BSc. (Hons) in Software Engineering
Faculty of Computing

Online Book Shop System

Individual Assignment

Madhurangi G.N.P - IT/IFLS/B003/0028

Module: Advanced Web Technology

Submission Date: 2025/05/07

Table of Content

1. System Overview

The Online Bookshop System is a web application that allows users to browse, search, and purchase books. The system integrates with external book APIs to provide a comprehensive catalog and implements secure user authentication and shopping cart functionality.

Key Features:

- User registration and authentication
- Book browsing with search and filters
- Shopping cart management
- Secure checkout process
- Order history tracking

2. Technical Architecture

Frontend

- Framework: React.js
- State Management: Redux
- Styling: CSS3 with Bootstrap for responsive design
- Form Handling: Formik with Yup validation

Backend

- Framework: Node.js with Express
- Database: MySql
- Authentication: JWT (JSON Web Tokens)
- API Integration: Axios for external API calls

External APIs

- Google Books API: For book data and covers

3.API Documentation

Internal API Endpoints

Authentication

- POST /api/auth/register - User registration
- POST /api/auth/login - User login
- GET /api/auth/me - Get current user profile (protected)

Books

- GET /api/books - Get all books (with optional query parameters)
- GET /api/books/:id - Get single book details

Cart

- POST /api/cart - Add item to cart (protected)
- GET /api/cart - Get user's cart (protected)
- PUT /api/cart/:id - Update cart item (protected)
- DELETE /api/cart/:id - Remove item from cart (protected)

Orders

- POST /api/orders - Create new order (protected)

- GET /api/orders - Get user's order history (protected)

External API Integration

GET https://www.googleapis.com/books/v1/volumes?q={searchTerm}&key={API_KEY}

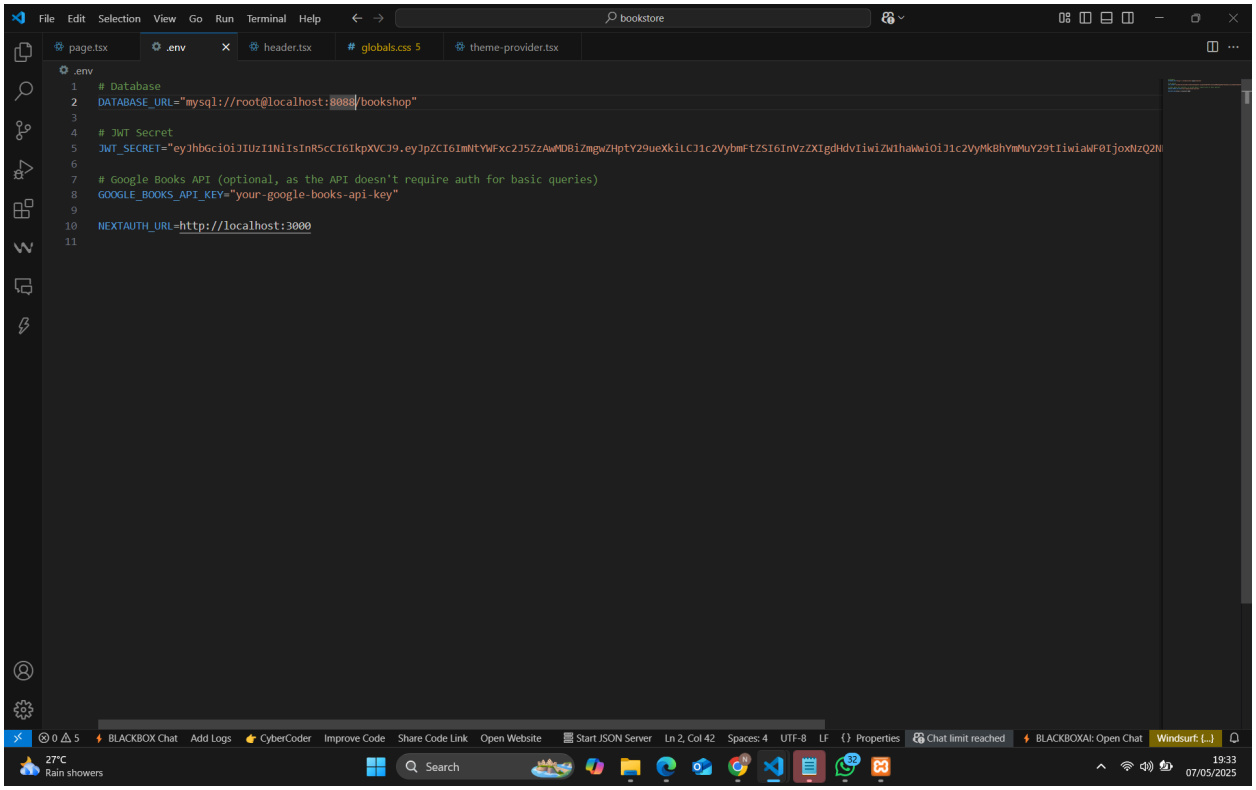
4.Database Schema

- My SQL

5.Setup Guide

```bash
npm run dev
# or
yarn dev
# or
pnpm dev
# or
bun dev
```

Create a .env file in the backend directory with the following variables:



bash

```
npm start
```

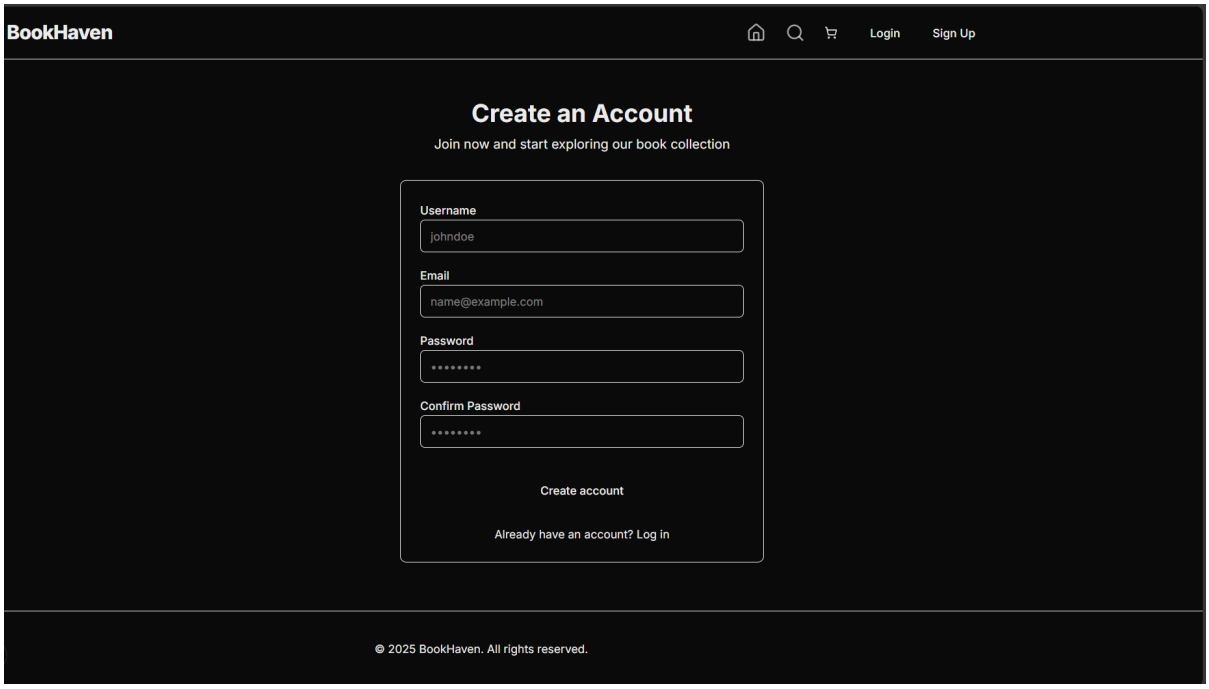
bash

```
npm install --legacy-peer-deps
npm run dev
```

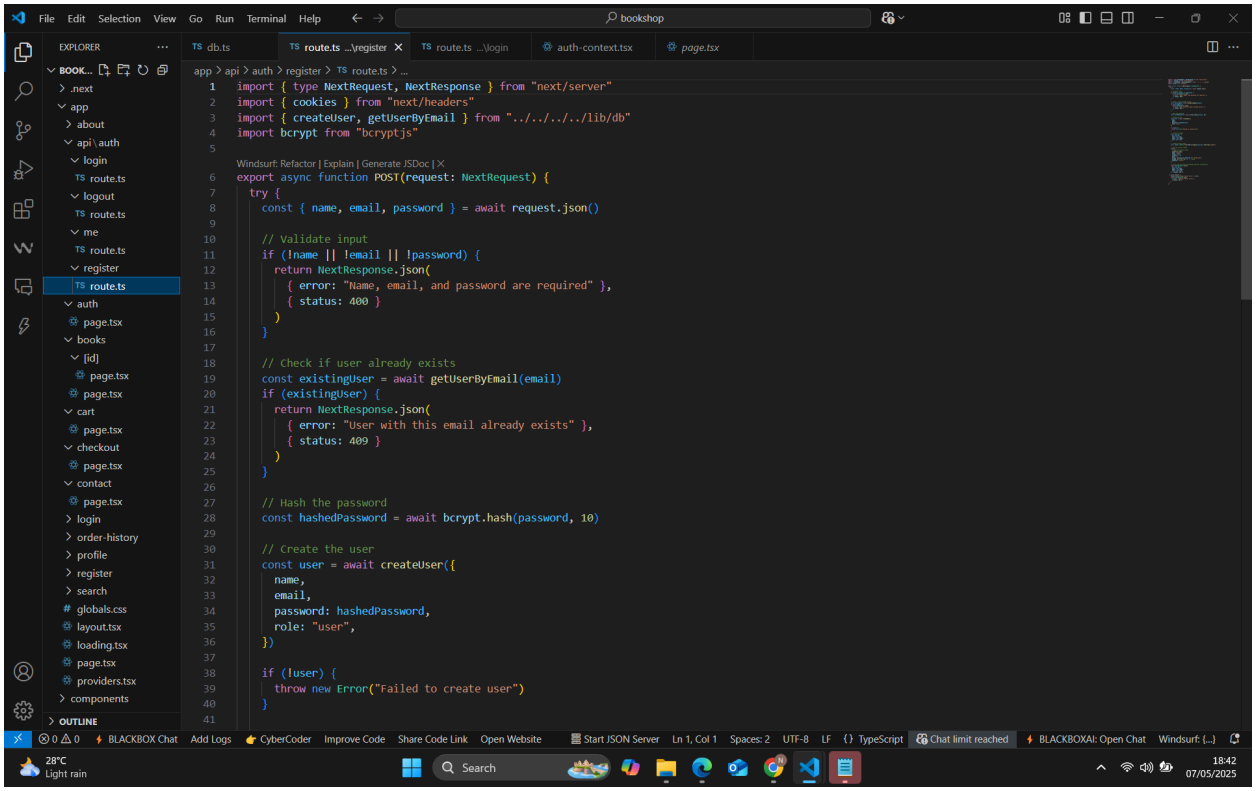
6.User Interface Documentation

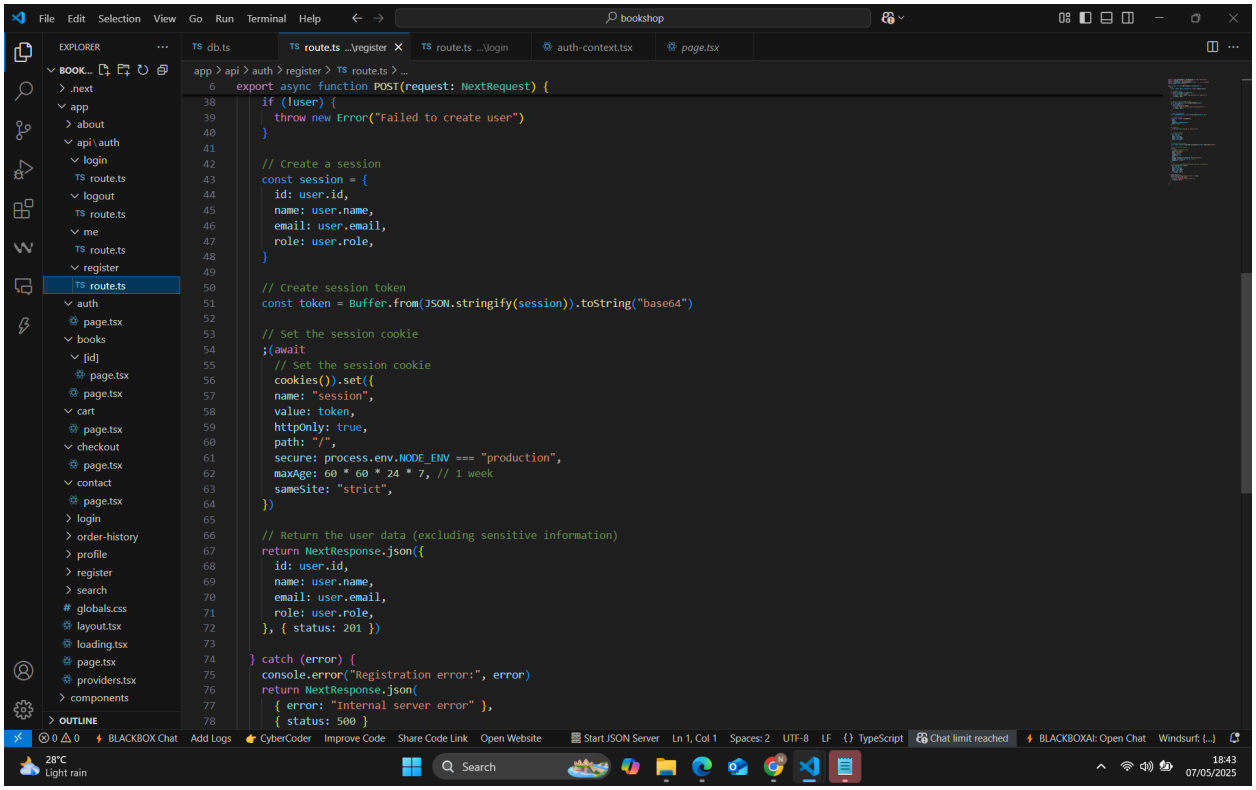
User Authentication Module

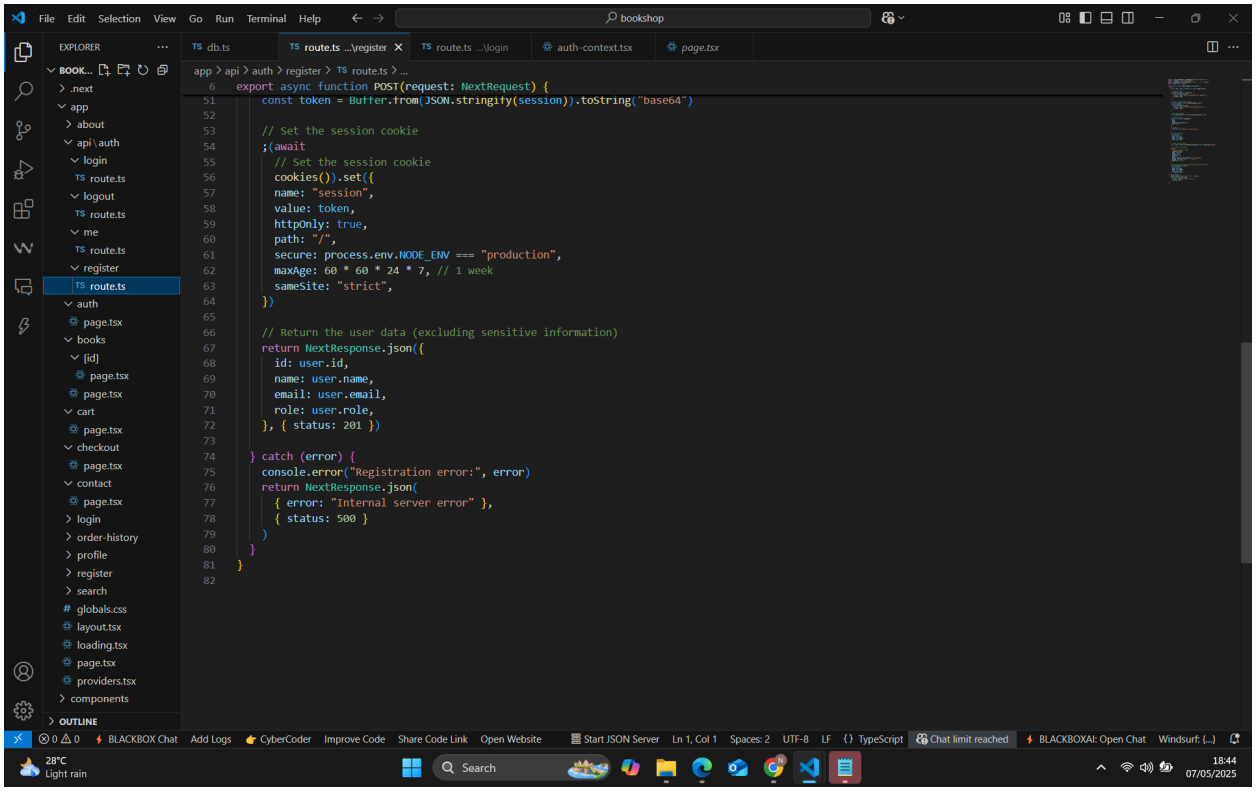
- Registration Page


BookHaven
Login
Sign Up
Create an Account
Join now and start exploring our book collection
Username
johndoe
Email
name@example.com
Password
Confirm Password
Create account
Already have an account? Log in
© 2025 BookHaven. All rights reserved.

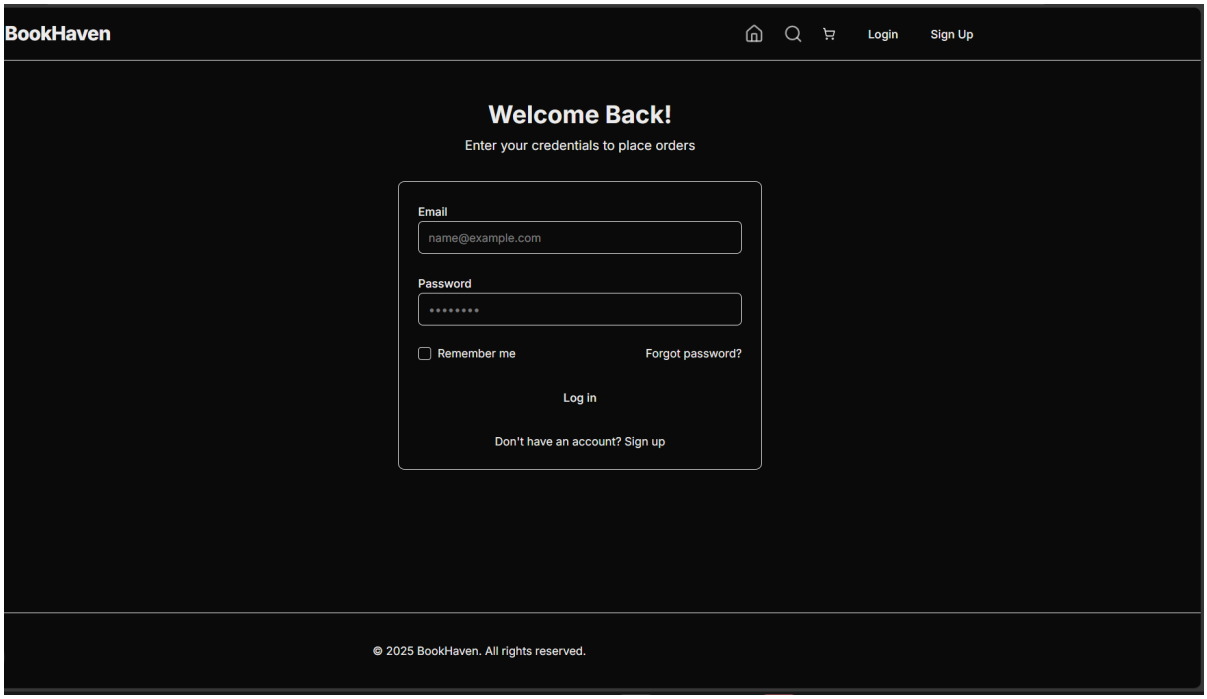
Code

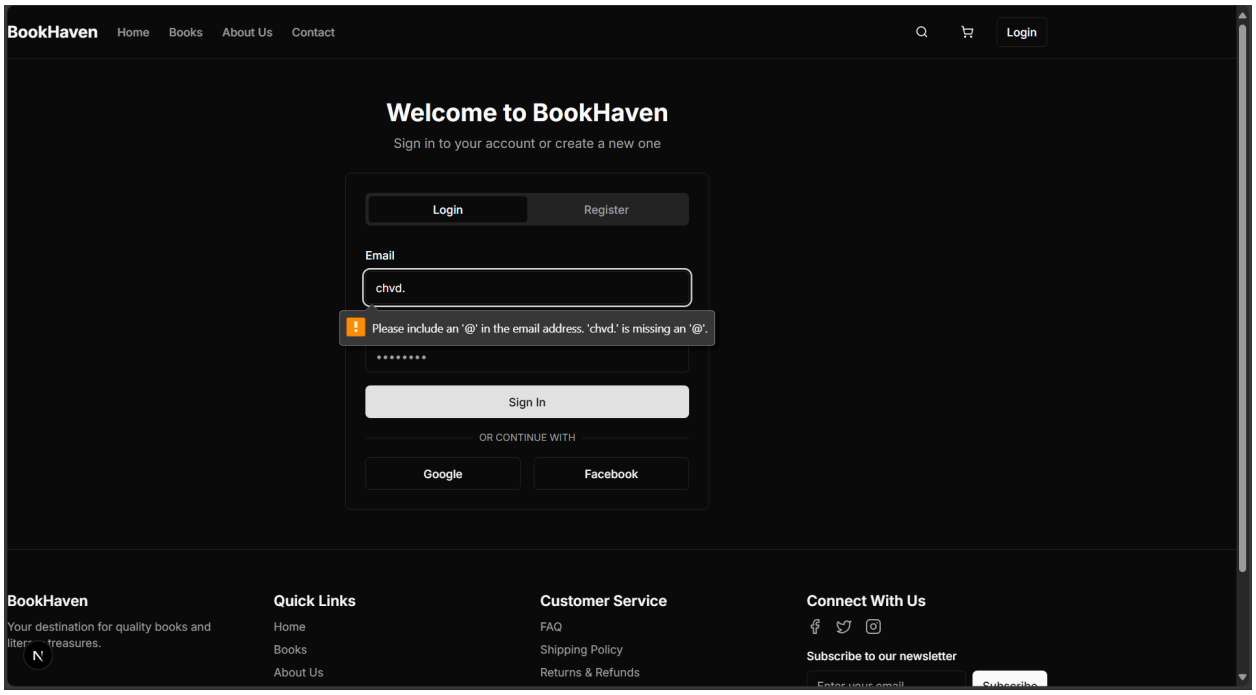

File Edit Selection View Go Run Terminal Help
bookshop
EXPLORER
db.ts
route.ts ...\register
route.ts ...\login
auth-context.tsx
page.tsx
app > api > auth > register > route.ts > ...
import { type NextRequest, NextResponse } from "next/server"
import { cookies } from "next/headers"
import { createUser, getUserByEmail } from "../../../../lib/db"
import bcrypt from "bcryptjs"
Windsurf: Refactor | Explain | Generate JSDoc |
export async function POST(request: NextRequest) {
try {
const { name, email, password } = await request.json()
// Validate input
if (!name || !email || !password) {
return NextResponse.json(
{ error: "Name, email, and password are required" },
{ status: 400 }
)
}
// Check if user already exists
const existingUser = await getUserByEmail(email)
if (existingUser) {
return NextResponse.json(
{ error: "User with this email already exists" },
{ status: 409 }
)
}
// Hash the password
const hashedPassword = await bcrypt.hash(password, 10)
// Create the user
const user = await createUser({
name,
email,
password: hashedPassword,
role: "user",
})
if (!user) {
throw new Error("Failed to create user")
}
OUTLINE
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Share Code Link
Open Website
Start JSON Server
Ln 1, Col 1
Spaces: 2
UTF-8
LF
TypeScript
Chat limit reached
BLACKBOXAI: Open Chat
28°C
Light rain
Search
18:42
07/05/2025


File Edit Selection View Go Run Terminal Help
bookshop
EXPLORER
db.ts
route.ts ...\register
route.ts ...\login
auth-context.tsx
page.tsx
app > api > auth > register > route.ts > ...
export async function POST(request: NextRequest) {
if (!user) {
throw new Error("Failed to create user")
}
// Create a session
const session = {
id: user.id,
name: user.name,
email: user.email,
role: user.role,
}
// Create session token
const token = Buffer.from(JSON.stringify(session)).toString("base64")
// Set the session cookie
;(await
// Set the session cookie
cookies()).set({
name: "session",
value: token,
httpOnly: true,
path: "/",
secure: process.env.NODE_ENV === "production",
maxAge: 60 * 60 * 24 * 7, // 1 week
sameSite: "strict",
})
// Return the user data (excluding sensitive information)
return NextResponse.json({
id: user.id,
name: user.name,
email: user.email,
role: user.role,
}, { status: 201 })
} catch (error) {
console.error("Registration error:", error)
return NextResponse.json(
{ error: "Internal server error" },
{ status: 500 }
28°C
Light rain
Search
18:43
07/05/2025

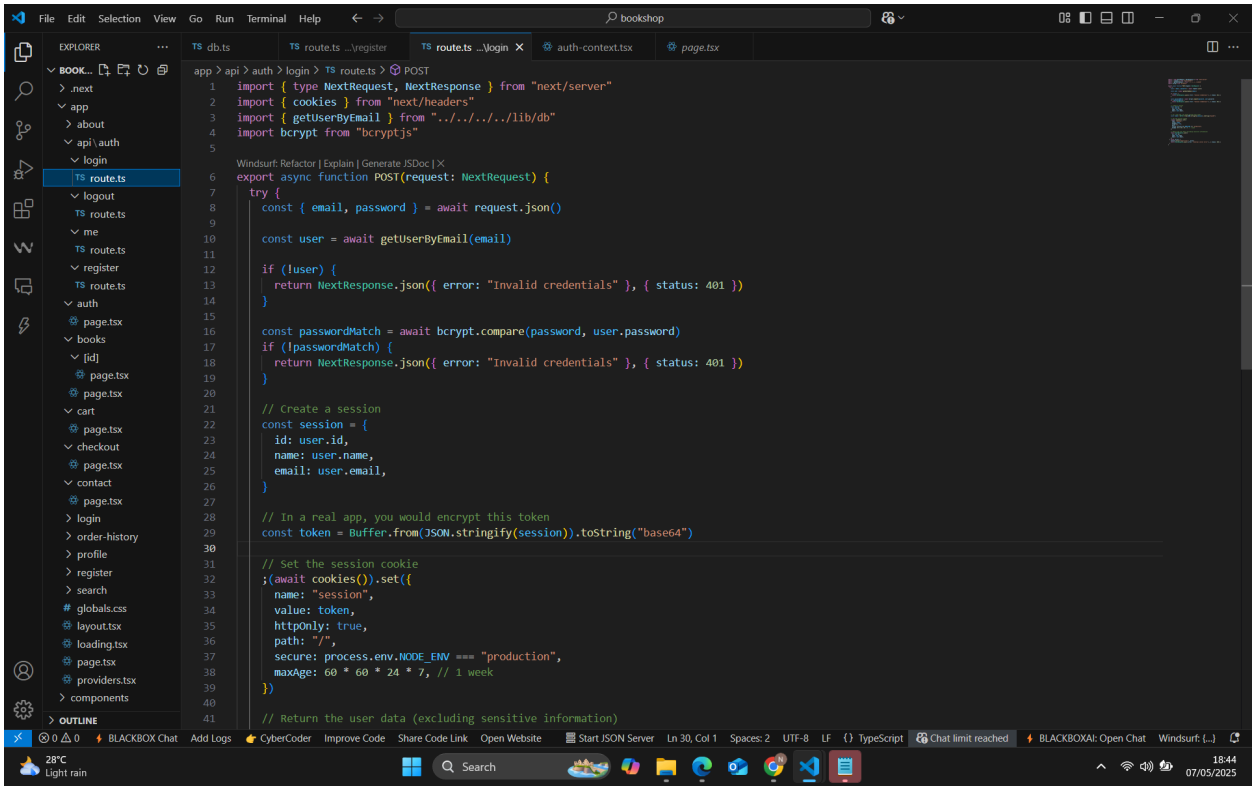

File Edit Selection View Go Run Terminal Help
bookshop
EXPLORER
db.ts
route.ts ...\register
route.ts ...\login
auth-context.tsx
page.tsx
app > api > auth > register > route.ts > ...
export async function POST(request: NextRequest) {
const token = Buffer.from(JSON.stringify(session)).toString("base64")
// Set the session cookie
;(await
// Set the session cookie
cookies()).set({
name: "session",
value: token,
httpOnly: true,
path: "/",
secure: process.env.NODE_ENV === "production",
maxAge: 60 * 60 * 24 * 7, // 1 week
sameSite: "strict",
})
// Return the user data (excluding sensitive information)
return NextResponse.json({
id: user.id,
name: user.name,
email: user.email,
role: user.role,
}, { status: 201 })
} catch (error) {
console.error("Registration error:", error)
return NextResponse.json(
{ error: "Internal server error" },
{ status: 500 }
)
}
}
28°C
Light rain
Search
18:44
07/05/2025

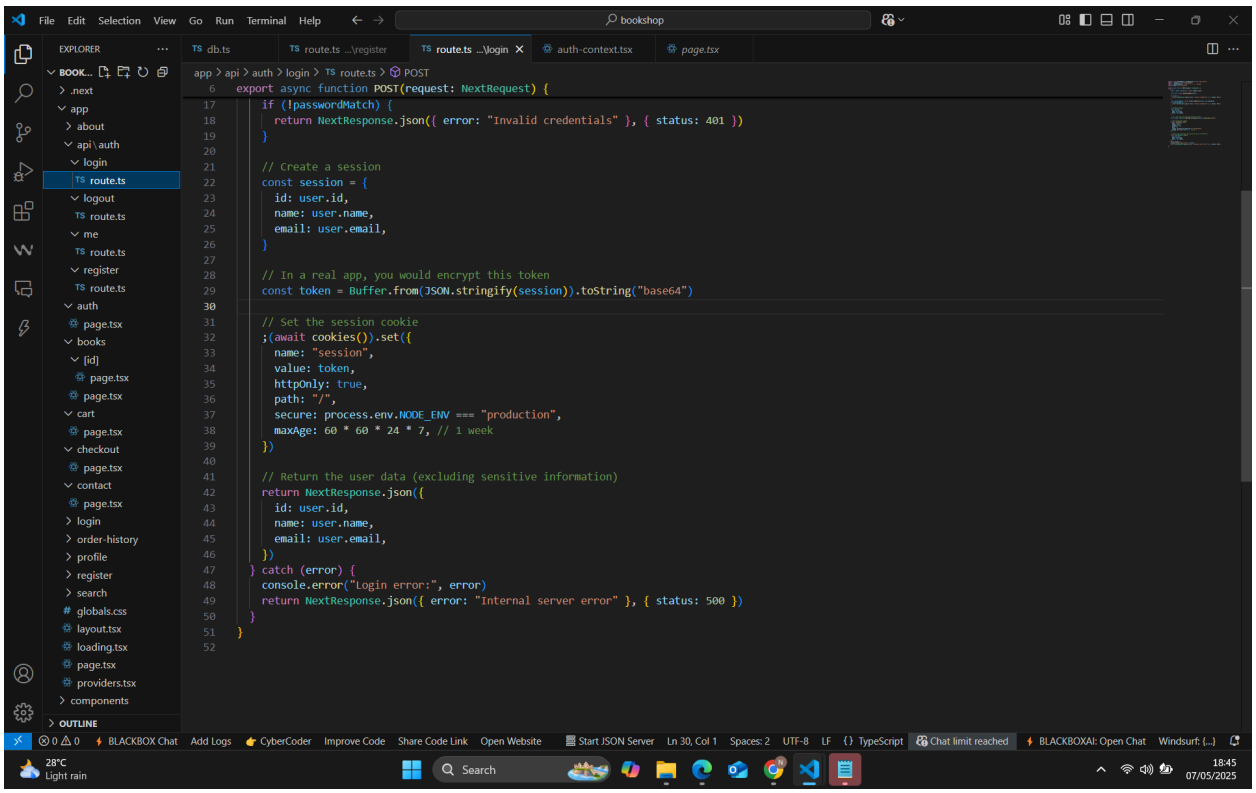
- Login Page


BookHaven
Login
Sign Up
Welcome Back!
Enter your credentials to place orders
Email
name@example.com
Password
Remember me
Forgot password?
Log in
Don't have an account? Sign up
© 2025 BookHaven. All rights reserved.


BookHaven
Home
Books
About Us
Contact
Login
Welcome to BookHaven
Sign in to your account or create a new one
Login
Register
Email
chvd.
Please include an '@' in the email address. 'chvd.' is missing an '@'.
Sign In
OR CONTINUE WITH
Google
Facebook
BookHaven
Quick Links
Home
Books
About Us
Customer Service
FAQ
Shipping Policy
Returns & Refunds
Connect With Us
Subscribe to our newsletter

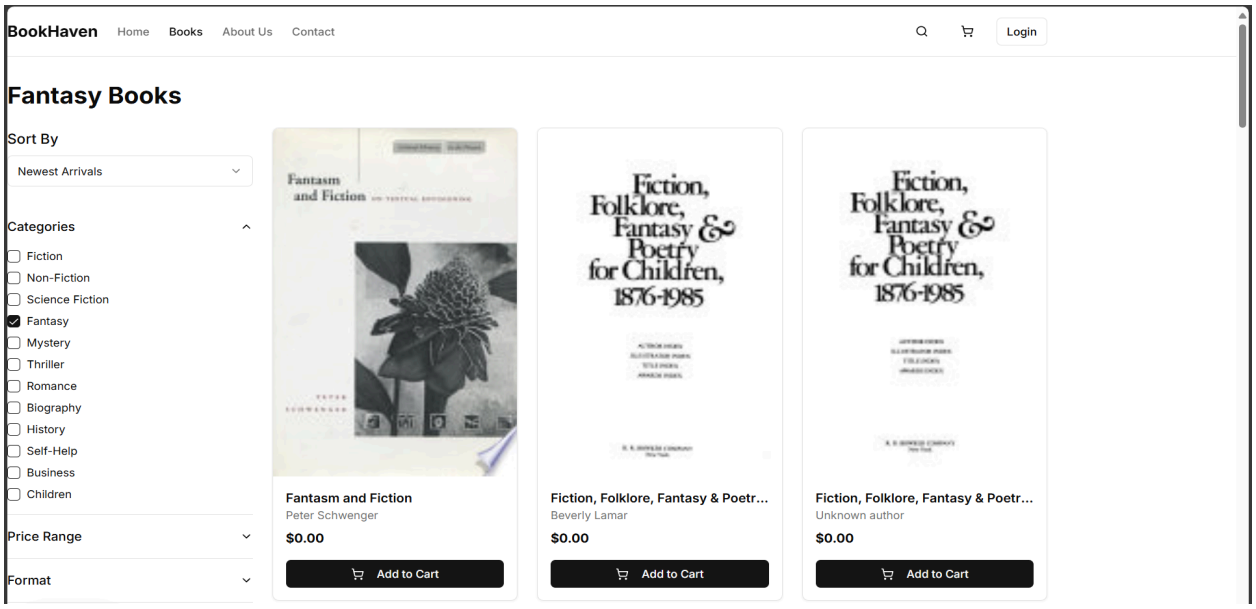
Code



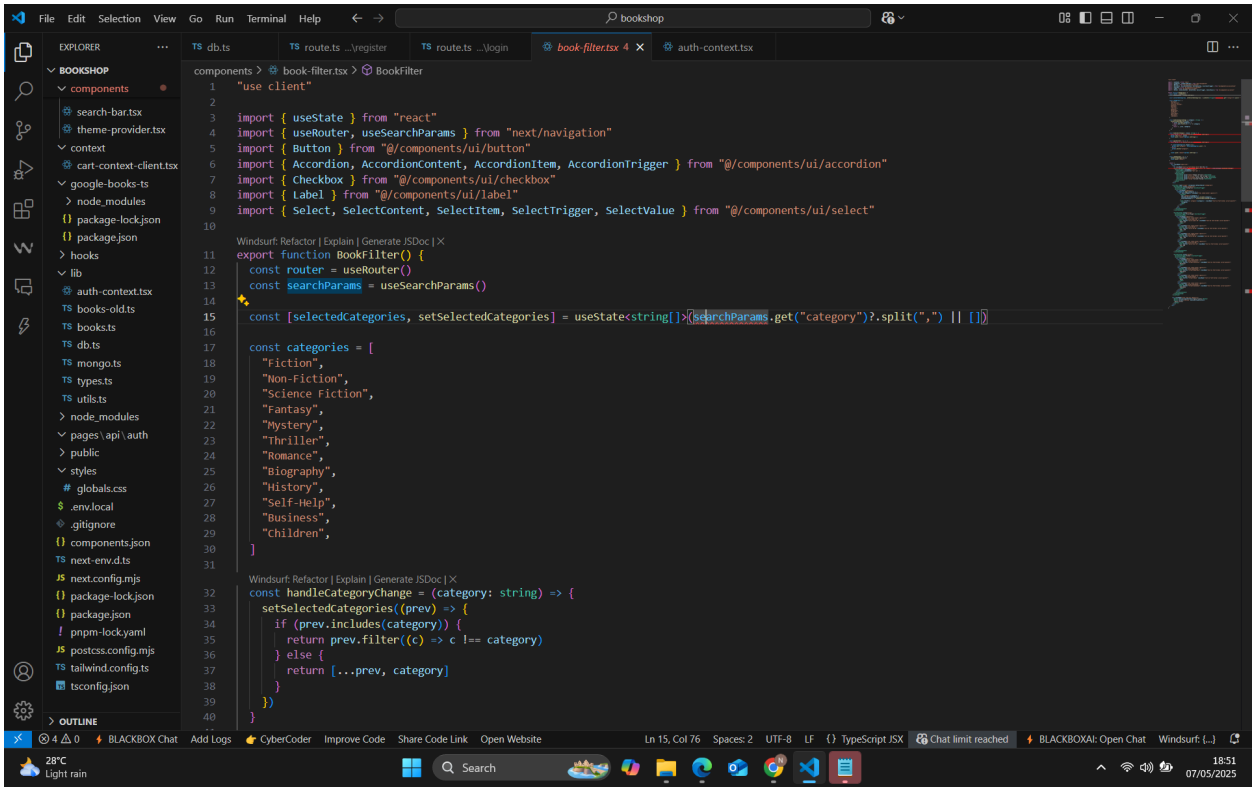


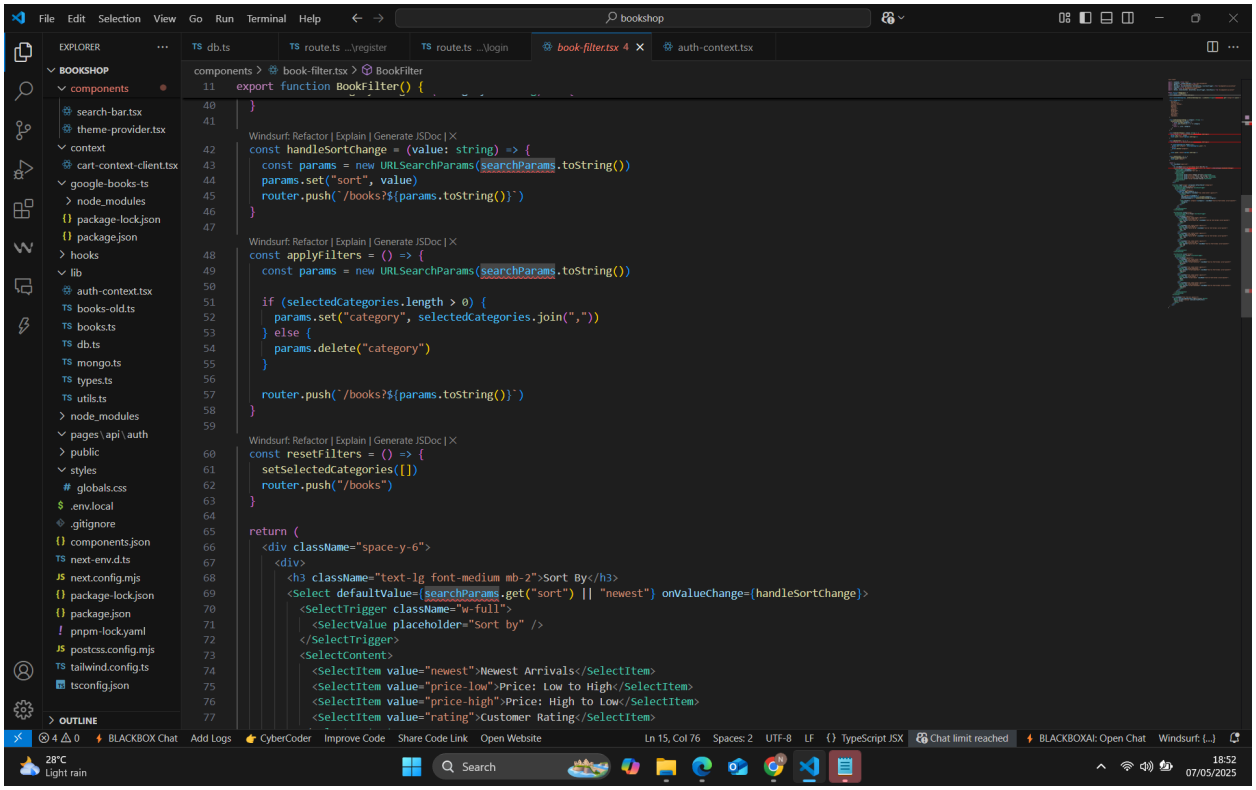
Book Management Module

- Book Listing and search

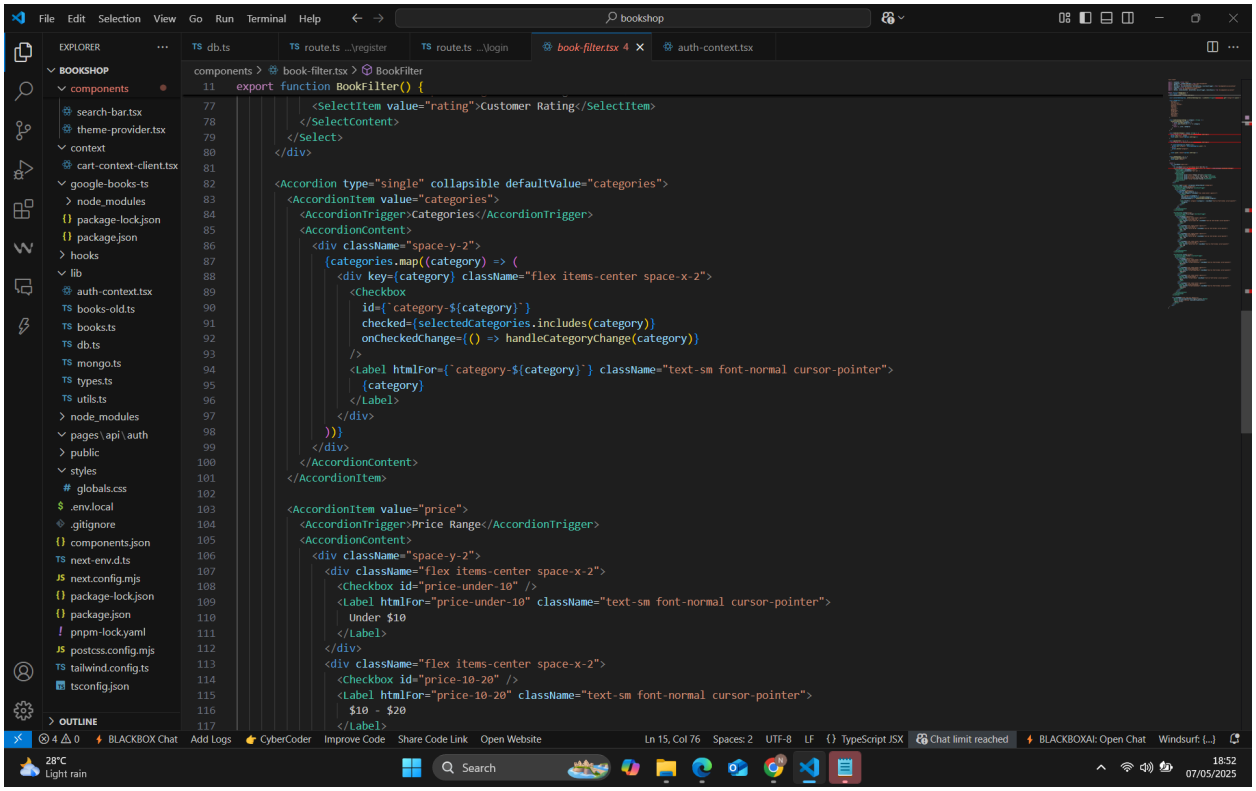


Code




File Edit Selection View Go Run Terminal Help
bookshop
EXPLORER
BOOKSHOP
components
search-bar.tsx
theme-provider.tsx
context
cart-context-client.tsx
google-books-ts
node_modules
package-lock.json
package.json
hooks
lib
auth-context.tsx
books-old.ts
books.ts
db.ts
mongo.ts
types.ts
utils.ts
node_modules
pages\api\auth
public
styles
globals.css
.env.local
.gitignore
components.json
next-env.d.ts
next.config.mjs
package-lock.json
package.json
pnpm-lock.yaml
postcss.config.mjs
tailwind.config.ts
tsconfig.json
OUTLINE
db.ts
route.ts ...\register
route.ts ...\login
book-filter.tsx 4
auth-context.tsx
components > book-filter.tsx > BookFilter
export function BookFilter() {
Windsurf: Refactor | Explain | Generate JSDoc |
const handleSortChange = (value: string) => {
const params = new URLSearchParams(searchParams.toString())
params.set("sort", value)
router.push(`/books?${params.toString()}`)
}
Windsurf: Refactor | Explain | Generate JSDoc |
const applyFilters = () => {
const params = new URLSearchParams(searchParams.toString())
if (selectedCategories.length > 0) {
params.set("category", selectedCategories.join(","))
} else {
params.delete("category")
}
router.push(`/books?${params.toString()}`)
}
Windsurf: Refactor | Explain | Generate JSDoc |
const resetFilters = () => {
setSelectedCategories([])
router.push("/books")
}
return (

<h3 className="text-lg font-medium mb-2">Sort By</h3>
<Select defaultValue={searchParams.get("sort") || "newest"} onValueChange={handleSortChange}>
<SelectTrigger className="w-full">
<SelectValue placeholder="Sort by" />
</SelectTrigger>
<SelectContent>
<SelectItem value="newest">Newest Arrivals</SelectItem>
<SelectItem value="price-low">Price: Low to High</SelectItem>
<SelectItem value="price-high">Price: High to Low</SelectItem>
<SelectItem value="rating">Customer Rating</SelectItem>
Ln 15, Col 76 Spaces: 2 UTF-8 LF {} TypeScript JSX Chat limit reached BLACKBOXAI: Open Chat Windsurf: (...)
28°C
Light rain
Search
18:52
07/05/2025


File Edit Selection View Go Run Terminal Help
bookshop
<SelectItem value="rating">Customer Rating</SelectItem>
</SelectContent>
</Select>

<Accordion type="single" collapsible defaultValue="categories">
<AccordionItem value="categories">
<AccordionTrigger>Categories</AccordionTrigger>
<AccordionContent>

{categories.map((category) => (

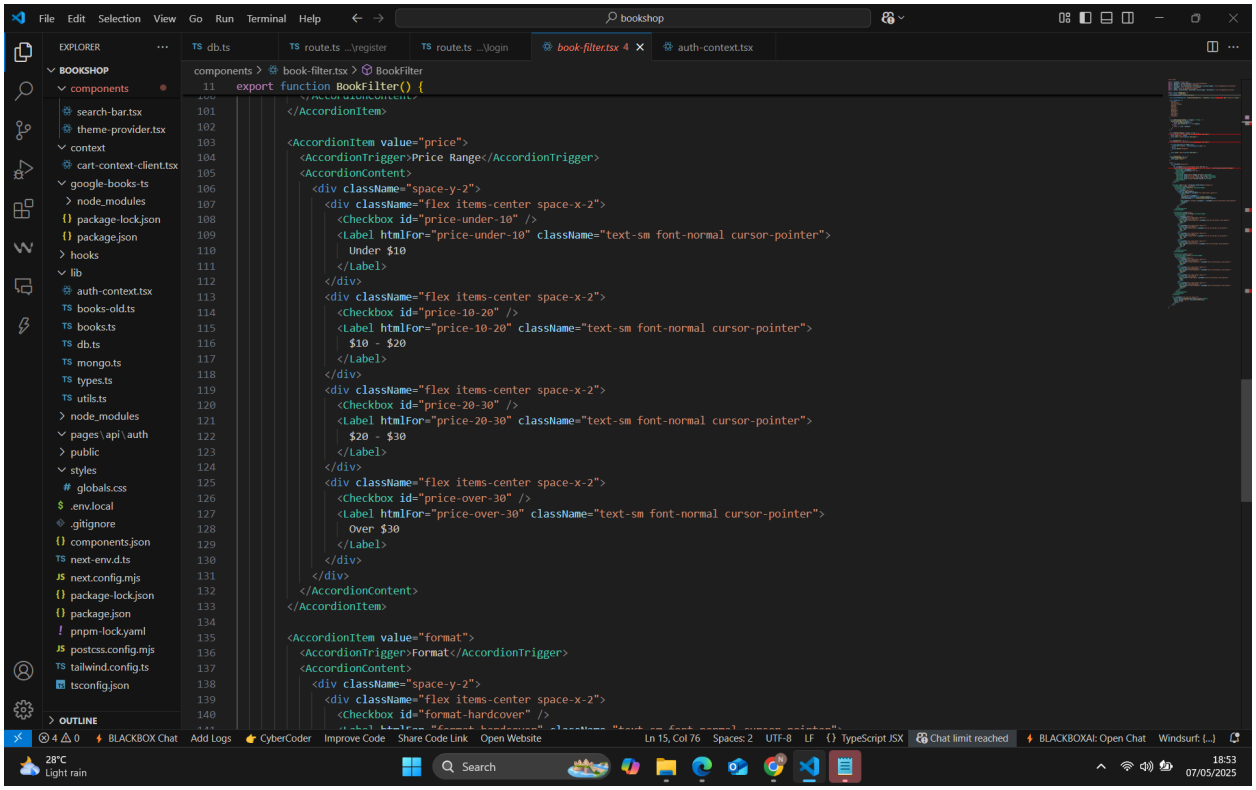
<Checkbox
id={`category-${category}`}
checked={selectedCategories.includes(category)}
onCheckedChange={() => handleCategoryChange(category)}
/>
<Label htmlFor={`category-${category}`} className="text-sm font-normal cursor-pointer">
{category}
</Label>

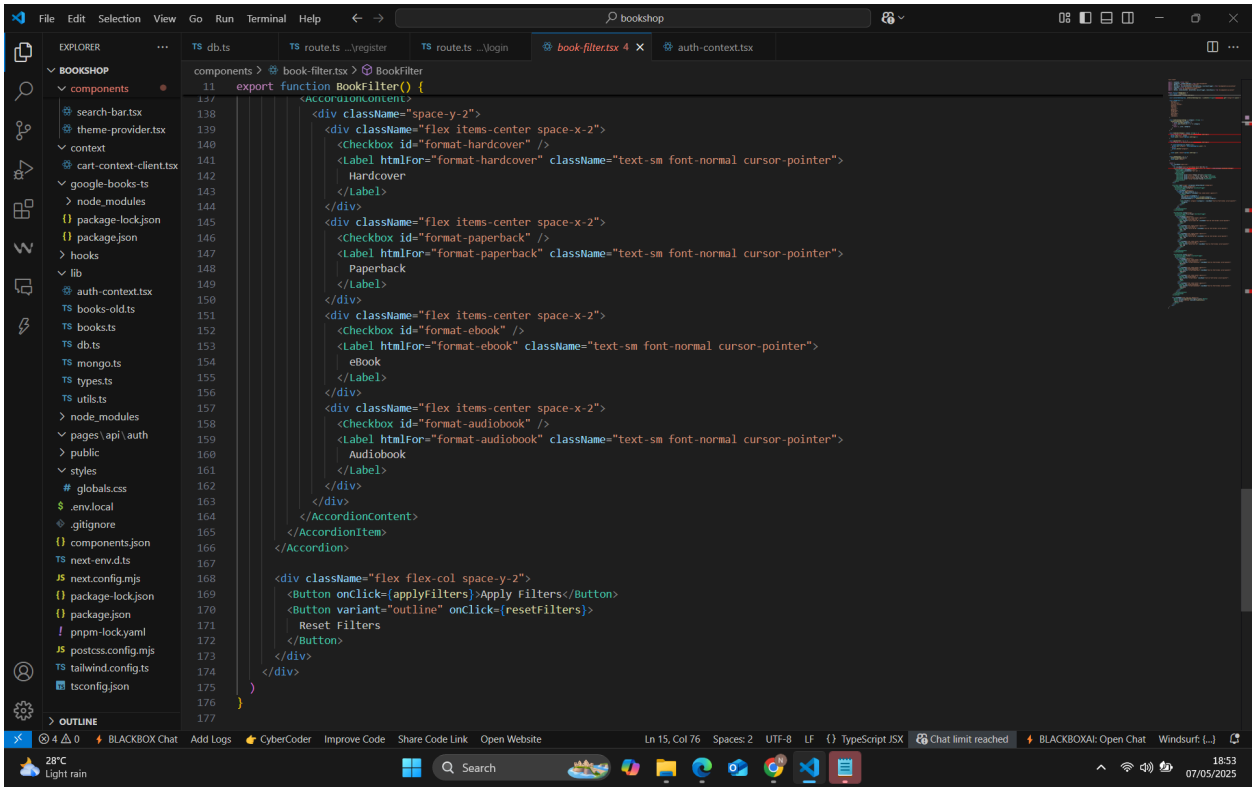
))}

</AccordionContent>
</AccordionItem>
<AccordionItem value="price">
<AccordionTrigger>Price Range</AccordionTrigger>
<AccordionContent>

<Checkbox id="price-under-10" />
<Label htmlFor="price-under-10" className="text-sm font-normal cursor-pointer">
Under $10
</Label>

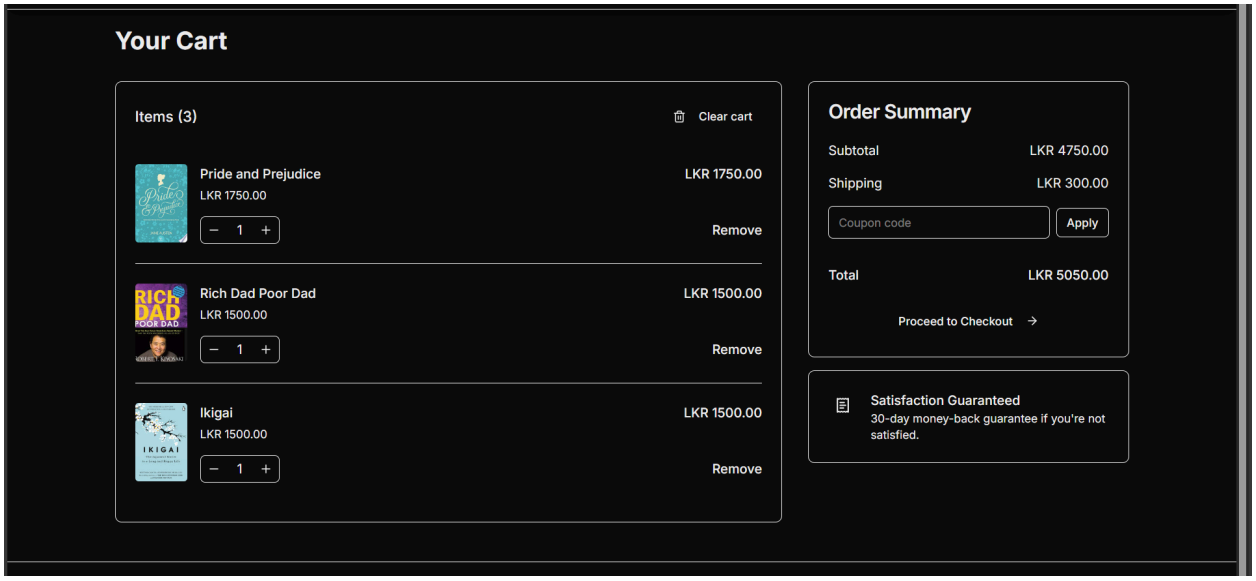
<Checkbox id="price-10-20" />
<Label htmlFor="price-10-20" className="text-sm font-normal cursor-pointer">
$10 - $20
</Label>
Ln 15, Col 76 Spaces: 2 UTF-8 LF {} TypeScript JSX Chat limit reached BLACKBOXAI: Open Chat Windsurf: (...)
28°C
Light rain
Search
18:52
07/05/2025



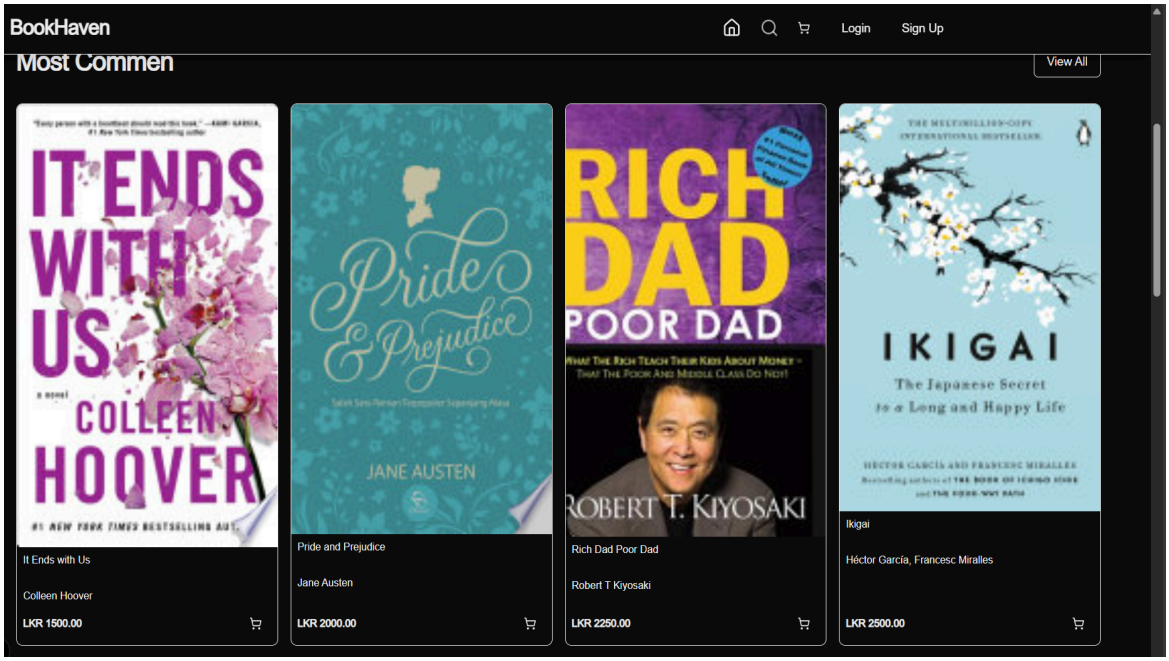


Shopping Module

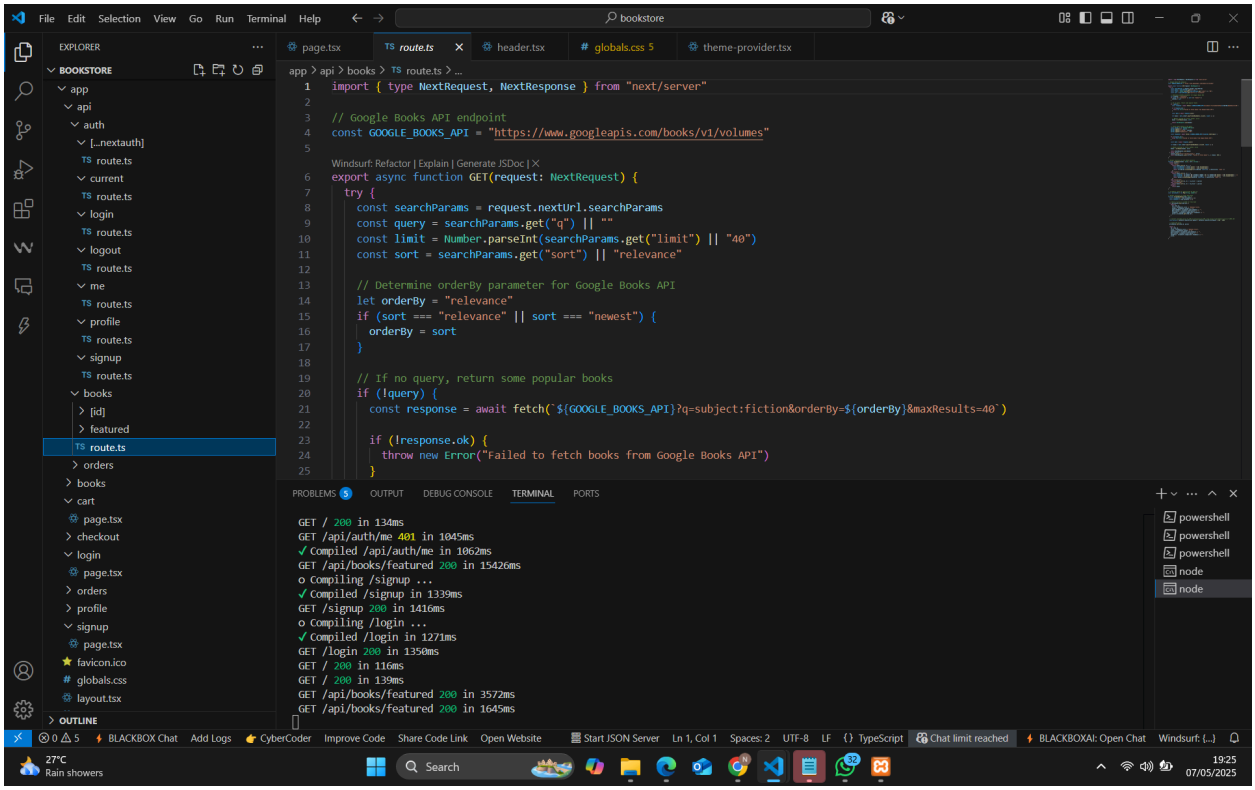
- Shopping Cart

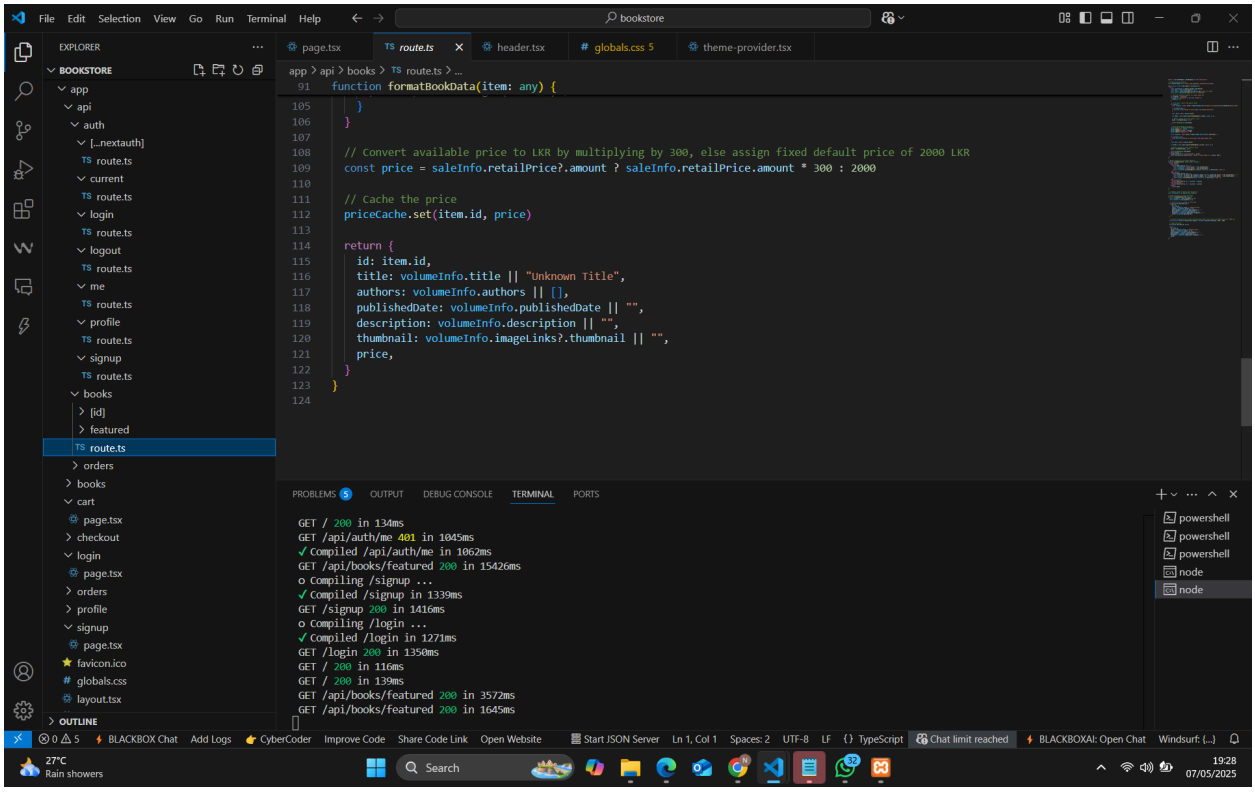


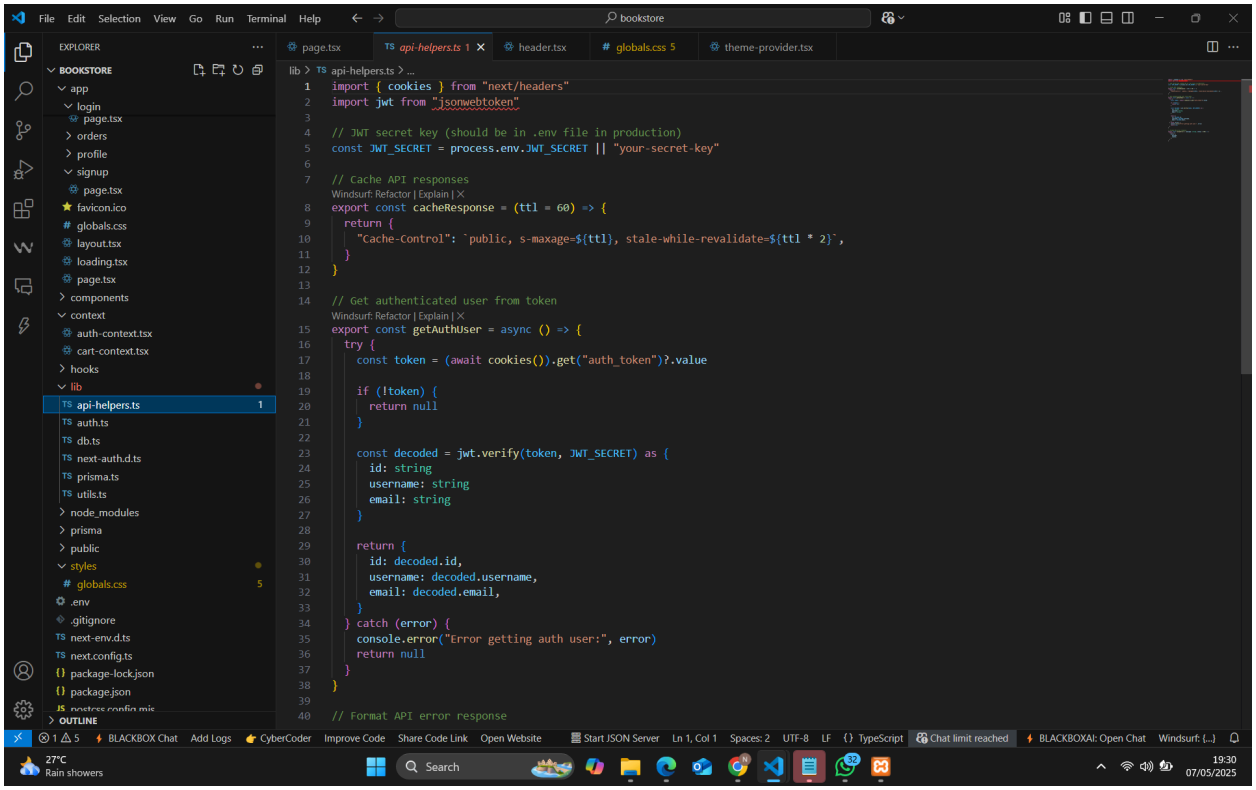
- Home page

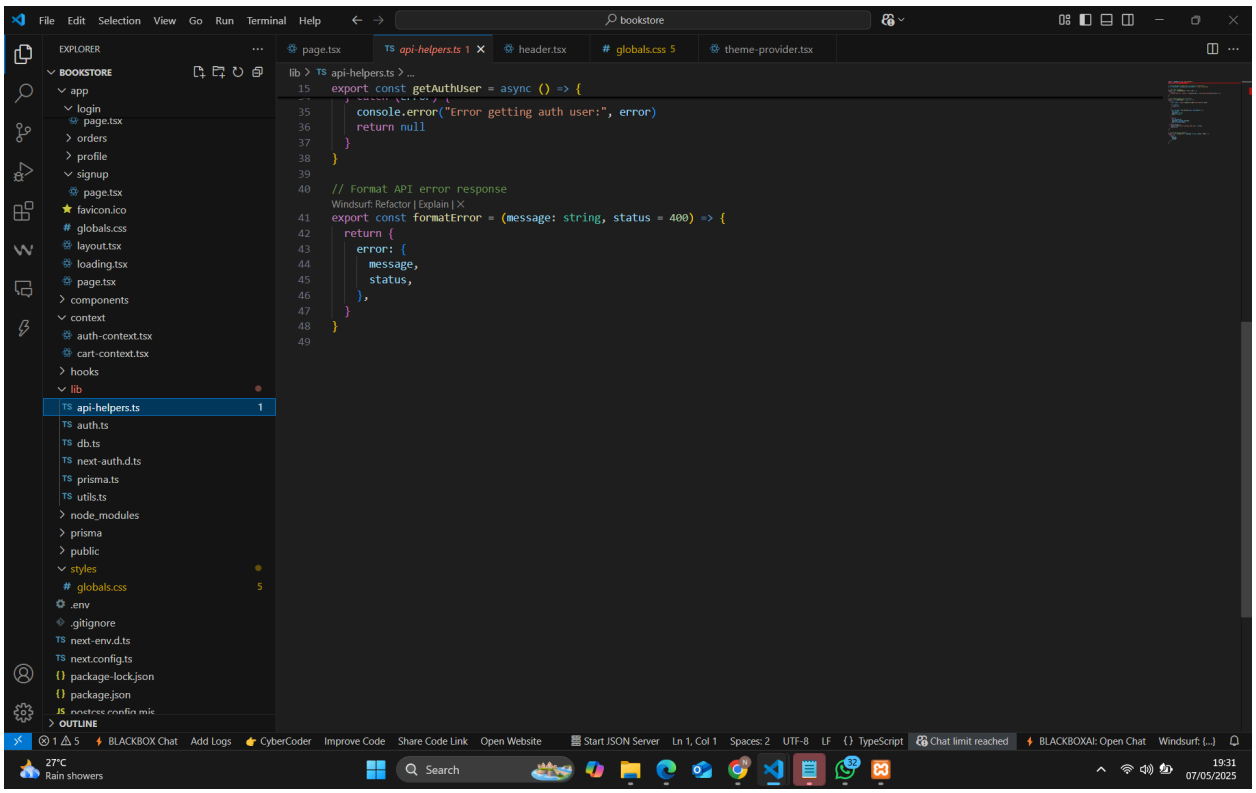


External API Integration


File Edit Selection View Go Run Terminal Help
bookstore
EXPLORER
BOOKSTORE
page.tsx
route.ts
header.tsx
globals.css 5
theme-provider.tsx
app > api > books > route.ts > ...
import { type NextRequest, NextResponse } from "next/server"
// Google Books API endpoint
const GOOGLE_BOOKS_API = "https://www.googleapis.com/books/v1/volumes"
Windsurf: Refactor | Explain | Generate JSDoc | X
export async function GET(request: NextRequest) {
try {
const searchParams = request.nextUrl.searchParams
const query = searchParams.get("q") || ""
const limit = Number.parseInt(searchParams.get("limit") || "40")
const sort = searchParams.get("sort") || "relevance"
// Determine orderBy parameter for Google Books API
let orderBy = "relevance"
if (sort === "relevance" || sort === "newest") {
orderBy = sort
}
// If no query, return some popular books
if (!query) {
const response = await fetch(`${GOOGLE_BOOKS_API}?q=subject:fiction&orderBy=${orderBy}&maxResults=40`)
if (!response.ok) {
throw new Error("Failed to fetch books from Google Books API")
}
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GET / 200 in 134ms
GET /api/auth/me 401 in 1045ms
✓ Compiled /api/auth/me in 1062ms
GET /api/books/featured 200 in 15426ms
o Compiling /signup ...
✓ Compiled /signup in 1339ms
GET /signup 200 in 1416ms
o Compiling /login ...
✓ Compiled /login in 1271ms
GET /login 200 in 1350ms
GET / 200 in 116ms
GET / 200 in 139ms
GET /api/books/featured 200 in 3572ms
GET /api/books/featured 200 in 1645ms
powershell
powershell
powershell
node
node
OUTLINE
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Share Code Link
Open Website
Start JSON Server
Ln 1, Col 1
Spaces: 2
UTF-8
LF
TypeScript
Chat limit reached
BLACKBOXAI: Open Chat
Windsurf: (...)
27°C
Rain showers
Search
19:25
07/05/2025

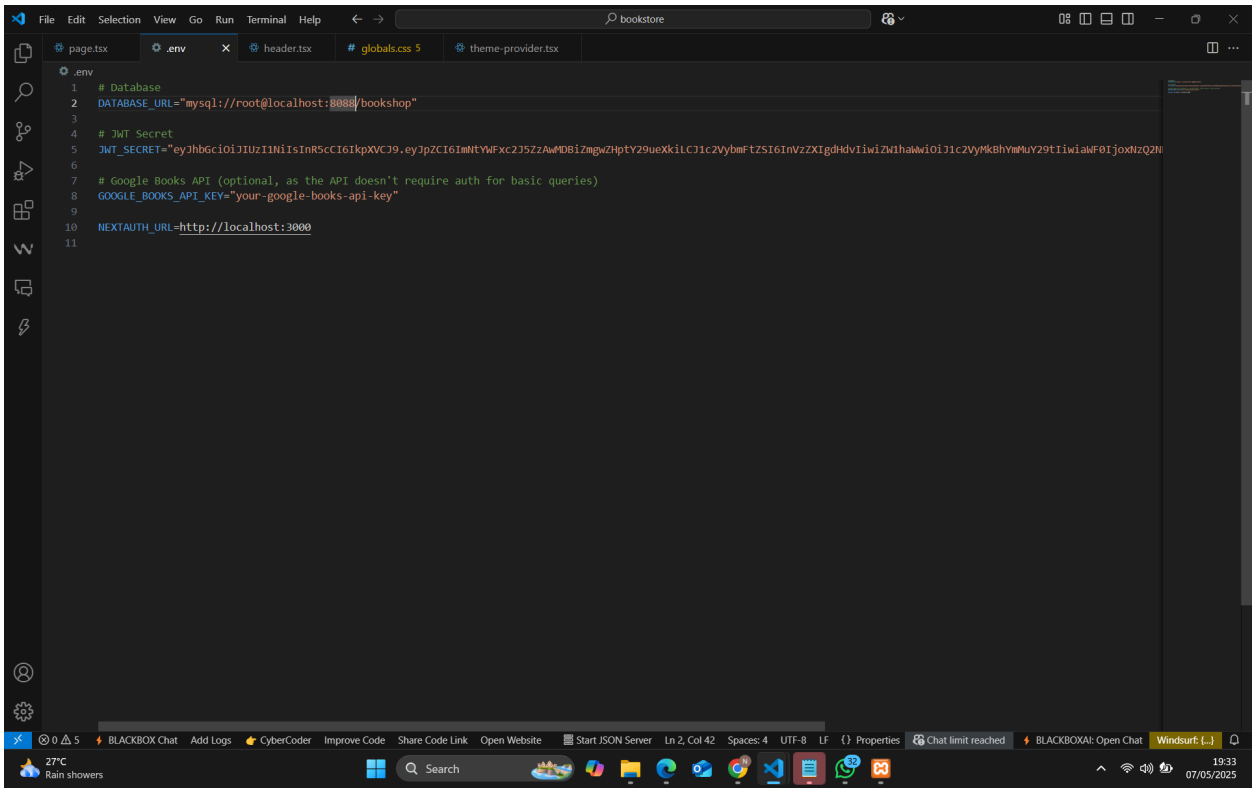

File Edit Selection View Go Run Terminal Help
bookstore
EXPLORER
BOOKSTORE
page.tsx
route.ts
header.tsx
globals.css 5
theme-provider.tsx
app > api > books > route.ts > ...
function formatBookData(item: any) {
}
}
// Convert available price to LKR by multiplying by 300, else assign fixed default price of 2000 LKR
const price = saleInfo.retailPrice?.amount ? saleInfo.retailPrice.amount * 300 : 2000
// Cache the price
priceCache.set(item.id, price)
return {
id: item.id,
title: volumeInfo.title || "Unknown Title",
authors: volumeInfo.authors || [],
publishedDate: volumeInfo.publishedDate || "",
description: volumeInfo.description || "",
thumbnail: volumeInfo.imageLinks?.thumbnail || "",
price,
}
}
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GET / 200 in 134ms
GET /api/auth/me 401 in 1045ms
✓ Compiled /api/auth/me in 1062ms
GET /api/books/featured 200 in 15426ms
o Compiling /signup ...
✓ Compiled /signup in 1339ms
GET /signup 200 in 1416ms
o Compiling /login ...
✓ Compiled /login in 1271ms
GET /login 200 in 1350ms
GET / 200 in 116ms
GET / 200 in 139ms
GET /api/books/featured 200 in 3572ms
GET /api/books/featured 200 in 1645ms
powershell
powershell
powershell
node
node
OUTLINE
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Share Code Link
Open Website
Start JSON Server
Ln 1, Col 1
Spaces: 2
UTF-8
LF
TypeScript
Chat limit reached
BLACKBOXAI: Open Chat
Windsurf: (...)
27°C
Rain showers
Search
19:28
07/05/2025


File Edit Selection View Go Run Terminal Help
bookstore
EXPLORER
BOOKSTORE
page.tsx
api-helpers.ts 1
header.tsx
globals.css 5
theme-provider.tsx
lib > api-helpers.ts > ...
import { cookies } from "next/headers"
import jwt from "jsonwebtoken"
// JWT secret key (should be in .env file in production)
const JWT_SECRET = process.env.JWT_SECRET || "your-secret-key"
// Cache API responses
Windsurf: Refactor | Explain |
export const cacheResponse = (ttl = 60) => {
return {
"Cache-Control": `public, s-maxage=${ttl}, stale-while-revalidate=${ttl * 2}`,
}
}
// Get authenticated user from token
Windsurf: Refactor | Explain |
export const getAuthUser = async () => {
try {
const token = (await cookies()).get("auth_token")?.value
if (!token) {
return null
}
const decoded = jwt.verify(token, JWT_SECRET) as {
id: string
username: string
email: string
}
return {
id: decoded.id,
username: decoded.username,
email: decoded.email,
}
} catch (error) {
console.error("Error getting auth user:", error)
return null
}
}
// Format API error response
OUTLINE
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Share Code Link
Open Website
Start JSON Server
Ln 1, Col 1
Spaces: 2
UTF-8
LF
TypeScript
Chat limit reached
BLACKBOXAI: Open Chat
Windsurf: (...)
27°C
Rain showers
Search
19:30
07/05/2025


File Edit Selection View Go Run Terminal Help
bookstore
EXPLORER
BOOKSTORE
page.tsx
api-helpers.ts 1
header.tsx
globals.css 5
theme-provider.tsx
lib > api-helpers.ts > ...
export const getAuthUser = async () => {
console.error("Error getting auth user:", error)
return null
}
}
// Format API error response
Windsurf: Refactor | Explain |
export const formatError = (message: string, status = 400) => {
return {
error: {
message,
status,
},
}
}
OUTLINE
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Share Code Link
Open Website
Start JSON Server
Ln 1, Col 1
Spaces: 2
UTF-8
LF
TypeScript
Chat limit reached
BLACKBOXAI: Open Chat
Windsurf: (...)
27°C
Rain showers
Search
19:31
07/05/2025

7. Security Implementation

JWT tokens for session management



8. Testing

User Registration

Test Case 1.1: Successful Registration

Description: Verify that a new user can register with valid credentials

Preconditions: No existing user with the test email

Test Steps:

Navigate to registration page

Enter valid username, email, and password

Submit the form

Expected Result:

User is redirected to home page

Success message is displayed

New user record exists in database

Test Case 1.2: Registration with Existing Email

Description: Verify system prevents duplicate email registration

Preconditions: User with test@example.com already exists

Test Steps:

Navigate to registration page

Enter username and existing email (test@example.com)

Submit the form

Expected Result:

Form displays error "Email already in use"

No new user created

Test Case 1.3: Registration with Invalid Password

Description: Verify password complexity requirements

Test Steps:

Navigate to registration page

Enter valid username and email

Enter weak password (e.g., "123")

Submit the form

Expected Result:

Form displays password requirements error

Form submission prevented

Login and Authentication

Test Case 2.1: Successful Login

Description: Verify valid user can login

Preconditions: Test user exists in database

Test Steps:

Navigate to login page

Enter valid email and password

Submit the form

Expected Result:

User is redirected to home page

Auth token is stored in cookies/local storage

User menu shows authenticated state

Test Case 2.2: Login with Invalid Credentials

Description: Verify system rejects invalid credentials

Test Steps:

Navigate to login page

Enter invalid email or password

Submit the form

Expected Result:

Error message "Invalid credentials" displayed

No auth token stored

User remains unauthenticated

9. Future Enhancements

- Email Verification: Implement email confirmation for new registrations
- Wish Lists: Allow users to create and manage book wishlists
- Reviews and Ratings: Enable users to review purchased books
- Recommendation Engine: Suggest books based on user's browsing history
- Mobile App: Develop native mobile applications for iOS and Android