0

**TRƯỜNG ĐẠI HỌC VĂN LANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ**

**Ỳ**

Website bán sách

***Môn học:***

**Lập trình ứng dụng Web**

**Phạm Minh Quân**

**LỚP: 241\_71ITSE30503\_0301**

**GVHD: Nguyễn Văn Trung**

**Đặng Đình Hòa**

**TP. Hồ Chí Minh – 11/2024**

**TP. Hồ Chí Minh – 3/2023**

**TP. Hồ Chí Minh – 3/2023**

**TP. Hồ Chí Minh – 3/2023**

****![A picture containing background pattern

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADIAAAA/CAMAAACVf2tdAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAGUExURf///wAAAFXC034AAAACdFJOU/8A5bcwSgAAAAlwSFlzAAAh1QAAIdUBBJy0nQAAABpJREFUSEvtwTEBAAAAwqD1T20LLyAAAIC/GgyNAAGHlV64AAAAAElFTkSuQmCC)

**MỤC LỤC**

[**I. Cấu trúc hệ thống website bán sách** 2](#_Toc182959903)

[**II. Front-end** 3](#_Toc182959904)

[**1. Công nghệ xây dựng website** 3](#_Toc182959905)

[**2. Components** 4](#_Toc182959906)

[**3. App.js** 4](#_Toc182959907)

[**4. Package.json** 7](#_Toc182959908)

[**5. Giao diện của Website** 8](#_Toc182959909)

[5.1 Header 8](#_Toc182959910)

[5.2 Footer 9](#_Toc182959911)

[5.3 Trang đăng ký 10](#_Toc182959912)

[**5.4 Trang đăng nhập** 12](#_Toc182959913)

[**5.5 Trang chủ** 14](#_Toc182959914)

[5.6 Trang thông tin chi tiết sách 15](#_Toc182959915)

[5.7 Trang giỏ hàng 17](#_Toc182959916)

[**III. Back-end** 19](#_Toc182959917)

[**1. Database** 19](#_Toc182959918)

[**2. books.php** 20](#_Toc182959919)

[**3. bookdetail.php** 21](#_Toc182959920)

[**4. genre.php** 22](#_Toc182959921)

[**5. login.php** 24](#_Toc182959922)

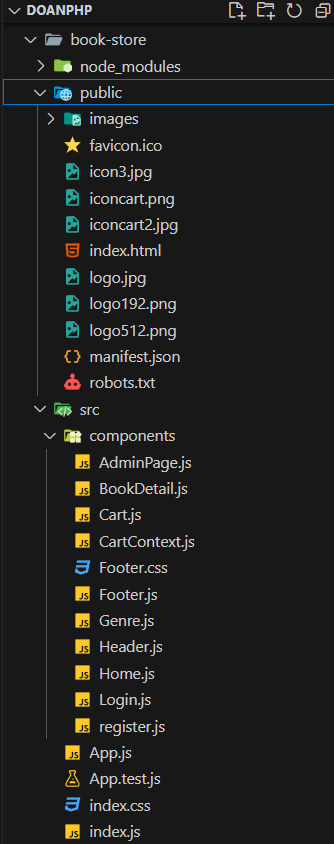
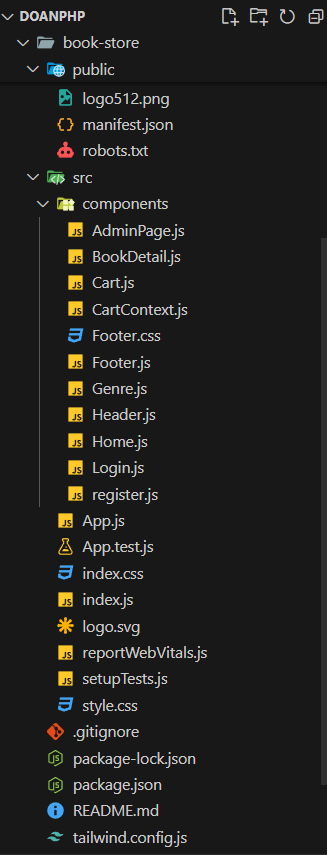
[**6. register.php** 26](#_Toc182959923)

[**IV. Database** 27](#_Toc182959924)

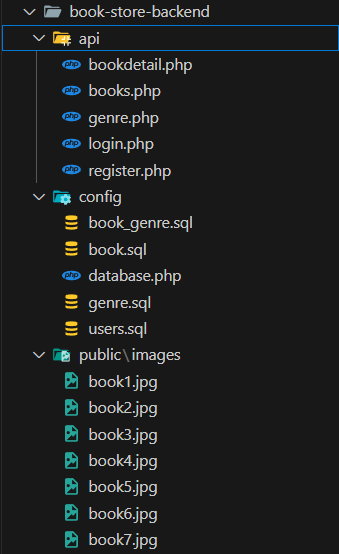
[**1. XAMPP** 27](#_Toc182959925)

[**2. Thành phần của cơ sở dữ liệu** 28](#_Toc182959926)

# **I. Cấu trúc hệ thống website bán sách**

**- Front-end:**

**- Back-end:**

****

# **II. Front-end**

## **1. Công nghệ xây dựng website**

Phần Front-end của hệ thống website bán sách online được xây dựng bằng các công nghệ sau:

- **React.js** là một thư viện JavaScript mã nguồn mở, được phát triển bởi Facebook, dùng để xây dựng giao diện người dùng (UI) cho ứng dụng web. React cho phép xây dựng ứng dụng đơn trang (SPA) với cách tiếp cận dựa trên component, giúp mã nguồn dễ duy trì, mở rộng, và tái sử dụng. React sử dụng Virtual DOM để tối ưu hóa việc cập nhật giao diện, giúp tăng hiệu suất.

- **JavaScript (ES6)**: là một ngôn ngữ lập trình phổ biến được sử dụng chủ yếu để phát triển các ứng dụng web động. ES6 (ECMAScript 2015) là một phiên bản nâng cấp quan trọng của JavaScript, mang lại nhiều tính năng mới như:  
+ Let/const thay cho var để khai báo biến.

+ Arrow functions giúp cú pháp ngắn gọn và dễ đọc.

+ Template literals hỗ trợ định dạng chuỗi linh hoạt.

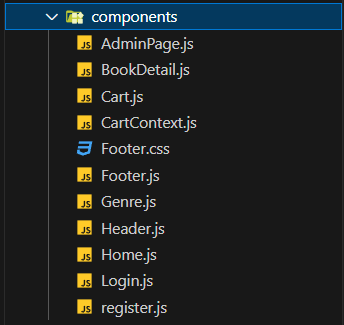
+ Destructuring giúp tách các giá trị từ object hoặc array dễ dàng.

+ Modules để chia mã thành các module riêng biệt.

- **Tailwind CSS**: là một framework CSS theo kiểu utility-first, cho phép xây dựng giao diện người dùng trực tiếp bằng cách kết hợp các lớp tiện ích (utility classes) thay vì viết CSS tùy chỉnh. Nó giúp giảm bớt việc viết mã CSS thủ công và tăng tốc độ phát triển, đặc biệt khi thiết kế giao diện có tính linh hoạt và thay đổi nhanh chóng. Tailwind cung cấp một loạt các lớp đã được định nghĩa sẵn, giúp tạo ra các thiết kế đẹp mắt mà không cần phải lo lắng nhiều về chi tiết CSS.

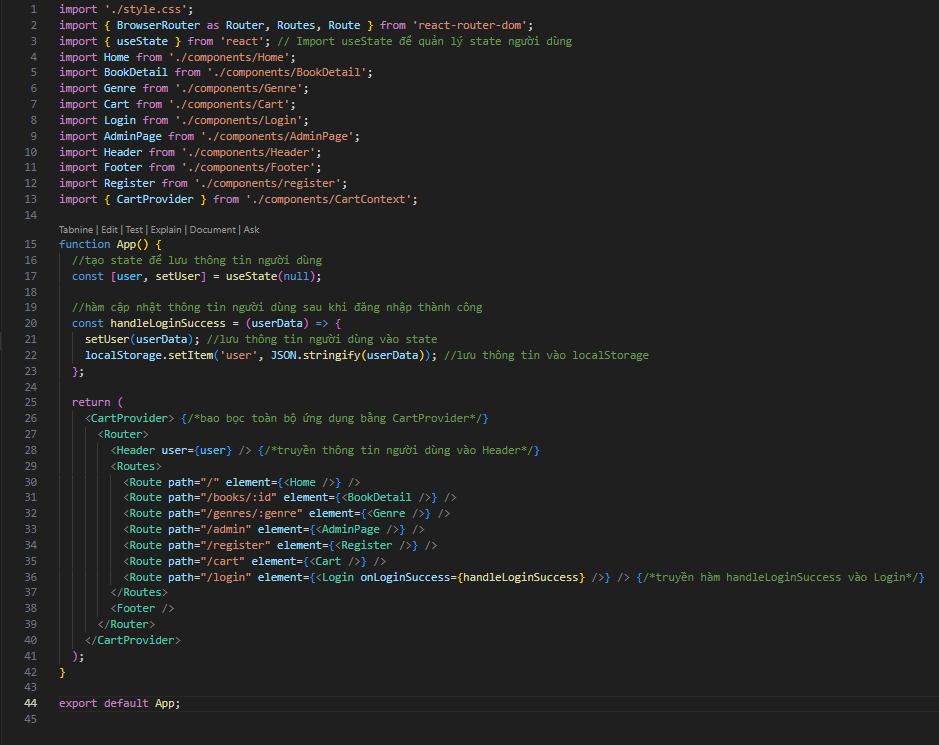
**- FetchAPI**: là một API trong JavaScript giúp thực hiện các yêu cầu HTTP (GET, POST, PUT, DELETE, v.v.) từ trình duyệt. Fetch cung cấp cú pháp dựa trên Promise, giúp xử lý bất đồng bộ dễ dàng hơn so với các phương thức cũ như XMLHttpRequest. Với Fetch, bạn có thể lấy dữ liệu từ các API và xử lý kết quả trả về một cách linh hoạt, dễ đọc. Nó thay thế các thư viện như jQuery để xử lý các yêu cầu HTTP trong các ứng dụng web hiện đại.

## **2. Components**

Danh sách các components của hệ thống:  
****

## **3. App.js**

là điểm khởi đầu của ứng dụng React, nơi bạn định nghĩa cấu trúc và các tuyến đường (routes) trong ứng dụng. Đây là nơi các component quan trọng được kết nối lại với nhau và giúp xây dựng trải nghiệm người dùng.

****

**- Import các thư viện và components:**

import './style.css';

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

import { useState } from 'react'; // Import useState để quản lý state người dùng

import Home from './components/Home';

import BookDetail from './components/BookDetail';

import Genre from './components/Genre';

import Cart from './components/Cart';

import Login from './components/Login';

import AdminPage from './components/AdminPage';

import Header from './components/Header';

import Footer from './components/Footer';

import Register from './components/register';

import { CartProvider } from './components/CartContext';

**- Các hàm xử lí thao tác của người dùng:**

const [user, setUser] = useState(null);

const handleLoginSuccess = (userData) => {

setUser(userData);

localStorage.setItem('user', JSON.stringify(userData));

};  
**Cấu trúc của ứng dụng (Routing)**:return (

<CartProvider> {/\*bao bọc toàn bộ ứng dụng bằng CartProvider\*/}

<Router>

<Header user={user} /> {/\*truyền thông tin người dùng vào Header\*/}

<Routes>

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

<Route path="/books/:id" element={<BookDetail />} />

<Route path="/genres/:genre" element={<Genre />} />

<Route path="/admin" element={<AdminPage />} />

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

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

<Route path="/login" element={<Login onLoginSuccess={handleLoginSuccess} />} /> {/\*truyền hàm handleLoginSuccess vào Login\*/}

</Routes>

<Footer />

</Router>

</CartProvider>

);

- <CartProvider>: Bọc toàn bộ ứng dụng bằng context CartProvider. Điều này cho phép các component trong ứng dụng có thể truy cập và cập nhật thông tin giỏ hàng.

- <Router>: Đây là thành phần chính của React Router, chịu trách nhiệm cung cấp các khả năng điều hướng cho ứng dụng. Tất cả các route (tuyến đường) sẽ nằm trong <Router>.

- <Header user={user} />: Component Header nhận thông tin người dùng từ state user để có thể hiển thị tên người dùng hoặc thông tin liên quan.

- <Routes> và <Route>: Đây là cấu trúc điều hướng chính trong ứng dụng:

- <Route path="/" element={<Home />} />: Điều hướng người dùng đến trang chủ khi họ truy cập URL /.

- <Route path="/books/:id" element={<BookDetail />} />: Điều hướng đến trang chi tiết sách, nơi :id là tham số động (ID của sách).

- <Route path="/x" element={<y />} />: Điều hướng đến trang y bằng đường dẫn localhost:3000/x

**4. Package.json**

package.json là một file cấu hình trong các dự án Node.js và JavaScript. Nó giúp xác định các thông tin quan trọng liên quan đến dự án, bao gồm:

* Tên dự án
* Phiên bản của dự án
* Các phụ thuộc mà dự án yêu cầu để chạy
* Các tập lệnh (scripts) để chạy dự án hoặc kiểm tra mã nguồn
* Cấu hình công cụ như ESLint, Webpack, Babel, v.v.

Đây là một trong những file quan trọng nhất của bất kỳ dự án Node.js nào và đóng vai trò như là "trái tim" của việc quản lý các phụ thuộc (dependencies), cấu hình của dự án và các tập lệnh (scripts).



## **5. Giao diện của Website**

### 5.1 Header

book-store\src\components\Header.js



- Giỏ hàng

+ Hiển thị số lượng sản phẩm trong giỏ hàng: Mỗi khi có thay đổi trong giỏ hàng (cart), cartItemCount sẽ được tính toán lại bằng cách giảm dần số lượng sản phẩm trong giỏ. Nếu cartItemCount > 0, một badge sẽ hiển thị số lượng sản phẩm trong giỏ hàng

+ Chuyển hướng đến trang giỏ hàng: Khi người dùng nhấp vào biểu tượng giỏ hàng, họ sẽ được chuyển đến trang giỏ hàng (/cart).

**- Tìm kiếm sản phẩm:** Người dùng có thể nhập từ khóa tìm kiếm vào ô input để tìm sản phẩm.

- **Quản lý tài khoản người dùng: Hiển thị thông tin tài khoản (nếu người dùng đã đăng nhập):** Nếu có thông tin người dùng trong localStorage, tên người dùng (username) sẽ được hiển thị trên giao diện, kèm theo biểu tượng tài khoản.

**- Mở dropdown menu khi nhấp vào tên người dùng:** Khi người dùng nhấp vào tên của họ, một dropdown sẽ xuất hiện, hiển thị các tùy chọn như:

**Thông tin tài khoản:** Liên kết đến trang thông tin tài khoản (/account).

**Đăng xuất:** Cung cấp tùy chọn để đăng xuất khỏi tài khoản. Khi nhấn vào "Đăng xuất", thông tin người dùng sẽ bị xóa khỏi localStorage và người dùng sẽ được điều hướng về trang chủ (/).

**- Điều hướng:**

**Trang chủ:** Người dùng có thể nhấn vào logo để quay lại trang chủ.

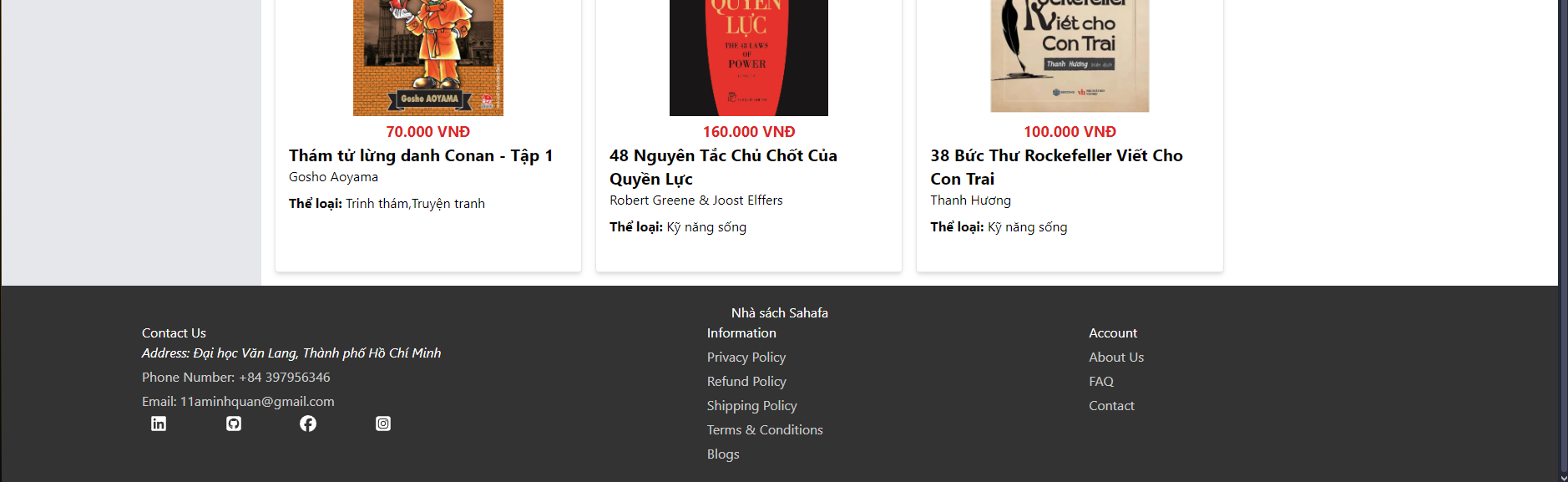
**- Dropdown menu:** Khi người dùng nhấn vào tên tài khoản, dropdown sẽ bật lên (hoặc ẩn đi) tùy theo trạng thái isDropdownOpen.

**- Trạng thái không đăng nhập:** Nếu không có người dùng trong localStorage, thay vì hiển thị tên tài khoản, sẽ có một liên kết Tài khoản dẫn đến trang đăng nhập (/login).

**- Tình huống không có giỏ hàng:** Nếu giỏ hàng không có sản phẩm, phần badge với số lượng sản phẩm sẽ không được hiển thị.

### 5.2 Footer

book-store\src\components\Footer.js

****

Thông tin liên hệ (Contact Us):

* Phone Number: Khi người dùng click vào số điện thoại (+84 397956346), trình duyệt sẽ mở ứng dụng gọi điện (hoặc trình gọi điện của thiết bị di động nếu người dùng sử dụng điện thoại).
* Email: Khi người dùng click vào địa chỉ email (11aminhquan@gmail.com), trình duyệt sẽ mở ứng dụng email mặc định (như Gmail hoặc Outlook) để soạn email mới gửi tới địa chỉ đó.
* Social Media Icons:
  + LinkedIn: Người dùng click vào biểu tượng LinkedIn sẽ được chuyển hướng tới trang LinkedIn của Nhà sách Sahafa (hoặc trang mạng xã hội liên quan).
  + GitHub: Người dùng click vào biểu tượng GitHub sẽ được chuyển hướng tới trang GitHub của Nhà sách Sahafa.
  + Facebook: Người dùng click vào biểu tượng Facebook sẽ được chuyển hướng tới trang Facebook của Nhà sách Sahafa.
  + Instagram: Người dùng click vào biểu tượng Instagram sẽ được chuyển hướng tới trang Instagram của Nhà sách Sahafa.

Thông tin (Information):

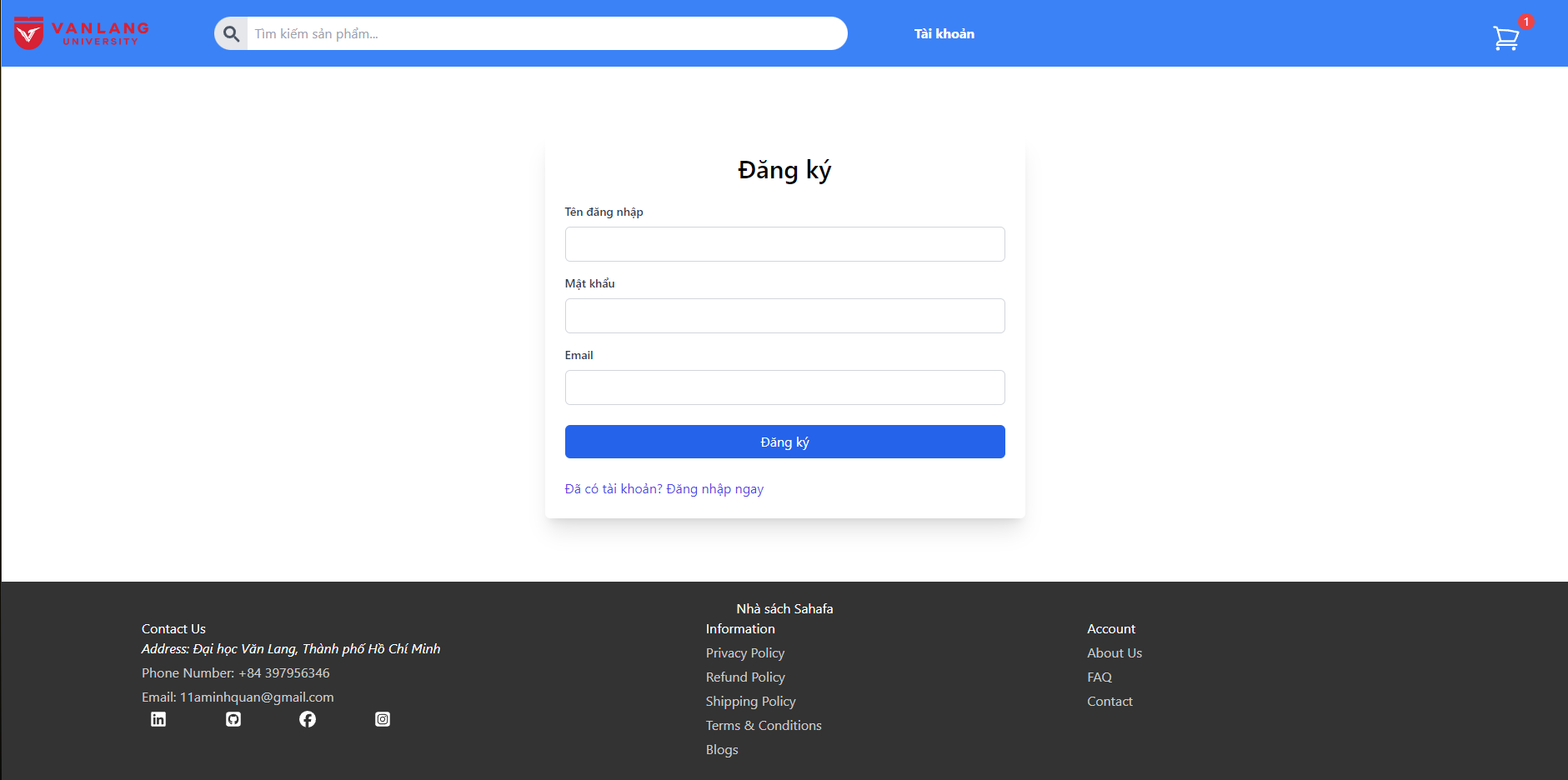
* Privacy Policy: Khi người dùng click vào liên kết "Privacy Policy", họ sẽ được chuyển hướng tới một trang liên quan đến Chính sách bảo mật của Nhà sách Sahafa (có thể là một trang mới hoặc một đoạn mô tả trong cùng một trang).
* Refund Policy: Khi người dùng click vào liên kết "Refund Policy", họ sẽ được chuyển hướng tới trang chính sách hoàn trả sản phẩm.
* Shipping Policy: Người dùng click vào "Shipping Policy" sẽ được chuyển hướng tới trang chính sách vận chuyển.
* Terms & Conditions: Người dùng click vào "Terms & Conditions" sẽ được chuyển hướng tới trang điều khoản và điều kiện.
* Blogs: Người dùng click vào "Blogs" sẽ được chuyển hướng tới trang hoặc bài viết blog của Nhà sách Sahafa.

Tài khoản (Account):

* About Us: Khi người dùng click vào liên kết "About Us", họ sẽ được chuyển hướng tới trang giới thiệu về Nhà sách Sahafa.
* FAQ: Khi người dùng click vào "FAQ", họ sẽ được chuyển hướng tới trang Câu hỏi thường gặp (FAQ) để tìm kiếm các câu hỏi và câu trả lời liên quan đến dịch vụ của Nhà sách Sahafa.
* Contact: Người dùng click vào liên kết "Contact" sẽ được chuyển hướng đến trang liên hệ để gửi yêu cầu hoặc câu hỏi.

### 5.3 Trang đăng ký

book-store\src\components\register.js

****

Điền thông tin vào các trường nhập liệu (input fields):

* Tên đăng nhập (username): Người dùng nhập tên đăng nhập vào ô "Tên đăng nhập". Dữ liệu nhập vào sẽ được cập nhật trong formData.username thông qua sự kiện handleChange.
* Mật khẩu (password): Người dùng nhập mật khẩu vào ô "Mật khẩu". Dữ liệu nhập vào sẽ được cập nhật trong formData.password.
* Email: Người dùng nhập email vào ô "Email". Dữ liệu nhập vào sẽ được cập nhật trong formData.email.

Nhấn nút "Đăng ký":

* Khi người dùng nhấn nút "Đăng ký", form sẽ gửi yêu cầu đăng ký tới API thông qua phương thức fetch. Dữ liệu từ form (bao gồm username, password, và email) sẽ được gửi dưới dạng JSON.
* API sẽ phản hồi với kết quả đăng ký. Tùy vào kết quả này, có thể có các tình huống sau:
  + Đăng ký thành công: Nếu API trả về { success: true }, thông báo thành công sẽ được hiển thị trên giao diện. Sau 1 giây, người dùng sẽ được điều hướng đến trang đăng nhập.
  + Tài khoản đã tồn tại: Nếu API trả về thông điệp "Tài khoản đã tồn tại", lỗi này sẽ được hiển thị trên giao diện dưới dạng thông báo lỗi.
  + Đăng ký thất bại: Nếu API trả về một thông điệp lỗi khác (hoặc không trả về thông tin success), lỗi sẽ được hiển thị dưới dạng thông báo lỗi trên giao diện.

Hiển thị thông báo lỗi và thông báo thành công:

* Nếu có lỗi xảy ra trong quá trình gửi form (ví dụ: tài khoản đã tồn tại hoặc lỗi mạng), thông báo lỗi sẽ được hiển thị với nội dung tương ứng.
* Nếu đăng ký thành công, một thông báo thành công sẽ được hiển thị trên giao diện.

Điều hướng người dùng:

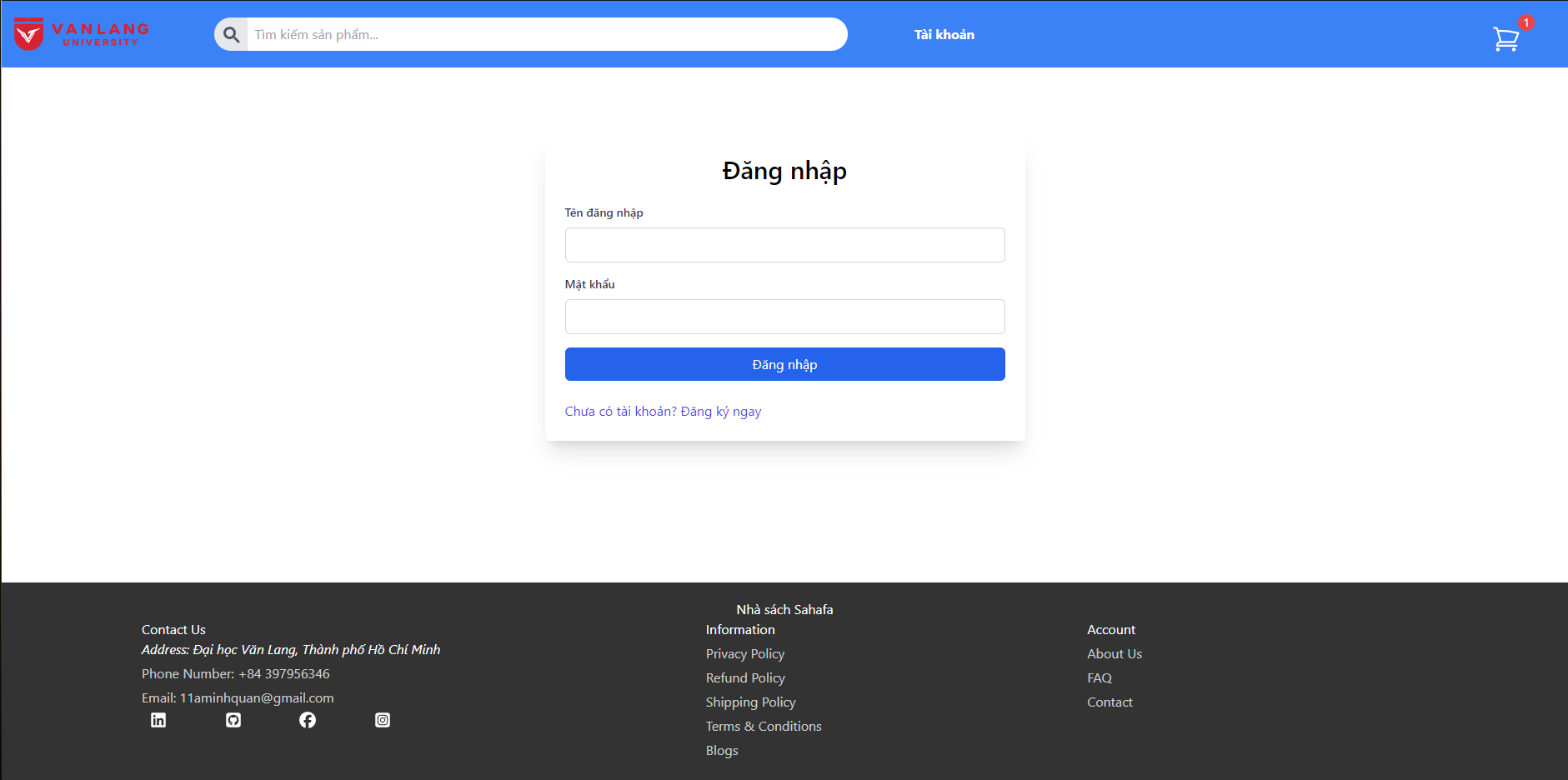
* Điều hướng đến trang đăng nhập: Nếu đăng ký thành công (dữ liệu trả về từ API có success: true), sau 1 giây, người dùng sẽ được điều hướng đến trang đăng nhập (/login).
* Điều hướng đến trang đăng nhập khi người dùng đã có tài khoản: Người dùng có thể nhấn vào nút "Đã có tài khoản? Đăng nhập ngay" để điều hướng đến trang đăng nhập.

Xử lý lỗi trong quá trình đăng ký:

* Nếu có lỗi xảy ra khi gửi yêu cầu API (ví dụ, lỗi mạng hoặc server không phản hồi), một thông báo lỗi tổng quát ("Đã xảy ra lỗi khi đăng ký") sẽ được hiển thị.

## 5.4 Trang đăng nhập

book-store\src\components\Login.js

****

**Điền thông tin vào các trường nhập liệu (input fields):**

* **Tên đăng nhập (username):** Người dùng nhập tên đăng nhập vào ô "Tên đăng nhập". Dữ liệu nhập vào sẽ được lưu trong state username thông qua setUsername.
* **Mật khẩu (password):** Người dùng nhập mật khẩu vào ô "Mật khẩu". Dữ liệu nhập vào sẽ được lưu trong state password thông qua setPassword.

**Nhấn nút "Đăng nhập":**

* Khi người dùng nhấn nút "Đăng nhập", form sẽ ngừng hành động mặc định (ngừng việc tải lại trang) thông qua e.preventDefault().
* Nếu người dùng chưa nhập đủ thông tin (thiếu tên đăng nhập hoặc mật khẩu), thông báo lỗi "Vui lòng điền đầy đủ thông tin" sẽ được hiển thị.
* Nếu người dùng nhập đầy đủ thông tin, một yêu cầu đăng nhập sẽ được gửi đến API (http://localhost/book-store-backend/api/login.php) với phương thức POST. Dữ liệu đăng nhập (username và password) sẽ được gửi dưới dạng JSON.

**Nhận và xử lý phản hồi từ API:**

* **Đăng nhập thành công:**
  + Nếu API trả về { success: true }, đăng nhập sẽ thành công.
  + Thông tin người dùng (dự kiến có các trường như id, username, role, ...) sẽ được lưu vào localStorage (dưới dạng JSON) để có thể sử dụng lại trong các trang khác.
  + Hàm onLoginSuccess sẽ được gọi, truyền thông tin người dùng lên component cha (nếu có), để cập nhật thông tin người dùng trong ứng dụng.
  + Kiểm tra quyền hạn của người dùng qua data.user.role:
    - Nếu người dùng có vai trò là admin, người dùng sẽ được điều hướng đến trang /admin.
    - Nếu người dùng có vai trò khác, họ sẽ được điều hướng đến trang chủ (/).
* **Đăng nhập thất bại (Sai tài khoản hoặc mật khẩu):**
  + Nếu API trả về { success: false }, thông báo lỗi "Sai thông tin tài khoản hoặc mật khẩu" sẽ được hiển thị.
* **Lỗi API (ví dụ, lỗi mạng hoặc server không phản hồi):**
  + Nếu có lỗi trong quá trình gọi API, lỗi này sẽ được hiển thị trên console, nhưng không có thông báo lỗi hiển thị trên giao diện người dùng.

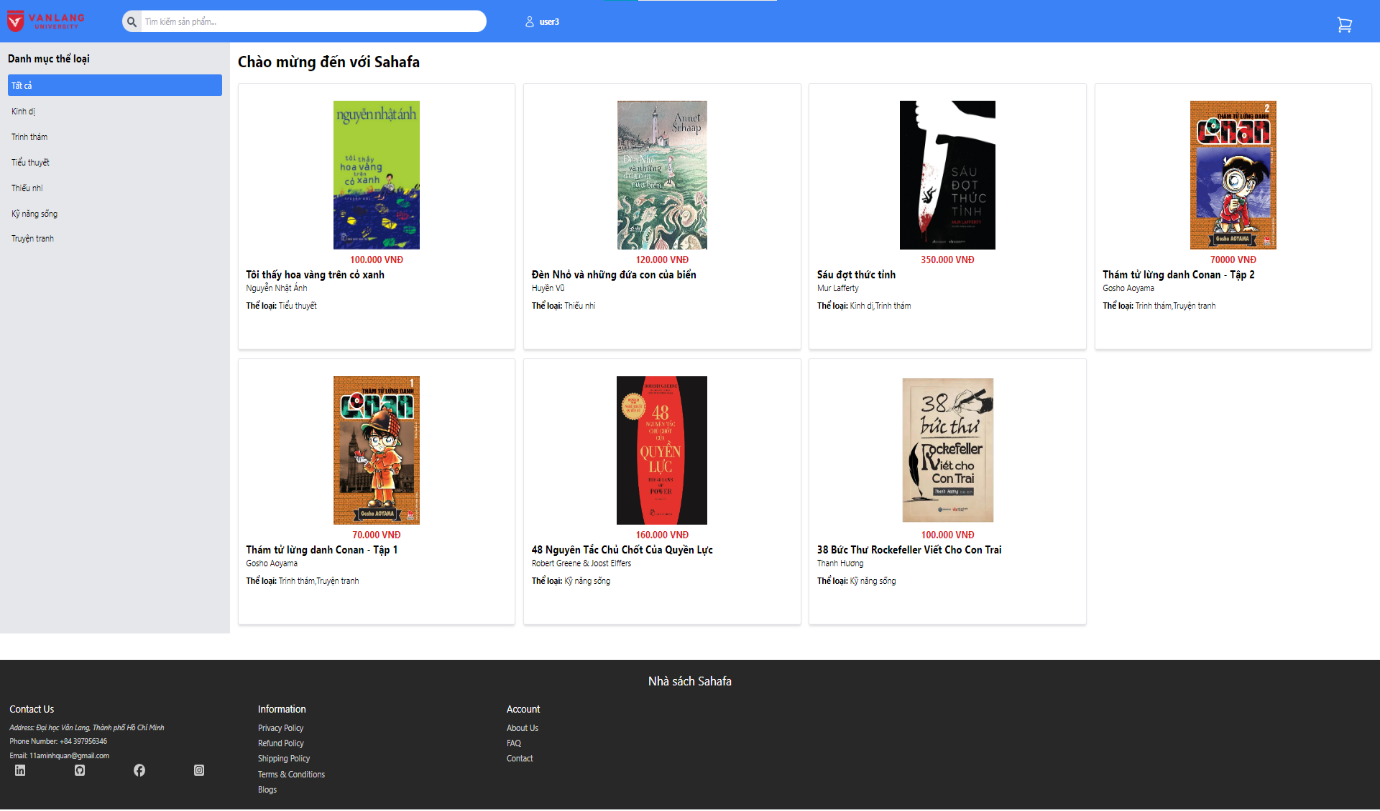
**Hiển thị thông báo lỗi:**

* Nếu có lỗi xảy ra trong quá trình đăng nhập (như sai tên đăng nhập hoặc mật khẩu), thông báo lỗi sẽ được hiển thị dưới dạng văn bản màu đỏ ở dưới các trường nhập liệu.
* Lỗi sẽ được cập nhật trong state error và hiển thị trong giao diện người dùng.

**Điều hướng người dùng:**

* **Chuyển hướng đến trang đăng ký:**
  + Nếu người dùng chưa có tài khoản, họ có thể nhấn vào nút "Chưa có tài khoản? Đăng ký ngay" để điều hướng đến trang đăng ký (/register).
* **Chuyển hướng sau khi đăng nhập thành công:**
  + Sau khi đăng nhập thành công và kiểm tra vai trò người dùng, người dùng sẽ được điều hướng đến trang tương ứng (trang admin nếu là quản trị viên, hoặc trang chủ nếu là người dùng thông thường).

## 5.5 Trang chủ

**** book-store\src\components\Home.js

**Hiển thị danh sách thể loại sách (genres)**

* **Thao tác:**
  + **Mở ứng dụng:** Khi người dùng mở trang Home, component sẽ tự động gọi API để lấy danh sách các thể loại sách từ http://localhost/book-store-backend/api/genre.php. Dữ liệu trả về từ API sẽ được lưu vào state genres.
  + **Chọn thể loại:** Người dùng có thể chọn một thể loại sách từ sidebar (cột bên trái). Mỗi thể loại sẽ có một button tương ứng trong danh sách. Khi người dùng nhấn vào một thể loại, giá trị selectedGenre sẽ được cập nhật, và trang sẽ hiển thị các sách thuộc thể loại đó.
  + **Chọn "Tất cả":** Khi người dùng chọn nút "Tất cả", selectedGenre sẽ được đặt thành chuỗi rỗng (''), làm cho tất cả sách được hiển thị, không phân loại theo thể loại.
* **Tác động:**
  + Sau khi người dùng chọn thể loại, useEffect sẽ được kích hoạt lại và gọi API với tham số thể loại đã chọn. Nếu không có thể loại nào được chọn, API sẽ gọi để lấy tất cả sách.
  + Nếu dữ liệu trả về từ API có danh sách thể loại sách, các thể loại này sẽ được hiển thị trong sidebar.

**Hiển thị danh sách sách**

* **Thao tác:**
  + Sau khi danh sách thể loại được tải thành công, các sách sẽ được hiển thị theo thể loại mà người dùng đã chọn.
  + Mỗi sách sẽ có hình ảnh, tên sách, tác giả, giá tiền và thể loại. Người dùng có thể nhấp vào bất kỳ sách nào để chuyển đến trang chi tiết của sách đó.
  + **Không có sách:** Nếu không có sách nào trong thể loại đã chọn, sẽ có thông báo "Không có sách nào" hiển thị.
* **Tác động:**
  + Khi người dùng nhấn vào một cuốn sách, trang sẽ chuyển hướng đến đường dẫn /books/${book.id}, nơi book.id là ID của cuốn sách được chọn. Điều này sẽ dẫn đến trang chi tiết của sách đó.

**Tương tác với sidebar (Danh mục thể loại)**

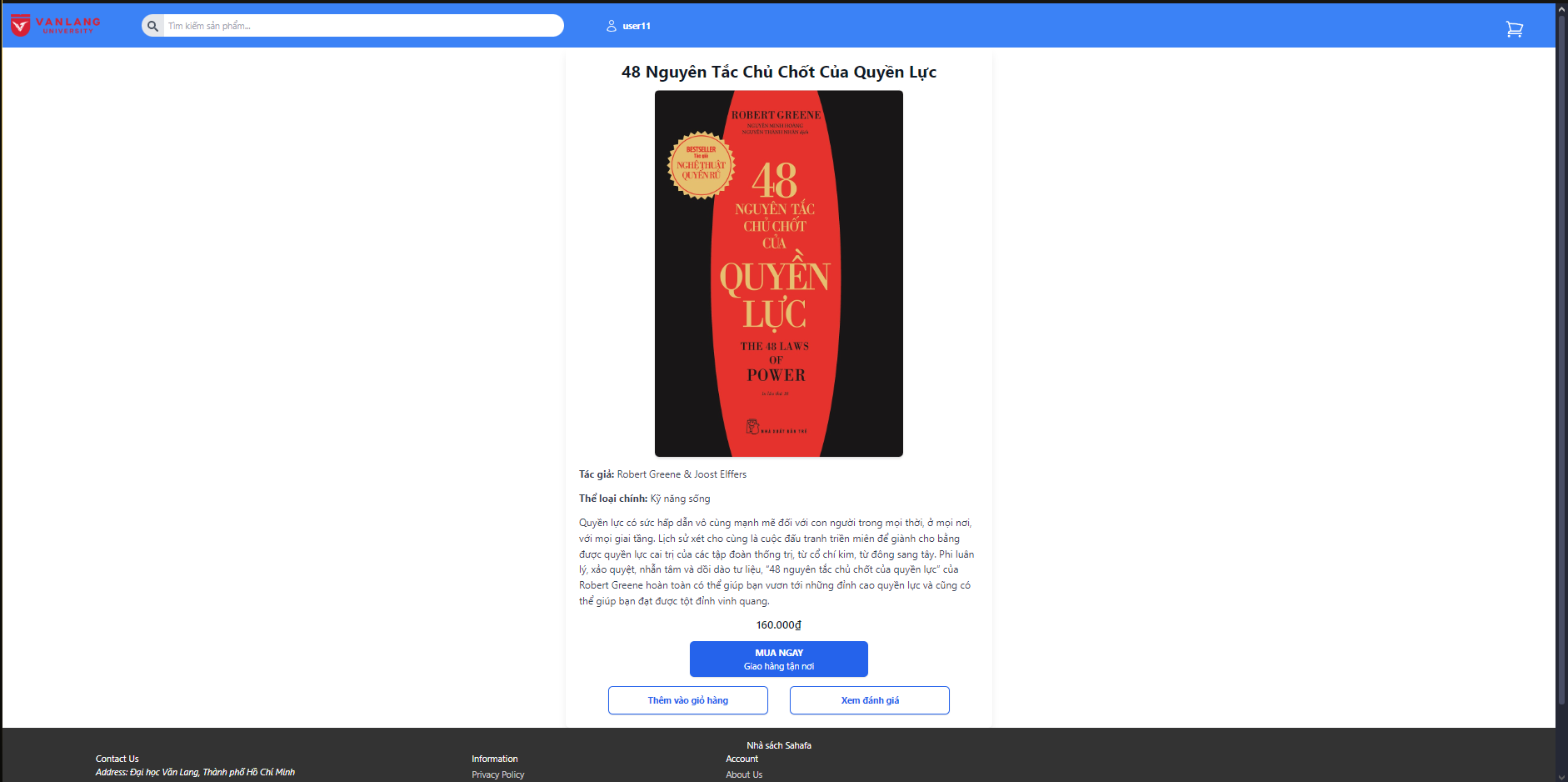
* **Thao tác:**
  + **Chọn thể loại:** Người dùng có thể chọn bất kỳ thể loại nào trong danh sách sidebar. Sau khi chọn, API sẽ được gọi lại để tải danh sách sách của thể loại đó.
  + **Thao tác hover và chọn:** Khi người dùng di chuột qua các thể loại sách, các nút thể loại sẽ thay đổi màu sắc (hover:bg-blue-200). Khi người dùng chọn thể loại, nút sẽ có nền màu xanh (bg-blue-500) và chữ trắng.
  + **Chọn "Tất cả":** Khi người dùng chọn "Tất cả", ứng dụng sẽ không lọc sách theo thể loại nào, và tất cả sách sẽ được hiển thị.
* **Tác động:**
  + Khi người dùng thay đổi thể loại (chọn thể loại khác hoặc "Tất cả"), selectedGenre sẽ được cập nhật, kích hoạt một yêu cầu mới đến API để lấy danh sách sách phù hợp.

**Xử lý dữ liệu từ API**

* **Thao tác:**
  + **Lấy danh sách thể loại:** Khi ứng dụng lần đầu tiên tải, API sẽ được gọi để lấy danh sách thể loại sách. Nếu có lỗi xảy ra khi gọi API, lỗi sẽ được ghi vào console (console.error('Error:', error)).
  + **Lấy danh sách sách theo thể loại:** Mỗi khi selectedGenre thay đổi, một yêu cầu API sẽ được gửi đi để lấy danh sách sách của thể loại đã chọn. Nếu không có thể loại được chọn, ứng dụng sẽ hiển thị tất cả sách.
  + **Xử lý lỗi:** Nếu có lỗi trong quá trình gọi API (ví dụ: không kết nối được với server hoặc API gặp sự cố), lỗi sẽ được ghi vào console, và danh sách sách sẽ không được cập nhật.

### **5**.****6 Trang thông tin chi tiết sách

book-store\src\components\BookDetail.js

****

**Trạng thái ban đầu**

* **Khi tải trang chi tiết sách:**
  + **Lấy dữ liệu** từ API (bằng useEffect) sử dụng ID sách từ URL.
  + **Trạng thái loading** sẽ được hiển thị trong khi chờ dữ liệu trả về từ API.
  + **Lỗi (error)** sẽ được xử lý nếu API không trả về kết quả hoặc có vấn đề trong quá trình tải.

Trạng thái ban đầu này bao gồm:

* **Loading:** Nếu dữ liệu vẫn đang được tải, người dùng sẽ thấy thông báo "Loading...".
* **Lỗi:** Nếu có lỗi trong quá trình tải, người dùng sẽ thấy thông báo lỗi (e.g., "Error: Book not found").
* **Không tìm thấy sách:** Nếu không có sách phù hợp với ID, người dùng sẽ thấy thông báo "Book not found."

**Hiển thị chi tiết sách**

Khi dữ liệu sách được tải thành công:

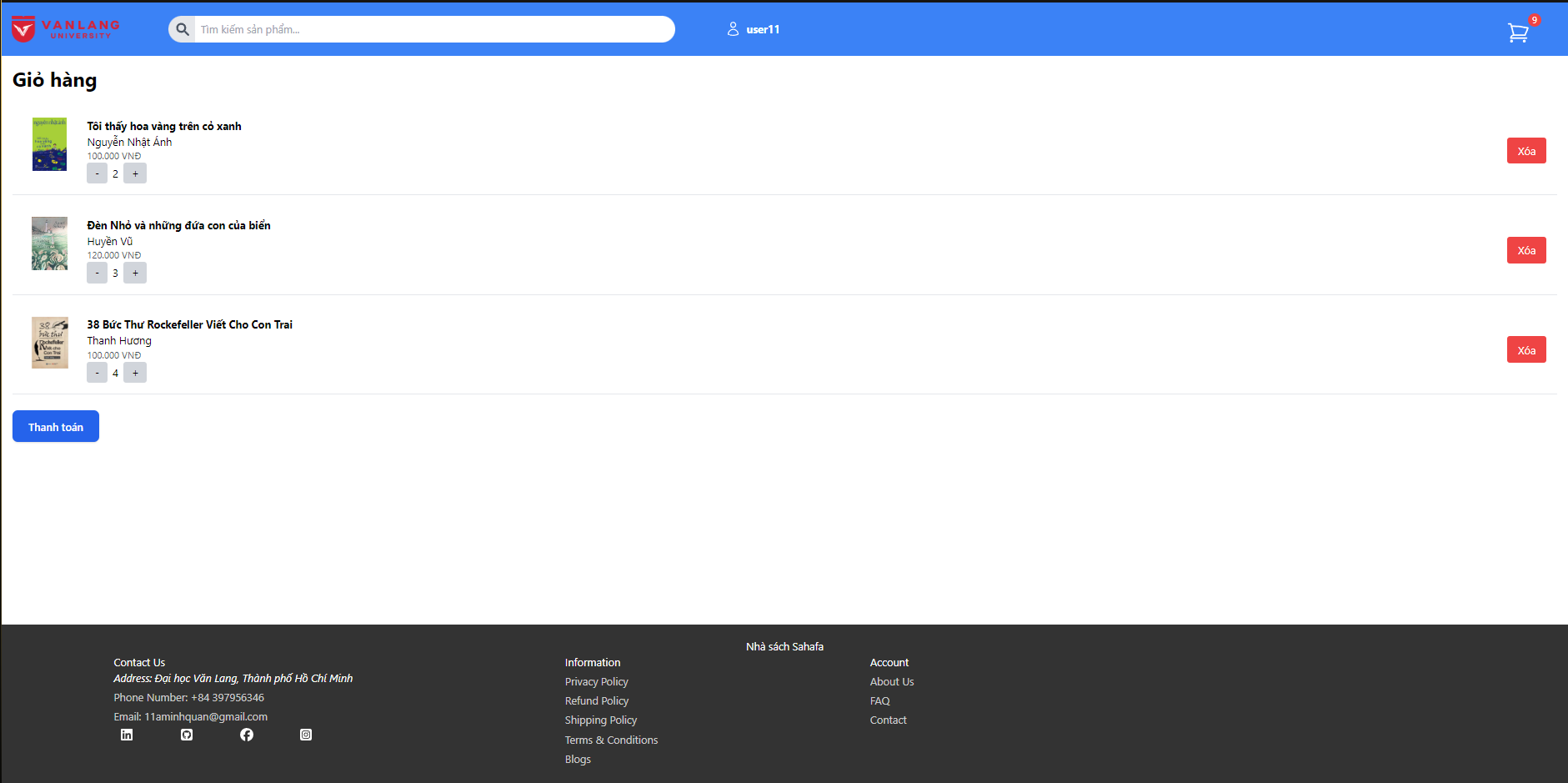
* **Hiển thị tên sách**: Tên sách sẽ được hiển thị trên đầu trang với kích thước lớn.
* **Hình ảnh sách:** Ảnh của sách sẽ được hiển thị ở giữa, và nếu có lỗi tải ảnh, sẽ có một lỗi console xuất hiện.
* **Thông tin sách:** Bao gồm tác giả, thể loại chính, và mô tả chi tiết về sách.
* **Giá sách:** Giá sách sẽ được hiển thị với định dạng tiền tệ Việt Nam (₫).

**Thao tác của người dùng**

* **Nhấn nút "Mua ngay":**
  + Nút này là để đặt hàng sách. Chuyển hướng đến trang thanh toán và đặt hàng
* **Thêm vào giỏ hàng:**
  + Nút "Thêm vào giỏ hàng" sử dụng hàm addToCart từ CartContext để thêm sách vào giỏ hàng.
  + Khi người dùng nhấn nút này, sẽ có một thông báo (hoặc hành động) cho biết sách đã được thêm vào giỏ hàng.
* **Xem đánh giá:**
  + Nút "Xem đánh giá" bật modal hiển thị các đánh giá của sách.

### 5.7 Trang giỏ hàng

book-store\src\components\Cart.js

****

**Trạng thái ban đầu**

* **Khi giỏ hàng trống:**
  + Nếu giỏ hàng (cart) không có bất kỳ sản phẩm nào, người dùng sẽ thấy thông báo "Giỏ hàng của bạn hiện đang trống!".
* **Khi giỏ hàng có sản phẩm:**
  + Nếu giỏ hàng có sản phẩm, mỗi sản phẩm sẽ được hiển thị trong một danh sách, gồm các thông tin như tên sách, tác giả, hình ảnh, giá, số lượng, và nút thao tác (thêm, bớt số lượng và xóa sản phẩm).

**Danh sách sản phẩm trong giỏ hàng**

* **Thông tin sản phẩm:**
  + **Hình ảnh sách**: Được hiển thị với kích thước cố định (w-20 h-20), dùng để giúp người dùng nhận diện nhanh sản phẩm.
  + **Tên sách**: Được hiển thị dưới dạng **font-bold**.
  + **Tác giả**: Được hiển thị ngay dưới tên sách.
  + **Giá sách:** Được hiển thị với định dạng "VNĐ" và kích thước chữ nhỏ (text-sm text-gray-600).
* **Số lượng sách:**
  + **Tăng giảm số lượng**: Người dùng có thể thay đổi số lượng sản phẩm trong giỏ hàng bằng cách nhấn vào nút "-" hoặc "+". Sau khi nhấn, số lượng sẽ được cập nhật.
  + **Thao tác tăng số lượng**: Nhấn vào nút "+" sẽ gọi hàm increaseQuantity(item.title) để tăng số lượng sản phẩm.
  + **Thao tác giảm số lượng**: Nhấn vào nút "-" sẽ gọi hàm decreaseQuantity(item.title) để giảm số lượng sản phẩm.
  + **Hiển thị số lượng**: Số lượng sản phẩm hiện tại sẽ được hiển thị giữa hai nút "+" và "-".
* **Xóa sản phẩm khỏi giỏ hàng:**
  + **Xóa sản phẩm**: Có một nút "Xóa" màu đỏ bên cạnh mỗi sản phẩm. Khi nhấn vào nút này, hàm removeFromCart(item.title) sẽ được gọi để xóa sản phẩm khỏi giỏ hàng.

**Thanh toán**

* **Nút thanh toán:**
  + Nếu giỏ hàng có sản phẩm, người dùng có thể nhấn vào nút "Thanh toán". Khi nhấn, hàm handleCheckout sẽ được gọi.
  + Trong phiên bản hiện tại, hàm handleCheckout chỉ hiển thị một thông báo "Thanh toán thành công!"

**Các thao tác của người dùng**

* **Khi giỏ hàng trống:**
  + Người dùng sẽ thấy thông báo "Giỏ hàng của bạn hiện đang trống!". Trong trường hợp này, họ không thể thực hiện thanh toán hoặc các thao tác khác.
* **Khi giỏ hàng có sản phẩm:**
  + **Thêm số lượng sản phẩm:** Khi người dùng nhấn vào nút "+", số lượng sản phẩm sẽ tăng lên và giỏ hàng sẽ được cập nhật (dựa trên increaseQuantity).
  + **Giảm số lượng sản phẩm**: Khi người dùng nhấn vào nút "-", số lượng sản phẩm sẽ giảm đi và giỏ hàng sẽ được cập nhật (dựa trên decreaseQuantity).
  + **Xóa sản phẩm**: Khi người dùng nhấn nút "Xóa", sản phẩm sẽ bị xóa khỏi giỏ hàng, và giỏ hàng sẽ được cập nhật (dựa trên removeFromCart).
  + **Thanh toán**: Khi người dùng nhấn vào nút "Thanh toán", hệ thống sẽ hiển thị một thông báo "Thanh toán thành công!".

# **III. Back-end**

## **1. Database**

$servername: Địa chỉ máy chủ cơ sở dữ liệu. Trong trường hợp này, giá trị là "localhost", tức là cơ sở dữ liệu đang chạy trên cùng một máy chủ với ứng dụng PHP.

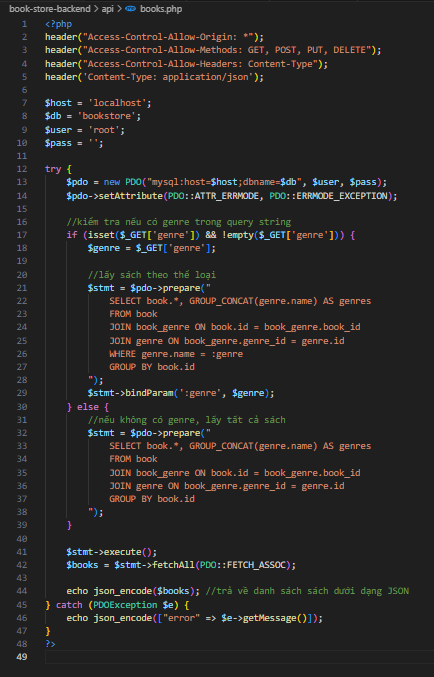
$username: Tên người dùng để kết nối với cơ sở dữ liệu MySQL. Trong ví dụ này, người dùng là "root", đây là tài khoản mặc định trong nhiều môi trường phát triển.

$password: Mật khẩu của người dùng MySQL. Trong ví dụ này, mật khẩu là một chuỗi rỗng (""), có nghĩa là không có mật khẩu.

$dbname: Tên cơ sở dữ liệu cần kết nối. Trong ví dụ này, cơ sở dữ liệu có tên là "bookstore", đây là cơ sở dữ liệu chứa thông tin về các cuốn sách.

## **2. books.php**

book-store-backend\api\books.php



- Lấy Dữ Liệu Sách Từ Cơ Sở Dữ Liệu: Đây là API giúp frontend có thể lấy danh sách sách từ cơ sở dữ liệu, có thể lọc theo thể loại hoặc lấy tất cả sách nếu không có tham số thể loại. Điều này rất hữu ích trong việc xây dựng giao diện tìm kiếm, phân loại sách trên website bán sách.

- Cung Cấp Dữ Liệu Dưới Dạng JSON: Việc trả về dữ liệu dưới dạng JSON giúp các ứng dụng frontend (như ReactJS, VueJS, Angular) dễ dàng xử lý và hiển thị thông tin sách.

- Tính Linh Hoạt với Tham Số genre: API này rất linh hoạt, có thể trả về tất cả sách hoặc chỉ các sách thuộc thể loại cụ thể, dựa trên tham số genre trong URL. Điều này giúp người dùng có thể tìm kiếm các sách theo thể loại một cách hiệu quả.

- Hỗ Trợ Tích Hợp CORS: Tiêu đề Access-Control-Allow-Origin: \* cho phép hệ thống này hỗ trợ các ứng dụng từ các miền khác nhau, giúp API có thể được truy cập từ bất kỳ đâu, từ web hoặc ứng dụng di động.

## **3. bookdetail.php**

book-store-backend\api\bookdetail.php

API Truy Cập Dữ Liệu Sách:

* Đây là một API giúp hệ thống frontendc ó thể truy vấn thông tin chi tiết về sách từ cơ sở dữ liệu.
* Khi người dùng yêu cầu xem thông tin chi tiết của một cuốn sách, hệ thống frontend gửi yêu cầu GET với tham số id tới API này. API sau đó sẽ trả về thông tin chi tiết của sách dưới dạng JSON.

Ứng Dụng Cơ Sở Dữ Liệu:

* API này giúp hệ thống quản lý sách dễ dàng truy xuất thông tin sách từ cơ sở dữ liệu, bao gồm các thông tin liên quan như tên sách, tác giả, giá, thể loại, mô tả và hình ảnh.
* Việc sử dụng LEFT JOIN cho phép lấy thông tin sách ngay cả khi không có thể loại được gán cho sách, giúp hệ thống linh hoạt hơn khi quản lý dữ liệu sách.

Hỗ Trợ Frontend:

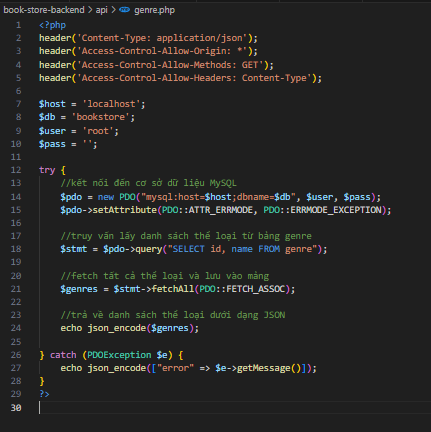
* Hệ thống frontend có thể gửi yêu cầu GET tới API này với ID của sách và nhận lại dữ liệu cần thiết để hiển thị cho người dùng. Điều này giúp tách biệt việc xử lý logic từ frontend và backend, tạo nên một cấu trúc dễ bảo trì và mở rộng.

Tiện Ích Cho Các Tính Năng Khác:

* API này có thể được mở rộng để phục vụ cho các tính năng khác của hệ thống như tìm kiếm sách, lọc sách theo thể loại, cập nhật thông tin sách, v.v.
* Ví dụ: API có thể được mở rộng để nhận nhiều tham số hơn như author, price\_range, genre, và trả về danh sách sách phù hợp với các tiêu chí tìm kiếm.

## **4. genre.php**

book-store-backend\api\genre.php



API Lấy Danh Sách Thể Loại:

* Đoạn mã này cung cấp một API để frontend có thể yêu cầu danh sách thể loại sách từ backend. API này sẽ trả về dữ liệu dưới dạng JSON, giúp hệ thống frontend dễ dàng sử dụng để hiển thị cho người dùng.

Hỗ Trợ Tính Năng Tìm Kiếm và Lọc:

* API này được sử dụng trong tính năng tìm kiếm sách theo thể loại, lọc sách theo thể loại, hoặc tạo các danh mục thể loại sách trên trang web bán sách.
* Khi người dùng muốn tìm sách thuộc thể loại cụ thể, hệ thống frontend có thể gửi yêu cầu đến API này để lấy danh sách thể loại và sau đó hiển thị chúng dưới dạng danh sách hoặc menu cho người dùng chọn.

Cấu Trúc Dữ Liệu Linh Hoạt và Mở Rộng:

* Dữ liệu trả về có cấu trúc đơn giản với hai trường id và name cho mỗi thể loại sách. Điều này giúp hệ thống dễ dàng mở rộng hoặc thay đổi trong tương lai, như thêm các trường thông tin mới nếu cần thiết.

Phân Tách Backend và Frontend:

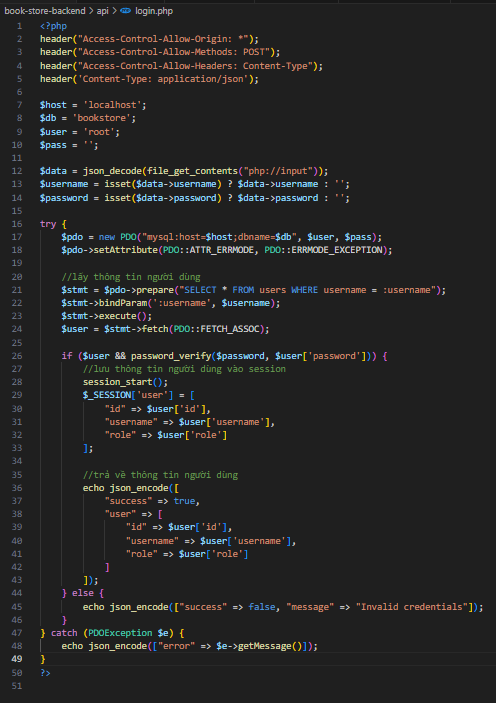
* Đoạn mã này thể hiện mô hình phát triển phân tách giữa frontend và backend. Frontend chỉ cần gửi yêu cầu GET và xử lý dữ liệu trả về dưới dạng JSON mà không cần quan tâm đến chi tiết cài đặt của cơ sở dữ liệu hoặc cách xử lý phía backend. Điều này giúp tăng tính bảo mật và dễ bảo trì.

Hỗ Trợ CORS:

* Việc sử dụng CORS cho phép API này có thể được truy cập từ các nguồn khác nhau (domain khác), điều này rất quan trọng khi frontend và backend nằm trên các máy chủ khác nhau.

## **5. login.php**

book-store-backend\api\login.php

****

API Đăng Nhập Người Dùng:

* Đoạn mã này cung cấp một API để xử lý việc đăng nhập người dùng. Frontend sẽ gửi yêu cầu POST chứa tên đăng nhập và mật khẩu, và backend sẽ xác thực người dùng thông qua cơ sở dữ liệu.
* Nếu đăng nhập thành công, backend trả về thông tin người dùng (bao gồm id, username, và role) và đánh dấu đăng nhập là thành công với "success": true.
* Nếu đăng nhập thất bại, một thông báo lỗi "Invalid credentials" sẽ được trả về.

Bảo Mật:

* Mã sử dụng password hashing để bảo mật mật khẩu người dùng. Mật khẩu được mã hóa trước khi lưu vào cơ sở dữ liệu và sau đó được xác thực bằng cách so sánh mật khẩu người dùng nhập vào với mật khẩu đã mã hóa trong cơ sở dữ liệu sử dụng password\_verify(). Điều này giúp bảo vệ mật khẩu người dùng ngay cả khi cơ sở dữ liệu bị xâm nhập.

Session Management:

* Sau khi người dùng đăng nhập thành công, thông tin người dùng sẽ được lưu vào session ($\_SESSION['user']). Điều này cho phép hệ thống lưu trạng thái đăng nhập của người dùng trong suốt phiên làm việc của họ, giúp người dùng không phải đăng nhập lại cho mỗi yêu cầu.
* Thông tin người dùng lưu trong session có thể được sử dụng để phân quyền (quản lý quyền hạn của người dùng) hoặc tùy chỉnh giao diện dựa trên vai trò người dùng (role).

Cấu Trúc RESTful API:

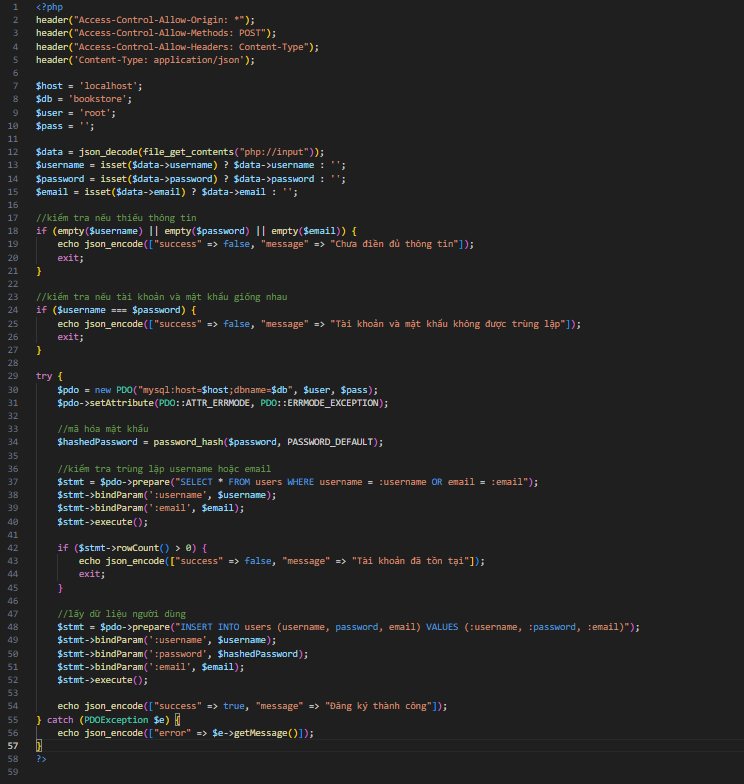
* API này được thiết kế theo kiến trúc RESTful, sử dụng phương thức POST để gửi yêu cầu đăng nhập và trả về dữ liệu dưới dạng JSON. Điều này giúp tách biệt frontend và backend, làm cho hệ thống linh hoạt và dễ bảo trì hơn.

Tiện Ích Cho Frontend:

* Frontend có thể dễ dàng sử dụng API này để thực hiện chức năng đăng nhập. Dữ liệu trả về (dưới dạng JSON) có thể được sử dụng để hiển thị thông tin người dùng hoặc điều hướng người dùng tới các trang khác sau khi đăng nhập thành công.

## **6. register.php**

book-store-backend\api\register.php

****

API Đăng Ký Người Dùng:

* Đoạn mã này cung cấp một API để người dùng đăng ký tài khoản mới. Frontend sẽ gửi yêu cầu POST chứa thông tin tên đăng nhập, mật khẩu và email, và backend sẽ xử lý và lưu trữ thông tin đó vào cơ sở dữ liệu.

Bảo Mật Mật Khẩu:

* Mã sử dụng **password hashing** để bảo mật mật khẩu người dùng trước khi lưu vào cơ sở dữ liệu. Điều này giúp bảo vệ mật khẩu ngay cả khi cơ sở dữ liệu bị xâm nhập. Quá trình xác thực trong đăng nhập (được mô tả trong đoạn mã trước đó) sẽ sử dụng phương thức password\_verify() để xác minh mật khẩu người dùng.

Kiểm Tra Dữ Liệu Đầu Vào:

* API thực hiện các kiểm tra bảo mật cơ bản như: kiểm tra thiếu thông tin, kiểm tra nếu tên đăng nhập và mật khẩu giống nhau, và kiểm tra trùng lặp tên đăng nhập và email trong cơ sở dữ liệu. Điều này giúp đảm bảo dữ liệu người dùng hợp lệ và tránh việc đăng ký tài khoản không hợp lệ.

Tránh Tình Trạng Đăng Ký Trùng Lặp:

* API kiểm tra xem tên đăng nhập và email có bị trùng lặp trong cơ sở dữ liệu không. Điều này giúp tránh việc người dùng đăng ký nhiều lần với cùng một thông tin tài khoản, đảm bảo tính duy nhất của mỗi tài khoản trong hệ thống.

Cấu Trúc RESTful API:

* API này được thiết kế theo kiến trúc **RESTful**, sử dụng phương thức POST để gửi yêu cầu đăng ký và trả về dữ liệu dưới dạng JSON. Điều này giúp tách biệt frontend và backend, giúp hệ thống dễ bảo trì và mở rộng hơn.

Tiện Ích Cho Frontend:

* Frontend có thể dễ dàng sử dụng API này để thực hiện chức năng đăng ký. Dữ liệu trả về (dưới dạng JSON) sẽ giúp hiển thị thông báo lỗi nếu có vấn đề trong quá trình đăng ký hoặc thông báo thành công nếu người dùng đăng ký thành công.

# **IV. Database**

## **1. XAMPP**

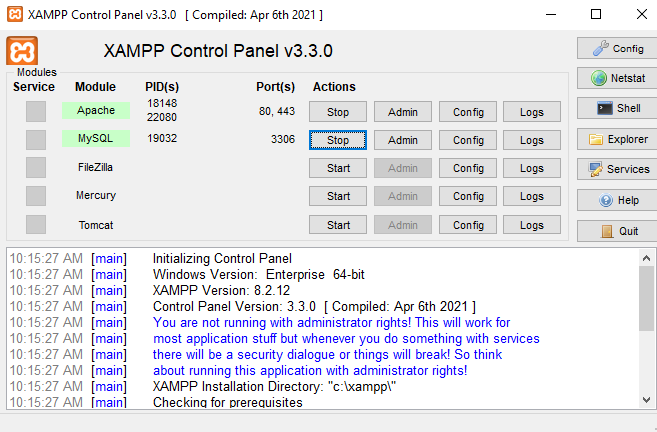
**- XAMPP** là một phần mềm mã nguồn mở tích hợp, được thiết kế để tạo môi trường máy chủ cục bộ (local server) trên máy tính cá nhân. Nó rất phổ biến trong phát triển web và học tập nhờ tính dễ cài đặt, sử dụng, và cung cấp đầy đủ các công cụ cần thiết cho một máy chủ web.

- XAMPP biến máy tính cá nhân thành một web server hoàn chỉnh để thử nghiệm ứng dụng hoặc website.

Giúp lập trình viên kiểm tra và phát triển ứng dụng web mà không cần kết nối internet hoặc thuê hosting.

- Tích hợp đầy đủ Apache, PHP, và MySQL, XAMPP là công cụ lý tưởng cho phát triển ứng dụng web sử dụng ngôn ngữ PHP và cơ sở dữ liệu MySQL.

- phpMyAdmin hỗ trợ quản lý cơ sở dữ liệu thông qua giao diện trực quan, giảm bớt khó khăn khi dùng dòng lệnh.



## **2. Thành phần của cơ sở dữ liệu**

Website bán sách bao gồm 4 bảng: book, book\_genre, genre, users

- book: Lưu thông tin sách.

Các cột:

id (int, khóa chính, tự động tăng):  
Mã định danh duy nhất cho từng cuốn sách.

title (varchar(255), NOT NULL):  
Tên của cuốn sách.

price (varchar(255), NOT NULL):  
Giá của cuốn sách.

author (varchar(255), NOT NULL):  
Tên tác giả của cuốn sách.

description (text, NOT NULL):  
Mô tả chi tiết về nội dung cuốn sách.

images (varchar(255), NOT NULL):  
Tên hoặc đường dẫn tệp hình ảnh đại diện của cuốn sách.

genre\_id (int, NOT NULL):  
Khóa ngoại liên kết với bảng genre, dùng để xác định thể loại sách.

- book\_genre: Đây là bảng trung gian many-to-many giữa bảng book và bảng genre. Một cuốn sách (book) có thể thuộc nhiều thể loại (genre), và ngược lại, một thể loại có thể áp dụng cho nhiều cuốn sách.

Các cột:

book\_id: int(11), (NOT NULL), Là khóa ngoại, liên kết với cột id trong bảng book.

genre\_id: int(11)

Kiểu dữ liệu: int(11), (NOT NULL), Là khóa ngoại, liên kết với cột id trong bảng genre.

- genre:Bảng genre được dùng để lưu trữ thông tin về thể loại sách.Mỗi thể loại có một ID duy nhất và một tên mô tả (ví dụ: "Kinh dị", "Trinh thám",...).

Các cột:

id**:** int(11, khóa chính (primary key), xác định duy nhất một thể loại. Cột này có tính năng AUTO\_INCREMENT, tức là giá trị sẽ tự động tăng mỗi khi có dòng mới được thêm.

name**:** varchar(255) (chuỗi ký tự). Tên của thể loại, ví dụ: "Kinh dị", "Trinh thám". Cột này bắt buộc phải có giá trị (NOT NULL).

- users:Bảng users được sử dụng để quản lý thông tin người dùng trong hệ thống.Lưu trữ các thông tin cơ bản như **tên đăng nhập, mật khẩu, email,** và **vai trò** của người dùng.

Các cột:

id**:** int(11). Là khóa chính của bảng (PRIMARY KEY), đảm bảo mỗi người dùng có một ID duy nhất. Có tính năng AUTO\_INCREMENT, tự động tăng giá trị khi thêm dòng mới.

username**:** varchar(50). Là tên đăng nhập của người dùng, không được trùng lặp (UNIQUE KEY). Bắt buộc phải có giá trị (NOT NULL).

password: varchar(255). Lưu trữ mật khẩu người dùng dưới dạng mã hóa. Sử dụng bcrypt để mã hóa mật khẩu.

email: varchar(100). email người dùng, không bắt buộc phải có (DEFAULT NULL). Không được trùng lặp (UNIQUE KEY).

role: enum('admin', 'user'). Xác định vai trò của người dùng. Giá trị mặc định: 'user'.