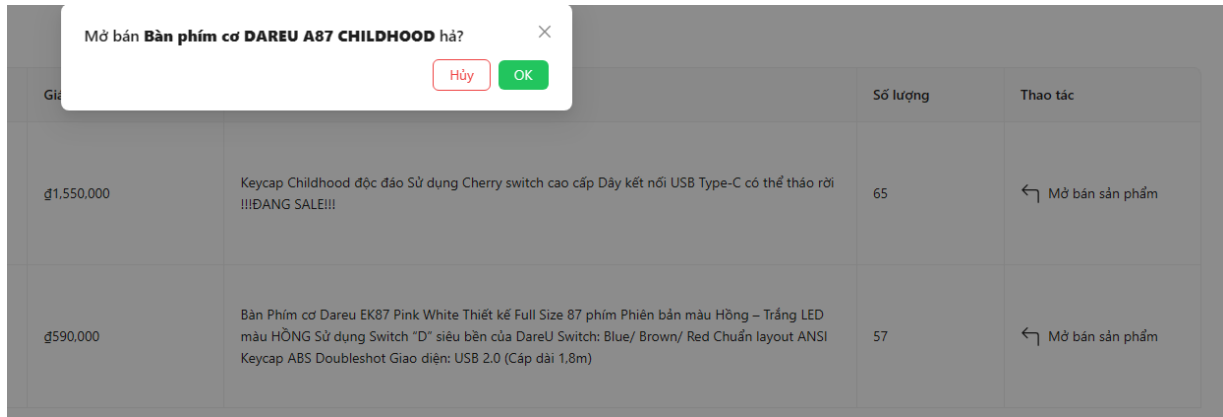


Hình 17. Giao diện xác nhận xóa sản phẩm

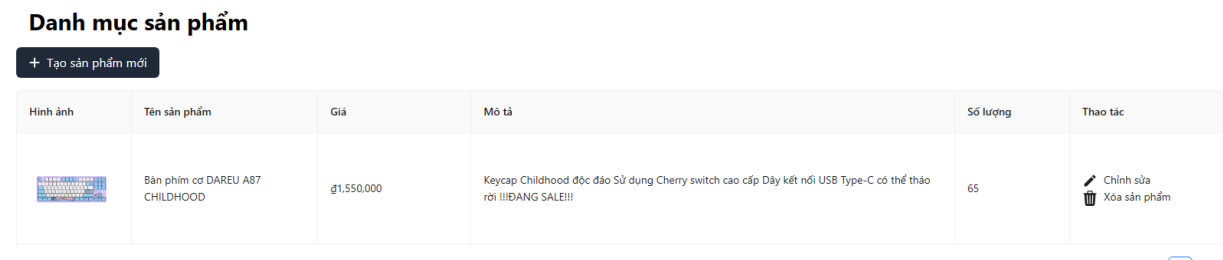


Hình 18. Giao diện sản phẩm chưa mở bán

Trong giao diện sản phẩm chưa mở bán, có thể ấn vào mở bán sản phẩm để đưa sản phẩm vào lại phần sản phẩm mở bán:

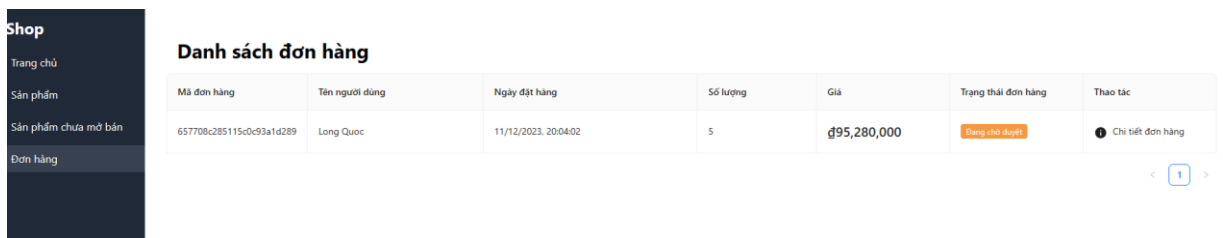


Hình 19. Giao diện xác nhận mở bán sản phẩm



Hình 20. Hình ảnh sản phẩm được mở bán lại

Quản lý đơn hàng:



Hình 21. Giao diện quản lý đơn hàng

Shop có thể ấn vào chi tiết đơn hàng để xem và duyệt đơn hoặc chỉnh sửa trạng thái đơn hàng đó:

Chi tiết đơn hàng

Mã đơn hàng: 657708c285115c0c93a1d289

Tên người dùng	Long Quoc	Email	longquoc47krb@gmail.com	Ngày đặt hàng	11/12/2023, 20:04:02
Tổng tiền	₫95,280,000	Số lượng hàng	5	Trạng thái	Đang chờ duyệt

Lưu thay đổi

Danh sách các sản phẩm

Mã sản phẩm	Tên sản phẩm	Số lượng	Giá
65755263b21e49f73d76fa45	MICROSOFT SURFACE LAPTOP GO	3	₫19,100,000
65755968b21e49f73d76fc66	Apple MacBook Air M1 256GB 2020	2	₫18,990,000

< 1 >

Hình 22. Giao diện chi tiết đơn hàng

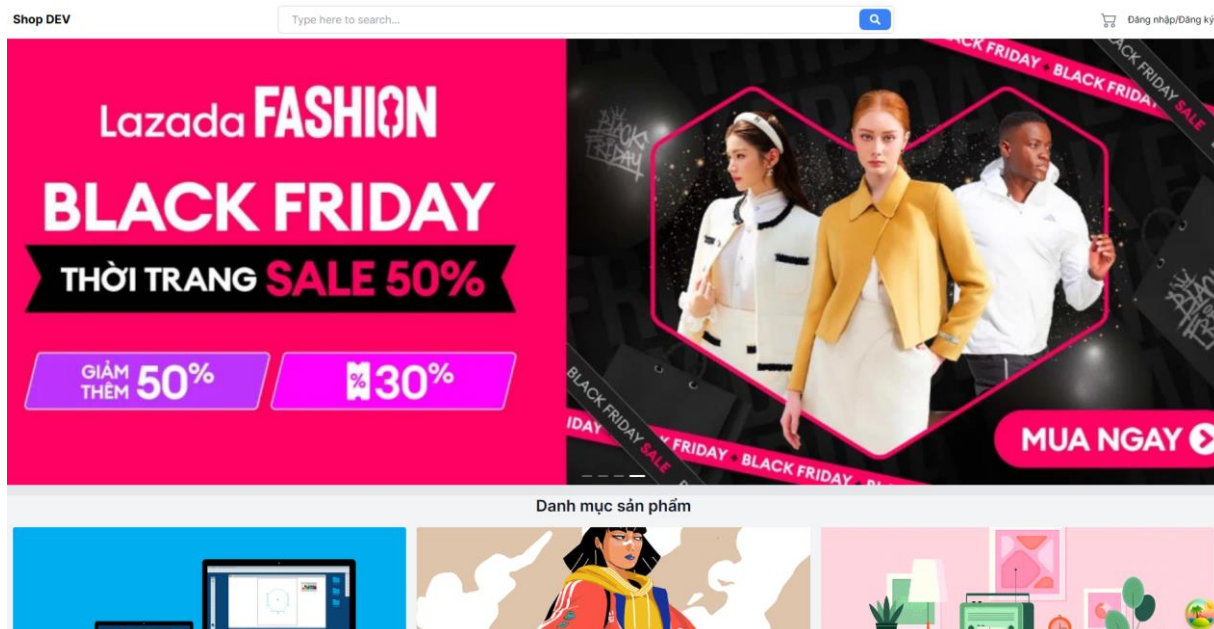
Danh sách đơn hàng						
Mã đơn hàng	Tên người dùng	Ngày đặt hàng	Số lượng	Giá	Trạng thái đơn hàng	Thao tác
657708c285115c0c93a1d289	Long Quoc	11/12/2023, 20:04:02	5	₫95,280,000	Đã xác nhận	Chi tiết đơn hàng

< 1 >

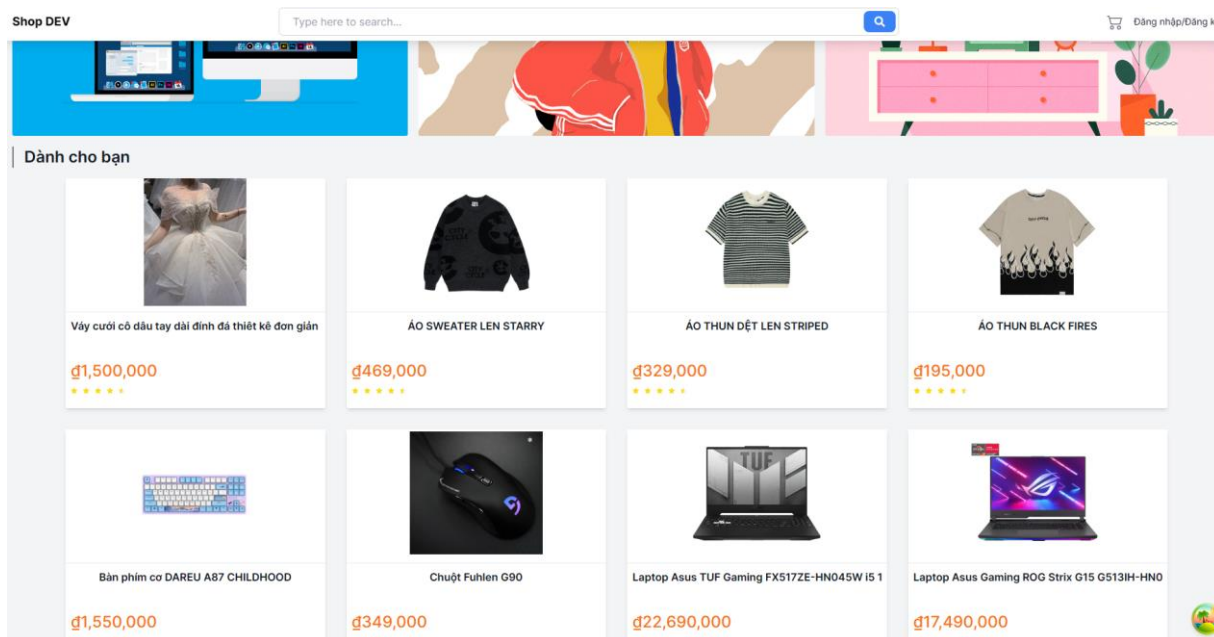
Hình 23. Giao diện sau khi thay đổi trạng thái đơn hàng

3.4.2. Thiết kế giao diện phía người mua hàng

Trang chủ của web được thiết kế như sau:



Hình 24. Giao diện trang chủ



Hình 25. Giao diện trang chủ kèm danh sách sản phẩm

Người mua có thể đăng ký tài khoản mới:

Đăng ký

Tên

Email

Mật khẩu

Nhập lại mật khẩu

Đăng ký

Đã có tài khoản? [Đăng nhập](#)

Hình 26. Giao diện trang đăng ký tài khoản

Sau khi đăng ký tài khoản, người dùng có thể đăng nhập:

Đăng nhập

Email

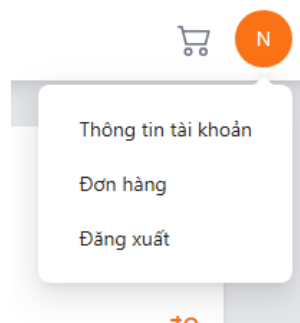
Mật khẩu

Đăng nhập

Chưa có tài khoản? [Đăng ký](#)

Hình 27. Giao diện trang đăng nhập

Người dùng có thể xem lại và chỉnh sửa thông tin tài khoản của mình trong mục Thông tin tài khoản:



Hình 28. Giao diện chức năng chỉnh sửa thông tin tài khoản

Khi ấn vào, trang chỉnh sửa thông tin tài khoản sẽ hiện ra:

Shop DEV

Type here to search...

THÔNG TIN NGƯỜI DÙNG

Tên
Quoc Lan Nguyen

Email
quoclan.nguyen2506@gmail.com

Số điện thoại
0947420012

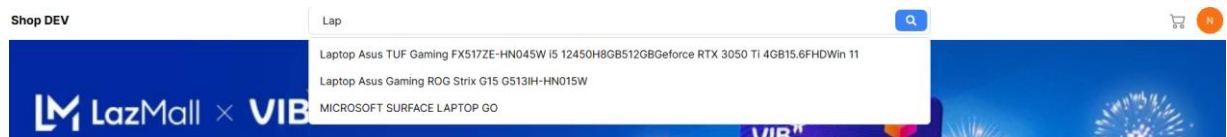
Địa chỉ (Số nhà, tên đường hoặc thôn, ấp)
123/12 Cộng Hòa

Tỉnh Điện Biên Huyện Tuần GiáoXã Ta Ma

Chỉnh Sửa

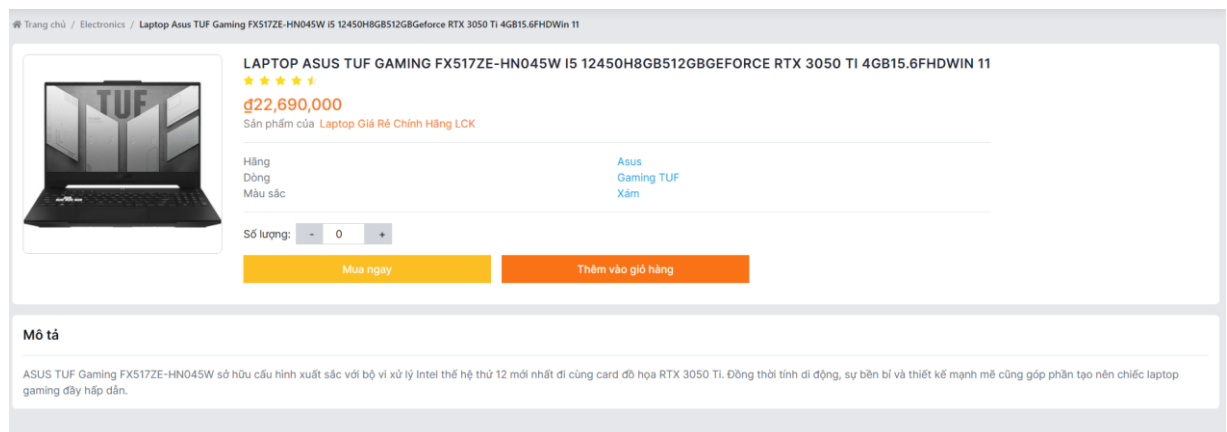
Hình 29. Giao diện chỉnh sửa thông tin người dùng

Ngoài ra, người dùng có thể xem sản phẩm bằng cách ấn vào sản phẩm muốn xem hoặc người dùng có thể search sản phẩm trên thanh tìm kiếm:



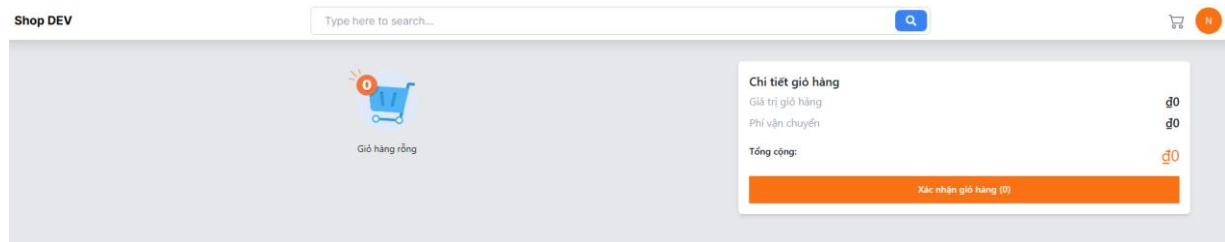
Hình 30. Giao diện thanh tìm kiếm

Khi ấn vào kết quả tìm kiếm, ứng dụng sẽ tự động chuyển tới trang sản phẩm đó:



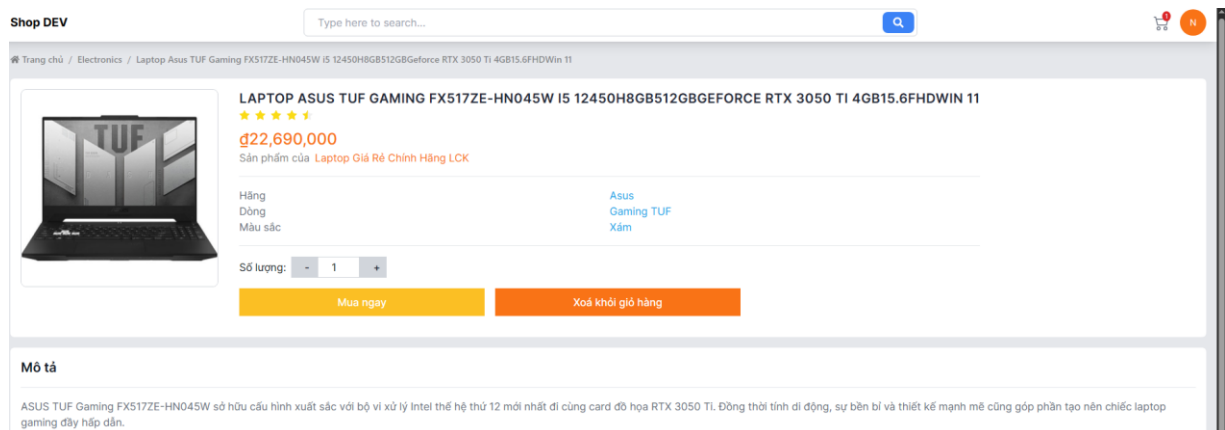
Hình 31. Giao diện một trang sản phẩm

Giao diện giỏ hàng được thiết kế như sau:

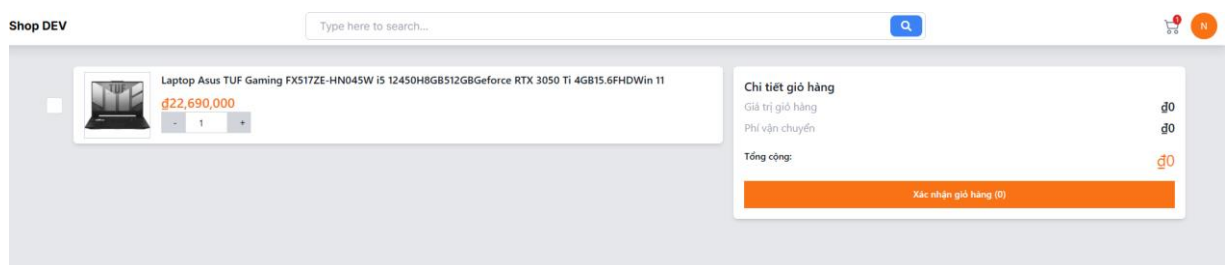


Hình 32. Giao diện giỏ hàng rỗng

Sau khi ấn nút “+” và ấn mua ngay, mặt hàng sẽ được thêm vào giỏ hàng của người dùng:

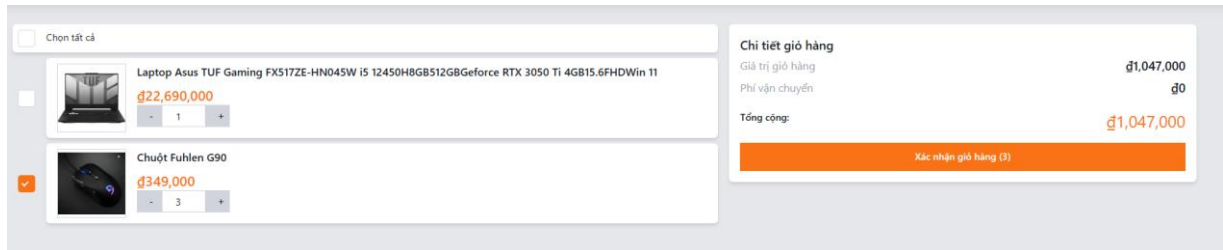


Hình 33. Giao diện sản phẩm khi thêm vào giỏ hàng

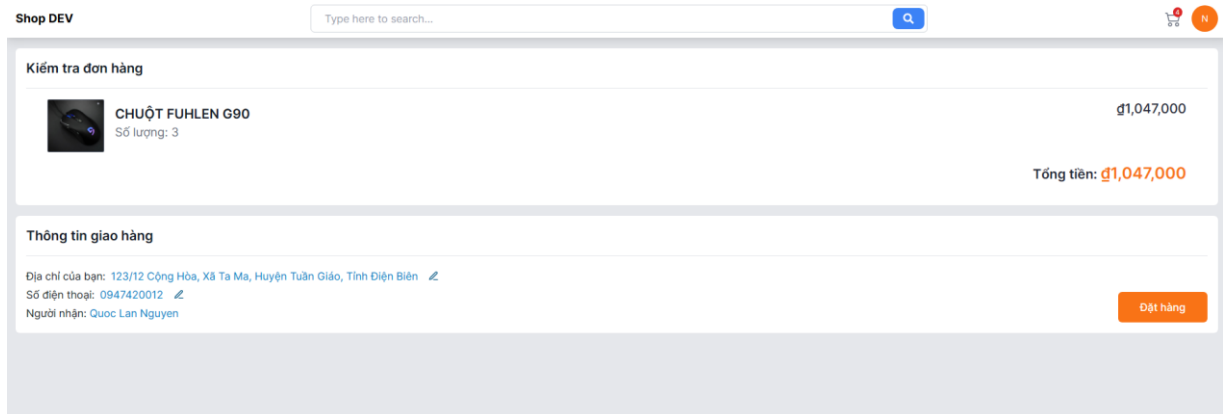


Hình 34. Giao diện giỏ hàng sau khi thêm vào giỏ

Sau khi thêm vào giỏ hàng, người dùng có thể thực hiện thanh toán bằng cách tích vào những mặt hàng muốn thanh toán:

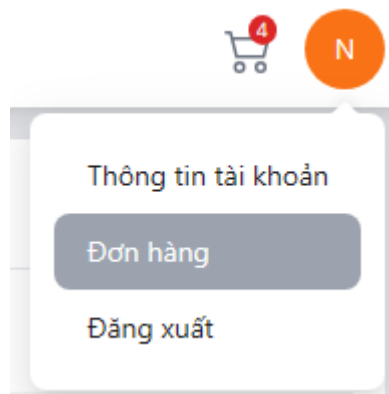


Hình 35. Giao diện thanh toán tại giỏ hàng

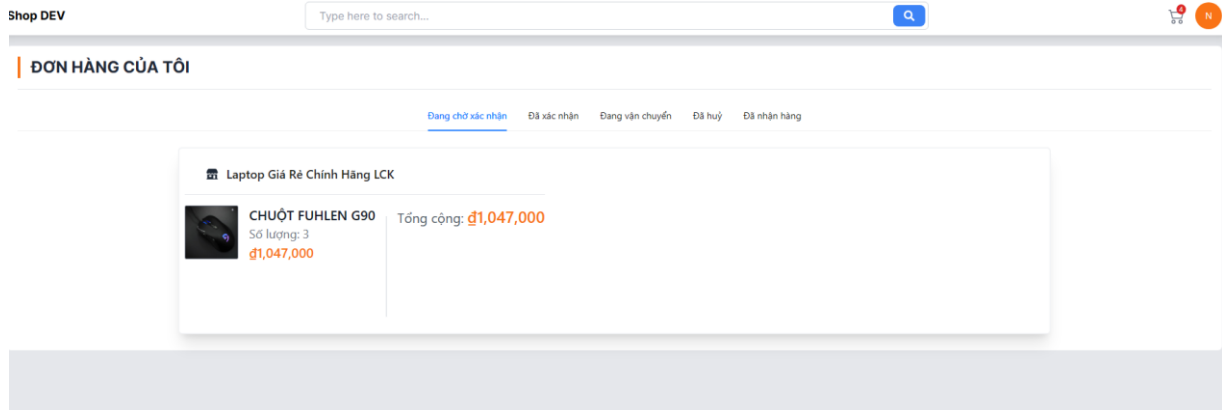


Hình 36. Giao diện hiển thị thông tin giao hàng

Sau khi đặt hàng thành công, đơn hàng có thể được xem lại trong mục xem lại đơn hàng:



Hình 37. Giao diện xem lại đơn hàng



Hình 38. Giao diện xem lại thông tin chi tiết đơn hàng

Khi đó, phía bên shop sẽ nhận được đơn hàng như sau:

Danh sách đơn hàng

Mã đơn hàng	Tên người dùng	Ngày đặt hàng	Số lượng	Giá	Trạng thái đơn hàng	Thao tác
657708c285115c0c93a1d289	Long Quoc	11/12/2023, 20:04:02	5	₫95,280,000	Đã xác nhận	Chi tiết đơn hàng
65774e76ef9adb60a016abc0	Quoc Lan Nguyen	12/12/2023, 01:01:26	3	₫1,047,000	Đang chờ duyệt	Chi tiết đơn hàng

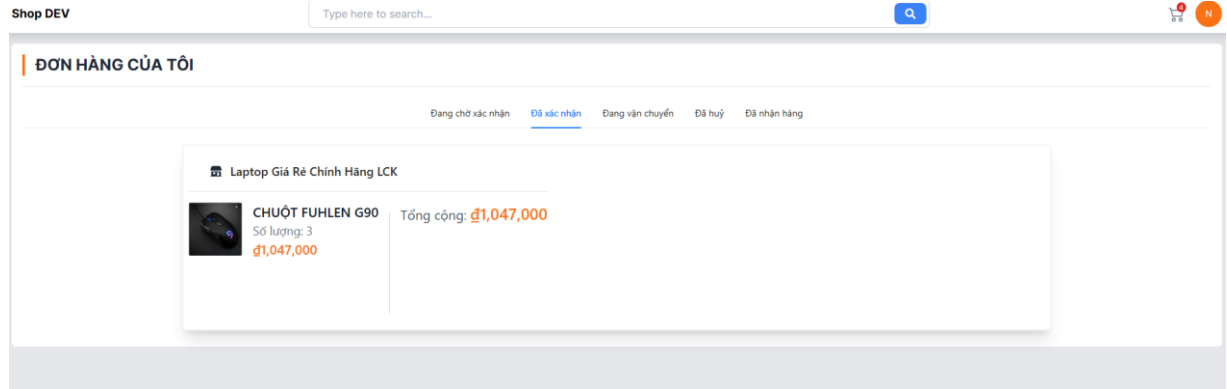
Hình 39. Giao diện đơn hàng được cập nhật

Danh sách đơn hàng

Mã đơn hàng	Tên người dùng	Ngày đặt hàng	Số lượng	Giá	Trạng thái đơn hàng	Thao tác
657708c285115c0c93a1d289	Long Quoc	11/12/2023, 20:04:02	5	₫95,280,000	Đã xác nhận	Chi tiết đơn hàng
65774e76ef9adb60a016abc0	Quoc Lan Nguyen	12/12/2023, 01:01:26	3	₫1,047,000	Đã xác nhận	Chi tiết đơn hàng

Hình 40. Giao diện chỉnh sửa trạng thái đơn hàng

Khi shop đã thay đổi trạng thái đơn hàng, phía người mua sẽ thấy đơn hàng của mình được cập nhật như sau:



Hình 41. Giao diện đơn hàng đã được xác nhận

CHƯƠNG 4. PHÂN CÔNG CÔNG VIỆC

4.1. Bảng phân công công việc

Bảng 12. Phân công công việc

STT	Tên công việc đã làm	Thành viên thực hiện
1	Thiết kế giao diện, màn hình phía shop quản lý	Nguyễn Quốc Lâm
2	Thiết kế giao diện, màn hình phía client	Nguyễn Quốc Lâm Nguyễn Anh Hào
3	Thiết kế cơ sở dữ liệu	Đỗ Anh Khoa Nguyễn Anh Hào
4	Viết API cho hệ thống	Đỗ Anh Khoa
5	Tìm hiểu nghiên cứu về AI trong phát triển website	Nguyễn Anh Hào
6	Tổng kết và viết báo cáo	Đỗ Anh Khoa Nguyễn Quốc Lâm

4.2. Bảng tỷ lệ hoàn thành công việc

Bảng 13. Tỷ lệ đóng góp

STT	Tên thành viên nhóm	Tỷ lệ hoàn thành
1	Nguyễn Quốc Lâm	100%
2	Đỗ Anh Khoa	100%
3	Nguyễn Anh Hào	100%

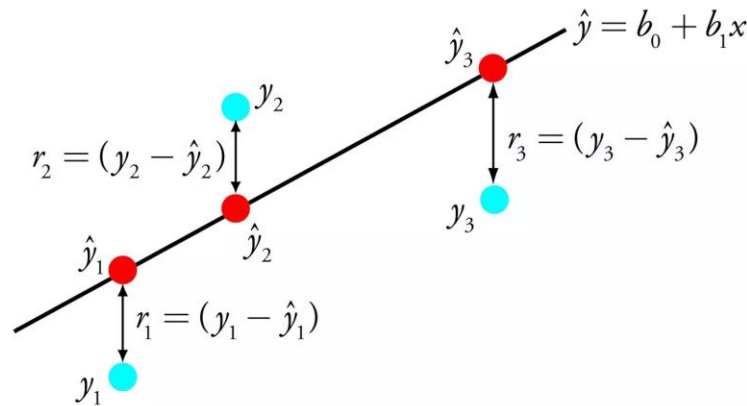
CHƯƠNG 5. NGHIÊN CỨU VỀ AI TRONG PHÁT TRIỂN WEBSITE

5.1. Một vài thuật toán cơ bản trong Machine Learning

5.1.1. Linear Regression

Hồi quy tuyến tính là một trong những thuật toán nổi tiếng nhất trong Machine Learning. Đây là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại.

Hồi quy tuyến tính được phát minh vào khoảng hơn 200 năm trước và được nghiên cứu rộng rãi. Đây là một thuật toán tốt, nhanh chóng và dễ sử dụng.



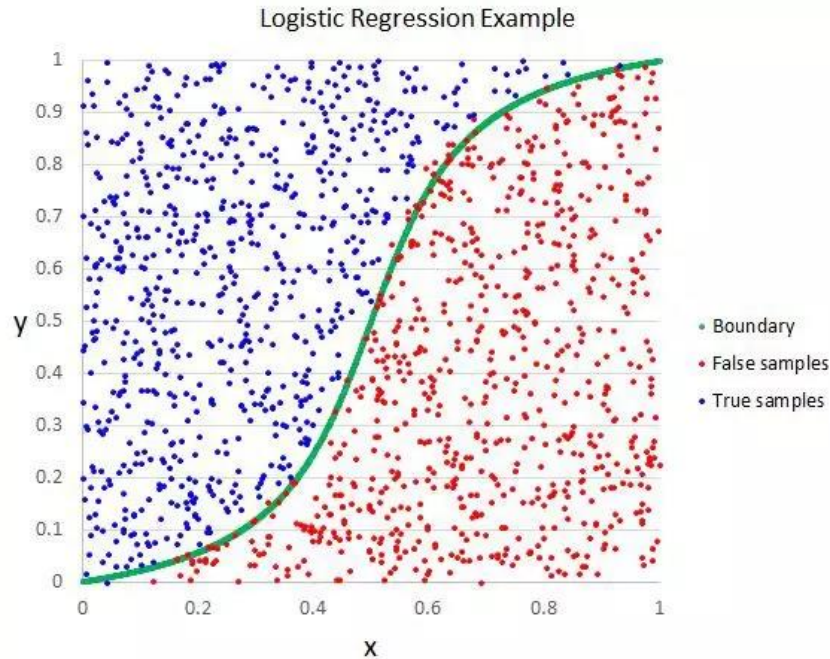
Hình 42. Đường hồi quy

5.1.2. Logistic Regression

Logistic Regression (hay còn được gọi là hồi quy logistic) được sử dụng để ước tính các giá trị rời rạc (thường là các giá trị nhị phân như 0/1) từ một tập hợp các biến độc lập. Hồi quy logistic giúp dự đoán xác suất của một sự kiện bằng cách khớp dữ liệu với một hàm logit.

Tương tự hồi quy tuyến tính, hồi quy logistic sẽ hoạt động tốt hơn khi loại bỏ các thuộc tính không liên quan đến biến đầu ra hoặc các thuộc tính tương tự nhau.

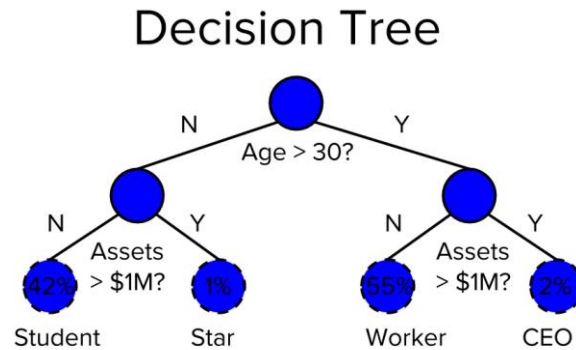
Đây là mô hình có thể học được nhanh và có hiệu quả với các vấn đề phân loại nhị phân.



Hình 43. Đồ thị phân nhóm dữ liệu theo đường logistic

5.1.3. Decision Tree

Decision Tree là một thuật toán học tập có giám sát được sử dụng để phân loại các vấn đề. Decision Tree hoạt động tốt khi phân loại cho cả biến phụ thuộc phân loại và biến phụ thuộc liên tục. Trong thuật toán này có thể chia tổng thể thành hai hoặc nhiều tập đồng nhất dựa trên các thuộc tính hoặc biến độc lập quan trọng nhất.

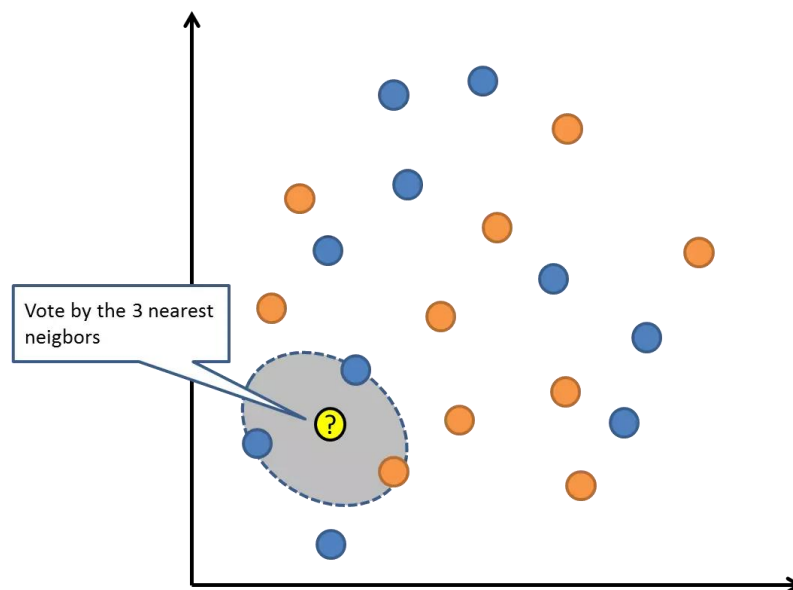


Hình 44. Một cây quyết định đơn giản

5.1.4. Thuật toán K-Nearest Neighbors

Thuật toán này có thể được áp dụng cho cả bài toán phân loại và bài toán hồi quy. Đây là một thuật toán đơn giản lưu trữ tất cả các trường hợp có sẵn và phân loại bất kỳ trường hợp mới nào bằng cách lấy đa số “phiếu bầu” (vote) của K láng giềng (neighbor). Sau đó ta gán điểm chung nhất thông qua đo khoảng cách.

Chẳng hạn như biểu đồ dưới đây, ta chọn $K = 3$. Khi đó 3 láng giềng này sẽ vote ra một điểm chung nhất là dấu chấm hỏi thông qua việc đo khoảng cách từ đó có thể thực hiện hồi quy hay phân loại tùy ý.



Hình 45. Trường hợp $K = 3$ trong KNN

Tuy nhiên, cần cân nhắc vài điều trước khi chọn KNN như: Các biến phải được chuẩn hóa, nếu không các biến có phạm vi cao hơn có thể làm sai lệch thuật toán; dữ liệu vẫn cần được xử lý trước.

5.1.5. Thuật toán Random Forest

Random Forest là một tập gồm nhiều Decision Tree. Để phân lớp một đối tượng mới dựa vào thuộc tính mà nó có, mỗi Decision Tree sẽ được phân loại và “bầu” cho lớp đó.

Cụ thể, thuật toán này diễn ra theo các bước sau:

1. Chọn các mẫu ngẫu nhiên từ tổng thể đã cho.
2. Thiết lập Decision Tree cho từng mẫu và nhận kết quả dự đoán mỗi Decision Tree.
3. Bỏ phiếu bầu cho mỗi kết quả dự đoán.
4. Chọn kết quả được bỏ phiếu nhiều nhất làm kết quả.



Hình 46. Mô hình Random Forest

5.2. Tiềm năng của Machine Learning ứng dụng vào website

Các mô hình được đề cập ở trên có thể được áp dụng vào phát triển website như sau:

- *Linear Regression (Hồi quy tuyến tính)*: Dự đoán các yếu tố như thời gian duyệt web, số lượng truy cập, doanh số bán hàng dựa trên các biến độc lập như chiến dịch tiếp thị, quảng cáo.
- *Logistic Regression (Hồi quy logistic)*: Phân loại người dùng hoặc khách hàng có khả năng chuyển đổi (ví dụ: đăng ký, mua hàng) dựa trên hành vi trực trạng trang web.
- *Decision Tree (Cây quyết định)*: Tối ưu hóa trải nghiệm người dùng bằng cách đưa ra quyết định về nội dung hiển thị, dựa trên các đặc trưng như khu vực địa lý, loại thiết bị, hoặc nguồn truy cập.
- *K-Nearest Neighbors (KNN)*: Xác định nhóm người dùng tương tự để đề xuất nội dung hoặc sản phẩm tương tự mà họ có thể quan tâm.
- *Random Forest*: Dự đoán xu hướng và thị trường dựa trên dữ liệu lịch sử, giúp định hình chiến lược kinh doanh trên trang web.

Trong thực tế, các công ty và tổ chức hàng đầu thế giới cũng đã và đang phát triển website và ứng dụng có tích hợp và được phát triển bằng AI, Machine Learning, ví dụ như

- *Amazon* sử dụng ML để cá nhân hóa trải nghiệm mua sắm của người dùng. Amazon sử dụng dữ liệu mua sắm của người dùng để đề xuất các sản phẩm phù hợp với sở thích của người dùng.
- *Google* sử dụng AI để tạo ra các chatbot thông minh. Google Assistant là một chatbot thông minh có thể trả lời các câu hỏi của người dùng và thực hiện các yêu cầu của người dùng.
- *Facebook* sử dụng ML để phát hiện các hoạt động bất thường trên mạng xã hội. Facebook sử dụng ML để phát hiện các tài khoản giả mạo, các bài đăng có nội dung khiêu dâm, hoặc các hành vi vi phạm chính sách của Facebook.

5.3. Hạn chế của Machine Learning và Artificial Intelligence trong phát triển website

Dù có tiềm năng lớn là vậy, tuy nhiên việc ứng dụng ML vào quá trình phát triển website cũng gặp phải không ít khó khăn. Thứ nhất, cần phải có nguồn dữ liệu chất lượng thông qua tiền xử lý (thu thập dữ liệu, bias trong dữ liệu). Thứ hai, khả năng mở rộng trở nên

hạn chế do bản chất của quá trình training AI, ML đòi hỏi rất nhiều tài nguyên như điện, làm mát, chạy máy ảo, ... Do đó, nếu ngân sách hạn chế thì khó có thể áp dụng được ML vào website. Thứ ba, rủi ro an ninh bảo mật có thể xảy ra, vì việc training AI có thể trở thành mục tiêu bị tấn công nếu sử dụng đám mây công cộng đối với các tổ chức thiếu thốn về ngân sách. Và cuối cùng, việc duy trì và cập nhật mô hình ML, AI phải được thực hiện thường xuyên thậm chí là realtime, do đó cần đội ngũ kỹ sư AI bảo trì thường xuyên dẫn tới tốn kém, cần nhân lực có trình độ cao.

CHƯƠNG 6. KẾT LUẬN

6.1. Tổng kết

Trong quá trình thực hiện đồ án, từ thời điểm nhận đề tài cho đến nay, nhóm đã gặp nhiều khó khăn và không tránh khỏi những mâu thuẫn, tranh cãi xảy ra. Tuy nhiên cuối cùng nhóm đã hoàn thành tốt những nhiệm vụ được giao và hoàn thiện được đồ án.

6.2. Kết quả đạt được

- Học hỏi được công nghệ mới: MERN stack và biết cách ứng dụng vào đồ án môn học
- Rèn luyện được kỹ năng tự học, kỹ năng tìm kiếm tài liệu, kỹ năng tìm kiếm giải pháp để giải quyết những khúc mắc gặp phải trong quá trình thực hiện đồ án
- Nâng cao kỹ năng làm việc, thảo luận nhóm, phân chia công việc giữa các thành viên trong nhóm
- Hoàn thiện được ứng dụng web là nền tảng để phát triển thương mại điện tử cho các cửa hàng.

6.3. Ưu điểm và nhược điểm

6.3.1. Ưu điểm

- Website có giao diện dễ nhìn, thân thiện với người dùng
- Hỗ trợ người mua dễ dàng liên lạc với người bán, dễ dàng mua hàng
- Hỗ trợ người bán dễ dàng trở thành nhà phân phối sản phẩm

6.3.2. Nhược điểm

- Nhóm sử dụng cơ sở dữ liệu là MongoDB để lưu trữ dữ liệu với dung lượng cho phép là 512MB, do đó khá hạn chế về mặt lưu trữ dữ liệu
- Chưa tối ưu được việc re-render các component

- Còn nhiều chức năng chưa hoàn thiện do chưa đủ kiến thức và thời gian như thanh toán qua thẻ ngân hàng, chọn vị trí giao hàng qua GoogleMaps, ...

6.4. Khó khăn

6.4.1. Khó khăn về công nghệ

Bảng 14. Khó khăn về công nghệ

STT	Khó khăn	Khắc phục
1	MERN stack là công nghệ mới đồng thời nhóm chưa có nhiều kinh nghiệm nên trong quá trình thực hiện đồ án, nhóm gặp khó khăn trong bước đầu tiếp cận và tìm hiểu về công nghệ	Nghiên cứu và tham khảo thêm nhiều tài liệu từ Google, Udemy
2	Khó khăn trong việc lưu trữ hình ảnh và file của người dùng	Tìm giải pháp lưu hình ảnh trên clouinary và lưu file ở Firebase.

6.4.2. Khó khăn về quy trình thực hiện

Bảng 15. Khó khăn về quy trình thực hiện

STT	Khó khăn	Khắc phục
1	Bị lỗi conflict khi pull hoặc push code trên github	Sử dụng source control để quản lý nhánh cá nhân thành viên
2	Gặp khó khăn trong việc thống nhất ý kiến	Lắng nghe ý kiến của nhau, từ

	của các thành viên trong nhóm	đó đưa ra quyết định chung
--	-------------------------------	----------------------------

6.5. Bài học kinh nghiệm

- Học được cách quản lý source code trên github
- Biết lắng nghe ý kiến của thành viên trong nhóm, cùng nhóm tìm ra giải pháp tốt nhất cho đề tài

6.6. Hướng phát triển

- Xây dựng thêm nhiều giải pháp cho người dùng thanh toán hơn thay vì chỉ có ship COD
- Xây dựng chức năng chọn địa chỉ giao hàng thông qua GoogleMaps, GPS.
- Xây dựng chức năng gợi ý sản phẩm thông qua AI
- Xây dựng chức năng chatbot
- Xây dựng chức năng theo dõi quá trình đơn hàng (tới kho, đang chuyển,...)
- Xây dựng các chức năng thống kê cho shop

TÀI LIỆU THAM KHẢO

- [1].<https://www.mongodb.com/languages/mern-stack-tutorial> *How to Use MERN Stack: A Complete Guide*, MongoDB, date accessed: 11/12/2023
- [2].<https://www.freecodecamp.org/news/how-to-build-a-react-project-with-create-react-app-in-10-steps/> *How to Build a React Project with Create React App in 10 Steps*, Reed Barger, date accessed: 11/12/2023
- [3].<https://tailwindcss.com/docs/guides/create-react-app> *Install Tailwind CSS with Create React App*, TailwindCSS document, date accessed: 11/12/2023