

BỘ GIAO THÔNG VẬN TẢI
HỌC VIỆN HÀNG KHÔNG VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN



**ĐỒ ÁN CHUYÊN NGÀNH
CÔNG NGHỆ THÔNG TIN**

**“SỬ DỤNG REACT, NODEJS VÀ MONGODB XÂY DỰNG
WEBSITE MUA VÀ ĐỌC TRUYỆN TRỰC TUYẾN”**

Giảng viên hướng dẫn: TS. Tô Bá Lâm, ThS. Huỳnh Thanh Sơn

Nhóm sinh viên thực hiện: Nhóm 05

Mã số sinh viên:

Lớp: 010100086402

TP.Hồ Chí Minh, tháng 12/2024

BỘ GIAO THÔNG VẬN TẢI
HỌC VIỆN HÀNG KHÔNG VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN



**ĐỒ ÁN CHUYÊN NGÀNH
CÔNG NGHỆ THÔNG TIN**

**“SỬ DỤNG REACT, NODEJS VÀ MONGODB XÂY DỰNG
WEBSITE MUA VÀ ĐỌC TRUYỆN TRỰC TUYẾN”**

Giảng viên hướng dẫn: TS. Tô Bá Lâm, ThS. Huỳnh Thanh Sơn

Nhóm sinh viên thực hiện: Nhóm 05.....

Mã số sinh viên:

Lớp: 010100086402

Thành phố Hồ Chí Minh, tháng 12/2024

Danh sách Nhóm:

STT	Họ và tên	MSSV	Lớp	Ghi chú
1	Nguyễn Hồ Trường Giang	2154810037	21ĐHTT01	Nhóm Trường
2	Trần Đình Đạt	2154810039	21ĐHTT01	Thành Viên
3	Nguyễn Văn Sơn	2254810168	22ĐHTT04	Thành Viên
4	Vũ Thành Đạt	2254810175	22ĐHTT04	Thành Viên
5	Nguyễn Gia Hy	2154810005	21ĐHTT01	Thành Viên

Cán bộ chấm thi 1 <i>(ký và ghi rõ họ tên)</i>	Cán bộ chấm thi 2 <i>(ký và ghi rõ họ tên)</i>
Cán bộ chấm thi phúc khảo 1 <i>(ký và ghi rõ họ tên)</i>	Cán bộ chấm thi phúc khảo 2 <i>(ký và ghi rõ họ tên)</i>

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu, chữ viết tắt	Chữ viết đầy đủ
MVC	Model-View-Controller
DOM	Document Object Model
ID	Identification

DANH MỤC HÌNH ẢNH

Hình 2.1. Framework NodeJS	3
Hình 2.2. Framework ExpressJS	4
Hình 2.3. Framework ReactJS	4
Hình 2.4. Công nghệ DOM trong ReactJS	5
Hình 2.5. Cơ sở dữ liệu mongoDB	5
Hình 3.1. Mô hình hệ thống Web đọc truyện tranh.....	7
Hình 3.2. Sơ đồ Use-case New Reader và Reader	8
Hình 3.3. Sơ đồ Use-case Admin	8
Hình 3.4. Bảng Users.....	9
Hình 3.5. Bảng Comics	9
Hình 3.6. Bảng Chapters	10
Hình 3.7. Bảng Carts	10
Hình 3.8. Bảng Payments	11
Hình 3.9. Bảng Vouchers	11
Hình 3.10. Bảng Orders.....	13
Hình 3.11. Sơ đồ quan hệ thực thể	13
Hình 3.12. Giao diện trang chủ	14
Hình 3.13. Giao diện đăng nhập	15
Hình 3.14. Giao diện đăng ký.....	15
Hình 3.15. Giao diện tìm kiếm truyện.....	17
Hình 3.16. Giao diện yêu thích.....	18
Hình 3.17. Giao diện thông tin người dùng.....	19
Hình 3.18. Giao diện liên hệ.....	20
Hình 3.19. Giao diện chi tiết truyện	22
Hình 3.20. Giao diện chương truyện	23
Hình 3.21. Giao diện chương truyện	24
Hình 3.22. Giao diện giỏ hàng.....	25
Hình 3.23. Giao diện thông tin giao hàng	26
Hình 3.24. Giao diện xác nhận giao hàng	28
Hình 3.25. Giao diện chi tiết đơn hàng.....	30

Hình 3.26. Phương thức thanh toán.....	32
Hình 3.27. Thông tin thanh toán.....	32
Hình 3.28. Xác thực thanh toán.....	32
Hình 3.29. Thanh toán thành công	33
Hình 3.30. Giao diện đăng nhập của Admin	33
Hình 3.31. Giao diện bảng của trang Admin.....	35
Hình 3.32. Giao diện chỉnh sửa truyện của trang Admin.....	36
Hình 3.33. Giao diện chỉnh sửa chương của Admin	38
Hình 3.34. Giao diện chỉnh sửa voucher của trang Admin	41
Hình 3.35. Giao diện Admin quản lý đơn hàng user.....	44
Hình 3.36. Giao diện Admin quản lý đơn hàng user.....	44

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT	i
DANH MỤC HÌNH ẢNH	ii
MỞ ĐẦU	vi
CHƯƠNG 1. GIỚI THIỆU	1
1.1. Lý do chọn đề tài	1
1.2. Mục tiêu đề tài	1
1.3. Phạm vi đề tài	1
1.4. Đối tượng nghiên cứu.....	2
1.5. Phương pháp nghiên cứu.....	2
1.6. Bố cục đề tài	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	3
2.1. Node.js	3
2.2. ExpressJS	3
2.3. Framework React.....	4
2.3.1 React là gì	4
2.3.2 Lý do chọn React trong đề tài.....	5
2.4. MongoDB	5
CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM	7
3.1. Phân tích hệ thống.....	7
3.1.1. Mô hình hệ thống.....	7
3.1.2. Use case	8
3.2. Thiết kế cơ sở dữ liệu	9
3.2.1. Thiết kế bảng	9
3.2.2. Sơ đồ quan hệ thực thể	13
3.3. Xây dựng giao diện sản phẩm.....	13
3.3.1 Giao diện trang chủ.....	13
3.3.2 Giao diện trang chi tiết truyện	15
3.3.3 Giao diện tìm kiếm truyện	16
3.3.4 Giao diện yêu thích truyện.....	18
3.3.5 Giao diện thông tin người dùng	19
3.3.6 Giao diện liên hệ	20
3.3.7 Giao diện trang chi tiết truyện	22

3.3.8 Giao diện xem chương truyện	23
3.3.9 Giao diện giờ hàng	25
3.3.10 Giao diện giao hàng	26
3.3.11 Giao diện xác nhận đơn hàng	27
3.3.12 Giao diện chi tiết đơn hàng.....	29
3.3.13 Giao diện thanh toán	31
3.3.14 Giao diện Admin Login	33
3.3.15 Giao diện Admin Dashboard	34
3.3.16 Giao diện Admin Chính sửa truyện tranh.....	36
3.3.17 Giao diện Admin chỉnh sửa chương truyện.....	38
3.3.18 Giao diện Admin chỉnh sửa voucher	41
3.3.19 Giao diện Admin quản lý đơn hàng user	43
KẾT LUẬN	45
DANH MỤC TÀI LIỆU THAM KHẢO	46

MỞ ĐẦU

Trong một thế giới ngày tiên tiến, nhu cầu giải trí là điều không thể phủ nhận. Việc tìm kiếm và thưởng thức những bộ truyện tranh mới hay đang nổi tiếng trên các nền tảng xã hội đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Tuy nhiên, chúng em đã nhận thấy một thực tế đáng chú ý rằng trong khi một số tư nhân hoặc doanh nghiệp đã áp dụng hình thức đặt mua và đọc truyện trực tuyến thông qua website, nhưng vẫn còn rất nhiều doanh nhân hoặc tổ chức cá nhân khác vẫn duy trì thói quen mua sách báo tại các nhà sách truyền thống. Điều này thúc đẩy nhóm chúng em đặt ra câu hỏi: Tại sao không tận dụng sức mạnh của Internet, khi mà mọi người đều có một chiếc điện thoại hoặc một chiếc laptop trong người?

Chính vì lẽ đó, nhóm của chúng em quyết định thực hiện dự án nghiên cứu về Node.js, React và MongoDB, ba công nghệ hiện đang rất phát triển và được sử dụng rộng rãi trong lĩnh vực phát triển phần mềm. Mục tiêu của chúng em là áp dụng những kiến thức từ các công nghệ này vào việc xây dựng một hệ thống ứng dụng đặt mua và đọc truyện trực tuyến, giúp người dùng có thể trải nghiệm các nhu cầu giải trí một cách thuận tiện và nhanh chóng.

CHƯƠNG 1. GIỚI THIỆU

1.1. Lý do chọn đề tài

Dưới sự cạnh tranh ngày càng khốc liệt của thị trường truyện tranh, việc xây dựng một trang web truyện tranh không chỉ là bước đi sáng tạo mà còn có tiềm năng phát triển, giúp cho người dùng có thể dễ dàng tiếp cận với nhiều bộ truyện hơn so với phương pháp đọc truyện truyền thống trên sách. Ngoài việc tạo ra một trang web có nhiều truyện để thu hút người đọc, việc trang web có giao diện bắt mắt và dễ sử dụng cũng là một điểm cộng để lôi kéo người dùng đọc truyện. Do đó, nhóm em đã lựa chọn đề tài này để có thể áp dụng những kiến thức đã học để xây dựng một trang web với những tiêu chí trên. Đồng thời đề tài giúp chúng em nâng cao khả năng làm việc nhóm thông qua những công việc như là thiết kế giao diện, xây dựng cơ sở dữ liệu đến tối ưu trang web, những công việc này cần kỹ năng giao tiếp, chia sẻ ý tưởng và cùng nhau giải quyết vấn đề, tạo tiền đề cho việc mài dũa kỹ năng cho mỗi người.

1.2. Mục tiêu đề tài

- Nghiên cứu và xây dựng ứng dụng đọc truyện trực tuyến thông qua việc sử dụng các công nghệ: Node.js, React và MongoDB.
- Phân tích và thiết kế các chức năng cần thiết cho ứng dụng đọc truyện tranh.
- Xây dựng ứng dụng đẹp mắt, dễ sử dụng và mang lại trải nghiệm tốt cho người dùng.

1.3. Phạm vi đề tài

- Phát triển nền tảng: Xây dựng một nền tảng đọc truyện trực tuyến, đa dạng về nội dung và tính năng, giao diện thân thiện và dễ sử dụng cho người dùng.
- Đa dạng thể loại, nội dung: Cung cấp đa dạng nội dung từ nhiều loại truyện khác nhau để đáp ứng nhu cầu giải trí của người dùng.
- Trải nghiệm người dùng: Phát triển các tính năng khác như là đánh giá, bình luận, chia sẻ để giúp người dùng cảm thấy tiện lợi hơn.

1.4. Đối tượng nghiên cứu

Tìm hiểu, nghiên cứu và áp dụng các kiến thức như: sơ đồ MVC, sơ đồ Use case, sơ đồ tuần tự,... và các công cụ hoặc framework như Node.js, Flutter và MongoDB vào việc xây dựng ứng dụng đọc truyện tranh của nhóm.

1.5. Phương pháp nghiên cứu

- Phương pháp thu thập thông tin: Tìm hiểu, đọc các nguồn tài liệu trên mạng.
- Phương pháp xử lý thông tin: Thu thập, khảo sát ý kiến từ những người quen, thân cận bên ngoài.
- Phương pháp thực nghiệm: Lập trình thực nghiệm trên môi trường lập trình VSCode.

1.6. Bố cục đề tài

Phần còn lại của báo cáo đồ án chuyên ngành được tổ chức như sau:

Chương 2 nhóm 6 sẽ trình bày các cơ sở lý thuyết về những công nghệ và framework có liên quan đến việc xây dựng giao diện người dùng (front-end) và các tác vụ xử lý bên dưới (back-end) của sản phẩm đề tài.

Tiếp đến trong Chương 3, nhóm em sẽ giới thiệu chi tiết về sơ đồ use-case của từng chức năng như đăng nhập/đăng ký, đọc truyện..., sơ đồ tuần tự của các chức năng chính và cách thiết kế cơ sở dữ liệu của trang web mua và đọc truyện trực tuyến.

Cuối cùng ở phần Kết luận, nhóm sẽ tóm tắt lại những chức năng đã xây dựng được cho trang web, những gì còn chưa làm được và đưa ra hướng phát triển của trang web trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Node.js

Node.js là một môi trường chạy Javascript miễn phí được xây dựng dựa trên công cụ Javascript V8, phần cốt lõi của Google Chrome. Nó cho phép phát triển các ứng dụng hiệu năng cao với khả năng mở rộng tốt, xử lý các yêu cầu đồng thời mà không làm nghẽn bộ nhớ [1].



Hình 2.1. Framework NodeJS

NodeJS được lựa chọn sử dụng cho việc phát triển giao diện Website của nhóm nhờ vào các ưu điểm sau [2]:

- + **Hiệu năng cao:** Một trong những tính năng nổi bật của Node.js là khả năng tạo ra một ứng dụng có khả năng phản hồi kết quả chỉ trong vài giây. Không những thế, Node.js còn cung cấp khả năng đa nhiệm, thử cực kỳ phù hợp cho việc phát triển website và ứng dụng di động. Do được thiết kế theo hướng sự kiện và đơn luồng nên nó có thể xử lý nhiều tác vụ yêu cầu từ phía Clients mà không bị tắc nghẽn RAM.
- + **Khả năng mở rộng:** Node.js hỗ trợ phát triển các hệ thống phân tán và có khả năng mở rộng mạnh mẽ, giúp ứng dụng có thể xử lý hàng triệu kết nối đồng thời.
- + **Cộng đồng hỗ trợ lớn:** NPM là một gói quản lý thư viện toàn diện nhất thế giới; nó cung cấp cho các lập trình viên rất nhiều công cụ và thư viện mà ta có thể sử dụng ngay bên trong dự án đang được thực hiện. Ngoài ra, nhiều công ty công nghệ lớn như Amazon, Facebook, Netflix đã đóng góp đáng kể cho cộng đồng Node.js.

2.2. ExpressJS

Express.js là một framework tối giản cho Node.js, giúp đơn giản hóa việc phát triển server bằng cách cung cấp các công cụ, tiện ích để xây dựng các API RESTful và xử lý HTTP request [3].



Hình 2.2. Framework ExpressJS

Chức năng:

- Định nghĩa các route để xử lý các yêu cầu HTTP (GET, POST, PUT, DELETE).
- Cung cấp các middleware để xử lý dữ liệu từ các request (JSON parsing, URL encoding, etc.).

Lợi ích khi dùng Expressjs:

- Express.js là một framework nhẹ và nhanh, dễ dàng mở rộng thông qua các middleware.
- Hỗ trợ mô hình MVC (Model-View-Controller), giúp dễ dàng tổ chức và quản lý code.

2.3. Framework React



Hình 2.3. Framework ReactJS

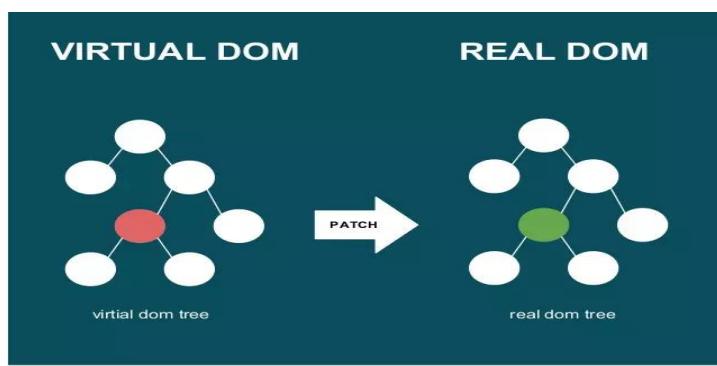
2.3.1 React là gì

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. là một thư viện UI hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được, React được sử dụng tại Facebook trong production, Instagram được viết hoàn toàn trên React [4].

2.3.2 Lý do chọn React trong đề tài

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM [5].

Công nghệ DOM ảo giúp tăng hiệu năng cho ứng dụng. Việc chỉ node gốc mới có trạng thái và khi nó thay đổi sẽ tái cấu trúc lại toàn bộ, đồng nghĩa với việc DOM tree cũng sẽ phải thay đổi một phần, điều này sẽ ảnh hưởng đến tốc độ xử lý. React JS sử dụng Virtual DOM (DOM ảo) để cải thiện vấn đề này. Virtual DOM là một object Javascript, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi nó sẽ tính toán sự thay đổi giữa object và tree thật, điều này sẽ giúp tối ưu hóa việc re-render DOM tree thật.



Hình 2.4. Công nghệ DOM trong ReactJS

React là một thư viện rất thú vị và được phát triển dựa trên rất nhiều cấu trúc phức tạp. Tuy nhiên thư viện này lại rất dễ sử dụng và thêm vào trong nhiều application khác nhau.

2.4. MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL, sử dụng cấu trúc dữ liệu dưới dạng document (tài liệu), mỗi document được lưu trữ dưới dạng BSON (Binary JSON). MongoDB thường được sử dụng cùng với Node.js thông qua các thư viện như Mongoose để kết nối, tương tác với cơ sở dữ liệu và tạo các API [6].



Hình 2.5. Cơ sở dữ liệu mongoDB

Ưu điểm của MongoDB:

- Khả năng mở rộng cao (scalable).
- Lưu trữ dữ liệu dưới dạng JSON-like, dễ dàng truy vấn và quản lý.
- Thích hợp cho các ứng dụng có cấu trúc dữ liệu không cố định.

Ứng dụng:

MongoDB được sử dụng trong hệ thống như một nơi lưu trữ dữ liệu người dùng, dữ liệu giao dịch, nội dung, và có thể đồng bộ hóa giữa ứng dụng di động và web thông qua các API.

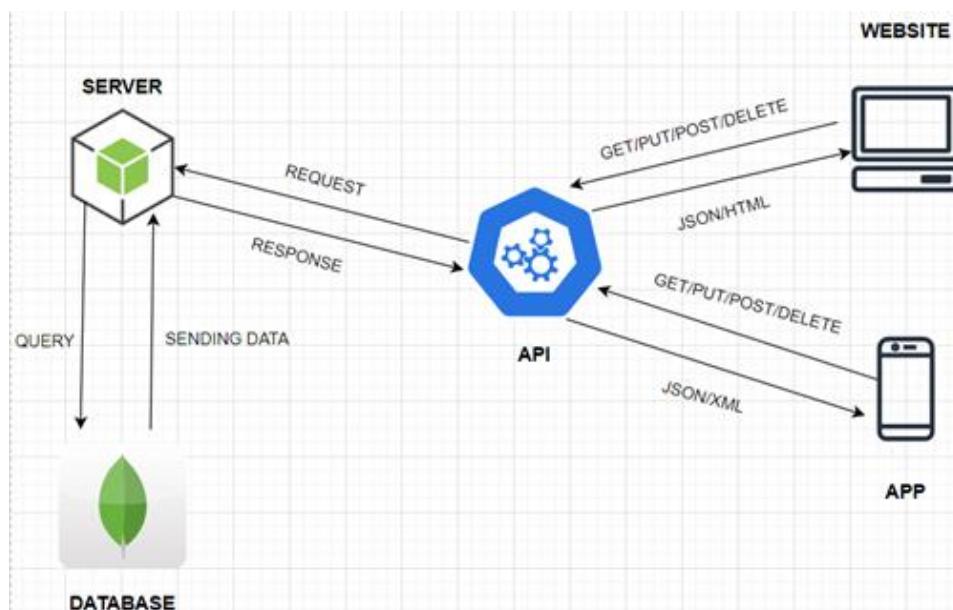
CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM

3.1. Phân tích hệ thống

3.1.1. Mô hình hệ thống

Mô hình hệ thống website sẽ bao gồm các thành phần chính như sau:

- Server: Chịu trách nhiệm nhận các yêu cầu (request) từ API như các phương thức GET/PUT/POST/DELETE và tiến hành truy vấn cơ sở dữ liệu, sau đó phản hồi về phía API các tập dữ liệu được truy vấn và cuối cùng là API sẽ trả về website/app một tập tin JSON, HTML hoặc XML tùy vào yêu cầu bên phía giao diện.
- API: Sử dụng các phương thức của HTTP như GET/PUT/POST từ phía Client. Là cầu nối trong việc gửi yêu cầu lên Server, tiếp nhận các phản hồi từ phía Server và trả về các file JSON/HTML/XML về phía giao diện.
- Client: Là các giao diện người dùng trên nền tảng Windows (Website) và các thiết bị di động (App). Chúng sẽ chịu trách nhiệm nhận các phản hồi từ Server và hiển thị các thông tin nhận được lên các giao diện người dùng.
- Database: Lưu trữ các thông tin dữ liệu như người dùng, truyền tranh. Phục vụ việc truy vấn dữ liệu và trả về các dữ liệu được truy vấn về phía Server.



Hình 3.1. Mô hình hệ thống Web đọc truyện tranh

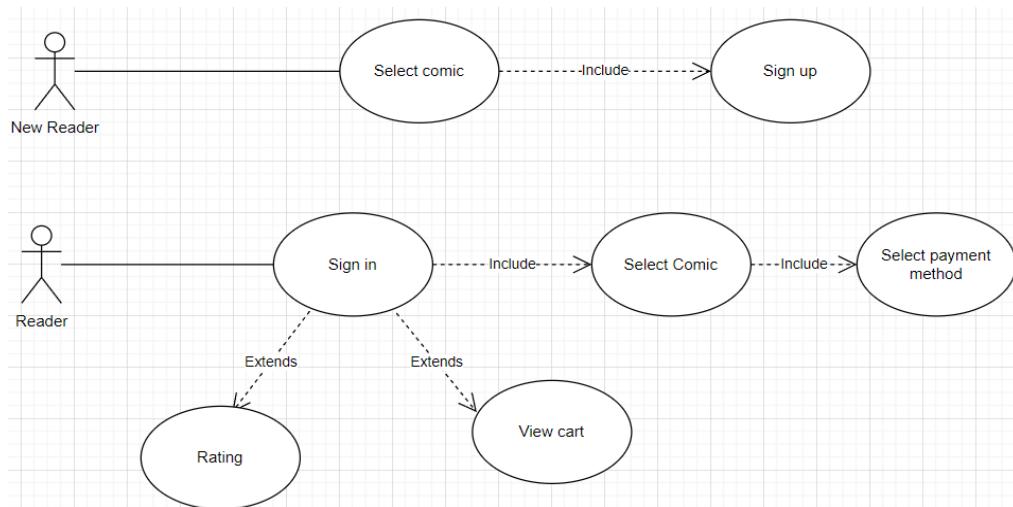
3.1.2. Use case

Sơ đồ use-case sẽ bao gồm 2 đối tượng chính là : *Reader* và *Admin*

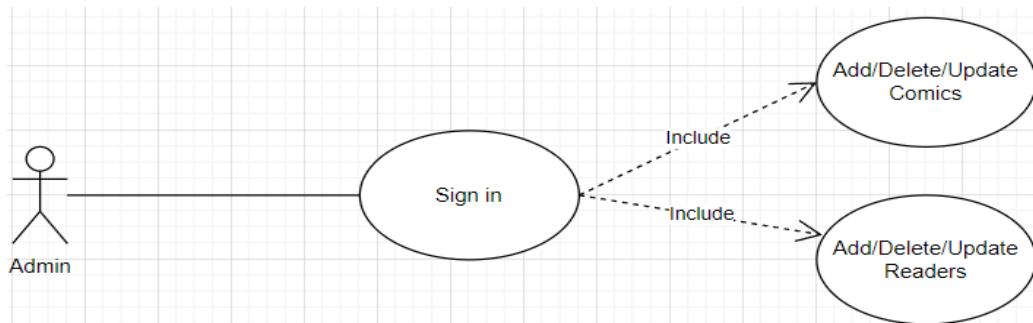
Reader: Được chia thành 2 đối tượng nhỏ hơn là *New Reader* và *Reader*.

- + *New Reader*: là các người dùng chưa từng hoặc lần đầu sử dụng các chức năng của hệ thống.
- + *Reader*: là những người dùng đã có tài khoản tồn tại trên database, role *Reader* sẽ có các quyền như: chọn truyện, mua và đọc truyện trực tuyến trên website.

Admin: Sở hữu các quyền quản lý như quản lý tài khoản *Reader* và quản lý truyện trên hệ thống website.



Hình 3.2. Sơ đồ Use-case New Reader và Reader



Hình 3.3. Sơ đồ Use-case Admin

3.2. Thiết kế cơ sở dữ liệu

3.2.1. Thiết kế bảng

Website mua và đọc truyện sẽ bao gồm 6 bảng như sau:

Users	
PK	UserID
	userName
	email
	passWord
	role
	createdAt

Hình 3.4. Bảng Users

- Bảng Users: được sử dụng trong việc lưu trữ các tài khoản của các đọc giả cũ và đọc giả mới vừa đăng ký trong lúc sử dụng các tính năng trên website của nhóm. Bảng sẽ bao gồm các trường như sau:
 - + username: Hiển thị tên người dùng trên website.
 - + email: Dùng để đăng nhập vào website.
 - + password: Sẽ được đưa vào bảng băm thành một chuỗi mã hóa và sau đó sẽ được lưu trên MongoDB.
 - + role: Phân quyền giữa đọc giả và quản trị viên.
 - + createdAt: Lưu trữ thời gian đăng ký tài khoản dưới dạng ngày tháng năm.

Comics	
PK	comicID
	title
	author
	description
	price
	cover_image
	reviews
	numReviews
	rating

Hình 3.5. Bảng Comics

- Bảng Comics: Lưu trữ các mẫu truyện tranh mà nhóm muốn đăng bán. Bảng sẽ bao gồm các trường như sau:
 - + title: Hiển thị tên truyện.
 - + author: Hiển thị tên tác giả của truyện.
 - + genre: Hiển thị thể loại lên trang web.
 - + description: Mô tả ngắn gọn nội dung của truyện.
 - + cover_image: Lưu trữ ảnh dùng để hiển thị trên trang web
 - + reviews: Danh sách đánh giá, chứa user, name, rating, comment.
 - + numReviews: Số lượng đánh giá.
 - + rating: Điểm đánh giá trung bình.

Chapters	
PK	chapterID
FK	comicID
	chapterNumber
*	title
	releaseDate
	pageCount
	url

Hình 3.6. Bảng Chapters

- Bảng Chapters: Lưu trữ các hình ảnh của mẫu truyện liên quan thông qua comicID. Bảng sẽ bao gồm các trường như sau:
 - + comicID: Dùng để hiển thị các trang truyện lên truyện có ID tương ứng.
 - + chapterNumber: Chương của truyện.
 - + title: Tên của chương truyện đó.
 - + releaseDate: Ngày phát hành chương.
 - + pageCount: Tổng số trang của chương.
 - + url: Lưu trữ đường dẫn hình ảnh chương để hiển thị trên trang web.

Carts	
PK	CartID
FK	userID
	items
	totalPrice
	createdAt
	updatedAt

Hình 3.7. Bảng Carts

- Bảng Carts: Lưu trữ thông tin giỏ hàng. Bảng sẽ bao gồm các trường như sau:
 - + cartID: ID của giỏ hàng.
 - + userID: ID của khách hàng sở hữu giỏ hàng tương ứng.
 - + items: Lưu trữ các truyện mà khách hàng muốn mua dưới dạng mảng.
 - + totalPrice: Tổng tiền phải thanh toán.
 - + createdAt: Ngày tạo giỏ hàng.
 - + updatedAt: Ngày cập nhật sản phẩm trên giỏ hàng.

Payments	
PK	PaymentID
FK	userID
FK	chapterID
	createdAt

Hình 3.8. Bảng Payments

- Bảng Payments: Lưu trữ thông tin đơn hàng. Bảng sẽ bao gồm các trường như sau:
 - + paymentID: ID của đơn hàng.
 - + userID: ID của khách hàng sở hữu đơn hàng tương ứng.
 - + chapterID: ID của chương truyện cần mua.
 - + createdAt: Ngày tạo đơn hàng.

Vouchers	
PK	voucherID
	code
	discount
	expiryDate

Hình 3.9. Bảng Vouchers

- Bảng Vouchers: Lưu trữ các voucher giảm giá. Bảng sẽ bao gồm các trường như sau:
 - + voucherID: ID của voucher.
 - + code: Mã code dùng để nhập và kích hoạt voucher.
 - + discount: Phần trăm giảm giá của voucher.

- + expiryDate: Ngày hết hạn.

Orders	
PK	orderID
FK	userID
	orderItems
	shippingAddress
	paymentMethod
	totalPrice
	voucherCode
	isPaid
	paidAt
	isDelivered
	deliveredAt

Hình 3.10. Bảng Orders

- Bảng Orders: Lưu trữ thông tin đơn hàng cho người dùng. Bảng sẽ bao gồm các trường như sau:

- + user: Người dùng.
- + orderItems: Danh sách sản phẩm, gồm:
name: Tên sản phẩm.

qty: Số lượng.

price: Giá mỗi sản phẩm.

product: Sản phẩm liên kết.

- + shippingAddress: Địa chỉ giao hàng, gồm:
street: Đường.

ward: Phường.

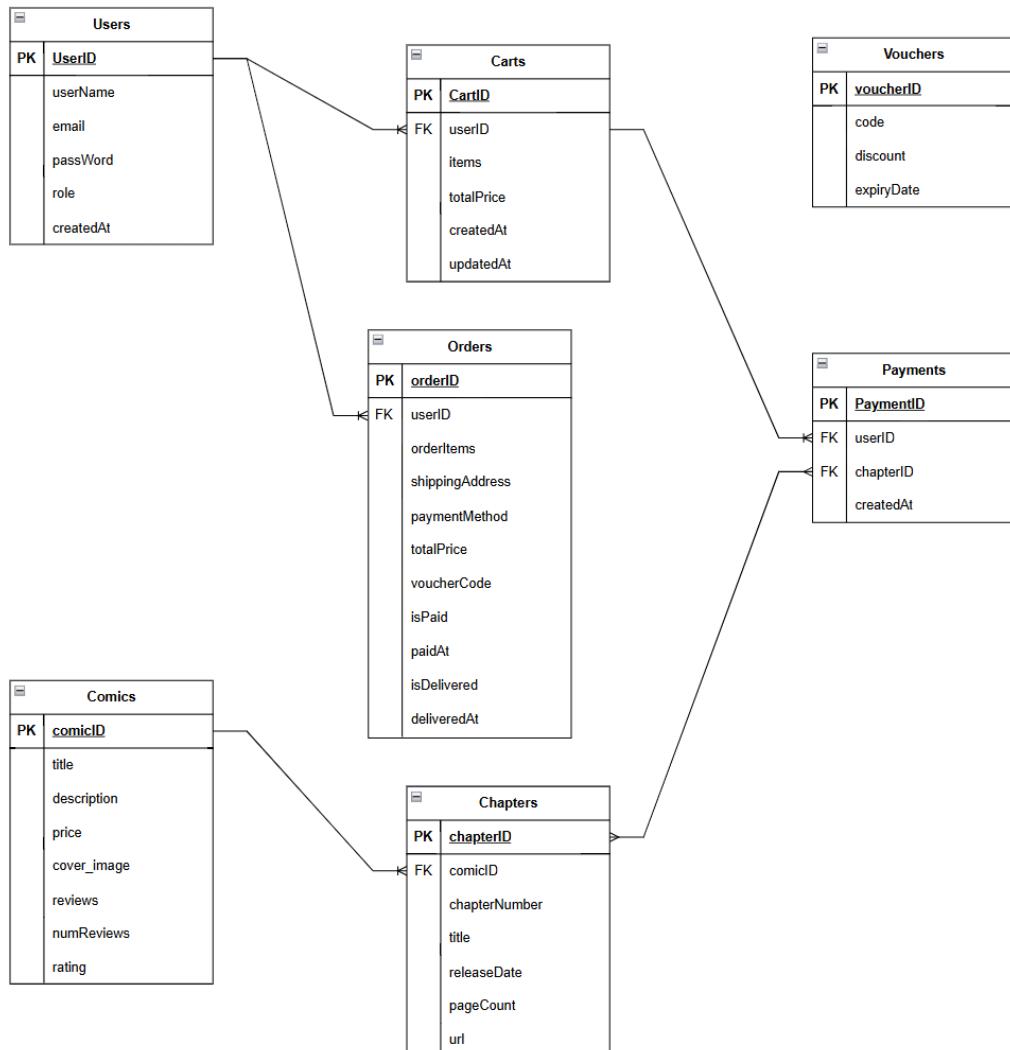
district: Quận/Huyện.

city: Thành phố.

- + paymentMethod: Phương thức thanh toán.
- + totalPrice: Tổng giá trị đơn hàng.

- + voucherCode: Mã giảm giá.
- + isPaid: Đã thanh toán.
- + paidAt: Thời gian thanh toán.
- + isDelivered: Đã giao hàng.
- + deliveredAt: Thời gian giao hàng (Date).

3.2.2. Sơ đồ quan hệ thực thể



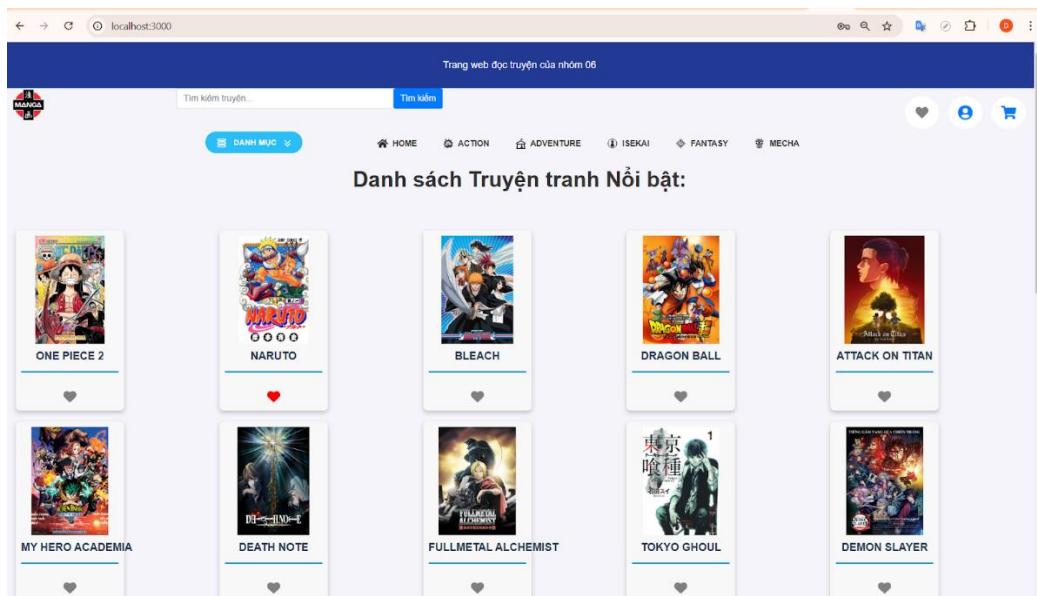
Hình 3.11. Sơ đồ quan hệ thực thể

3.3. Xây dựng giao diện sản phẩm

3.3.1 Giao diện trang chủ

Đến với giao diện trang chủ, sẽ chứa các mục: Logo dẫn đến trang Home, Menu tìm kiếm truyện, icon favorites nhằm đánh dấu các sản phẩm truyện yêu thích, icon users để dẫn đến trang đăng nhập hoặc đăng ký, và icon giỏ hàng chứa các sản

phẩm truyện cần thêm vào giỏ. Ngoài ra còn có Menu phân thể loại truyện gồm có thể loại “Action” , “Adventure” , “Isekai” , “Fantasy” , “Mecha”,...



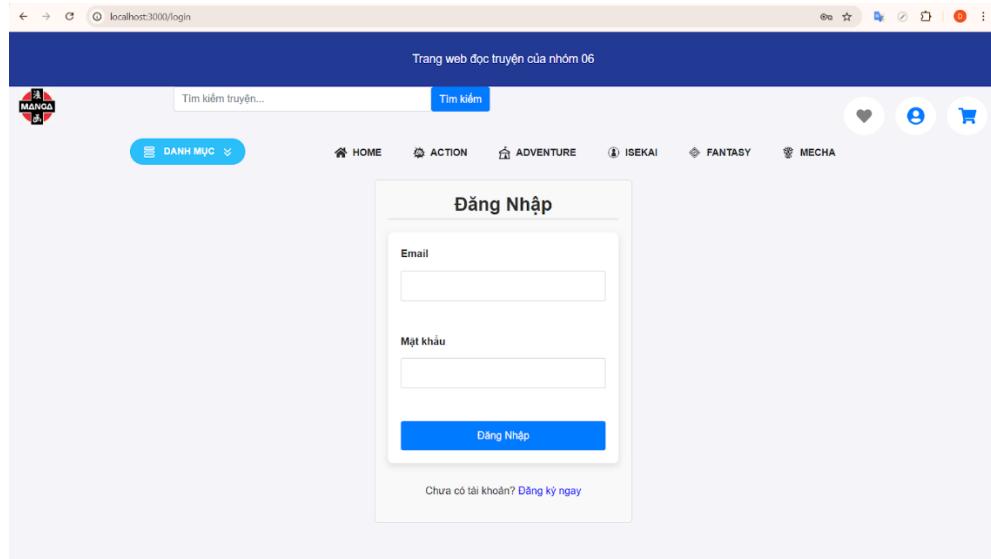
Hình 3.12. Giao diện trang chủ

Đây là code xử lí hiển thị các truyện ra trang chủ:

```
// Lấy danh sách truyện từ API
useEffect(() => {
  axios
    .get("http://localhost:8080/api/v1/comics/getAll")
    .then((response) => {
      setComics(response.data);
      setLoading(false);
    })
    .catch((error) => {
      console.error("Error fetching comics:", error);
      setLoading(false);
    });
}, []);
```

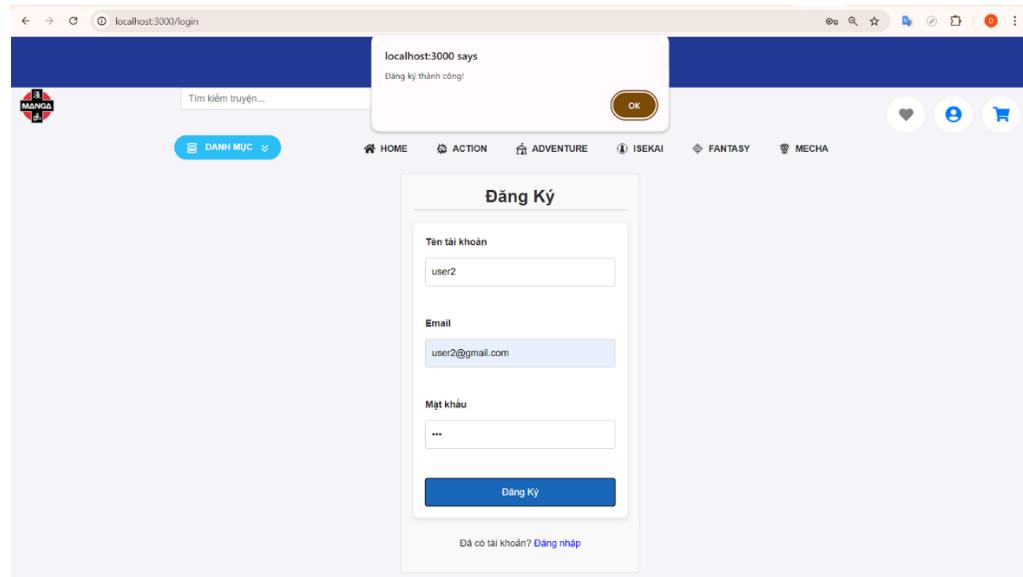
3.3.2 Giao diện trang chi tiết truyện

Khi người dùng muốn thực hiện đăng nhập vào tài khoản thì nhấp vào icon user.



Hình 3.13. Giao diện đăng nhập

Tại đây, người dùng cần điền vào các field yêu cầu như Email và Mật khẩu để thực hiện đăng nhập, hoặc nếu người dùng chưa có tài khoản thì thực hiện Đăng ký ngay:



Hình 3.14. Giao diện đăng ký

Khi thực hiện đăng ký, người dùng phải điền vào các trường bắt buộc như Tên tài khoản, Email và Mật khẩu. Sau khi thực hiện đăng ký thành công, sẽ hiển thị menu thông báo và điều hướng người dùng sang trang chủ.

Code xử lý trang Đăng nhập / Đăng ký:

```
const handleSubmit = async (e) => {
  e.preventDefault();
  const endpoint = isLoggedIn ? "/api/users/login" : "/api/users/signup";

  try {
    const response = await fetch(`http://localhost:8080${endpoint}`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(formData),
    });
    const data = await response.json();

    if (response.ok) {
      alert(isLoggedIn ? "Đăng nhập thành công!" : "Đăng ký thành công!");

      // Lưu token, user_id và username vào localStorage
      localStorage.setItem("token", data.token);
      localStorage.setItem("user_id", data.user_id); // Lưu user_id
      if (data.username) {
        | localStorage.setItem("username", data.username);
      }

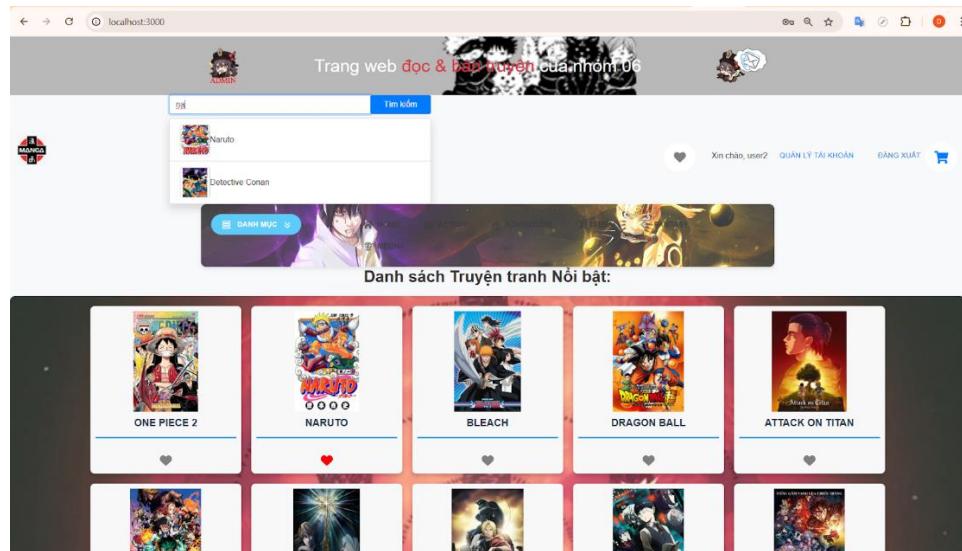
      console.log("Lưu user_id vào localStorage:", data.user_id);

      // Điều hướng về trang chủ và reload
      navigate("/");
      window.location.reload();
    } else {
      alert(data.message);
    }
  } catch (error) {
    console.error("Lỗi:", error);
    alert("Có lỗi xảy ra. Vui lòng thử lại.");
  }
};

<div className="login-signup">
  <h2>{isLoggedIn ? "Đăng Nhập" : "Đăng Ký"}</h2>
  <form onSubmit={handleSubmit}>
    {isLoggedIn && (
      <div>
        <label>Tên tài khoản</label>
        <input
          type="text"
          name="username"
          value={formData.username}
          onChange={handleInputChange}
          required={!isLoggedIn}
        />
      </div>
    )}
    <div>
      <label>Email</label>
      <input
        type="email"
        name="email"
        value={formData.email}
        onChange={handleInputChange}
        required
      />
    </div>
    <div>
      <label>Mật khẩu</label>
      <input
        type="password"
        name="password"
        value={formData.password}
        onChange={handleInputChange}
        required
      />
    </div>
    <button type="submit">{isLoggedIn ? "Đăng Nhập" : "Đăng Ký"}</button>
  </form>
  <p>{isLoggedIn ? "Chưa có tài khoản?" : "Đã có tài khoản?"}{" "}</p>
  <span
    style={({ cursor: "pointer", color: "blue" })
    onClick={() => setIsLogin(!isLoggedIn)}>
    {isLoggedIn ? "Đăng ký ngay" : "Đăng nhập"}
  </span>
</div>
```

3.3.3 Giao diện tìm kiếm truyện:

Đến với menu tìm kiếm truyện, người dùng có thể tìm kiếm truyện với bộ lọc có giá trị gần nhất, và chọn mẫu truyện cần tìm.



Hình 3.15. Giao diện tìm kiếm truyện

Sau khi click vào 1 mẩu truyện trên menu tìm kiếm, sẽ tự động điều hướng đến trang chi tiết truyện đó

Code xử lý chức năng tìm kiếm truyện:

```

useEffect(() => {
  const timer = setTimeout(() => {
    if (searchTerm.trim() !== '') {
      handleSearch();
    } else {
      setResults([]);
    }
  }, 500); // Debounce time (500ms)

  return () => clearTimeout(timer); // Cleanup on component unmount or searchTerm change
}, [searchTerm]);

const handleSearch = async () => {
  setLoading(true);
  try {
    const data = await searchComics(searchTerm);
    setResults(data);
    setLoading(false);

    if (data.length === 1) {
      navigate(`/comics/${data[0]._id}`);
    }
  } catch (error) {
    setLoading(false);
    console.error('Lỗi khi tìm kiếm:', error);
  }
};

const handleNavigateToDetail = (comicId) => {
  navigate(`/comics/${comicId}`);
};

<div className="search-box position-relative">
  <div className="input-group">
    <input type="text" className="form-control" placeholder="Tìm kiếm truyện..." value={searchTerm} onChange={(e) => setSearchTerm(e.target.value)} style={{ width: '300px' }} />
    <button
      className="btn btn-primary"
      onClick={handleSearch}
      disabled={loading} // Disable button while loading
    > Tìm kiếm </button>
  </div>
</div>

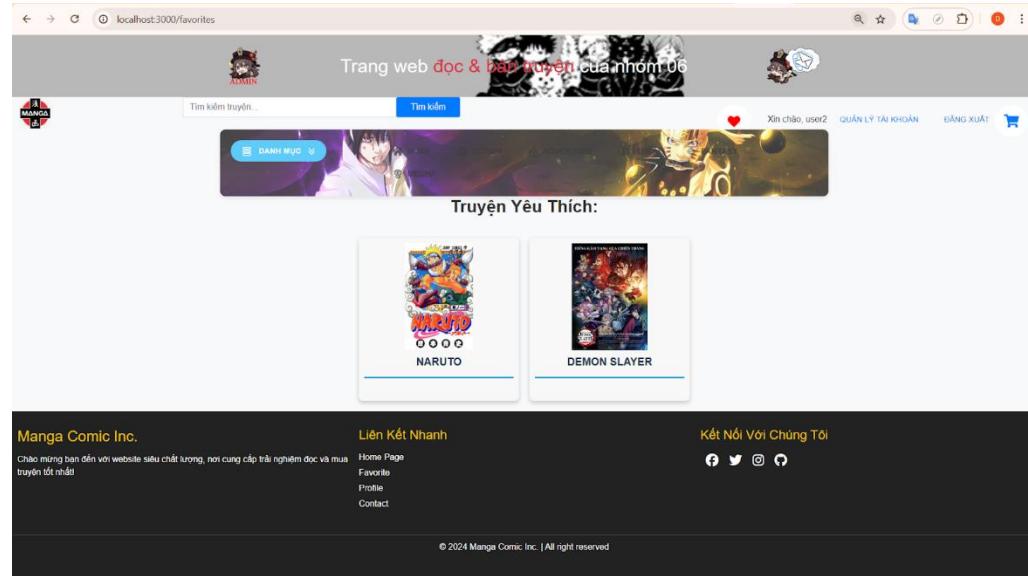
{loading && <p className="text-muted">Đang tìm kiếm...</p>}

{results.length > 0 && (
  <div className="search-results bg-light shadow mt-2 rounded">
    <ul className="list-group">
      {results.map((comic) => (
        <li key={comic._id}>
          <a href="#" onClick={() => handleNavigateToDetail(comic._id)} style={{ cursor: 'pointer' }}>
            <img src={comic.cover_image || 'http://localhost:8080/uploads/default.jpg'} alt={comic.title} className="me-1 rounded" style={{ width: '60px', height: '60px', objectFit: 'cover', border: '1px solid #ddd', }} />
            <span className="text-truncate" style={{ maxWidth: '300px' }}>{comic.title}</span>
          </a>
        </li>
      )));
    </ul>
  </div>
)
}

```

3.3.4 Giao diện yêu thích truyện:

Tại trang chủ, nếu người dùng đã thực hiện click vào icon yêu thích mẫu truyện nào, thì sang trang yêu thích sẽ hiển thị những mẫu truyện đó:



Hình 3.16. Giao diện yêu thích

Code xử lý của trang Yêu thích:

```
const FavoritesPage = () => {
  const [comics, setComics] = useState([]);
  const [favorites, setFavorites] = useState([]);
  const navigate = useNavigate();

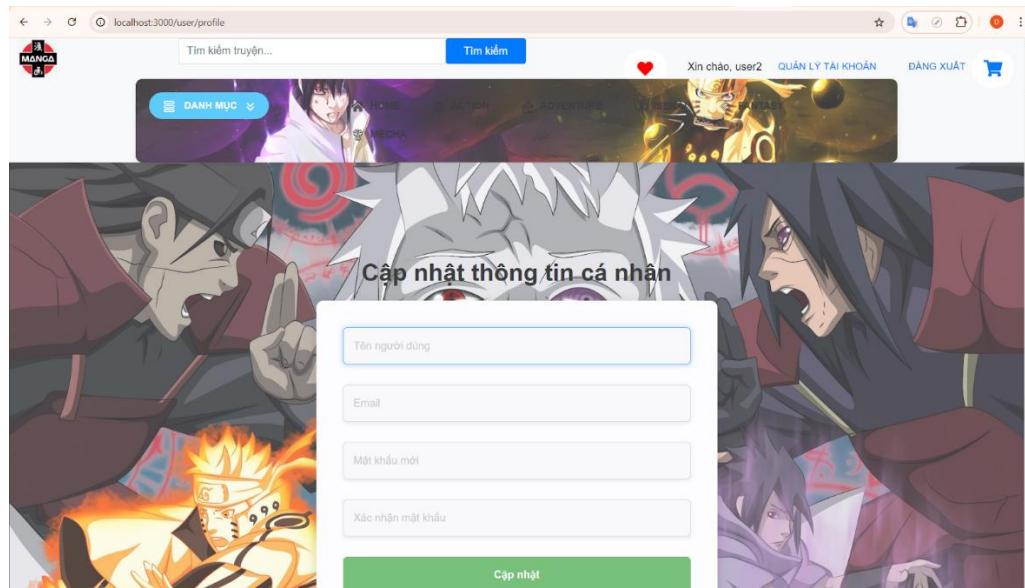
  useEffect(() => {
    const storedFavorites = JSON.parse(localStorage.getItem('favorites')) || [];
    setFavorites(storedFavorites);
  }, []);

  useEffect(() => {
    axios
      .get('http://localhost:8080/api/v1/comics/getAll')
      .then(response => {
        console.log('Comics data:', response.data);
        const favoriteComics = response.data.filter(comic =>
          favorites.includes(comic._id)
        );
        setComics(favoriteComics);
      })
      .catch(error => {
        console.error('Error fetching comics:', error);
      });
  }, [favorites]);

  return (
    <div>
      <h1>Truyện Yêu Thích:</h1>
      <div className="comic-list-container">
        {comics.length === 0 ? (
          <p>No favorite comics available</p>
        ) : (
          <div className="comic-list">
            {comics.map(comic => (
              <div key={comic._id} className="comic-item"
                onClick={() => navigate(`./comics/${comic._id}`)}
              >
                <img
                  src={`http://localhost:8080${comic.cover_image}`}
                  alt={comic.title}
                  className="comic-image"
                  onError={(e) => {
                    console.error('Error loading image:', e.target.src);
                    e.target.src = '/placeholder.jpg';
                  }}
                />
                <h2 className="comic-title">{comic.title}</h2>
              </div>
            )));
          </div>
        )}
      </div>
    </div>
  );
}
```

3.3.5 Giao diện thông tin người dùng:

Người dùng có thể thực hiện chỉnh sửa các thay đổi thông tin cá nhân, miễn là không bị trùng trường username và email của người khác:



Hình 3.17. Giao diện thông tin người dùng

Code xử lý trang thông tin người dùng:

```
const handleUpdate = async (e) => {
  e.preventDefault();
  if (password !== confirmPassword) {
    setMessage("Mật khẩu xác nhận không khớp");
    return;
  }
  if (!email || !username || !password) {
    setMessage("Vui lòng điền đầy đủ thông tin");
    return;
  }
  if (!validateEmail(email)) {
    setMessage("Email không hợp lệ");
    return;
  }
  setIsLoading(true);

  try {
    const token = localStorage.getItem("token");
    const config = { headers: { Authorization: `Bearer ${token}` } };
    const response = await axios.put("http://localhost:8080/api/users/update-profile",
      { username, email, password }, config);
    setMessage(response.data.message);
  } catch (error) {
    setMessage(error.response?.data?.message || "Cập nhật thất bại");
  } finally {
    setIsLoading(false);
  }
};

const validateEmail = (email) => {
  const re = /^[a-zA-Z0-9._%+]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
  return re.test(email);
};
```

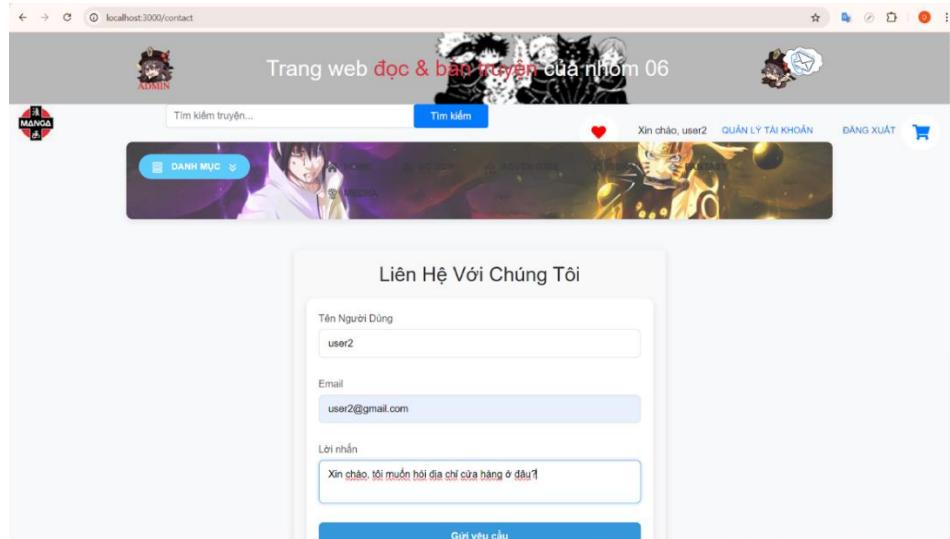
```

<div className="profile-container">
  <h2>Cập nhật thông tin cá nhân</h2>
  <form onSubmit={handleUpdate} className="profile-form">
    <input
      type="text"
      placeholder="Tên người dùng"
      value={username}
      onChange={(e) => setUsername(e.target.value)}
    />
    <input
      type="email"
      placeholder="Email"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
    />
    <input
      type="password"
      placeholder="Mật khẩu mới"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
    />
    <input
      type="password"
      placeholder="Xác nhận mật khẩu"
      value={confirmPassword}
      onChange={(e) => setConfirmPassword(e.target.value)}
    />
    <button type="submit" disabled={isLoading}>
      {isLoading ? "Đang cập nhật..." : "Cập nhật"}
    </button>
  </form>
  {message && <p className="message">{message}</p>}

```

3.3.6 Giao diện liên hệ:

Nếu có bất kỳ thắc mắc hoặc đề xuất gì cho trang web, người dùng có thể để lại lời nhắn trong trang liên hệ:



Hình 3.18. Giao diện liên hệ

Code xử lý trang Liên hệ:

```
// Hàm xử lý thay đổi giá trị input
const handleInputChange = (e) => [
  const { name, value } = e.target;
  setFormData((prevData) => ({
    ...prevData,
    [name]: value,
  }));
];

// Hàm kiểm tra lỗi trước khi gửi form
const validateForm = () => {
  let errors = {};
  if (!formData.username) errors.username = "Tên người dùng là bắt buộc.";
  if (!formData.email) errors.email = "Email là bắt buộc.";
  if (!/^\S+@\S+\.\S+$/.test(formData.email)) errors.email = "Email không hợp lệ.";
  if (!formData.message) errors.message = "Lời nhắn là bắt buộc.";
  return errors;
};

// Hàm xử lý khi gửi form
const handleSubmission = async (e) => {
  e.preventDefault();

  // Kiểm tra lỗi trước khi gửi form
  const errors = validateForm();
  if (Object.keys(errors).length > 0) {
    setFormErrors(errors);
    return; // Nếu có lỗi thì không gửi form
  }

  setIsLoading(true);
  setStatusMessage("");

  try {
    const response = await axios.post("http://localhost:8080/api/contact", formData);
    setStatusMessage(response.data.message);
    setFormData({ username: "", email: "", message: "" });
    setFormErrors({});
  } catch (error) {
    console.error("Lỗi khi gửi yêu cầu:", error);
    setStatusMessage("Đã có lỗi xảy ra. Vui lòng thử lại.");
  } finally {
    setIsLoading(false);
  }
};

<div className="contact-container">
  <h2>Liên Hệ Với Chúng Tôi!</h2>
  <form onSubmit={handleSubmission} className="contact-form">
    <div className="form-group">
      <label htmlFor="username">Tên Người Dùng</label>
      <input type="text" id="username" name="username"
        value={formData.username}
        onChange={handleInputChange}
        required
        placeholder="Nhập tên người dùng"
      />
      {formErrors.username && <p className="error">{formErrors.username}</p>}
    </div>

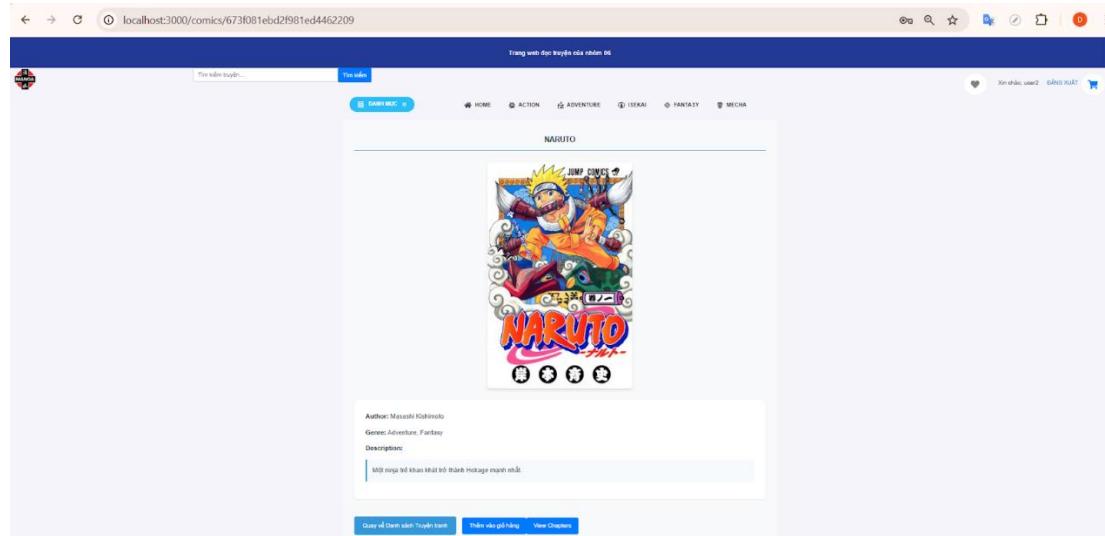
    <div className="form-group">
      <label htmlFor="email">Email</label>
      <input type="email" id="email" name="email"
        value={formData.email}
        onChange={handleInputChange}
        required
        placeholder="Nhập email của bạn"
      />
      {formErrors.email && <p className="error">{formErrors.email}</p>}
    </div>

    <div className="form-group">
      <label htmlFor="message">Lời Nhắn</label>
      <textarea id="message" name="message"
        value={formData.message}
        onChange={handleInputChange}
        required
        placeholder="Viết lời nhắn của bạn..."
      />
      {formErrors.message && <p className="error">{formErrors.message}</p>}
    </div>

    <button type="submit" disabled={isLoading}>
      {isLoading ? "Đang gửi..." : "Gửi yêu cầu"}
    </button>
  </form>
</div>
```

3.3.7 Giao diện trang chi tiết truyện

Khi nhập vào mẫu truyện bất kỳ trên trang web, sẽ điều hướng sang trang chi tiết truyện:



Hình 3.19. Giao diện chi tiết truyện

Trong trang chi tiết truyện sẽ có các trường thông tin dữ liệu như tên *tác giả*, *thể loại truyện*, và *mô tả của truyện*. Sau cùng là *button thêm vào giỏ hàng* nếu người dùng ưa thích sản phẩm, *button View Chapters* để thực hiện xem các chap truyện, hoặc quay về trang chủ để xem những mẫu truyện khác.

Code xử lý hiển thị giao diện truyện chi tiết truyện:

```
const ComicDetailPage = () => {
  const { id } = useParams();
  const [comic, setComic] = useState(null);

  useEffect(() => {
    axios
      .get(`http://localhost:8080/api/v1/comics/${id}`)
      .then((response) => {
        setComic(response.data);
      })
      .catch((error) => {
        console.error("Error fetching comic details:", error);
      });
  }, [id]);
```

```

<div className="comic-detail-container">
  {/* Tiêu đề */}
  <h1 className="comic-title">{comic.title}</h1>

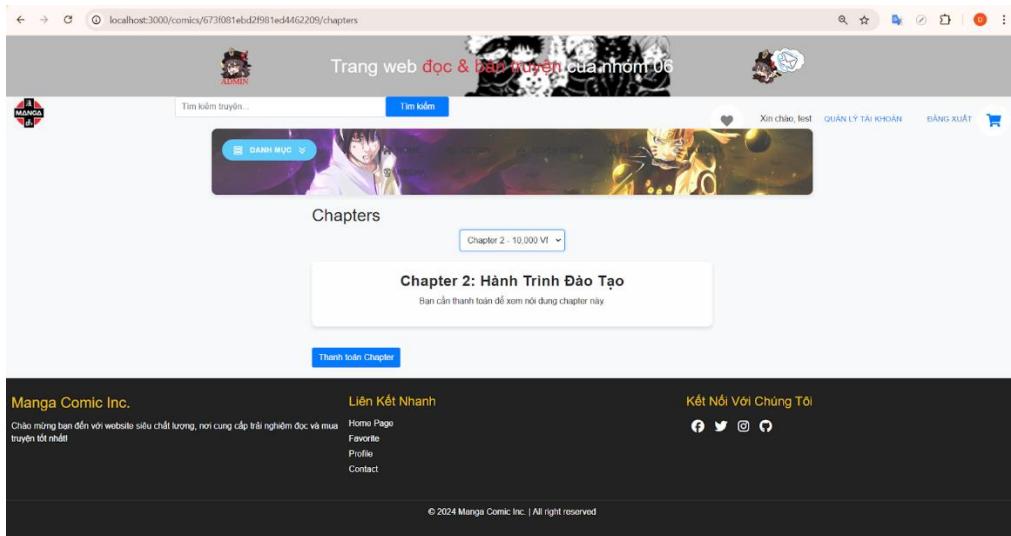
  {/* Hình ảnh bìa */}
  <img src={imageUrl} alt={comic.title} className="comic-cover" />

  {/* Chi tiết truyện */}
  <div className="comic-details">
    <p>
      | <strong>Author:</strong> {comic.author}
    </p>
    <p>
      | <strong>Genre:</strong> {comic.genre}
    </p>
    <p>
      | <strong>Description:</strong>
    </p>
    <div className="comic-description">{comic.description}</div>
  </div>
</div>

```

3.3.8 Giao diện xem chương truyện

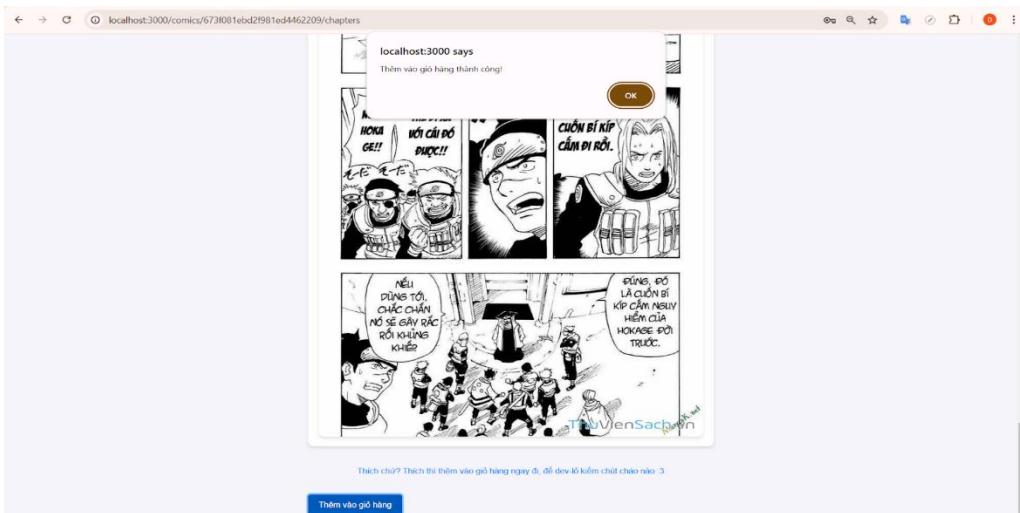
Sau khi nhập vào View chapter ở một mẫu truyện bất kỳ trong trang chi tiết truyện, sẽ điều hướng sang trang chapter của truyện đó:



Hình 3.20. Giao diện chương truyện

Đầu tiên, người dùng cần thực hiện thanh toán chapter để được xem chapter đó - trị giá 10.000 vnđ cho mỗi một chapter:

Sau khi thực hiện thanh toán chapter xong người dùng có thể đọc chapter đó, hiện tại là đang xem chapter 1, hoặc cũng có thể xem chapter khác nếu đã thanh toán. Đến cuối chapter, nếu người dùng cảm thấy thích mẫu truyện này, thì có thể thêm truyện vào giỏ hàng và thực hiện thanh toán để ủng hộ.



Hình 3.21. Giao diện chương truyện

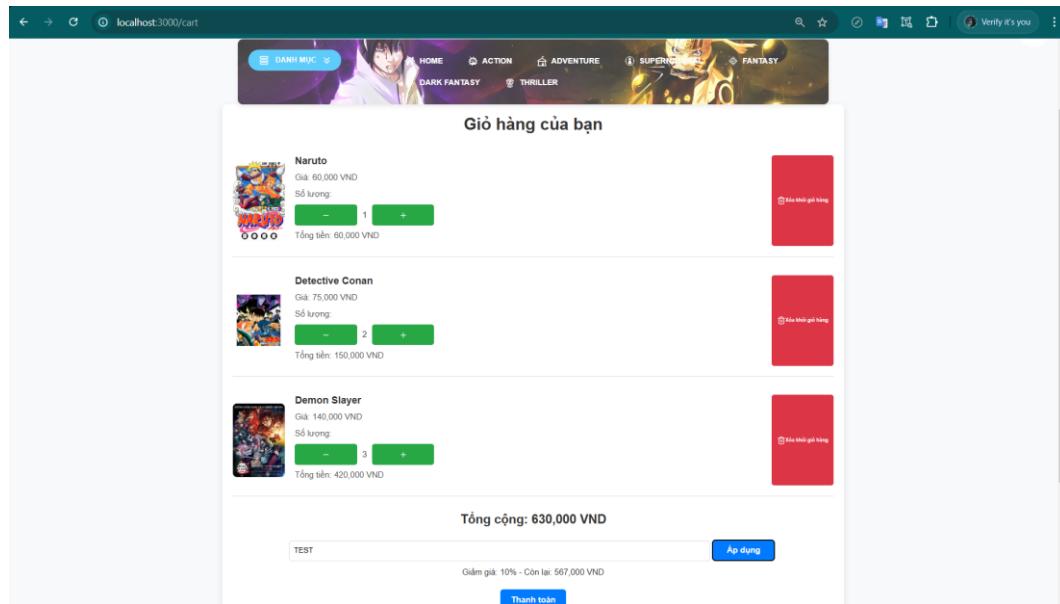
Code xử lí lấy nội dung chương truyện và hiển thị hình ảnh:

```
useEffect(() => [
  const fetchChapters = async () => {
    const token = localStorage.getItem("token");
    if (!token) {
      console.error("User token is missing");
      navigate("/login"); // Redirect to user login if token is missing
      return;
    }

    try {
      const response = await axios.get(`/api/chapters/byComic/${id}`, {
        headers: { Authorization: `Bearer ${token}` },
      });
      setChapters(response.data);
      setSelectedChapter(response.data[0]); // Chọn chapter đầu tiên làm mặc định
    } catch (error) {
      console.error("Error fetching chapters:", error.response?.data || error.message);
      if (error.response?.status === 401 || error.response?.status === 403) {
        navigate("/login"); // Redirect to user login if unauthorized
      }
    } finally {
      setLoading(false);
    }
  };
}

/* Hiển thị nội dung chapter được chọn */
{selectedChapter && (
  <div className="chapter-item">
    <h3 className="chapter-title">
      Chapter {selectedChapter.chapter_number}: {selectedChapter.title}
    </h3>
    {isPaid ? (
      <>
        <p className="chapter-content">{selectedChapter.content}</p>
        <div className="chapter-images-container">
          {selectedChapter.images.map((url, index) => (
            <img
              key={index}
              src={url}
              alt={`Chapter ${selectedChapter.chapter_number} Image ${index + 1}`}
            />
          )));
        </div>
      ) : (
        <p>Bạn cần thanh toán để xem nội dung chapter này.</p>
      )
    </div>
  )
)}
```

3.3.9 Giao diện giỏ hàng



Hình 3.22. Giao diện giỏ hàng

Các sản phẩm mà người dùng muốn thêm vào giỏ hàng sẽ tập trung tại đây. Trong trang này, người dùng có thể tùy chỉnh số lượng sản phẩm truyện trong giỏ, hoặc xóa mẫu truyện khỏi giỏ hàng, nhập mã giảm giá để nhận giảm giá ưu đãi theo %.

Code xử lý dữ liệu và hiển thị sản phẩm ra giỏ hàng:

```
useEffect(() => {
  const token = localStorage.getItem("token");
  if (!token) {
    alert("Bạn cần đăng nhập để xem giỏ hàng!");
    navigate("/login");
    return;
  }

  const fetchCart = async () => {
    try {
      const response = await axios.get("http://localhost:8080/api/cart", {
        headers: { Authorization: `Bearer ${token}` },
      });
      setCart(response.data || { items: [] });
    } catch (error) {
      console.error("Error fetching cart:", error);
      alert("Không thể tải dữ liệu giỏ hàng. Vui lòng thử lại sau!");
    } finally {
      setLoading(false);
    }
  };

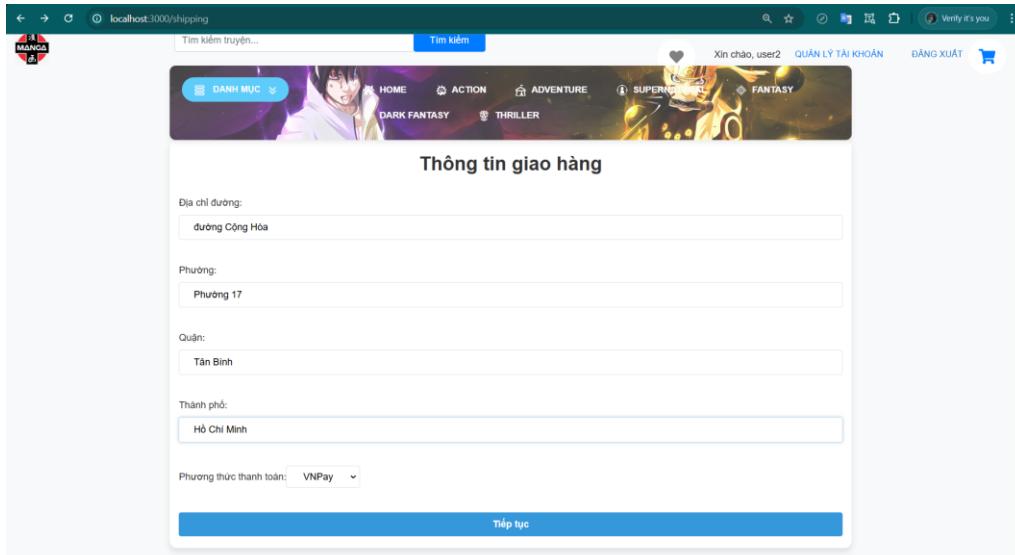
  fetchCart();
}, [navigate]);
```

```

<h1>Giỏ hàng của bạn</h1>
<ul>
  {cart.items.map((item) => (
    <li key={item.productId._id} className="cart-item">
      <div className="cart-item-details">
        <img
          src={`http://localhost:8080${item.productId.cover_image}`}
          alt={item.productId.title}
          className="cart-item-image"
        />
        <div className="cart-item-info">
          <h2>[item.productId.title]</h2>
          <p>Giá: {formatPrice(item.productId.price)} VND</p>
          <p>Số lượng:</p>
          <div className="quantity-controls">
            <button
              onClick={() =>
                handleChangeQuantity(
                  item.productId._id,
                  item.quantity - 1
                )
              }
              className="quantity-btn"
            >
              <Minus size={16}> />
            </button>
            <span>{item.quantity}</span>
            <button
              onClick={() =>
                handleChangeQuantity(
                  item.productId._id,
                  item.quantity + 1
                )
              }
              className="quantity-btn"
            >
              <Plus size={16}> />
            </button>
          </div>
          <p>
            Tổng tiền: {formatPrice(item.productId.price * item.quantity)}{" "}
          </p>
        </div>
      </div>
    </li>
  ))}

```

3.3.10 Giao diện thông tin giao hàng



Hình 3.23. Giao diện thông tin giao hàng

Khi thực hiện button Thanh toán, sẽ điều hướng user đến form để cập nhật thông tin giao hàng, gồm có địa chỉ đường, phường, quận, thành phố; và phương thức thanh toán hỗ trợ là VNPay.

Code xử lý dữ liệu và hiển thị thông tin giao hàng:

```
Tabnine | Edit | Explain
const Shipping = () => {
  const [address, setAddress] = useState({
    street: '',
    ward: '',
    district: '',
    city: ''
  });
  const [paymentMethod, setPaymentMethod] = useState("VNPay");
  const [errors, setErrors] = useState({});
  const { state } = useLocation();
  const navigate = useNavigate();

  const validateForm = () => {
    const newErrors = {};

    if (!address.street.trim()) {
      newErrors.street = "Địa chỉ đường không được để trống.";
    }
    if (!address.ward.trim()) {
      newErrors.ward = "Vui lòng nhập phường.";
    }
    if (!address.district.trim()) {
      newErrors.district = "Quận không được bỏ trống.";
    }
    if (!address.city.trim()) {
      newErrors.city = "Vui lòng nhập tên thành phố.";
    }

    setErrors(newErrors);
  }

  // Trả về true nếu không có lỗi
  return Object.keys(newErrors).length === 0;
};

const handleSubmit = (e) => {
  e.preventDefault();

  if (validateForm()) {
    localStorage.setItem("shippingAddress", JSON.stringify(address));
    localStorage.setItem("paymentMethod", paymentMethod);

    navigate("/placeorder", {
      state: {
        discountedTotal: state?.discountedTotal, // Truyền tiếp giá đã giảm
        voucher: state?.voucher, // Truyền tiếp voucher
      },
    });
  }
};

<form onSubmit={handleSubmit} className="container mx-auto mt-8 p-6">
  <h1 className="text-2xl font-semibold mb-6">Thông tin giao hàng</h1>
  <div className="mb-4">
    <label className="block mb-2 font-medium">Địa chỉ đường:</label>
    <input
      type="text"
      placeholder="Địa chỉ đường"
      value={address.street}
      onChange={(e) => setAddress({ ...address, street: e.target.value })}
      className="w-full px-4 py-2 border rounded ${errors.street ? "border-red-500" : "border-gray-300"}"
    />
    {errors.street && (
      <p className="text-red-500 text-sm mt-2">{errors.street}</p>
    )}
  </div>
  <div className="mb-4">...
  </div>
  <div className="mb-4">...
  </div>
  <div className="mb-4">...
  </div>
  <div className="mb-4">
    <label className="block mb-2 font-medium">Phương thức thanh toán:</label>
    <select
      value={paymentMethod}
      onChange={(e) => setPaymentMethod(e.target.value)}
      className="w-full px-4 py-2 border rounded border-gray-300"
    >
      <option value="VNPay">VNPay</option>
      {/* <option value="PayPal">PayPal</option> */}
    </select>
  </div>

  <button
    type="submit"
    className="bg-blue-500 text-white px-6 py-2 rounded font-medium hover:bg-blue-600 transition"
  >
    Tiếp tục
  </button>
</form>
```

3.3.11 Giao diện xác nhận đơn hàng

The screenshot shows a web browser window with the URL `localhost:3000/placeorder`. The title bar says "Xác nhận đơn hàng".

Thông tin giao hàng

Địa chỉ: đường Cộng Hòa, Phường 17, Tân Bình, Hồ Chí Minh

Chi tiết sản phẩm

HÌNH ẢNH	SẢN PHẨM	SỐ LƯỢNG	GIÁ	TỔNG CỘNG
	Naruto	1	60,000 VND	60,000 VND
	Detective Conan	2	75,000 VND	150,000 VND
	Demon Slayer	3	140,000 VND	420,000 VND

Tóm tắt đơn hàng

Mã giảm giá:	TEST
Phương thức thanh toán:	VNPay
Tổng sản phẩm (đã bao gồm mã giảm giá):	567,000 VND

Đặt hàng

Hình 3.24. Giao diện xác nhận giao hàng

Sau khi hoàn tất form thông tin giao hàng, sẽ tổng kết lại thông tin người dùng mua hàng về thông tin sản phẩm và thông tin đặt đã cung cấp. Tóm tắt đơn hàng có phương thức thanh toán đã chọn, áp mã giảm giá vào và tính tổng giá trị đơn hàng, cuối cùng là thực hiện Đặt hàng

Code xử lý dữ liệu và hiển thị xác nhận giao hàng:

```

const handlePlaceOrder = async () => {
  const token = localStorage.getItem("token");

  if (!token) {
    alert("Bạn chưa đăng nhập. Vui lòng đăng nhập lại.");
    return;
  }

  const order = {
    orderItems: cartItems.map((item) => ({
      product: item.productId._id,
      qty: item.quantity,
      price: item.productId.price,
    })),
    shippingAddress,
    paymentMethod,
    totalPrice: discountedTotal || 0,
    voucherCode: voucher || "",
  };

  try {
    const response = await fetch("http://localhost:8080/api/orders", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`,
      },
      body: JSON.stringify(order),
    });

    if (!response.ok) {
      const responseData = await response.json();
      throw new Error(responseData.message || "Failed to create order");
    }

    const data = await response.json();
    navigate(`order/${data._id}`);
  } catch (error) {
    console.error("Error creating order:", error);
    alert("Đã xảy ra lỗi khi tạo đơn hàng. Vui lòng thử lại.");
  }
}

/* Xác nhận đơn hàng */


## Thông tin giao hàng



Địa chỉ: (shippingAddress.street), (shippingAddress.district), (shippingAddress.city)



## Chi tiết sản phẩm



| Hình ảnh | Sản phẩm | Số lượng | Mô tả | Tổng cộng |
|----------|----------|----------|-------|-----------|
|----------|----------|----------|-------|-----------|


  </div>


/* Tóm tắt đơn hàng */


## Tóm tắt đơn hàng



- Đã bao gồm mã giảm giá: {voucher || "không có"}
- Phương thức thanh toán: {paymentMethod}
- Tổng sản phẩm (đã bao gồm mã giảm giá): {discountedTotal?.toLocaleString() || "0"} VNĐ


  <button type="button" class="btn btn-primary w-100 mt-4" onClick={handlePlaceOrder}>Đặt hàng</button>


```

3.3.12 Giao diện chi tiết đơn hàng

The screenshot shows a web browser window with the URL `localhost:3000/order/67601fd16f7bf16ea8fb218f`. The page title is "Chi tiết đơn hàng".

HÌNH ÁNH	SẢN PHẨM	SỐ LƯỢNG	ĐƠN GIÁ	TỔNG CỘNG	NGÀY TẠO	THANH TOÁN	GIAO HÀNG
	Naruto	1	60,000 VND	60,000 VND	12/16/2024	Chưa thanh toán	Chưa giao hàng
	Detective Conan	2	75,000 VND	150,000 VND	12/16/2024	Chưa thanh toán	Chưa giao hàng
	Demon Slayer	3	140,000 VND	420,000 VND	12/16/2024	Chưa thanh toán	Chưa giao hàng

Tóm tắt đơn hàng

Mã giảm giá: TEST
Mã đơn hàng: 67601fd16f7bf16ea8fb218f
Người đặt: user2
Email: user2@gmail.com
Địa chỉ giao hàng: đường Cộng Hòa, Phường 17, Tân Bình, Hồ Chí Minh
Phương thức thanh toán: VNPay
Tổng cộng (đã bao gồm mã giảm giá): 567,000 VND

Thanh toán ngay

Thanh toán để, rồi để lại Đánh giá tích cực cho sản phẩm của sôp Manga Comic nhé!

Hình 3.25. Giao diện chi tiết đơn hàng

Code xử lý dữ liệu và hiển thị chi tiết đơn hàng:

```

/* Xác nhận đơn hàng */
chi className="text-3xl font-bold mb-4">Xác nhận đơn hàng</h1>


/* Thông tin giao hàng */


Thông tin giao hàng
TEST



Mã đơn hàng:
67601fd16f7bf16ea8fb218f



Người đặt:
user2



Email:
user2@gmail.com



Địa chỉ giao hàng:
đường Cộng Hòa, Phường 17, Tân Bình, Hồ Chí Minh



Phương thức thanh toán:
VNPay



Tổng cộng (đã bao gồm mã giảm giá):
567,000 VND



Thanh toán ngay



Thanh toán để, rồi để lại Đánh giá tích cực cho sản phẩm của sôp Manga Comic nhé!


```

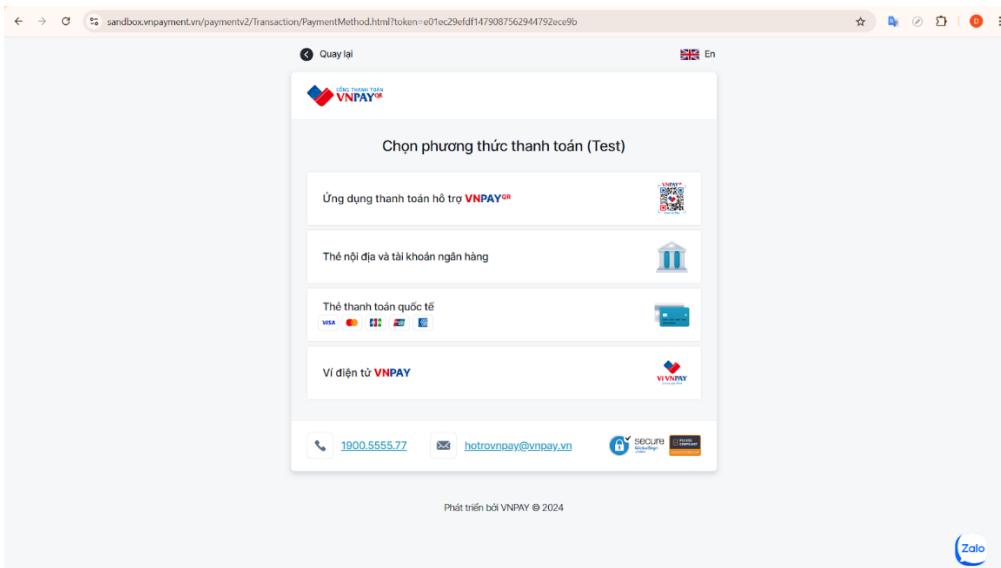
```


| Hình ảnh | Sản phẩm | Số lượng | Đơn giá | Tổng cộng | Ngày tạo | Thanh toán | Giao hàng |
|----------|----------|----------|---------|-----------|----------|------------|-----------|
|----------|----------|----------|---------|-----------|----------|------------|-----------|


```

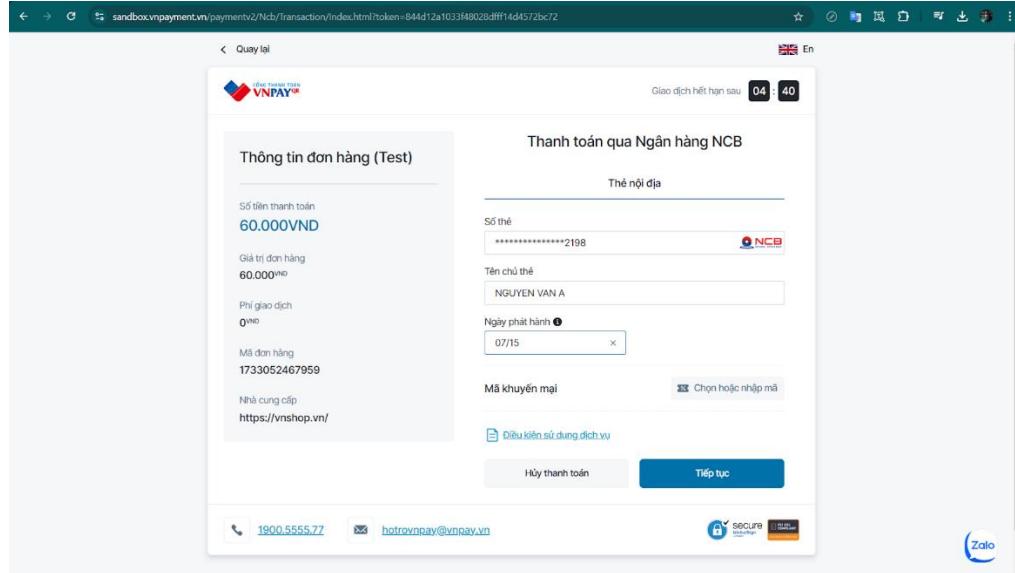
3.3.13 Giao diện thanh toán

Đến với giao diện thanh toán, ở đây chúng em sử dụng môi trường sandbox do VNPAY cung cấp để thực hiện thanh toán dành cho người dùng [7]:



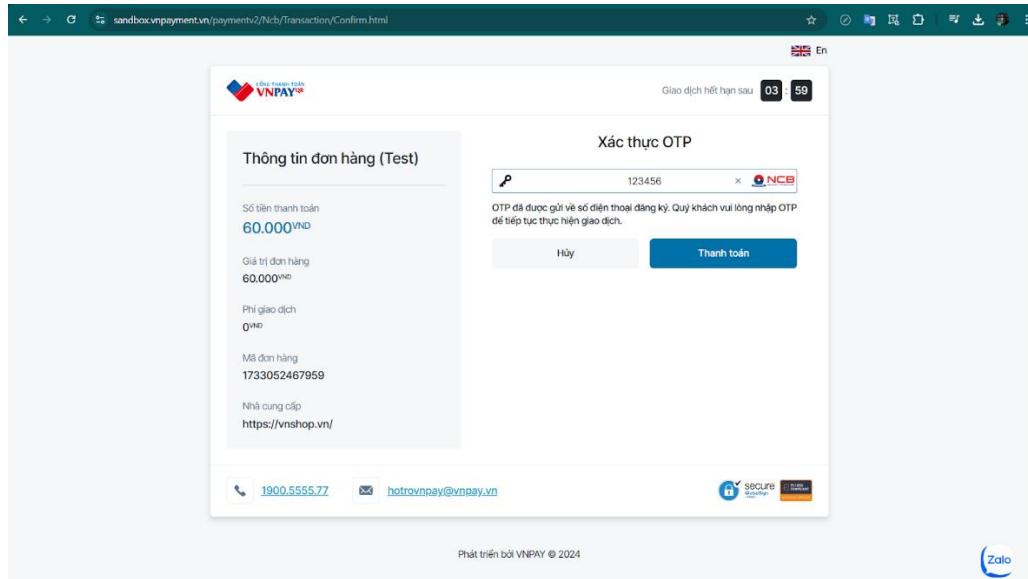
Hình 3.26. Phương thức thanh toán

Ở đây có các hình thức thanh toán như Quét mã QR qua ứng dụng Ngân hàng/Ví điện tử ; Thẻ nội địa và tài khoản ngân hàng ; Thẻ thanh toán quốc tế ; Ví điện tử VNPay... Chúng em chọn hình thức thanh toán là Thẻ nội địa và tài khoản ngân hàng và sử dụng ngân hàng NCB để thực hiện thanh toán:



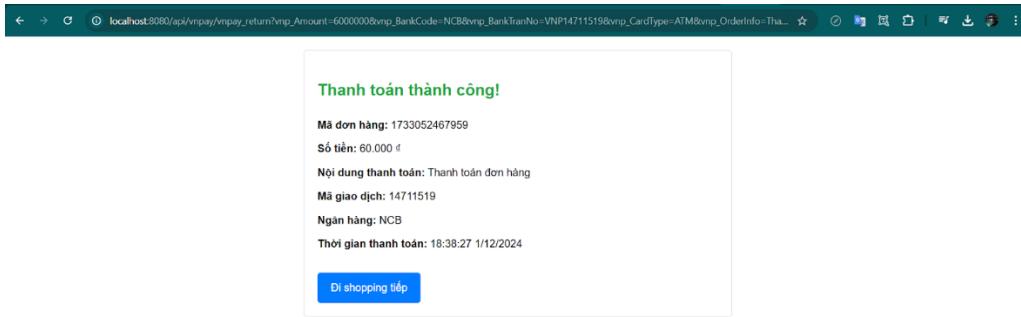
Hình 3.27. Thông tin thanh toán

Khi chọn được phương thức thanh toán, sẽ điều hướng đến nhập các thông tin của thẻ, và có 5 phút đếm ngược trong phiên thanh toán lần này:



Hình 3.28. Xác thực thanh toán

Người dùng cần nhập mã xác thực OTP để thực hiện hoàn tất thanh toán:

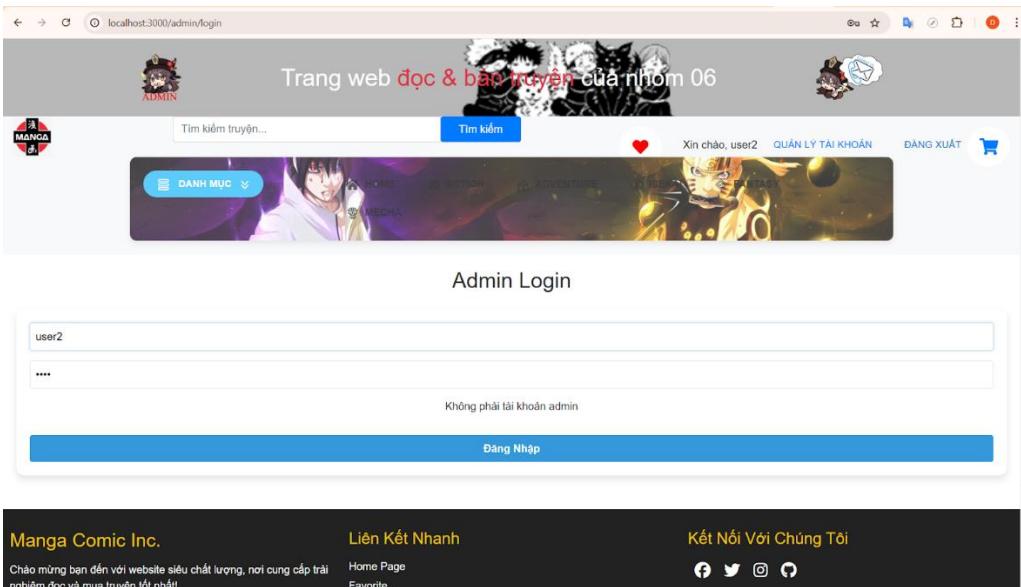


Hình 3.29. Thanh toán thành công

Sau khi thực hiện các bước trên, thanh toán sẽ được thực hiện thành công và xuất hiện mã đơn hàng, mã giao dịch được cung cấp để tra cứu chi tiết đơn hàng. Người dùng có thể tiếp tục mua các mẫu truyện ưa thích bằng cách nhấp vào button *Di shopping tiếp*.

3.3.14 Giao diện Admin Login:

Đến với giao diện admin, đầu tiên sẽ điền vào các trường yêu cầu như username và mật khẩu, sau đó xác thực tài khoản đó có phải là tài khoản admin hay không:



Hình 3.30. Giao diện đăng nhập của Admin

Code xử lý giao diện Admin Login:

```
// Cấu hình axios base URL ở ngay đầu file
axios.defaults.baseURL = "http://localhost:8080";

Tabnine | Edit | Explain
const AdminLogin = () => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post("/api/admin/login", {
        username,
        password,
      });

      // Lưu token
      localStorage.setItem("adminToken", response.data.token);

      // Chuyển hướng đến trang admin
      navigate("/admin/dashboard");
    } catch (err) {
      setError(err.response?.data?.message || "Đăng nhập thất bại");
    }
  };

<div className="min-h-screen flex items-center justify-center bg-gray-100">
  <div className="bg-white p-8 rounded shadow-md w-96">
    <h2 className="text-2xl mb-4 text-center">Admin Login</h2>
    <form onSubmit={handleLogin} className="space-y-4">
      <input
        type="text"
        value={username}
        onChange={(e) => setUsername(e.target.value)}
        placeholder="Tên đăng nhập"
        required
        className="w-full p-2 border rounded"
      />
      <input
        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        placeholder="Mật khẩu"
        required
        className="w-full p-2 border rounded"
      />
      {error && <p className="text-red-500 text-center">{error}</p>}
      <button
        type="submit"
        className="w-full bg-blue-500 text-white p-2 rounded hover:bg-blue-600"
      >
        Đăng Nhập
      </button>
    </form>
  </div>
</div>
```

3.3.15 Giao diện Admin Dashboard:

Tại menu Admin Dashboard, sẽ quản lý các thông tin về user và contact user. Ngoài ra còn có button điều hướng đến các phần quản lý khác, như Chính sửa truyện tranh, Chính sửa Chapter truyện, Chính sửa Voucher

The screenshot shows the Admin Dashboard of a manga application. At the top, there's a navigation bar with links for HOME, ACTION, ADVENTURE, SUPERNATURAL, and FANTASY. Below the navigation is a banner featuring manga characters. The main content area is titled "Admin Dashboard" and contains a table of users:

USERNAME	EMAIL	ROLE	ACTIONS
vuthanhdat	vtdhvn@gmail.com	admin	Xóa Thay đổi Role
test	test@gmail.com	reader	Xóa Thay đổi Role
cntt	cntt@gmail.com	reader	Xóa Thay đổi Role
user2	user2@gmail.com	reader	Xóa Thay đổi Role

Below the user table is a section titled "Danh Sách Yêu Cầu Liên Hệ" (List of Contact Requests) with the following data:

TÊN	EMAIL	LỜI NHẮN
user2	user2@gmail.com	Xin chào, tôi muốn hỏi địa chỉ cửa hàng ở đâu?
test	test@gmail.com	xin chào, tôi muốn nói rằng web này ngầu vkl III

Hình 3.31. Giao diện bảng của trang Admin

Code xử lý giao diện Admin Dashboard:

```

useEffect(() => {
  const fetchUsers = async () => {
    try {
      const token = LocalStorage.getItem("adminToken");
      const response = await axios.get("/api/admin/users", {
        headers: { Authorization: `Bearer ${token}` },
      });
      setUsers(response.data);
    } catch (error) {
      navigate("/admin/login");
    }
  };

  fetchUsers();
}, [navigate]);

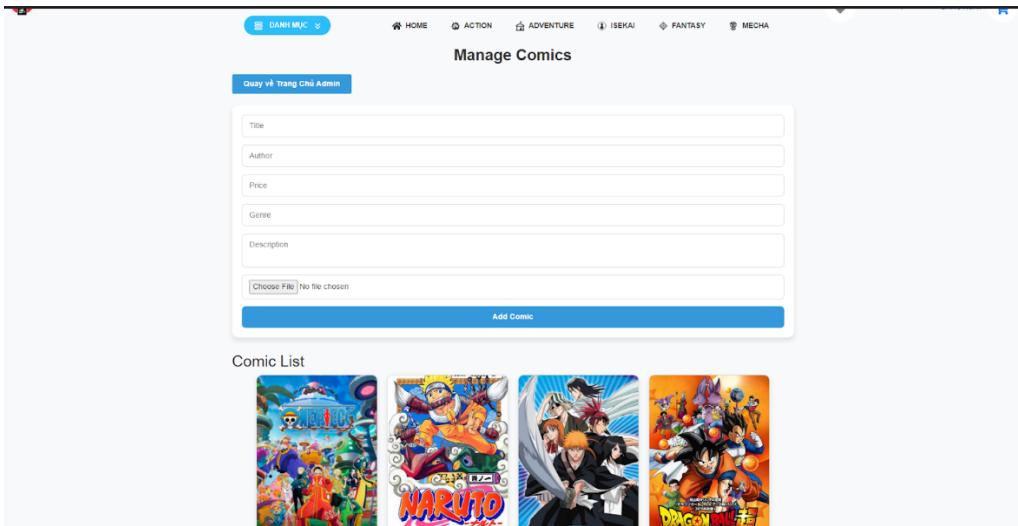
```

```

<div className="bg-white shadow-md rounded">
  <table>
    <thead>
      <tr>
        <th>Username</th>
        <th>Email</th>
        <th>Role</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {users.map((user) => [
        <tr key={user._id}>
          <td>{user.username}</td>
          <td>{user.email}</td>
          <td>{user.role}</td>
          <td className="p-3 flex justify-end gap-3">
            <button
              onClick={() => handleDeleteUser(user._id)}
              className="bg-red-500 text-white rounded"
            > Xóa </button>
            <button
              onClick={() => openRoleModal(user)}
              className="bg-blue-500 text-white rounded"
            > Thay đổi Role </button>
          </td>
        </tr>
      ])
    }
  </tbody>
</table>

```

3.3.16 Giao diện Admin Chính sửa truyện tranh:



Hình 3.32. Giao diện chỉnh sửa truyện của trang Admin

Code xử lí giao diện chỉnh sửa truyện:

```

  .then((res) => {
    if (editingComic) {
      setComics((prev) =>
        prev.map((comic) => (comic._id === editingComic ? res.data : comic))
      );
    } else {
      setComics((prev) => [...prev, res.data]);
    }

    setEditingComic(null);
    setForm({
      title: "",
      author: "",
      price: "",
      category_id: "",
      genre: "",
      description: "",
      cover_image: null,
    });
  })
  .catch((error) => {
    console.error("Error submitting comic:", error);
  });
}

```

```

<h1>Manage Comics</h1>
<button
  onClick={() => navigate("/admin/dashboard")}
  className="bg-blue-500 text-white px-4 py-2 rounded mb-4"
>
  Quay về Trang Chủ Admin
</button>
<form onSubmit={handleSubmit} encType="multipart/form-data">
  <input
    type="text"
    name="title"
    placeholder="Title"
    value={form.title}
    onChange={handleInputChange}
    required
  />
  <input
    type="text"
    name="author"
    placeholder="Author"
    value={form.author}
    onChange={handleInputChange}
    required
  />
  <input
    type="number"
    name="price"
    placeholder="Price"
    value={form.price}
    onChange={handleInputChange}
    required
  />
  <input
    type="number"
    name="price"
    placeholder="Price"
    value={form.price}
    onChange={handleInputChange}
    required
  />
  <input
    type="text"
    name="genre"
    placeholder="Genre"
    value={form.genre}
    onChange={handleInputChange}
  />
  <textarea
    name="description"
    placeholder="Description"
    value={form.description}
    onChange={handleInputChange}
  />
  <input type="file" name="cover_image" onChange={handleFileChange} />
  {form.cover_image && (
    <img
      src={URL.createObjectURL(form.cover_image)}
      alt="Preview"
      className="preview-image"
    />
  )}
  <button type="submit">{editingComic ? "Update" : "Add"} Comic</button>
</form>

```

3.3.17 Giao diện Admin chỉnh sửa chương truyện:

Add New Chapter

Chapter Number

Title

Content

Add Image URL

Enter image URL

Add Image

Price

Add Chapter

Reset Form

Hình 3.33. Giao diện chỉnh sửa chương của Admin

Code xử lý giao diện thêm chapter:

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const payload = {
      ...form,
      comic_id: selectedComic?._id,
    };

    if (editingChapterId) {
      await axios.put(`/api/admin/chapters/${editingChapterId}`, payload);
      alert("Chapter updated successfully");
    } else {
      await axios.post("/api/admin/chapters", payload);
      alert("Chapter added successfully");
    }

    resetForm();
    fetchComics();
  } catch (error) {
    console.error("Error submitting form:", error);
  }
};

/* Form thêm/sửa chapter */
{selectedComic && (
  <div className="form-section">
    <h2>{editingChapterId ? "Edit Chapter" : "Add New Chapter"}</h2>
    <form onSubmit={handleSubmit}>
      <div className="form-group">
        <label htmlFor="chapter_number">Chapter Number</label>
        <input
          type="number"
          id="chapter_number"
          value={form.chapter_number}
          onChange={(e) =>
            setForm({ ...form, chapter_number: e.target.value })
          }
          required
        />
      </div>
      <div className="form-group">
        <label htmlFor="title">Title</label>
        <input
          type="text"
          id="title"
          value={form.title}
          onChange={(e) => setForm({ ...form, title: e.target.value })}
          required
        />
      </div>
      <div className="form-group">
        <label htmlFor="content">Content</label>
        <textarea
          id="content"
          value={form.content}
          onChange={(e) =>

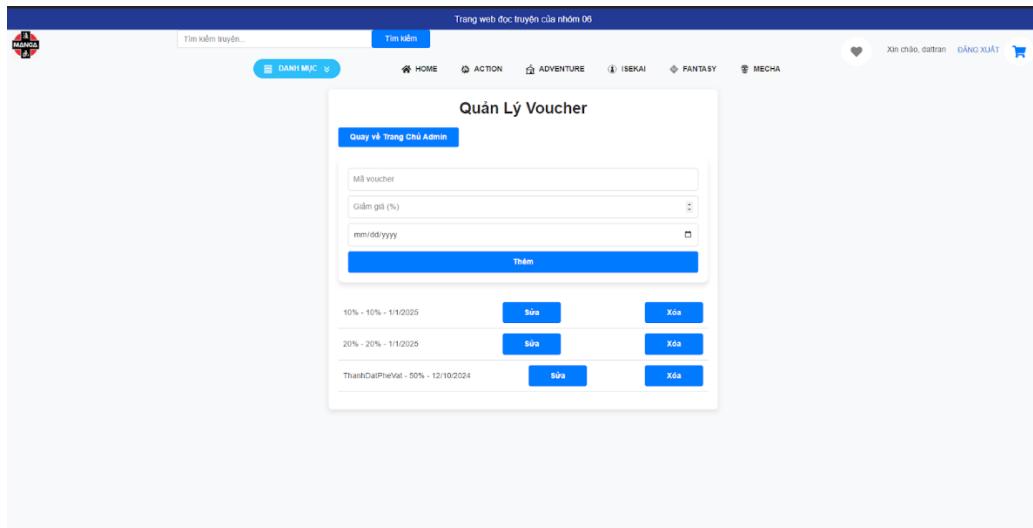
```

```

243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
    </ui>
    <div className="form-group">
      <label htmlFor="content">Content</label>
      <textarea
        id="content"
        value={form.content}
        onChange={(e) =>
          | setForm({ ...form, content: e.target.value })
        }
        rows="5"
      />
    </div>
    <div className="form-group">
      <label htmlFor="new-image">Add Image URL</label>
      <input
        type="text"
        id="new-image"
        value={newImage}
        onChange={(e) => setNewImage(e.target.value)}
        placeholder="Enter image URL"
      />
      <button
        type="button"
        onClick={addImageToList}
        className="add-image-button"
      >
        | Add Image
      </button>
    </div>
    <div className="image-preview">
      {form.images.map((image, index) => (
        <div key={index} className="image-item">
          <img
            | src={image}
            alt={`Preview ${index}`}
            className="preview-image"
          />
          <button
            type="button"
            onClick={() => removeImageFromList(index)}
            className="remove-image-button"
          >
            | Remove
          </button>
        </div>
      ))}
    </div>
    <div className="form-group">
      <label htmlFor="price">Price</label>
      <input
        type="number"
        id="price"
        value={form.price}
        onChange={(e) => setForm({ ...form, price: e.target.value })}
        required
      />
    </div>
    <div className="form-actions">
      <button type="submit" className="submit-button">
        | {editingChapterId ? "Update Chapter" : "Add Chapter"}
      </button>
      <button
        type="button"
        onClick={resetForm}
        className="reset-button"
      >

```

3.3.18 Giao diện Admin chỉnh sửa voucher:



Hình 3.34. Giao diện chỉnh sửa voucher của trang Admin
Code xử lí giao diện chỉnh sửa voucher:

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    if (isEditing) {
      await axios.put(`http://localhost:8080/api/admin/vouchers/${editId}`,
        form
      );
      setIsEditing(false);
      setEditId(null);
    } else {
      await axios.post("http://localhost:8080/api/admin/vouchers", form);
    }
    setForm({ code: "", discount: "", expiryDate: "" });
    fetchVouchers();
  } catch (error) {
    console.error("Error saving voucher:", error);
  }
};
```

```

return (
  <div className="voucher-manager">
    <h1>Quản Lý Voucher</h1>
    {/* Button to go back */}
    <button
      onClick={() => navigate("/admin/dashboard")}
      className="bg-gray-500 text-white px-4 py-2 rounded mb-4"
    >
      Quay về Trang Chủ Admin
    </button>
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        name="code"
        placeholder="Mã voucher"
        value={form.code}
        onChange={handleChange}
        required
      />
      <input
        type="number"
        name="discount"
        placeholder="Giảm giá (%)"
        value={form.discount}
        onChange={handleChange}
        required
      />
      <input
        type="date"
        name="expiryDate"
        placeholder="Ngày hết hạn"
        value={form.expiryDate}
        onChange={handleChange}
        required
      />
      <button type="submit">{isEditing ? "Cập nhật" : "Thêm"}</button>
    </form>
    <ul>
      {vouchers.map((voucher) => (
        <li key={voucher._id}>
          <span>
            {voucher.code} - {voucher.discount}% -{" "}
            {new Date(voucher.expiryDate).toLocaleDateString()}
          </span>
          <button onClick={() => handleEdit(voucher)}>Sửa</button>
          <button onClick={() => handleDelete(voucher._id)}>Xóa</button>
        </li>
      ))}
    </ul>
  </div>
);

```

3.3.19 Giao diện Admin quản lý đơn hàng user:

SẢN PHẨM	MÃ ĐƠN HÀNG	NGƯỜI DÙNG	NGÀY	TỔNG TIỀN	THANH TOÁN	GIAO HÀNG	HÀNH ĐỘNG
	675bc5c565fb6ea6b7e6948	test	2024-12-13	150,000 VND	Đã thanh toán	Đã giao	Xem chi tiết
	675bc71a65fb6ea6b7e6997	test	2024-12-13	130,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675bccca047451500c3e673a0	test	2024-12-13	1,070,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675bccce47451500c3e67421	test	2024-12-13	845,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675bcef47451500c3e67496	test	2024-12-13	925,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675bd10547451500c3e67504	test	2024-12-13	440,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675bdec247451500c3e67558	test	2024-12-13	300,000 VND	Chưa thanh toán	Chưa giao	Xem chi tiết
	675be12a47451500c3e67590	test	2024-12-13	240,000 VND	Đã thanh toán	Đã giao	Xem chi tiết

Hình 3.35. Giao diện Admin quản lý đơn hàng user
Tại đây admin có thể xem tình trạng thành toán và chi tiết đơn hàng của mỗi user

HÌNH ẢNH	SẢN PHẨM	SỐ LƯỢNG	ĐƠN GIÁ	TỔNG CỘNG	NGÀY TẠO	THANH TOÁN	GIAO HÀNG
	Bleach	3	70,000 VND	210,000 VND	12/13/2024	Đã thanh toán	Chưa giao hàng

Tóm tắt đơn hàng

Mã giảm giá:	2003
Mã đơn hàng:	675c429c7002ffa789d6aa5b
Người đặt:	user2
Email:	user2@gmail.com
Địa chỉ giao hàng:	đường 15, phường HBC, Thủ Đức, HCM
Phương thức thanh toán:	VNPay
Tổng cộng (đã bao gồm mã giảm giá):	73,500 VND

[Đánh dấu là đã giao hàng](#)

Hình 3.36. Giao diện Admin quản lý đơn hàng user

Đối với các đơn hàng đã thực hiện thanh toán từ người dùng, admin có thể thực hiện “Đánh dấu là đã giao hàng” khi người dùng nhận được sản phẩm.

KẾT LUẬN

Trong suốt quá trình thực hiện đề tài, nhóm em đã đạt được nhiều kết quả tích cực, đồng thời cũng rút ra được những bài học quý giá. Hệ thống giao diện frontend đã được xây dựng với đầy đủ các chức năng chính như quản lý admin (AdminDashboard, ChapterManagement, ManageComics, VoucherManager, OrderUser) và các trang dành cho người dùng (CartPage, DetailPage, ChapterPage, Favorites, Home, Profile, Contact, Order, Rating). Các tính năng như đăng nhập, đăng ký, tìm kiếm, quản lý và mua sắm truyện đã đáp ứng được các yêu cầu đề ra, đảm bảo sự linh hoạt và trải nghiệm người dùng tốt. Việc thiết kế giao diện theo hướng module hóa với các thành phần như Header, Footer, Navigation, SearchBox không chỉ giúp dễ dàng bảo trì và mở rộng, mà còn tăng tính nhất quán của hệ thống. Đặc biệt, các chức năng liên quan đến quản lý truyện và hiển thị chapter đã được phát triển chi tiết, mang lại giá trị sử dụng thực tiễn.

Tuy nhiên, trong quá trình thực hiện, vẫn còn một số điểm hạn chế. Hiệu năng của hệ thống chưa được tối ưu trên các trang quản lý khi xử lý dữ liệu lớn, dẫn đến tốc độ phản hồi chưa đạt kỳ vọng. Trang phân loại truyện chưa được tối ưu hóa và giao diện đa nền tảng dành cho người dùng như Flutter cần được cải thiện thêm để tạo cảm giác chuyên nghiệp hơn. Những hạn chế này cho thấy rằng, dù đã đạt được nhiều kết quả khả quan, nhưng vẫn còn cơ hội để phát triển và hoàn thiện hơn nữa.

Dựa trên những gì đã thực hiện, hướng phát triển trong tương lai sẽ tập trung vào việc tối ưu hóa và nâng cấp hệ thống. Trước tiên, cần tối ưu hiệu năng xử lý dữ liệu lớn trên các trang quản lý, nhằm nâng cao tốc độ phản hồi và trải nghiệm người dùng. Đồng thời, cải tiến giao diện để đảm bảo sự nhất quán và chuyên nghiệp. Bên cạnh đó, tích hợp các công nghệ mới như AI để gợi ý nội dung dựa trên sở thích, hoặc phát triển tính năng tìm kiếm thông minh cũng sẽ là những bước đi tiềm năng nhằm tăng giá trị của hệ thống. Ngoài ra, cải thiện khả năng responsive để giao diện hoạt động tốt trên mọi thiết bị và triển khai hệ thống lên các nền tảng cloud như AWS hoặc Azure sẽ giúp hệ thống ổn định, linh hoạt và dễ mở rộng hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] nodejs.org, "Introduction to Node.js," [Online]. Available: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.
- [2] V. Agarwal, "Advantages and Disadvantages of Node.js," 11 Jun 2024. [Online]. Available: <https://www.naukri.com/code360/library/advantages-and-disadvantages-of-nodejs>.
- [3] O. Foundation, "Express - Node.js web application," [Online]. Available: <https://expressjs.com>.
- [4] D. K. Toan, "Giới thiệu về ReactJS - Phần I (Các khái niệm cơ bản)," 2017. [Online]. Available: <https://viblo.asia/p/gioi-thieu-ve-reactjs-phan-icac-khai-niem-co-ban-V3m5WzjblO7>.
- [5] F. Official, "ReactJS 2022.," 2021. [Online]. Available: https://www.youtube.com/watch?v=x0fSBAgBrOQ&list=PL_-VfJajZj0UXjlKfBwFX73usByw3Ph9Q.
- [6] MongoDB, "Download MongoDB Community Edition," 2024. [Online]. Available: <https://www.mongodb.com/try/download/community>.
- [7] VN PAYMENT, "Demo Công thanh toán VN PAY," 2022. [Online]. Available: <https://sandbox.vnpayment.vn/apis/vnpay-demo/#:~:text=Hãy%20bắt%20đầu%20bằng%20việc%20sử%20dụng%20Demo,dưới%20và%20paste%20vào%20trình%20duyệt%20web%20http%3A%2F%2Fsandbox.vnpayment.vn%2Ftryitnow%2FHome%2FCreateOrder>.