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

BÁO CÁO KẾT THÚC MÔN

CÔNG NGHỆ PHẦN MỀM

HỌC KỲ II, NĂM HỌC 2025-2026

XÂY DỰNG ỨNG DỤNG BÁN SÁCH TRỰC TUYẾN

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Trần Ngọc Hành – 110122219
Lâm Trương Định – 110122052
Mai Trần Thanh Nhật – 110122017
Lớp: DA22TTA

Trà Vinh, tháng 7 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Trà Vinh, ngày …. . tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Trà Vinh, ngày …. . tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Lời nói đầu, Em xin cảm ơn thầy Nguyễn Bảo Ân đã hỗ trợ, hướng dẫn tận tình chúng em trong thời gian em làm môn phân tích và thiết kế hệ thống thông tin , những kiến thức mà thầy đã dạy chúng em sẽ là hành trang quý báo trên con đường học vấn và phát triển sự nghiệp tương lai rộng mở của chúng em. Thầy đã luôn kiên nhẫn, nhiệt tình trong việc truyền đạt kiến thức và kinh nghiệm quý báo, giúp chúng em vượt qua những khó khăn và thử thách trong quá trình học tập và nghiên cứu.

Những lời khuyên, góp ý của thầy không chỉ là kim chỉ nam cho sự phát triển của đồ án môn học này mà còn là nguồn động viên, khích lệ tinh thần lớn lao cho chúng em.

Chúng em xin hứa sẽ tiếp tục nỗ lực không ngừng để không phụ lòng thầy đã dành cho chúng em.

Xin chân thành cảm ơn Thầy.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: GIỚI THIỆU

1.1. Tên dự án và chủ đề

Dự án mang tên Web Bán Sách, là một nền tảng thương mại điện tử trực tuyến được thiết kế nhằm hỗ trợ người dùng trong việc tìm kiếm, lựa chọn và mua sắm sách một cách tiện lợi. Chủ đề của dự án tập trung vào việc xây dựng một hệ thống bán sách trực tuyến, đáp ứng nhu cầu ngày càng tăng của người dùng trong thời đại số hóa. Ứng dụng không chỉ cung cấp một kho sách đa dạng mà còn tích hợp các tính năng hiện đại như tìm kiếm thông minh, quản lý giỏ hàng, thanh toán an toàn và quản lý đơn hàng. Dự án được phát triển bởi nhóm DA22TTA, với mã nguồn được lưu trữ tại **https://github.com/NhatThanh115/DA-CNPM-DA22TTA**. Web Bán Sách hướng tới việc trở thành một giải pháp toàn diện, đáp ứng nhu cầu của cả người dùng phổ thông và những người yêu sách chuyên sâu.

1.2. Mục tiêu của ứng dụng

Mục tiêu chính của ứng dụng Web Bán Sách là tạo ra một nền tảng trực tuyến thân thiện, hiệu quả và an toàn để người dùng có thể dễ dàng tiếp cận và mua sắm sách. Cụ thể, ứng dụng được thiết kế để đạt được các mục tiêu sau:

- Cung cấp trải nghiệm mua sắm tiện lợi: Người dùng có thể duyệt qua danh mục sách phong phú, tìm kiếm sách theo các tiêu chí như tiêu đề, tác giả, thể loại hoặc giá cả, và thực hiện các giao dịch mua sắm một cách nhanh chóng.
- Đảm bảo tính bảo mật và xác thực: Hệ thống tích hợp cơ chế xác thực người dùng thông qua đăng nhập và đăng ký, sử dụng JSON Web Tokens (JWT) để bảo vệ thông tin cá nhân và lịch sử giao dịch.
- Quản lý đơn hàng hiệu quả: Cho phép người dùng theo dõi trạng thái đơn hàng, từ khi đặt hàng đến khi nhận sách, đồng thời cung cấp giao diện quản lý đơn hàng trực quan.
- Tối ưu hóa hiệu suất và khả năng tiếp cận: Ứng dụng sử dụng Vite để đảm bảo tốc độ tải trang nhanh, cùng với giao diện responsive tương thích trên cả thiết bị di động và máy tính, mang lại trải nghiệm liền mạch cho mọi đối tượng người dùng.
- Xây dựng nền tảng mở rộng: Sử dụng MongoDB và Docker, ứng dụng được thiết kế để dễ dàng mở rộng, hỗ trợ tích hợp thêm các tính năng mới như gợi ý sách cá nhân hóa hoặc phân tích hành vi người dùng trong tương lai.

Những mục tiêu này không chỉ hướng tới việc đáp ứng nhu cầu hiện tại của người dùng mà còn đặt nền tảng cho việc phát triển ứng dụng trong dài hạn, đảm bảo tính cạnh tranh trên thị trường thương mại điện tử sách.

1.3. Lý do chọn đề tài

Việc lựa chọn đề tài Web Bán Sách được nhóm cân nhắc kỹ lưỡng dựa trên các yếu tố thực tiễn, học thuật và xã hội. Đề tài không chỉ đáp ứng nhu cầu thực tế của thị trường mà còn mang lại cơ hội học tập, thử thách kỹ thuật và ý nghĩa xã hội sâu sắc. Dưới đây là năm lý do chính dẫn đến việc chọn đề tài này:

- Nhu cầu thị trường ngày càng tăng: Thị trường thương mại điện tử tại Việt Nam đang phát triển mạnh mẽ, với tốc độ tăng trưởng hàng năm đạt mức ấn tượng nhờ sự thay đổi trong hành vi mua sắm của người tiêu dùng. Theo các thống kê gần đây, nhu cầu mua sách trực tuyến tăng cao do sự tiện lợi và khả năng tiếp cận nguồn tài liệu đa dạng. Đặc biệt, người dùng trẻ tuổi và những người yêu sách đang tìm kiếm các nền tảng trực tuyến cung cấp trải nghiệm mua sắm mượt mà, với các tính năng như tìm kiếm nhanh, thanh toán an toàn và giao hàng tận nơi. Web Bán Sách được thiết kế để đáp ứng nhu cầu này, cung cấp một giải pháp toàn diện cho việc mua sắm sách trực tuyến, từ đó góp phần vào sự phát triển của ngành thương mại điện tử sách tại Việt Nam.
- Cơ hội học tập và phát triển kỹ năng: Dự án mang lại cơ hội quý báu để nhóm áp dụng và làm chủ các công nghệ hiện đại đang được sử dụng rộng rãi trong ngành công nghiệp phần mềm. Cụ thể, việc sử dụng React và TypeScript cho giao diện người dùng, Node.js cho backend, MongoDB cho cơ sở dữ liệu, và Docker cho triển khai ứng dụng giúp nhóm tiếp cận với quy trình phát triển phần mềm toàn diện. Ngoài ra, việc tích hợp các công cụ như JSON Web Tokens (JWT) cho xác thực và Swagger cho tài liệu API cung cấp kinh nghiệm thực tế trong việc xây dựng hệ thống an toàn và dễ bảo trì. Quá trình phát triển dự án không chỉ củng cố kiến thức lý thuyết mà còn rèn luyện các kỹ năng như lập trình, thiết kế hệ thống, và làm việc nhóm, chuẩn bị cho nhóm những kỹ năng cần thiết để bước vào môi trường làm việc chuyên nghiệp.
- Tính thực tiễn và khả năng ứng dụng: Web Bán Sách là một dự án có tính ứng dụng cao, với tiềm năng triển khai thực tế cho các nhà xuất bản, cửa hàng sách hoặc các nền tảng thương mại điện tử. Hệ thống được thiết kế để đáp ứng các nhu

cầu cơ bản của một nền tảng bán hàng trực tuyến, bao gồm duyệt sản phẩm, quản lý giỏ hàng, thanh toán và quản lý đơn hàng. Với kiến trúc linh hoạt sử dụng MongoDB và Docker, ứng dụng có thể dễ dàng được điều chỉnh để tích hợp thêm các tính năng như phân tích dữ liệu người dùng hoặc tích hợp với các hệ thống thanh toán phổ biến tại Việt Nam như Momo hoặc ZaloPay. Tính thực tiễn của dự án còn được thể hiện qua khả năng triển khai nhanh chóng trên các môi trường khác nhau, từ máy chủ cục bộ đến đám mây, đáp ứng nhu cầu của các doanh nghiệp vừa và nhỏ.

- Thách thức kỹ thuật và sự sáng tạo: Dự án đặt ra nhiều thách thức kỹ thuật thú vị, thúc đẩy nhóm tìm kiếm các giải pháp sáng tạo và tối ưu. Ví dụ, việc sử dụng Vite để tối ưu hóa tốc độ tải trang đòi hỏi nhóm phải hiểu rõ cách cấu hình và tối ưu hóa hiệu suất frontend. Tích hợp MongoDB với Node.js yêu cầu thiết kế cơ sở dữ liệu NoSQL hiệu quả, đồng thời đảm bảo 1 khả năng mở rộng khi lượng dữ liệu tăng. Ngoài ra, việc triển khai xác thực người dùng bằng JWT và tài liệu hóa API bằng Swagger đòi hỏi sự chính xác trong việc xử lý bảo mật và giao tiếp giữa các thành phần hệ thống. Những thách thức này không chỉ giúp nhóm nâng cao kỹ năng kỹ thuật mà còn khuyến khích sự sáng tạo trong việc thiết kế giao diện người dùng thân thiện và tối ưu hóa trải nghiệm người dùng.
- Ý nghĩa xã hội và văn hóa: Web Bán Sách không chỉ là một nền tảng thương mại mà còn mang ý nghĩa xã hội sâu sắc thông qua việc khuyến khích văn hóa đọc trong cộng đồng. Trong bối cảnh công nghệ số đang thay đổi cách con người tiếp cận tri thức, việc cung cấp một nền tảng trực tuyến để mua sắm sách giúp người dùng, đặc biệt là thế hệ trẻ, dễ dàng tiếp cận các nguồn tài liệu phong phú. Dự án góp phần thúc đẩy việc đọc sách, một hoạt động quan trọng để nâng cao tri thức và phát triển cá nhân. Hơn nữa, bằng cách cung cấp một giao diện dễ sử dụng và các tính năng như tìm kiếm thông minh, ứng dụng giúp giảm rào cản trong việc tiếp cận sách, từ đó đóng góp vào sự phát triển của giáo dục và văn hóa tại Việt Nam.

Những lý do trên không chỉ phản ánh sự phù hợp của đề tài với xu hướng công nghệ và thị trường mà còn thể hiện cam kết của nhóm trong việc xây dựng một sản phẩm có giá trị thực tiễn và ý nghĩa xã hội. Dự án Web Bán Sách là một cơ hội để nhóm thử

nghiệm, học hỏi và hoàn thiện các kỹ năng cần thiết, đồng thời tạo ra một nền tảng có tiềm năng ứng dụng cao trong thực tế.

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

2.1. Các chức năng chính của hệ thống (Functional Requirements)

Các chức năng chính của hệ thống được thiết kế để hỗ trợ người dùng trong toàn bộ quá trình mua sắm sách trực tuyến, từ việc tìm kiếm sách đến quản lý đơn hàng. Dưới đây là các chức năng chính của hệ thống, được mô tả chi tiết:

- **Duyệt và tìm kiếm sách:**

  – Người dùng có thể duyệt qua danh mục sách được phân loại theo thể loại (ví dụ: văn học, khoa học, kỹ thuật), tác giả hoặc mức giá.

  – Hệ thống cung cấp chức năng tìm kiếm thông minh, cho phép người dùng nhập từ khóa (tiêu đề, tác giả, ISBN) để lọc kết quả nhanh chóng.

  – Kết quả tìm kiếm có thể được sắp xếp theo các tiêu chí như giá (tăng dần, giảm dần), độ phổ biến hoặc ngày xuất bản.

  – Mỗi sách được hiển thị với thông tin chi tiết bao gồm tiêu đề, tác giả, giá, mô tả ngắn, và hình ảnh bìa sách (nếu có).

- **Quản lý giỏ hàng và thanh toán:**

  – Người dùng có thể thêm sách vào giỏ hàng từ trang chi tiết sách hoặc danh sách sách.

  – Giao diện giỏ hàng hiển thị danh sách sách đã chọn, số lượng, giá đơn vị, và tổng giá trị đơn hàng.

  – Người dùng có thể chỉnh sửa giỏ hàng (thêm, xóa, cập nhật số lượng sách) trước khi tiến hành thanh toán.

  – Hệ thống hỗ trợ quy trình thanh toán an toàn, tích hợp với các cổng thanh toán phổ biến (ví dụ: giả định tích hợp với Stripe hoặc cổng thanh toán nội địa như Momo, nếu được triển khai).

  – Sau khi thanh toán thành công, hệ thống tạo đơn hàng và gửi xác nhận qua giao diện người dùng hoặc email (nếu được tích hợp).

- **Xác thực người dùng:**

  – Hệ thống cung cấp chức năng đăng ký tài khoản, yêu cầu người dùng cung cấp thông tin như email, mật khẩu, và tên.

  – Chức năng đăng nhập sử dụng email và mật khẩu, với cơ chế xác thực dựa trên JSON Web Tokens (JWT) để đảm bảo bảo mật.

  – Hệ thống hỗ trợ khôi phục mật khẩu thông qua email (nếu được triển khai).

– Người dùng đã đăng nhập có thể truy cập các tính năng cá nhân hóa như lịch sử đơn hàng và thông tin tài khoản.

- **Quản lý đơn hàng:**

  – Người dùng có thể xem danh sách các đơn hàng đã đặt, bao gồm trạng thái (chờ xử lý, đang giao, đã giao), ngày đặt hàng, và chi tiết sách trong đơn hàng.

  – Hệ thống cho phép người dùng hủy đơn hàng (nếu đơn hàng chưa được xử lý) hoặc theo dõi tiến trình giao hàng (nếu được tích hợp với dịch vụ giao hàng).

  – Quản trị viên (nếu có) có thể truy cập giao diện quản lý để xử lý đơn hàng, cập nhật trạng thái, hoặc xem thống kê đơn hàng.

- **Quản lý nội dung sách (dành cho quản trị viên):**

  – Hệ thống cung cấp giao diện quản trị để thêm, sửa, xóa thông tin sách trong cơ sở dữ liệu.

  – Quản trị viên có thể cập nhật danh mục sách, giá, hoặc trạng thái tồn kho.

  – Chức năng này yêu cầu xác thực với vai trò quản trị viên để đảm bảo bảo mật.

Các chức năng trên được thiết kế để đáp ứng nhu cầu của cả đối tượng người dùng phổ thông (người mua sách) và quản trị viên (người quản lý hệ thống), đảm bảo trải nghiệm mua sắm liền mạch và hiệu quả.

2.2. Các yêu cầu phi chức năng (Non-functional Requirements)

Các yêu cầu phi chức năng xác định các thuộc tính chất lượng của hệ thống, đảm bảo ứng dụng đáp ứng các tiêu chuẩn về hiệu suất, bảo mật, khả năng mở rộng và trải nghiệm người dùng. Dưới đây là các yêu cầu phi chức năng chính:

- **Hiệu suất:**

  – Hệ thống phải đảm bảo thời gian phản hồi của các trang (như trang chủ, trang tìm kiếm) dưới 2 giây trong điều kiện tải bình thường (dưới 100 người dùng đồng thời).

  – Việc sử dụng Vite cho frontend giúp tối ưu hóa tốc độ tải trang, giảm thời gian xây dựng và triển khai mã nguồn.

  – Các truy vấn cơ sở dữ liệu MongoDB phải được tối ưu hóa để xử lý nhanh các yêu cầu tìm kiếm và lọc sách, ngay cả khi cơ sở dữ liệu chứa hàng nghìn bản ghi.

- **Tính tương thích và khả năng truy cập:**

  – Giao diện người dùng phải tương thích với các thiết bị di động, máy tính bảng và máy tính để bàn, sử dụng Tailwind CSS để đảm bảo thiết kế responsive.

– Hệ thống hỗ trợ các trình duyệt phổ biến như Chrome, Firefox, Safari và Edge (phiên bản mới nhất).

– Giao diện được thiết kế để dễ sử dụng, với các yếu tố điều hướng rõ ràng và trực quan, phù hợp với người dùng ở mọi độ tuổi và trình độ công nghệ.

- **Bảo mật:**

  – Thông tin người dùng (email, mật khẩu, lịch sử giao dịch) phải được mã hóa và lưu trữ an toàn trong MongoDB.

  – Cơ chế xác thực JWT đảm bảo rằng chỉ người dùng được ủy quyền mới có thể truy cập các tính năng cá nhân hóa hoặc quản trị.

  – Các API RESTful phải được bảo vệ chống lại các cuộc tấn công phổ biến như SQL injection hoặc cross-site scripting (XSS).

  – Kết nối giữa client và server sử dụng HTTPS (nếu triển khai trên môi trường sản xuất) để mã hóa dữ liệu truyền tải.

- **Khả năng mở rộng:**

  – Hệ thống được thiết kế để hỗ trợ mở rộng quy mô, với MongoDB cho phép xử lý dữ liệu lớn và Docker hỗ trợ triển khai trên các môi trường khác nhau.

  – Backend Node.js có thể được mở rộng để xử lý hàng nghìn người dùng đồng thời thông qua cân bằng tải (load balancing) nếu cần.

  – Hệ thống hỗ trợ tích hợp thêm các tính năng trong tương lai, như gợi ý sách dựa trên trí tuệ nhân tạo hoặc phân tích hành vi người dùng.

- **Độ tin cậy và bảo trì:**

  – Hệ thống phải đạt độ sẵn sàng (uptime) ít nhất 99% trong môi trường sản xuất, đảm bảo người dùng có thể truy cập dịch vụ liên tục.

  – Tài liệu API (thông qua Swagger) và mã nguồn được tổ chức rõ ràng để hỗ trợ bảo trì và phát triển tiếp theo.

  – Quy trình CI/CD sử dụng GitHub Actions đảm bảo mã nguồn được kiểm tra và triển khai tự động, giảm thiểu lỗi trong quá trình phát triển.

- **Trải nghiệm người dùng:**

  – Giao diện phải có thiết kế trực quan, với màu sắc hài hòa và phông chữ dễ đọc, sử dụng Noto Serif để hỗ trợ tiếng Việt chính xác.

  – Thời gian phản hồi của các hành động người dùng (như thêm vào giỏ hàng, thanh toán) phải dưới 1 giây để đảm bảo trải nghiệm mượt mà.

– Hệ thống cung cấp thông báo lỗi rõ ràng và thân thiện khi người dùng nhập sai thông tin hoặc gặp sự cố.

Các yêu cầu phi chức năng này đảm bảo rằng hệ thống không chỉ hoạt động đúng chức năng mà còn đáp ứng các tiêu chuẩn chất lượng cao về hiệu suất, bảo mật và trải nghiệm người dùng, phù hợp với một nền tảng thương mại điện tử hiện đại.

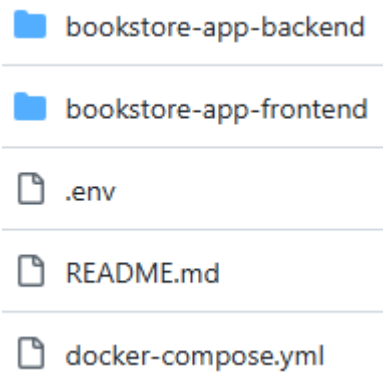
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

Hệ thống Web Bán Sách được thiết kế theo mô hình client-server, với các thành phần chính bao gồm giao diện người dùng (frontend), máy chủ xử lý logic nghiệp vụ (backend), và cơ sở dữ liệu. Dưới đây là mô tả chi tiết về kiến trúc hệ thống:

- Client (Frontend): Giao diện người dùng được xây dựng bằng React và TypeScript, sử dụng Vite làm công cụ xây dựng để tối ưu hóa tốc độ tải trang. Tailwind CSS được sử dụng để tạo giao diện responsive, đảm bảo tương thích trên các thiết bị di động và máy tính để bàn. Frontend giao tiếp với backend thông qua các API RESTful, sử dụng các thư viện như Axios để thực hiện các yêu cầu HTTP.
- Server (Backend): Backend được triển khai bằng Node.js với framework Express, cung cấp các API RESTful để xử lý các yêu cầu từ client. Backend xử lý các tác vụ như xác thực người dùng (sử dụng JWT), quản lý sách, và xử lý đơn hàng. Các API được tài liệu hóa bằng Swagger để hỗ trợ phát triển và kiểm thử.
- Cơ sở dữ liệu: MongoDB, một cơ sở dữ liệu NoSQL, được sử dụng để lưu trữ dữ liệu về sách, người dùng, và đơn hàng. MongoDB được chọn vì tính linh hoạt và khả năng mở rộng, phù hợp với các yêu cầu của một ứng dụng thương mại điện tử.
- Triển khai: Docker được sử dụng để đóng gói ứng dụng, với Docker Compose quản lý các dịch vụ (frontend, backend, MongoDB). Điều này đảm bảo tính nhất quán trong triển khai trên các môi trường khác nhau.

Sơ đồ kiến trúc hệ thống (hình minh họa sẽ được đính kèm trong báo cáo in) có thể được mô tả như sau: Client (trình duyệt hoặc thiết bị di động) gửi yêu cầu HTTP đến backend qua các API RESTful. Backend xử lý yêu cầu, tương tác với MongoDB để truy xuất hoặc cập nhật dữ liệu, và trả về phản hồi dưới dạng JSON. Các container Docker đảm bảo các dịch vụ hoạt động đồng bộ.



Hình 1: Kiến trúc tổng thể

3.2. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu được thiết kế sử dụng MongoDB, một cơ sở dữ liệu NoSQL, để hỗ trợ lưu trữ và truy xuất dữ liệu linh hoạt. Mô hình dữ liệu bao gồm các collection chính sau:

```
import dotenv from 'dotenv';
dotenv.config();

import mongoose from 'mongoose';

const connectDB = async () => {
  try {
    const mongoURI = process.env.MONGODB_URI;
    if (!mongoURI) {
      console.error('Error: MONGODB_URI is not defined in .env file');
      process.exit(1);
    }
    await mongoose.connect(mongoURI);
    console.log('MongoDB Connected...');
  } catch (err: any) {
    console.error('MongoDB Connection Error:', err.message);
    // Exit process with failure
    process.exit(1);
  }
};

export default connectDB;
```

Hình 2: MongoDB

bookstore-app-backend > .env

```
MONGODB_URI=mongodb+srv://thanhnhatff:wtDPlPjLvXQnzT3H@cluster0.api0w0p.mongodb.net/?retryWrite
JWT_SECRET=your-super-secret-jwt-key-that-should-be-very-long-and-random
```

Hình 3: Lưu trữ các biến môi trường

```
export interface IBook extends Document {
  title: string;
  author: string;
  description: string;
  price: number;
  imageUrl: string;
  category: string;
  rating: number;
  featured: boolean;
  inStock: boolean;
  publicationDate: Date;
}
```

Hình 4: Lưu trữ thông tin về sách

3.3. Thiết kế API

API của hệ thống Web Bán Sách được thiết kế theo chuẩn RESTful, sử dụng Node.js với framework Express để xử lý các yêu cầu từ client. Các endpoint chính hỗ trợ các chức năng như duyệt sách, xác thực người dùng, và quản lý đơn hàng. API được tài liệu hóa bằng Swagger, có thể truy cập tại http://localhost:3000/api-docs khi chạy backend. Dưới đây là mô tả ngắn gọn về các endpoint chính, cấu trúc request/response, và các đoạn mã nguồn chính.

3.3.1. Các endpoint chính

1. GET /api/books

```
* @swagger
 * /books:
 *   get:
 *     summary: Retrieve a list of books with optional filters and pagination
 *     tags: [Books]
 *     parameters:
 *       - in: query
 *         name: category
 *         schema: { type: string }
 *         description: Filter books by category slug (e.g., 'sci-fi')
 *       - in: query
 *         name: featured
 *         schema: { type: boolean }
 *         description: Filter for featured books
 *       - in: query
 *         name: page
 *         schema: { type: integer, default: 1 }
 *         description: Page number for pagination
 *       - in: query
 *         name: limit
 *         schema: { type: integer, default: 10 }
 *         description: Number of books per page
 *       - in: query
 *         name: sortBy
 *         schema: { type: string, enum: [createdAt, title, price, rating, publicationDate], default: createdAt }
 *         description: Field to sort by
 *       - in: query
 *         name: order
 *         schema: { type: string, enum: [asc, desc], default: desc }
 *         description: Sort order (ascending or descending)
 *     responses:
 *       200:
 *         description: A list of books
 *         content:
 *           application/json:
 *             schema: { $ref: '#/components/schemas/BooksPaginatedResponse' }
 *       500:
 *         description: Server error
 */
router.get('/', getBooks);
```

Hình 5: GET /api/books

2. POST /api/auth/register

```
* @swagger
 * tags:
 *   name: Authentication
 *   description: User authentication and management
 */

/**
 * @swagger
 * /auth/register:
 *   post:
 *     summary: Register a new user
 *     tags: [Authentication]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/UserInput'
 *     responses:
 *       201:
 *         description: User registered successfully
 *         content:
 *           application/json:
 *             schema:
 *               type: object
 *               properties:
 *                 token: { type: 'string' }
 *                 user: { $ref: '#/components/schemas/User' } # Assuming User schema is defined in swaggerConfig or User.ts model
 *       400:
 *         description: Bad request (e.g., user already exists, invalid input)
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Error'
 *       500:
 *         description: Server error
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Error'
 */
router.post('/register', registerUser);
```

Hình 6: POST /api/auth/register

3. POST /api/auth/login

```
* @swagger
 * /auth/login:
 *   post:
 *     summary: Log in an existing user
 *     tags: [Authentication]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/UserLogin'
 *     responses:
 *       200:
 *         description: User logged in successfully
 *         content:
 *           application/json:
 *             schema:
 *               type: object
 *               properties:
 *                 token: { type: 'string' }
 *                 user: { $ref: '#/components/schemas/User' }
 *       400:
 *         description: Invalid credentials
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Error'
 *       500:
 *         description: Server error
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Error'
 */
router.post('/login', loginUser);
```

Hình 7: POST /api/auth/login

4. POST /api/orders

```
* @swagger
 * tags:
 *   name: Users
 *   description: User profile and favorites management (requires authentication for all routes)
 * security:
 *   - bearerAuth: [] # Applies to all operations in this tag unless overridden
 */

router.use(authMiddleware); // All routes below are protected

/**
 * @swagger
 * /users/profile:
 *   get:
 *     summary: Get current user's profile
 *     tags: [Users]
 *     responses:
 *       200:
 *         description: Successfully retrieved user profile
 *         content:
 *           application/json:
 *             schema: { $ref: '#/components/schemas/User' } # Defined in swaggerConfig or User model
 *       401:
 *         description: Unauthorized
 *       500:
 *         description: Server error
 */
router.get('/profile', getUserProfile);

/**
 * @swagger
 * /users/profile:
 *   put:
 *     summary: Update current user's profile
 *     tags: [Users]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema: { $ref: '#/components/schemas/UserProfileUpdate' }
 *     responses:
 *       200:
 *         description: Profile updated successfully
 *         content:
 *           application/json:
 *             schema: { $ref: '#/components/schemas/User' }
 *       400:
 *         description: Invalid input data (e.g., username/email taken)
 *       401:
 *         description: Unauthorized
 *       500:
 *         description: Server error
 */
router.put('/profile', updateUserProfile);
```

Hình 8: POST /api/orders

3.3.2. Swagger

API được định nghĩa với các thông tin cơ bản như sau:

- Tiêu đề: BookHaven API.
- Phiên bản: 1.0.0.
- Mô tả: Tài liệu API cho ứng dụng sách trực tuyến BookHaven.
- Server phát triển: http://localhost:process.env.P ORT ||5001/api. API sử dụng cơ chế xác thực JWT (bearerAuth) cho các thao tác yêu cầu bảo mật, được áp dụng mặc định cho tất cả endpoint trừ khi ghi đè.

Schemas chính từ Swagger Các schema được định nghĩa trong swaggerConfig.ts để mô tả cấu trúc dữ liệu:

UserInput:

```
schemas: {
  // Define common schemas here to be referenced in your JSDoc comments
  UserInput: {
    type: 'object',
    properties: {
      username: { type: 'string', example: 'johndoe' },
      email: { type: 'string', format: 'email', example: 'johndoe@example.com' },
      name: { type: 'string', example: 'John Doe' },
      password: { type: 'string', format: 'password', example: 'password123' },
    },
    required: ['username', 'email', 'name', 'password'],
  },
```

Hình 9: UserInput

UserLogin:

```
UserLogin: {
  type: 'object',
  properties: {
    email: { type: 'string', format: 'email', example: 'johndoe@example.com' },
    password: { type: 'string', format: 'password', example: 'password123' },
  },
  required: ['email', 'password'],
},
```

Hình 10: UserLogin

Book:

```
Book: {
  type: 'object',
  properties: {
    _id: { type: 'string', example: '60c72b2f9b1e8c3b4c8e4f1a' },
    title: { type: 'string', example: 'The Great Gatsby' },
    author: { type: 'string', example: 'F. Scott Fitzgerald' },
    description: { type: 'string', example: 'A novel about the American dream.' },
    price: { type: 'number', format: 'float', example: 10.99 },
    imageUrl: { type: 'string', format: 'url', example: 'http://example.com/image.jpg' },
    category: { type: 'string', example: 'Fiction' },
    rating: { type: 'number', format: 'float', example: 4.5 },
    featured: { type: 'boolean', example: false },
    inStock: { type: 'boolean', example: true },
    publicationDate: { type: 'string', format: 'date', example: '1925-04-10' },
    createdAt: { type: 'string', format: 'date-time' },
    updatedAt: { type: 'string', format: 'date-time' },
  },
},
```

Hình 11: Book

3.3.3. Mã nguồn chính

Backend (Node.js/Express): Dưới đây là các đoạn mã nguồn chính cho backend, bao gồm cấu hình server, schema MongoDB, và xử lý API.

Cấu hình index (index.ts):

```ts
import express from 'express';
import type { Express, Request, Response } from 'express';
import dotenv from 'dotenv';
import cors from 'cors';
import connectDB from './config/db.js';
import authRoutes from './routes/authRoutes.js';
import bookRoutes from './routes/bookRoutes.js';
import userRoutes from './routes/userRoutes.js';
import swaggerUi from 'swagger-ui-express';
import swaggerSpec from './config/swaggerConfig.js';

dotenv.config();

const app: Express = express();
const port = process.env.PORT || 5001;

// Connect to Database
connectDB();

// Middleware
app.use(cors()); // Enable CORS for all routes
app.use(express.json()); // for parsing application/json

// Serve Swagger Docs
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));

// Define a simple route
app.get('/', (req: Request, res: Response) => {
  res.send('BookHaven Backend API is running! Access API docs at /api-docs');
});

// Define API routes (prefixed with /api)
app.use('/api/auth', authRoutes);
app.use('/api/books', bookRoutes);
app.use('/api/users', userRoutes);

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
  console.log(`Swagger API docs available at http://localhost:${port}/api-docs`);
});

export default app;
```

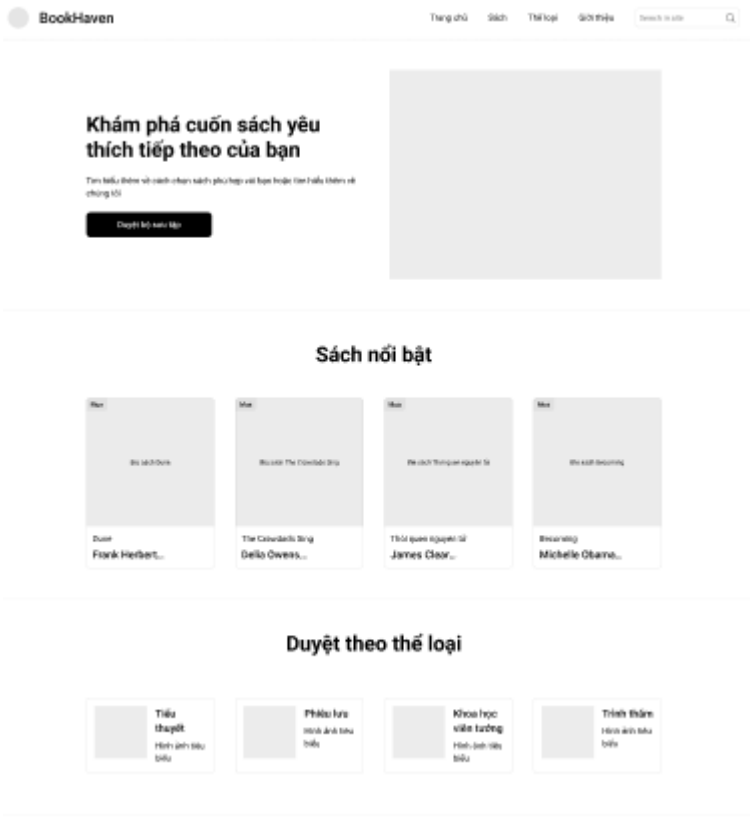
Hình 12: Cấu hình index

3.4. Thiết kế giao diện (UI/UX)

Hệ thống Web Bán Sách, được phát triển bằng React, TypeScript, và Tailwind CSS trong thư mục bookstore-app-frontend. Giao diện được thiết kế để đảm bảo tính trực quan, thân thiện, và tương thích trên các thiết bị khác nhau. Các màn hình chính được mô tả chi tiết, cùng với liên kết đến bản thiết kế Figma, nơi chứa các bố cục và nguyên mẫu giao diện.

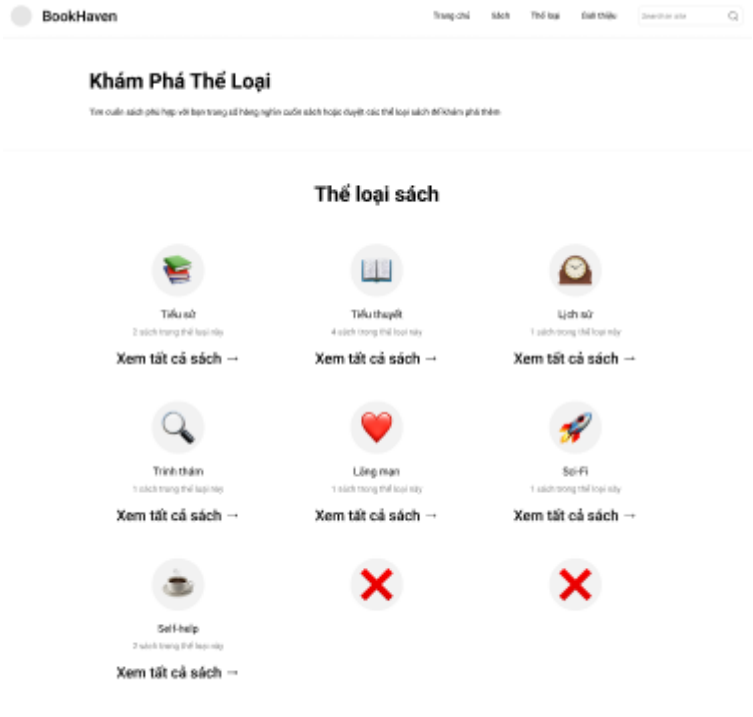
Mô tả các màn hình chính:

- Trang chủ sách:



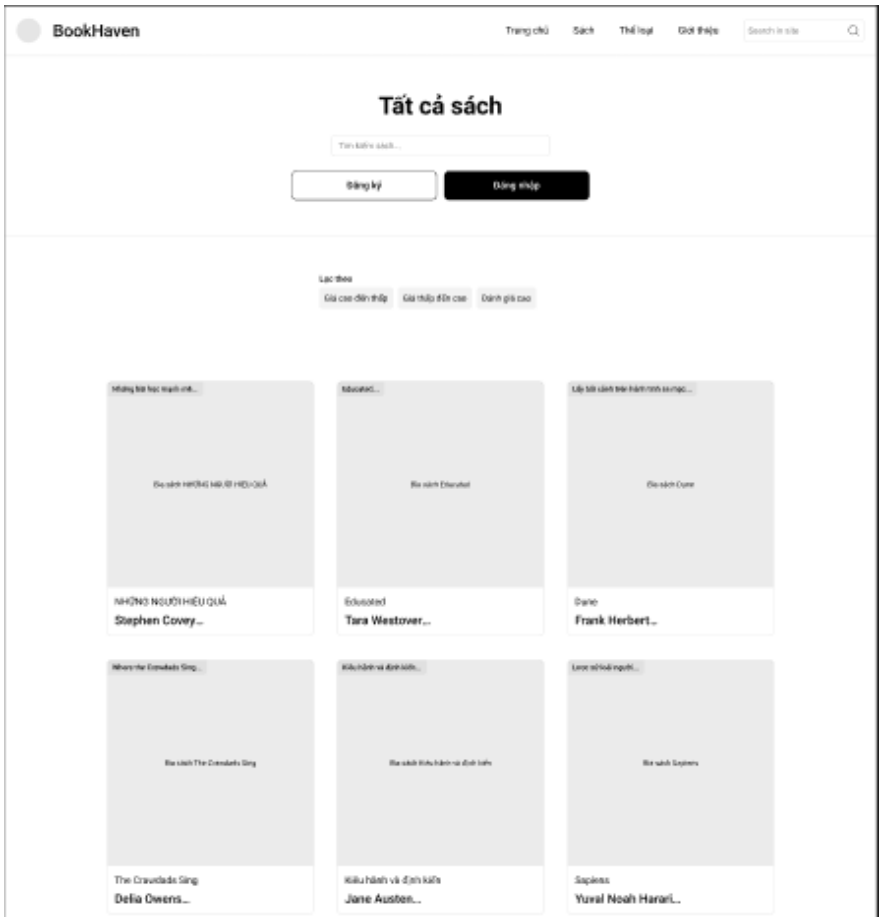
Hình 13: Trang chủ sách

- Trang thể loại sách:



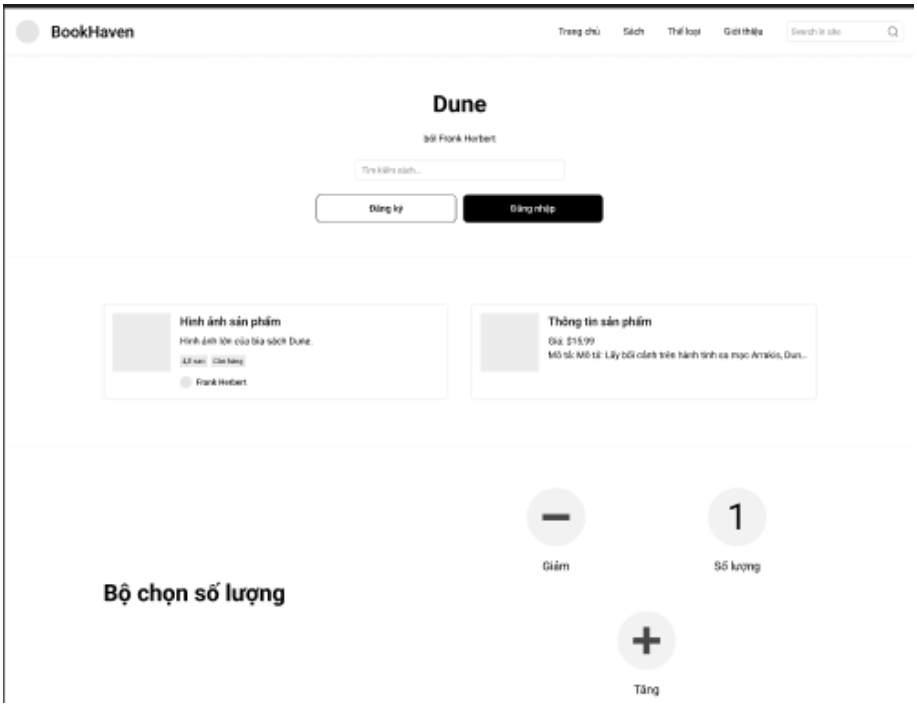
Hình 14: Trang thể loại sách

- Trang tất cả sách:



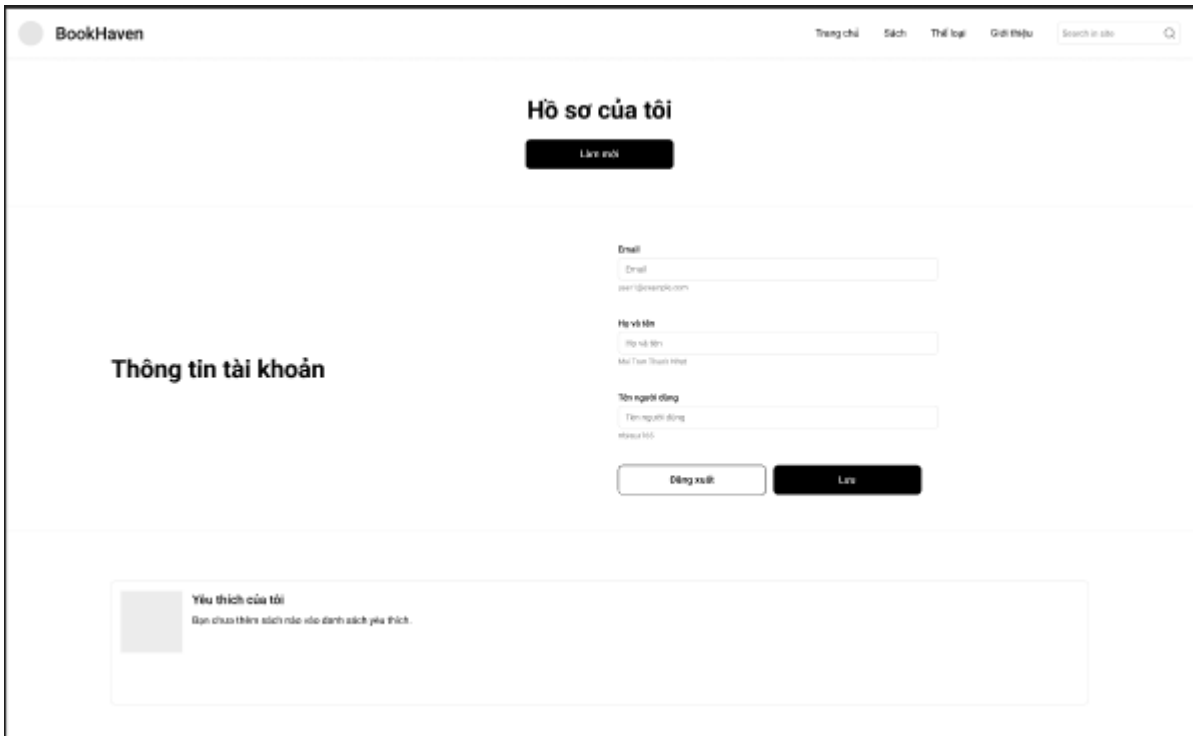
Hình 15: Trang tất cả sách

- Trang giỏ hàng:



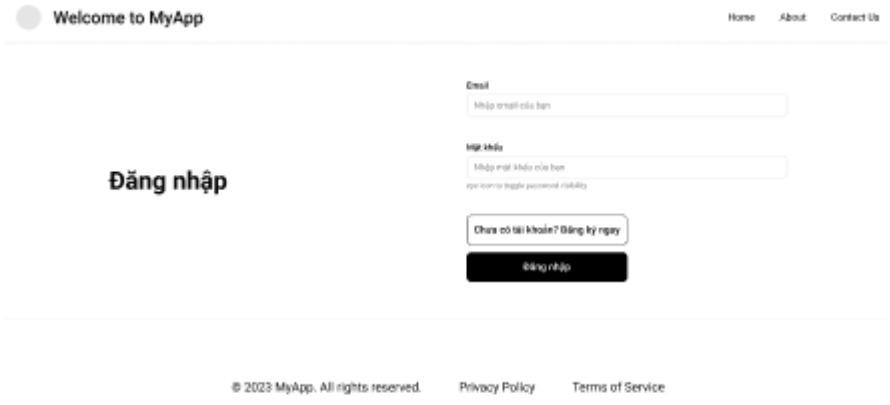
Hình 16: Trang giỏ hàng

- Trang hồ sơ và thông tin của khách hàng:



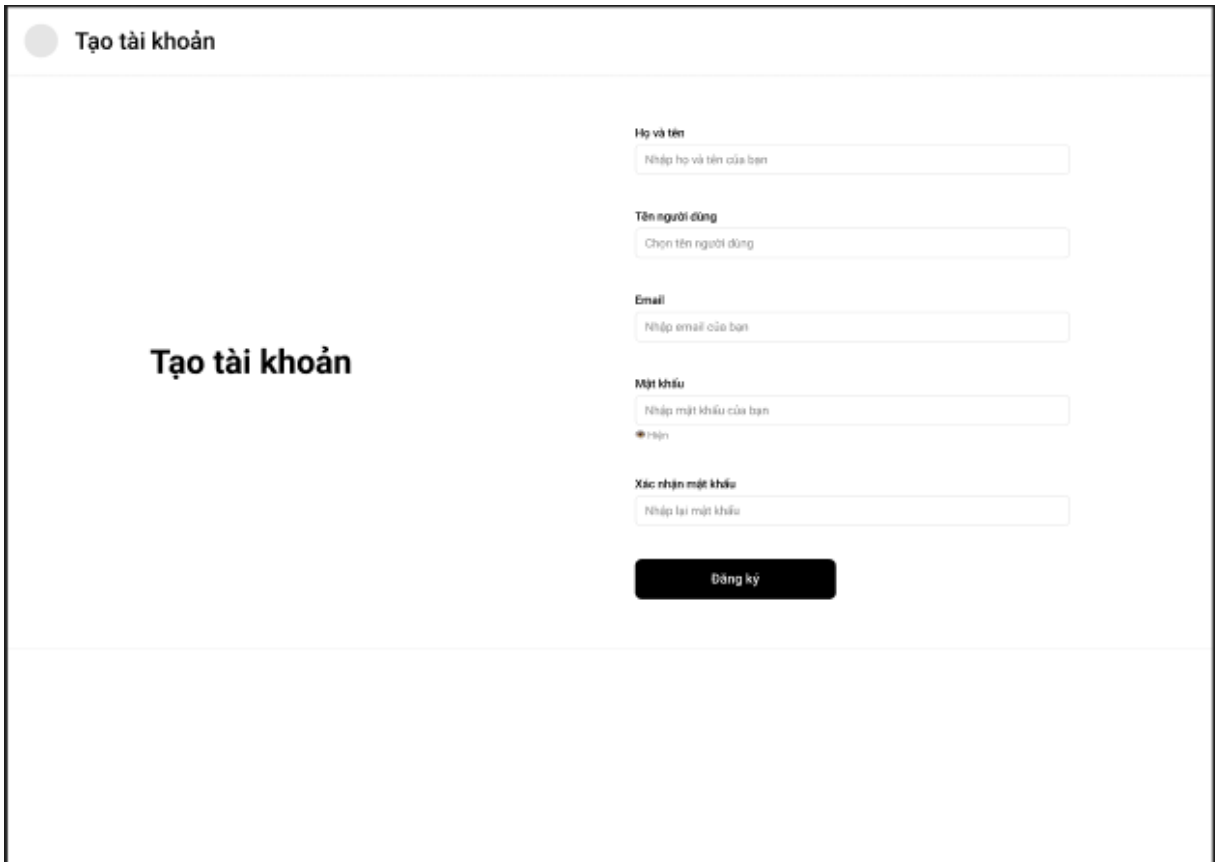
Hình 17: Trang hồ sơ và thông tin của tôi

- Trang đăng nhập:



Hình 18: Trang đăng nhập

- Trang đăng ký:



Hình 19: Trang đăng ký

CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

4.1. Danh sách các công nghệ đã sử dụng

Dự án sử dụng một tập hợp các công nghệ hiện đại để phát triển và triển khai, bao gồm:

Ngôn ngữ lập trình:

– JavaScript/TypeScript: Sử dụng cho cả backend và frontend, với TypeScript để tăng cường kiểm tra kiểu dữ liệu.

– Node.js: Được sử dụng làm môi trường runtime cho backend, hỗ trợ xử lý yêu cầu API.

Framework và thư viện:

– Express.js: Framework backend để xây dựng API RESTful trong thư mục bookstore-app-backend.

– React: Framework frontend để phát triển giao diện người dùng trong thư mục bookstore-app-frontend, tích hợp với TypeScript.

– Tailwind CSS: Thư viện CSS dùng để thiết kế giao diện đáp ứng (responsive) và tối ưu hóa hiệu suất.

Cơ sở dữ liệu:

– MongoDB: Cơ sở dữ liệu NoSQL được sử dụng để lưu trữ thông tin sách, người dùng, và đơn hàng, được cấu hình trong config/db.js.

Công cụ tự động hóa và triển khai:

– GitHub Actions: Công cụ CI/CD để tự động hóa xây dựng, kiểm thử, và triển khai.

– Docker: Sử dụng để container hóa ứng dụng, đảm bảo tính nhất quán giữa các môi trường phát triển và sản xuất.

Công cụ khác:

– Swagger: Dùng để tài liệu hóa API, được cấu hình trong swaggerConfig.ts.

– JWT (JSON Web Token): Sử dụng cho xác thực người dùng trong các route authRoutes.ts.

Các công nghệ trên được chọn để đảm bảo tính linh hoạt, hiệu suất cao, và khả năng mở rộng cho ứng dụng.

4.2. Quy trình CI/CD với GitHub Actions

Quy trình CI/CD được triển khai với GitHub Actions nhằm tự động hóa việc tích hợp mã nguồn, kiểm thử, và triển khai ứng dụng. Quy trình được định nghĩa trong tệp .github/workflows/ci-cd.yml trong repository, bao gồm các bước sau:

1. Kiểm tra mã nguồn:

- Khi có thay đổi được đẩy lên nhánh chính (main), GitHub Actions tự động kích hoạt workflow.

2. Xây dựng ứng dụng:

- Checkout mã nguồn bằng action actions/checkout@v3.
- Cài đặt các phụ thuộc Node.js bằng lệnh npm install.
- Xây dựng dự án frontend (React) và backend (Express) bằng npm run build.

3. Kiểm thử:

- Chạy các bài kiểm thử đơn vị và tích hợp (nếu có) bằng npm test, đảm bảo tính toàn vẹn của mã nguồn.

4. Xây dựng và đẩy Docker image:

- Đăng nhập vào Docker Hub bằng thông tin bí mật (secrets.DOCKER USERNAME)

5. Triển khai:

- Triển khai image Docker lên môi trường staging hoặc production (giả định sử dụng server riêng hoặc dịch vụ như AWS ECS), với các bước như kéo image và khởi động container.

Quy trình này đảm bảo rằng mọi thay đổi mã nguồn được kiểm tra và triển khai một cách tự động, giảm thiểu rủi ro lỗi và tăng tốc độ phát hành.

4.3. Cấu hình Docker và quy trình triển khai ứng dụng

Docker được sử dụng để container hóa ứng dụng, đảm bảo tính nhất quán giữa các môi trường phát triển, kiểm thử, và sản xuất. Dưới đây là cấu hình và quy trình triển khai:

4.3.1. Cấu hình Docker

Tệp Dockerfile được định nghĩa trong thư mục gốc của backend:

```
FROM node:20-alpine AS builder

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . .


RUN npm run build || { echo "Build failed!"; exit 1; }


RUN echo "--- Listing all files after build step ---" && ls -R


RUN test -d /app/dist || { echo "Error: /app/dist directory not found!"; exit 1; }

FROM node:20-alpine

WORKDIR /app

COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./package.json

RUN npm install --omit=dev

COPY --from=builder /app/dist ./dist


RUN echo "--- Listing all files in final stage ---" && ls -R

EXPOSE 5001

CMD ["node", "dist/index.js"]
```

Hình 20: Dockerfile

FROM node:20-alpine: Sử dụng image Node.js phiên bản 20 trên nền tảng Alpine nhẹ.

- COPY: Sao chép mã nguồn và phụ thuộc vào container.
- EXPOSE 5000: Mở cổng 5000 để phục vụ ứng dụng.
- CMD: Chạy ứng dụng bằng lệnh npm start.

Tệp docker-compose.yml có thể được sử dụng để định nghĩa các service, bao gồm backend và cơ sở dữ liệu MongoDB:

```yaml
services:
  backend:
    build:
      context: ./bookstore-app-backend
      dockerfile: Dockerfile
    ports:
      - "5001:5001" # Expose backend on host port 5001, mapping to container port 5001
    environment:
      - NODE_ENV=production
      - PORT=5001
      - MONGODB_URI=${MONGODB_URI} # These should be in a .env file at the root of your project
      - JWT_SECRET=${JWT_SECRET}
    volumes:
      - ./bookstore-app-backend:/usr/src/app # Mount local backend code for development (optional)
      - /usr/src/app/node_modules # Don't mount local node_modules over container's
    restart: unless-stopped
    depends_on:
      - mongodb # If you were running MongoDB in Docker

  frontend:
    build:
      context: ./bookstore-app-frontend
      dockerfile: Dockerfile
      args:
        VITE_API_BASE_URL: http://localhost:5001/api # URL for backend service inside Docker network
    ports:
      - "3000:3000" # Expose frontend Nginx on host port 3000, mapping to container port 80
    depends_on:
      - backend
    restart: unless-stopped

  mongodb:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db
    restart: unless-stopped

volumes:
    mongo-data: # Defines a named volume for MongoDB data persistence
```

Hình 21: docker-compose.yml

4.3.2. Quy trình triển khai ứng dụng

Quy trình triển khai ứng dụng được thực hiện như sau:

1. **Xây dựng image:** Chạy lệnh docker build -t username/bookhaven:latest . để tạo image từ Dockerfile.
2. **Đẩy image lên Docker Hub:** Sử dụng docker push username/bookhaven:latest sau khi đăng nhập.
3. **Triển khai container:** Trên server mục tiêu, kéo image bằng docker pull username/bookhaven:latest và khởi động container với docker run -d -p 5000:5000 username/bookhaven:latest.

CHƯƠNG 5: QUẢN LÝ DỰ ÁN VÀ KẾT QUẢ

5.1. Quản lý dự án

5.1.1. Lập kế hoạch và theo dõi tiến độ trên Jira

	Kiểu	Chìa khóa	Bản tóm tắt	Trạng thái	Chạy nước rút	Người được chuyển nhượng
☐	☑	SCRUM-1	Tìm hiểu nhu cầu, chức năng website	DONE	SCRUM Sprint 1	TH Tran Ngoc Hanh
☐	☑	SCRUM-2	Thiết kế giao diện web trên Figma	DONE	SCRUM Spri...	TH Tran Ngoc Hanh
☐	☑	SCRUM-3	Lập trình giao diện web	DONE	SCRUM Spri...	Unassigned
☐	☑	SCRUM-4	Lập trình chức năng web	DONE	SCRUM Spri...	MN Mai Tran Thanh Nhat
☐	☑	SCRUM-5	Chạy thử web, tìm và sửa lỗi	DONE	SCRUM Spri...	MN Mai Tran Thanh Nhat

Hình 22: Tiến độ trên Jira

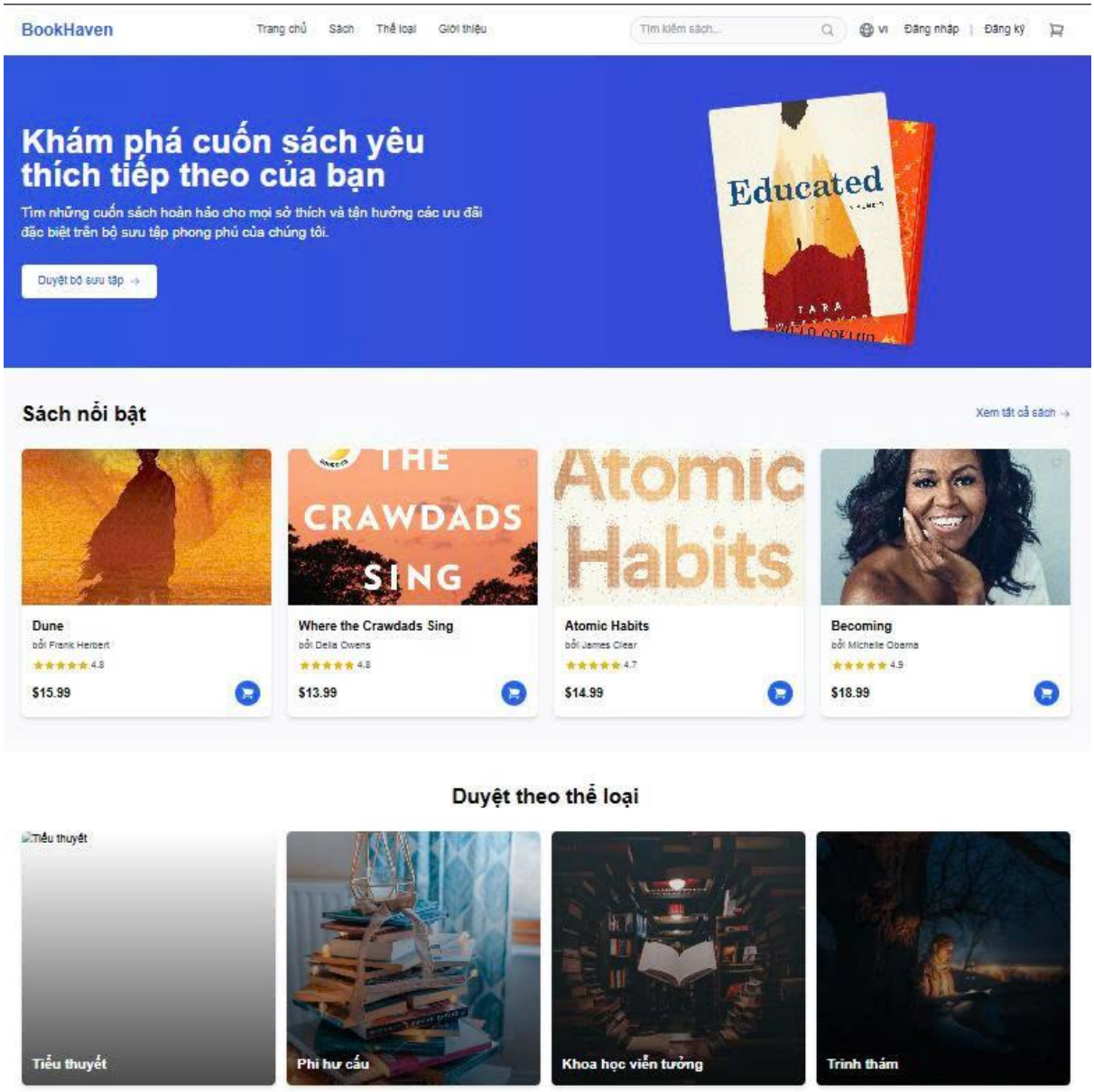
5.1.2. Phân công nhiệm vụ của từng thành viên trong nhóm

Bảng 1: Phân công nhiệm vụ

Họ Tên Sinh Viên	Công Việc
Trần Ngọc Hành	- Thiết kế giao diện web trên Figma - Viết báo cáo
Lâm Trương Định	- Lập trình giao diện Web - Thiết kế slide
Mai Trần Thanh Nhật	- Lập trình chức năng web - Chạy thử Web, tìm và sửa lỗi

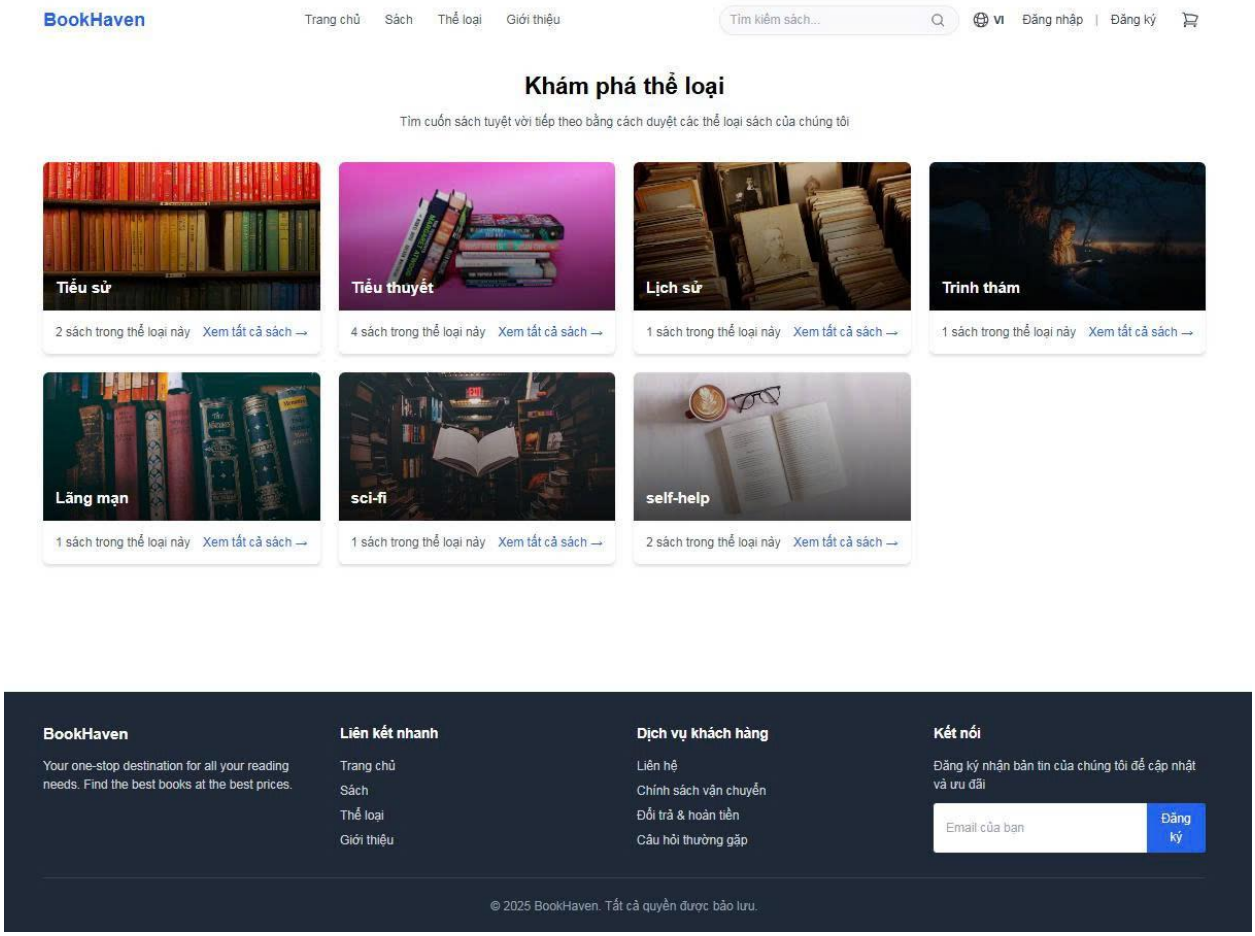
5.2. Kết Quả Chạy Web

Kết quả trang chủ web



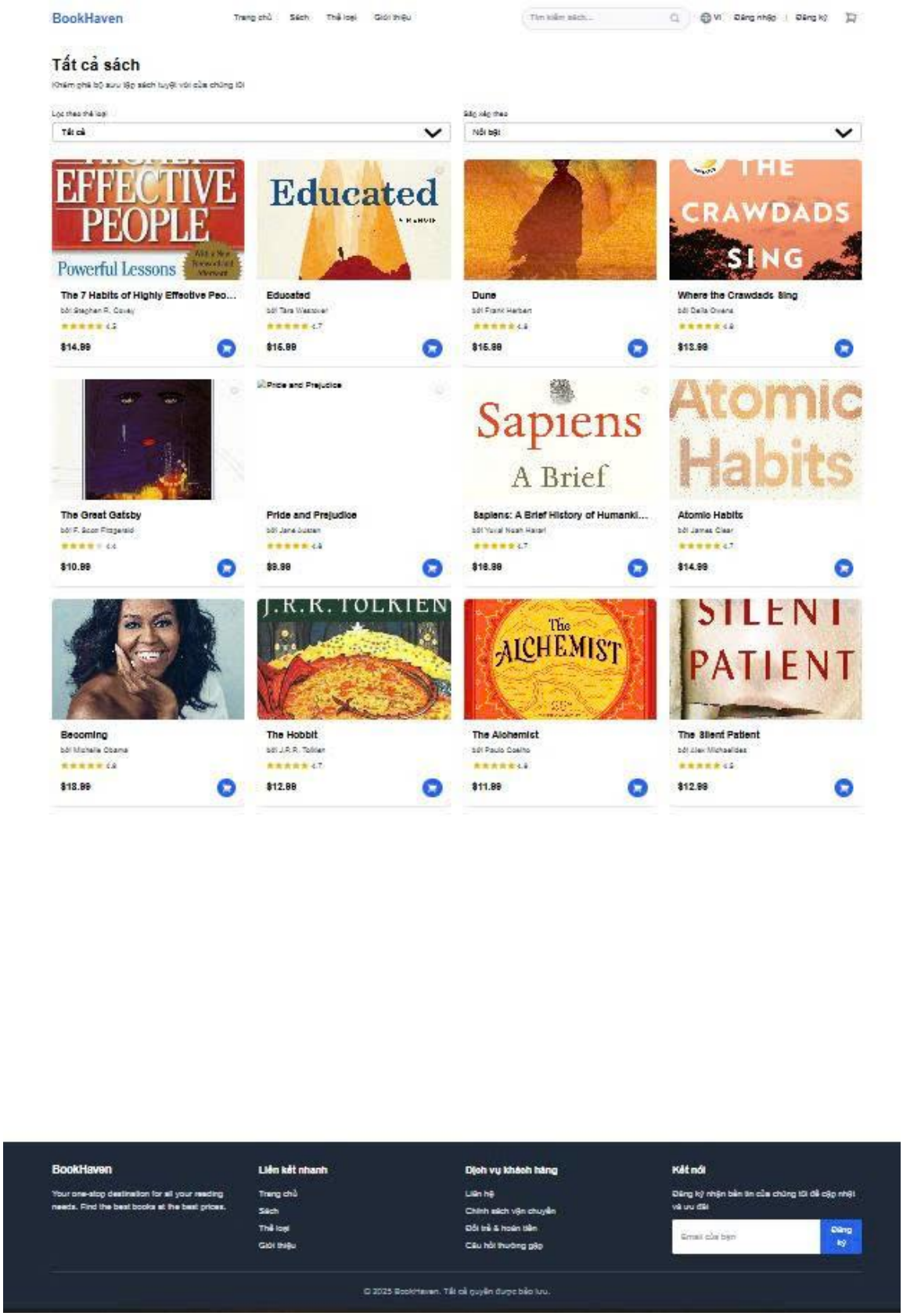
Hình 23: Kết quả trang chủ sách

Kết quả trang thể loại sách



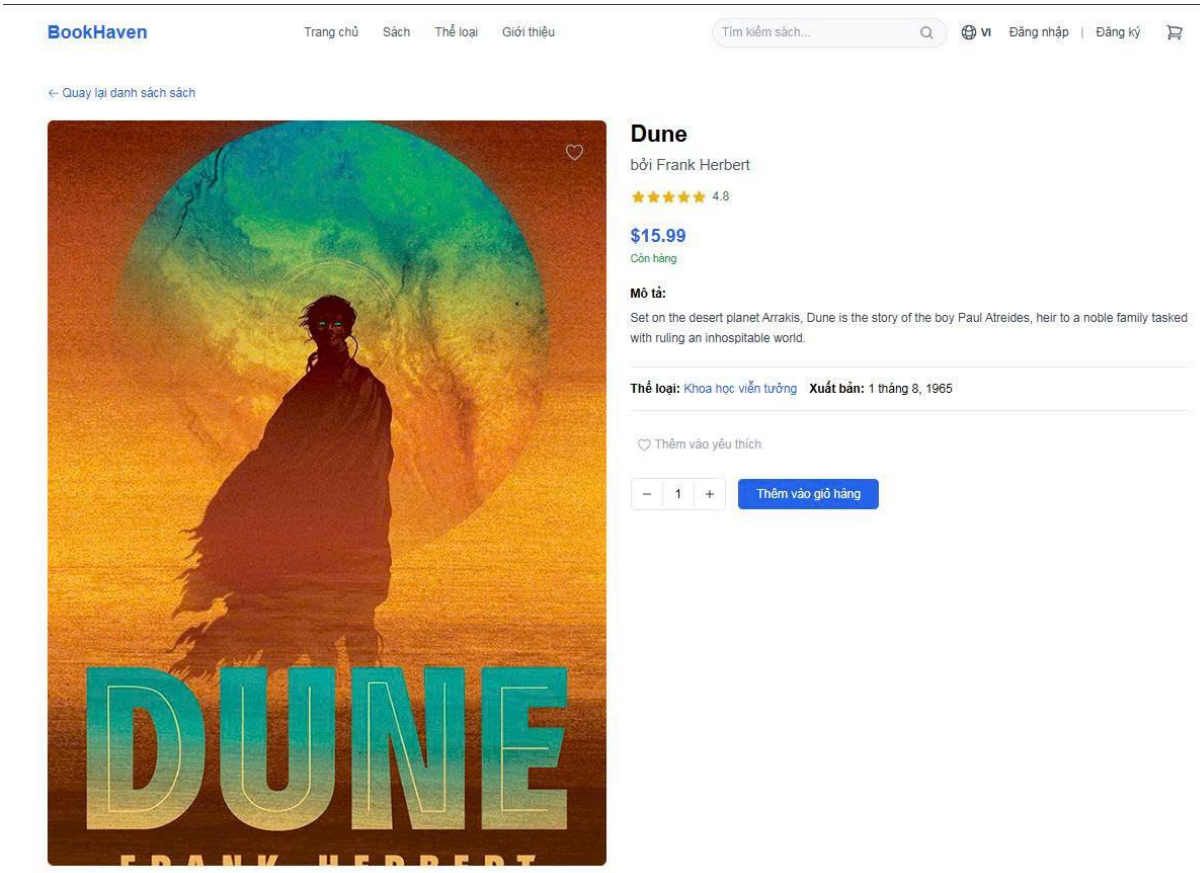
Hình 24: Kết quả trang thể loại sách

Kết quả trang tất cả sách



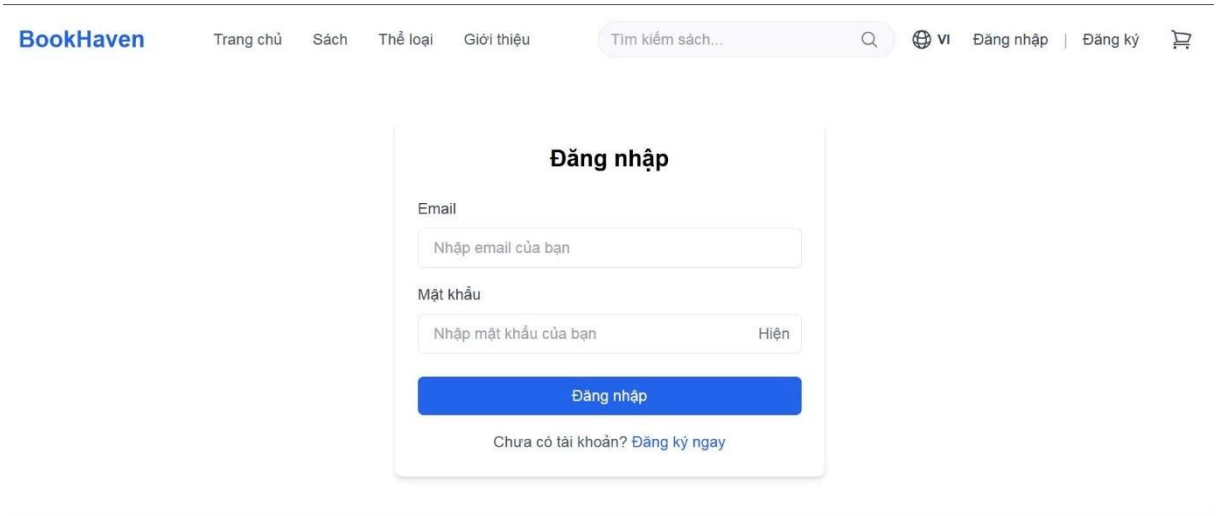
Hình 25: Kết quả trang tất cả sách

Kết quả trang giỏ hàng



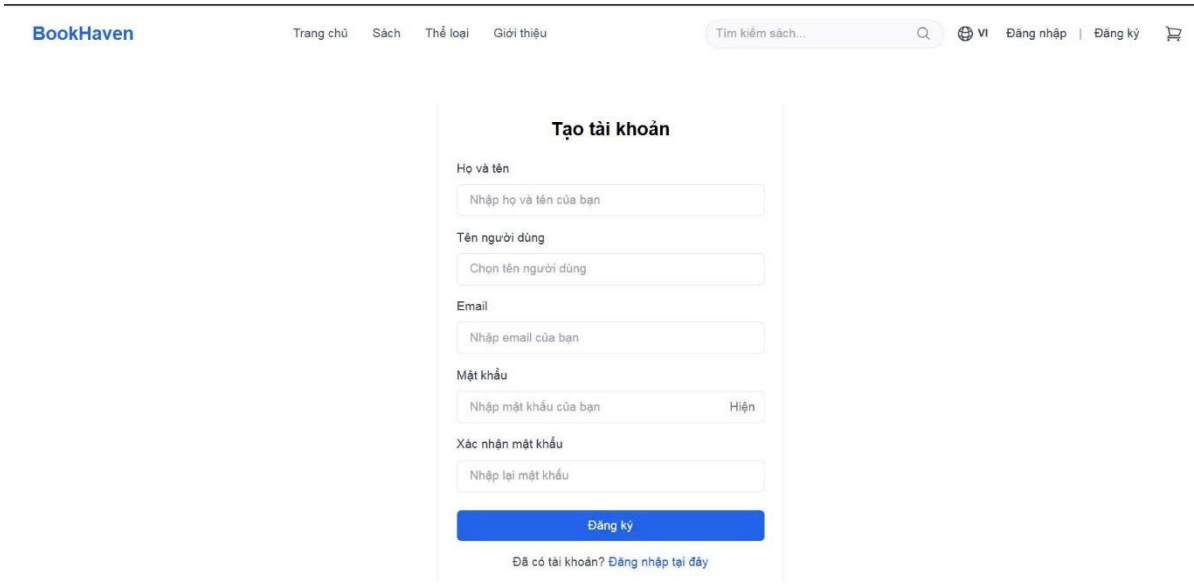
Hình 26: Kết quả trang giỏ hàng

Kết quả trang đăng nhập



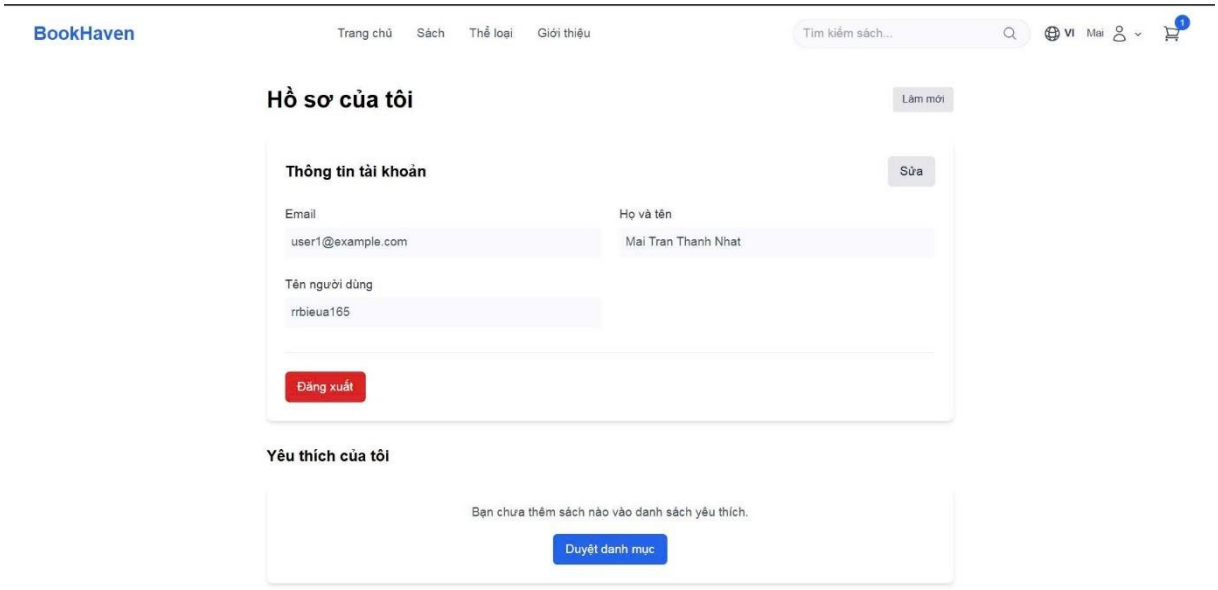
Hình 27: Kết quả trang đăng nhập

Kết quả trang đăng ký



Hình 28: Kết quả trang đăng ký

Kết quả hồ sơ của tôi



Hình 29: Kết quả hồ sơ và thông tin của tôi

CHƯƠNG 6: ĐÁNH GIÁ VÀ KẾT LUẬN

6.1. Những khó khăn gặp phải trong quá trình thực hiện

Trong quá trình phát triển dự án, nhóm đã đối mặt với một số thách thức đáng kể, bao gồm:

- Khó khăn trong tích hợp công nghệ: Việc kết hợp React, Node.js, và MongoDB đòi hỏi sự đồng bộ hóa giữa frontend và backend, dẫn đến các lỗi không nhất quán trong xử lý dữ liệu, đặc biệt khi triển khai API RESTful.
- Quản lý thời gian và tiến độ: Một số nhiệm vụ, như thiết kế giao diện responsive với Tailwind CSS và kiểm thử bằng Postman, đã vượt quá thời gian dự kiến do thiếu kinh nghiệm ban đầu trong việc ước lượng công sức.
- Vấn đề bảo mật và hiệu suất: Tích hợp JSON Web Token (JWT) và tối ưu hóa hiệu suất MongoDB gặp khó khăn do yêu cầu bảo mật cao và khối lượng dữ liệu thử nghiệm hạn chế.
- Triển khai Docker: Cấu hình Docker và tích hợp với GitHub Actions gặp trở ngại do lỗi môi trường và thiếu tài liệu chi tiết trong giai đoạn đầu.

Những khó khăn này đã ảnh hưởng đến tiến độ dự án, nhưng được giải quyết thông qua sự điều chỉnh và học hỏi liên tục trong quá trình thực hiện.

6.2. Bài học rút ra và đề xuất cải thiện trong tương lai

Từ quá trình thực hiện, nhóm đã rút ra các bài học quan trọng và đề xuất các cải tiến như sau:

Bài học rút ra:

– Tầm quan trọng của lập kế hoạch chi tiết trên Jira: Việc phân bổ thời gian hợp lý và cập nhật tiến độ thường xuyên giúp giảm thiểu rủi ro chậm trễ.

– Kinh nghiệm tích hợp công nghệ: Việc làm quen sớm với các công cụ như Docker và Swagger giúp tăng hiệu quả trong các giai đoạn sau.

– Hợp tác nhóm: Giao tiếp thường xuyên giữa các thành viên (backend, frontend, QA) là yếu tố then chốt để giải quyết các vấn đề kỹ thuật.

Đề xuất cải thiện trong tương lai:

– Tăng cường đào tạo trước dự án về các công nghệ mới (Docker, MongoDB) để giảm thời gian làm quen.

– Triển khai kiểm thử tải (load testing) bằng công cụ như JMeter để đánh giá hiệu suất hệ thống dưới áp lực lớn.

– Cải thiện tài liệu hóa, đặc biệt là hướng dẫn cấu hình môi trường và quy trình CI/CD, để hỗ trợ các thành viên mới hoặc dự án tiếp theo.

– Tích hợp thêm các tính năng như gợi ý sách cá nhân hóa hoặc hỗ trợ đa ngôn ngữ để nâng cao trải nghiệm người dùng.

PHỤ LỤC

- Clone repository:

- **git clone https://github.com/NhatThanh115/DA-CNPM-DA22TTA.git**

- Di chuyển vào thư mục dự án

- Khởi chạy npm install để cài đặt các gói mở rộng cần thiết cho dự án

- Cách để sử dụng Swagger

- Di chuyển đến folder backend

- **cd bookstore-app-backend**

- Cài đặt các gói phụ thuộc

- **npm install**

- Khởi chạy backend để sử dụng swagger

- **npm run dev**

- Triển khai Docker, ta chạy lệnh

- **docker-compose up --build**

* Lưu ý: nên chạy lệnh tại nơi chứa file docker-compose.yml

Link Github:

https://github.com/NhatThanh115/DA-CNPM-DA22TTA

Chỉnh Code thẳng trên github:

https://github.dev/NhatThanh115/DA-CNPM-DA22TTA

DANH MỤC TÀI LIỆU THAM KHẢO

1. https://react.dev/learn
2. https://www.w3schools.com/mongodb/
3. https://v2.tailwindcss.com/docs