



**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
2. Technical Architecture
  - Frontend
  - Backend
  - External APIs
3. API Documentation
  - Internal API Endpoints
    - Authentication
    - Books
    - Cart
    - Orders
  - External API Integration
4. Database Schema
5. Setup Guide
6. User Interface Documentation
7. Security Implementation
8. Testing
9. Future Enhancements

# 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}](https://www.googleapis.com/books/v1/volumes?q={searchTerm}&key={API_KEY})

## 4. Database Schema

- MySQL

## 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:**

The screenshot shows a code editor with several tabs open: `page.tsx`, `.env`, `header.tsx`, `globals.css`, and `theme-provider.tsx`. The `.env` file contains environment variables:

```
1 # Database
2 DATABASE_URL="mysql://root@localhost:8088/bookshop"
3
4 # JWT Secret
5 JWT_SECRET="eyJhbGciOiJIUzI1NiIsInR5cG1lklpXVCJ9.eyJpZC16IiMNTyWpxCzJSzzAwMDBiZmgwZHptY29ueXk1CJ1c2VybmlfLzS16InVzZX1gdHViIwi0iIjc2VyyMkBhYmMuY29tIiwiawF01joxNzQ2Ii
6
7 # Google Books API (optional, as the API doesn't require auth for basic queries)
8 GOOGLE_BOOKS_API_KEY="your-google-books-api-key"
9
10 NEXTAUTH_URL=http://localhost:3000
11
```

Below the code editor is a terminal window showing the command `npm start` being run. The terminal output includes:

```
27°C Rain showers
2025-07-05T19:33:00.000Z [main] [INFO] Starting application...
2025-07-05T19:33:00.000Z [main] [INFO] Application started at http://localhost:3000
2025-07-05T19:33:00.000Z [main] [INFO] Listening on port 3000
2025-07-05T19:33:00.000Z [main] [INFO] Ready for requests...
```

bash

```
npm start
```

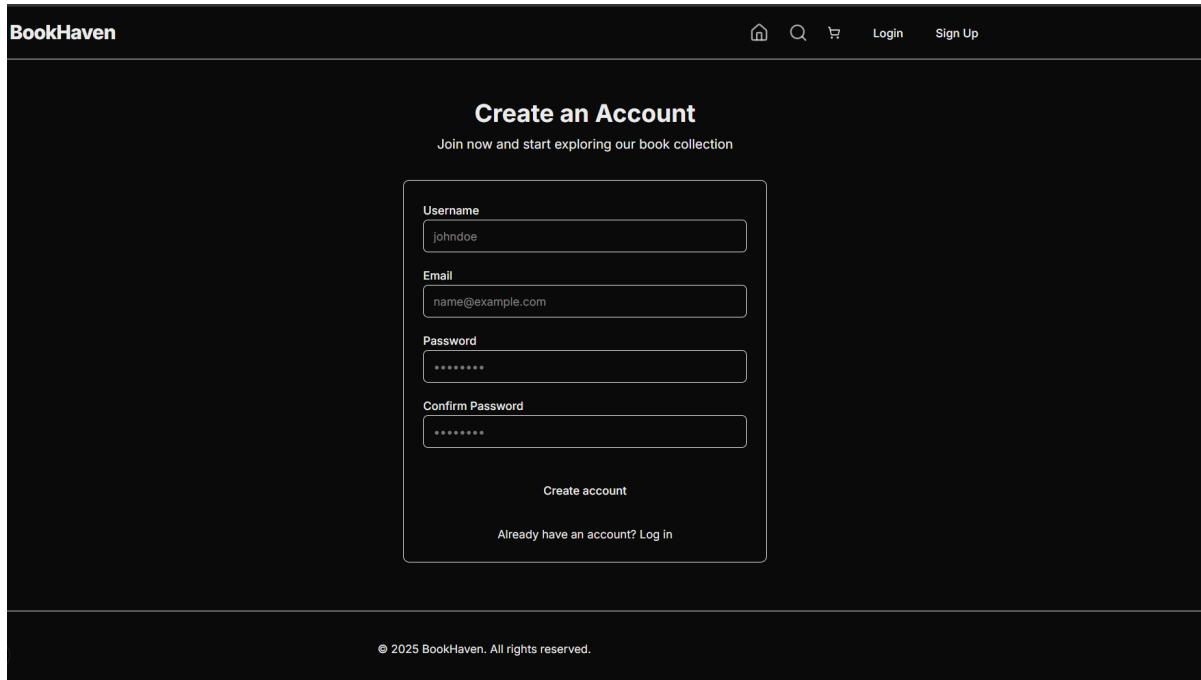
bash

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

## 6. User Interface Documentation

### User Authentication Module

- Registration Page

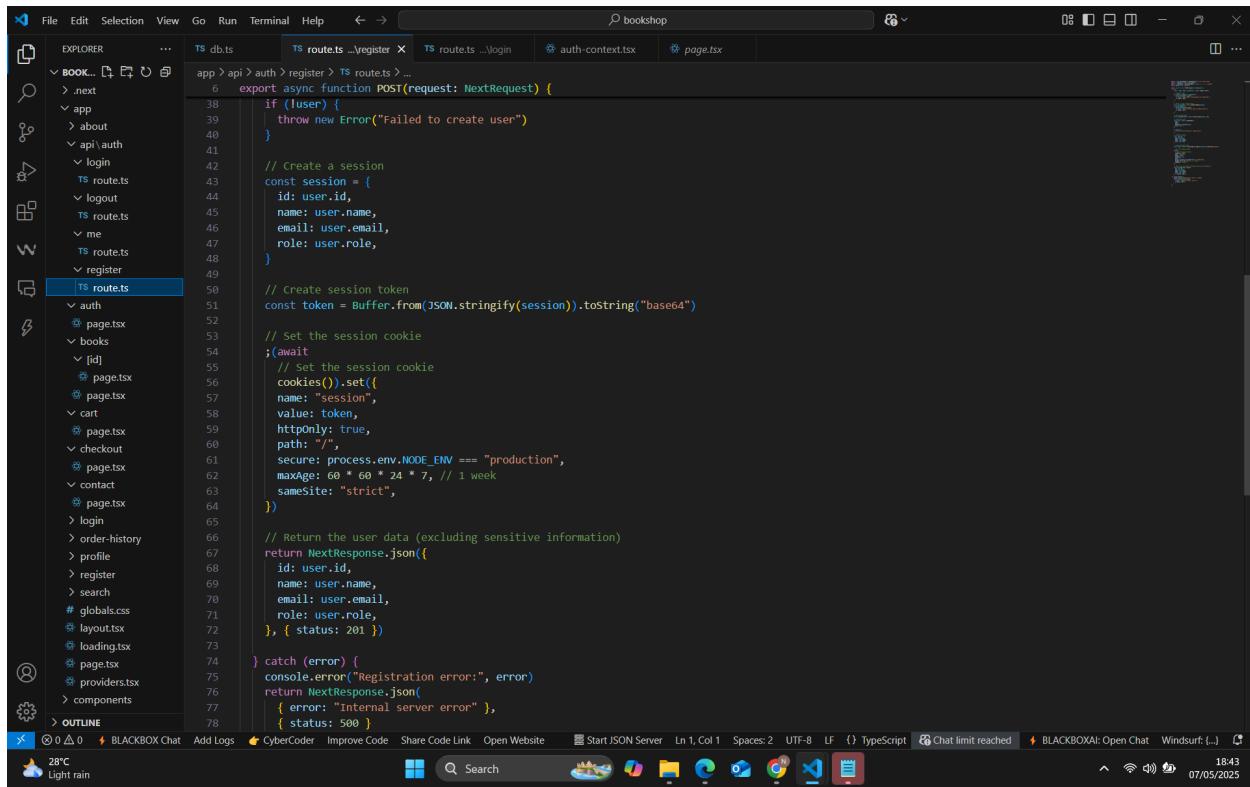


## Code

The screenshot shows a code editor interface with the file 'route.ts' selected in the left sidebar. The code in the editor is as follows:

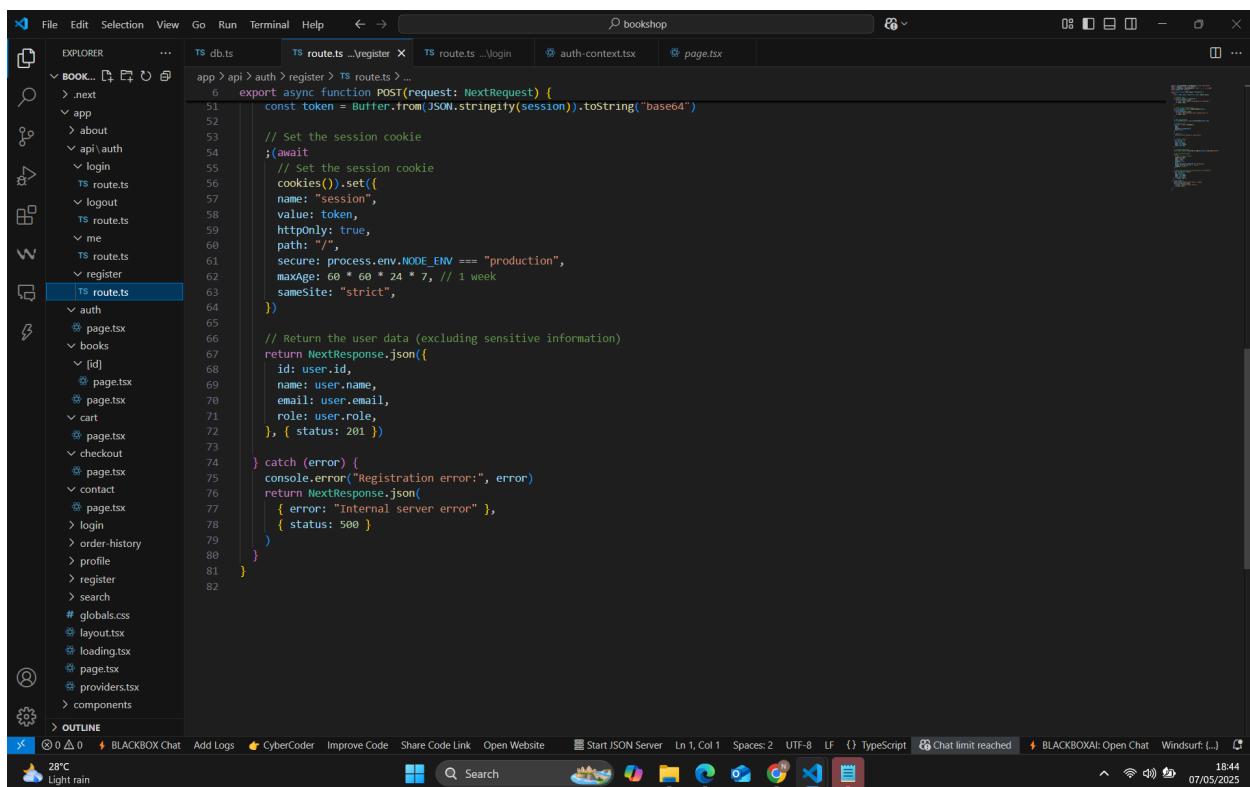
```
TS db.ts TS routes\_register TS routes\_login auth-context.ts page.tsx
app > auth > register > TS routes > ...
1 import { type NextRequest, NextResponse } from "next/server"
2 import { cookies } from "next/headers"
3 import { createUser, getUserByEmail } from "../../../../lib/db"
4 import bcrypt from "bcryptjs"
5
6 Windshaft Refactor | Explain | Generate JSDoc | X
7 export async function POST(request: NextRequest) {
8     try {
9         const { name, email, password } = await request.json()
10
11         // Validate input
12         if (!name || !email || !password) {
13             return NextResponse.json(
14                 { error: "Name, email, and password are required" },
15                 { status: 400 }
16             )
17         }
18
19         // Check if user already exists
20         const existingUser = await getUserByEmail(email)
21         if (existingUser) {
22             return NextResponse.json(
23                 { error: "User with this email already exists" },
24                 { status: 409 }
25             )
26
27         // Hash the password
28         const hashedPassword = await bcrypt.hash(password, 10)
29
30         // Create the user
31         const user = await createUser({
32             name,
33             email,
34             password: hashedPassword,
35             role: "user",
36         })
37
38         if (!user) {
39             throw new Error("Failed to create user")
40         }
41 }
```

The status bar at the bottom shows various toolbars and system information, including the date '07/05/2025' and time '18:42'.



```
File Edit Selection View Go Run Terminal Help < -> bookshop
EXPLORER ... TS db.ts TS route.ts ...register TS route.ts ...login auth-context.ts page.tsx
BOOK... .next app about api\auth login logout me route.ts register
TS route.ts
app > api > auth > register > TS route.ts > ...
6 export async function POST(request: NextRequest) {
38   if (!user) {
39     throw new Error("Failed to create user")
40   }
41
42   // Create a session
43   const session = {
44     id: user.id,
45     name: user.name,
46     email: user.email,
47     role: user.role,
48   }
49
50   // Create session token
51   const token = Buffer.from(JSON.stringify(session)).toString("base64")
52
53   // Set the session cookie
54   ;await
55   // Set the session cookie
56   cookies().set({
57     name: "session",
58     value: token,
59     httpOnly: true,
60     path: "/",
61     secure: process.env.NODE_ENV === "production",
62     maxAge: 60 * 60 * 24 * 7, // 1 week
63     sameSite: "strict",
64   })
65
66   // Return the user data (excluding sensitive information)
67   return NextResponse.json({
68     id: user.id,
69     name: user.name,
70     email: user.email,
71     role: user.role,
72   }, { status: 201 })
73
74 } catch (error) {
75   console.error("Registration error:", error)
76   return NextResponse.json(
77     { error: "Internal server error" },
78     { status: 500 }
79   )
80 }
81 }
```

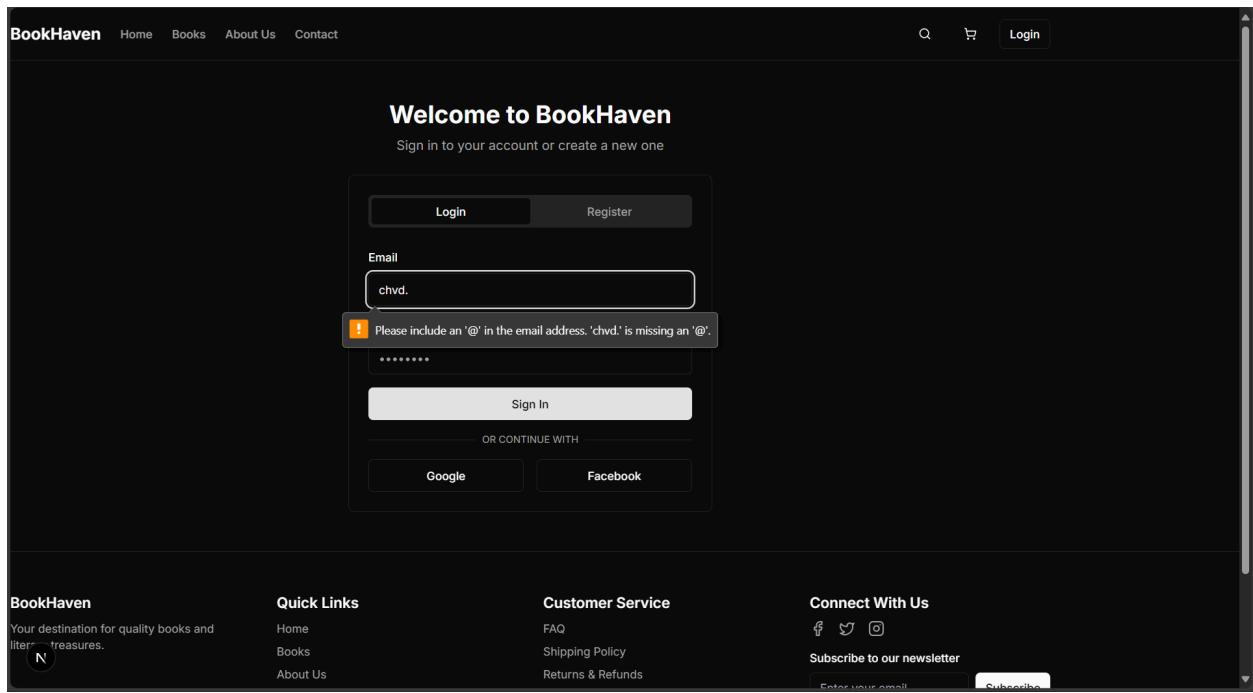
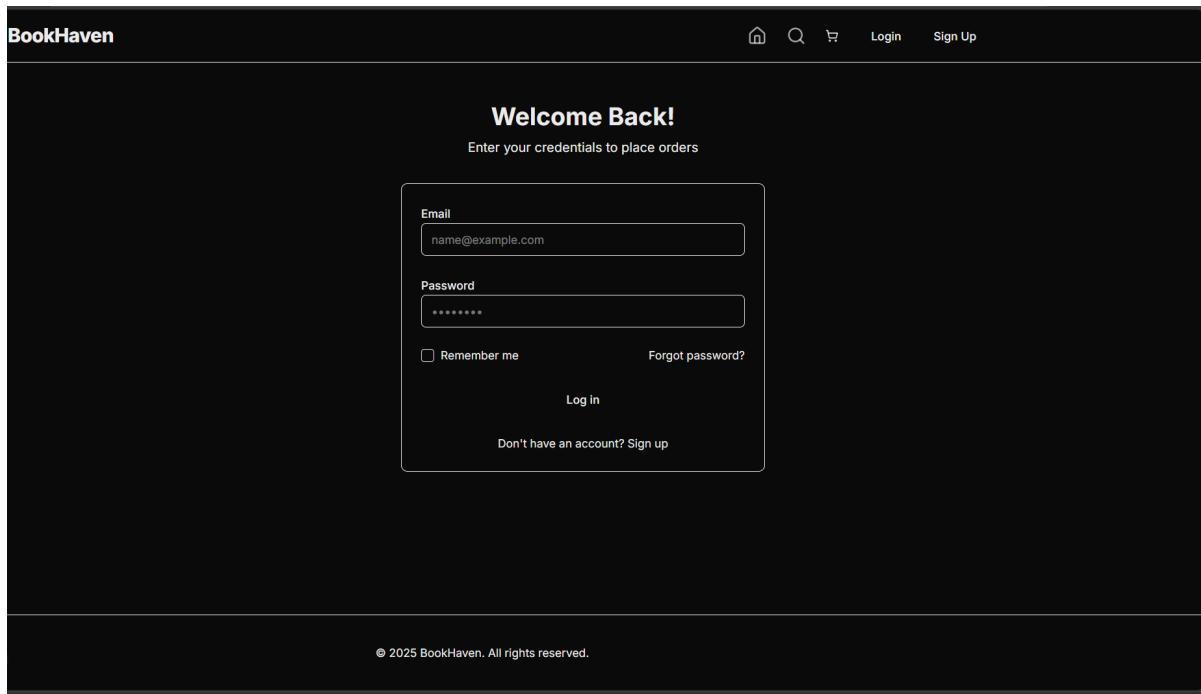
28°C Light rain



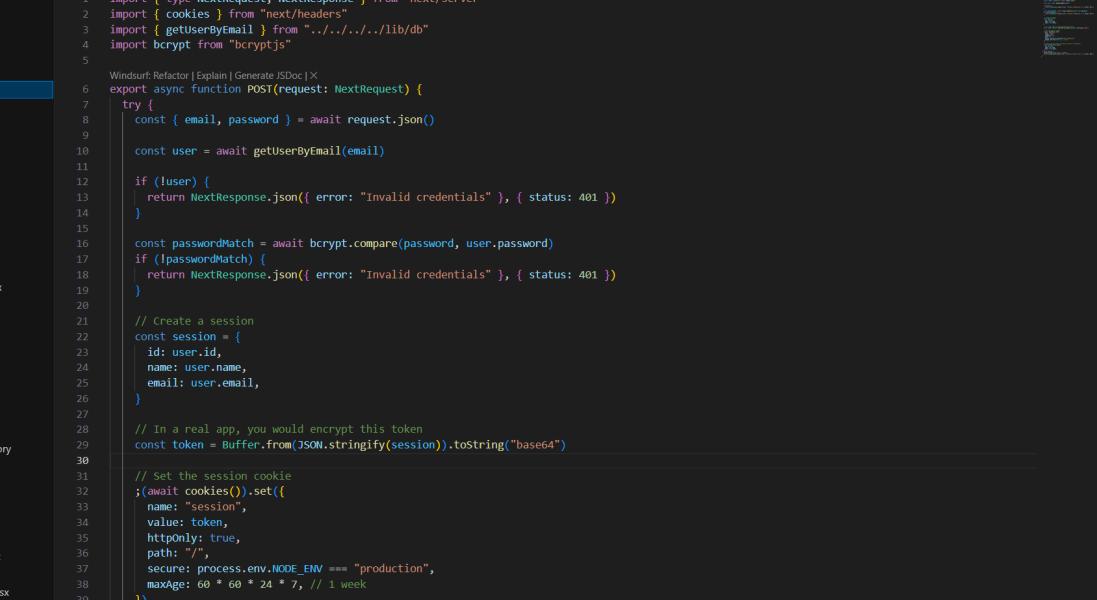
```
File Edit Selection View Go Run Terminal Help < -> bookshop
EXPLORER ... TS db.ts TS route.ts ...register TS route.ts ...login auth-context.ts page.tsx
BOOK... .next app about api\auth login logout me route.ts register
TS route.ts
app > api > auth > register > TS route.ts > ...
6 export async function POST(request: NextRequest) {
38   const token = Buffer.from(JSON.stringify(session)).toString("base64")
39
40   // Set the session cookie
41   ;await
42   // Set the session cookie
43   cookies().set({
44     name: "session",
45     value: token,
46     httpOnly: true,
47     path: "/",
48     secure: process.env.NODE_ENV === "production",
49     maxAge: 60 * 60 * 24 * 7, // 1 week
50     sameSite: "strict",
51   })
52
53   // Return the user data (excluding sensitive information)
54   return NextResponse.json({
55     id: user.id,
56     name: user.name,
57     email: user.email,
58     role: user.role,
59   }, { status: 201 })
60
61 } catch (error) {
62   console.error("Registration error:", error)
63   return NextResponse.json(
64     { error: "Internal server error" },
65     { status: 500 }
66   )
67 }
68 }
```

28°C Light rain

## ● Login Page



Code



The screenshot shows a Microsoft Edge browser window with the following details:

- Address Bar:** bookshop
- Page Content:** A list of books with titles: "The Great Gatsby", "War and Peace", "Pride and Prejudice", "Moby-Dick", "The Catcher in the Rye", "The Hobbit", "The Lord of the Rings", "The Da Vinci Code", "The Pillars of the Earth", and "The Handmaid's Tale".
- Developer Tools:** The F12 developer tools are open, showing the Network tab with several requests listed.
- Code Editor:** An integrated code editor on the left shows the file `route.ts` with TypeScript code for handling user login requests.

```
app > api > auth > login > TS route.ts ...login > POST
1 import { type NextRequest, NextResponse } from "next/server"
2 import { cookies } from "next/headers"
3 import { getUserByEmail } from "../../../../lib/db"
4 import bcrypt from "bcryptjs"
5
6 Windsurf:Refactor | Explain | Generate JSDoc | X
7 export async function POST(request: NextRequest) {
8     try {
9         const { email, password } = await request.json()
10
11         const user = await getUserByEmail(email)
12
13         if (!user) {
14             return NextResponse.json({ error: "Invalid credentials" }, { status: 401 })
15         }
16
17         const passwordMatch = await bcrypt.compare(password, user.password)
18         if (!passwordMatch) {
19             return NextResponse.json({ error: "Invalid credentials" }, { status: 401 })
20         }
21
22         // Create a session
23         const session = {
24             id: user.id,
25             name: user.name,
26             email: user.email,
27         }
28
29         // In a real app, you would encrypt this token
30         const token = Buffer.from(JSON.stringify(session)).toString("base64")
31
32         // Set the session cookie
33         ;(await cookies()).set({
34             name: "session",
35             value: token,
36             httpOnly: true,
37             path: "/",
38             secure: process.env.NODE_ENV === "production",
39             maxAge: 60 * 60 * 24 * 7, // 1 week
40         })
41
42         // Return the user data (excluding sensitive information)
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `db.ts`, `route.ts`, `register`, `login`, `auth-context.tsx`, and `page.tsx`.
- Editor (Center):** Displays the `route.ts` file content in TypeScript. The code handles a POST request for logging in, checks if the password matches, creates a session, sets a session cookie, and returns user data (excluding sensitive info). It also catches errors and returns a 500 Internal Server Error.
- Bottom Status Bar:** Includes icons for file operations (New, Open, Save, etc.), a search bar, a file browser, and various developer tools like Start JSON Server, Linting, and TypeScript support.
- Bottom Right:** Shows system status (28°C, Light rain), a Chat limit reached message, and a timestamp (07/05/2025).

## Book Management Module

- Book Listing and search

**Sort By**

Newest Arrivals

**Categories**

- Fiction
- Non-Fiction
- Science Fiction
- Fantasy
- Mystery
- Thriller
- Romance
- Biography
- History
- Self-Help
- Business
- Children

**Price Range**

**Format**

**Fantasm and Fiction**  
Peter Schwenger  
**\$0.00** Add to Cart

**Fiction, Folklore, Fantasy & Poetry for Children, 1876-1985**  
Beverly Lamar  
**\$0.00** Add to Cart

**Fiction, Folklore, Fantasy & Poetry for Children, 1876-1985**  
Unknown author  
**\$0.00** Add to Cart

## Code

```

use client"
import { useState } from "react"
import { useRouter, useSearchParams } from "next/navigation"
import { Button } from "@components/ui/button"
import { Accordion, AccordionContent, AccordionItem, AccordionTrigger } from "@components/ui/accordion"
import { Checkbox } from "@components/ui/checkbox"
import { Label } from "@components/ui/label"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@components/ui/select"

const [selectedCategories, setSelectedCategories] = useState<string[]>([searchParams.get("category")?.split(",") || []])

const categories = [
  "Fiction",
  "Non-Fiction",
  "Science Fiction",
  "Fantasy",
  "Mystery",
  "Thriller",
  "Romance",
  "Biography",
  "History",
  "Self-Help",
  "Business",
  "Children",
]

const handleCategoryChange = (category: string) => {
  setSelectedCategories((prev) => {
    if (prev.includes(category)) {
      return prev.filter((c) => c !== category)
    } else {
      return [...prev, category]
    }
  })
}

```

File Edit Selection View Go Run Terminal Help ⏵ ⏴ bookshop

EXPLORER BOOKSHOP

components > book-filter.tsx > BookFilter

components > search-bar.tsx > theme-provider.tsx

context > cart-context-client.tsx

books-books-ts > node\_modules > package-lock.json > package.json

hooks > hooks

lib > auth-context.tsx

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

4 △ 0 BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link Open Website

Ln 15, Col 76 Spaces: 2 UTF-8 LF TypeScript JSX Chat limit reached BLACKBOXAI: Open Chat Windsurf: (...) 18:51 07/05/2025

28°C Light rain

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists the project structure under "BOOKSHOP".
- Code Editor:** The main area displays the file "db.ts".
- Search Bar:** At the top center, it says "bookshop".
- Activity Bar:** At the bottom, it includes icons for "BLACKBOX Chat", "Add Logs", "CyberCoder", "Improve Code", "Share Code Link", "Open Website", "Ln 15, Col 76", "Spaces: 2", "UTF-8", "LF", "TypeScript JSX", "Chat limit reached", "BLACKBOXAI: Open Chat", and "Windsurf: (...)".
- Status Bar:** At the bottom right, it shows the date "07/05/2023".

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `book-filter.tsx`, `auth-context.tsx`, and various configuration and utility files.
- Editor Area (Center):** Displays the content of `book-filter.tsx`. The code uses JSX to render a rating selector and an accordion component for filtering by category and price range.
- Status Bar (Bottom):** Shows the following information:
  - File counts: 4 △ 0
  - File types: BLACKBOX Chat, Add Logs, CyberCoder, Improve Code, Share Code Link, Open Website
  - Editor settings: In 15, Col 76, Spaces: 2, UTF-8, LF
  - Language: TypeScript JSX
  - A warning: Chat limit reached
  - System status: 28°C, Light rain
  - System date: 07/05/2025

```
components > book-filter.tsx > BookFilter
11 export function BookFilter() {
100   </AccordionContent>
101   </AccordionItem>
102   <AccordionItem value="price">
103     <AccordionTrigger>Price Range</AccordionTrigger>
104     <AccordionContent>
105       <div className="space-y-2">
106         <div className="flex items-center space-x-2">
107           <Checkbox id="price-under-10" />
108           <Label htmlFor="price-under-10" className="text-sm font-normal cursor-pointer">
109             Under $10
110           </Label>
111         </div>
112         <div className="flex items-center space-x-2">
113           <Checkbox id="price-10-20" />
114           <Label htmlFor="price-10-20" className="text-sm font-normal cursor-pointer">
115             $10 - $20
116           </Label>
117         </div>
118         <div className="flex items-center space-x-2">
119           <Checkbox id="price-20-30" />
120           <Label htmlFor="price-20-30" className="text-sm font-normal cursor-pointer">
121             $20 - $30
122           </Label>
123         </div>
124         <div className="flex items-center space-x-2">
125           <Checkbox id="price-over-30" />
126           <Label htmlFor="price-over-30" className="text-sm font-normal cursor-pointer">
127             Over $30
128           </Label>
129         </div>
130       </AccordionContent>
131     </AccordionItem>
132     <AccordionItem value="format">
133       <AccordionTrigger>Format</AccordionTrigger>
134       <AccordionContent>
135         <div className="space-y-2">
136           <div className="flex items-center space-x-2">
137             <Checkbox id="format-hardcover" />
138           </div>
139         </AccordionContent>
140       </AccordionItem>
141     </AccordionItem>
142     <AccordionItem value="applyFilters">
143       <AccordionContent>
144         <div className="flex flex-col space-y-2">
145           <Button onClick={applyFilters}>Apply Filters</Button>
146           <Button variant="outline" onClick={resetFilters}>
147             Reset Filters
148           </Button>
149         </div>
150       </AccordionContent>
151     </AccordionItem>
152   </Accordion>
153 }
```

```
components > book-filter.tsx > BookFilter
11 export function BookFilter() {
100   </AccordionContent>
101   </AccordionItem>
102   <AccordionItem value="format">
103     <AccordionTrigger>Format</AccordionTrigger>
104     <AccordionContent>
105       <div className="space-y-2">
106         <div className="flex items-center space-x-2">
107           <Checkbox id="format-hardcover" />
108           <Label htmlFor="format-hardcover" className="text-sm font-normal cursor-pointer">
109             Hardcover
110           </Label>
111         </div>
112         <div className="flex items-center space-x-2">
113           <Checkbox id="format-paperback" />
114           <Label htmlFor="format-paperback" className="text-sm font-normal cursor-pointer">
115             Paperback
116           </Label>
117         </div>
118         <div className="flex items-center space-x-2">
119           <Checkbox id="format-ebook" />
120           <Label htmlFor="format-ebook" className="text-sm font-normal cursor-pointer">
121             eBook
122           </Label>
123         </div>
124         <div className="flex items-center space-x-2">
125           <Checkbox id="format-audiobook" />
126           <Label htmlFor="format-audiobook" className="text-sm font-normal cursor-pointer">
127             Audiobook
128           </Label>
129         </div>
130       </AccordionContent>
131     </AccordionItem>
132     <AccordionItem value="applyFilters">
133       <AccordionContent>
134         <div className="flex flex-col space-y-2">
135           <Button onClick={applyFilters}>Apply Filters</Button>
136           <Button variant="outline" onClick={resetFilters}>
137             Reset Filters
138           </Button>
139         </div>
140       </AccordionContent>
141     </AccordionItem>
142   </Accordion>
143 }
```

## Shopping Module

- Shopping Cart

## Your Cart

Items (3)

	Pride and Prejudice LKR 1750.00	LKR 1750.00	Remove
	Rich Dad Poor Dad LKR 1500.00	LKR 1500.00	Remove
	Ikigai LKR 1500.00	LKR 1500.00	Remove

[Clear cart](#)

**Order Summary**

Subtotal	LKR 4750.00
Shipping	LKR 300.00
<input style="width: 100%; height: 20px; border: 1px solid #ccc; margin-bottom: 5px;" type="text"/> Coupon code	<a href="#">Apply</a>
Total	LKR 5050.00

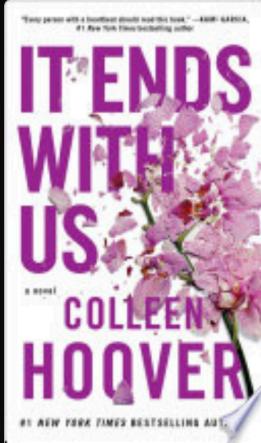
[Proceed to Checkout →](#)

**Satisfaction Guaranteed**  
30-day money-back guarantee if you're not satisfied.

- Home page

BookHaven

Most Common
[View All](#)

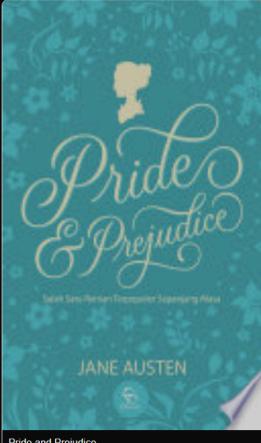


**IT ENDS WITH US**  
a novel  
**COLLEEN HOOPER**

"Every person with a breakable heart needs this book." —KAREN KAROLINA, #1 New York Times bestselling author

It Ends with Us  
Colleen Hoover

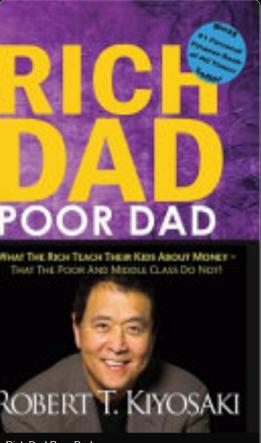
LKR 1500.00



**Pride & Prejudice**  
Sister Jane Austen's Feminist Reimagining Novel

Pride and Prejudice  
Jane Austen

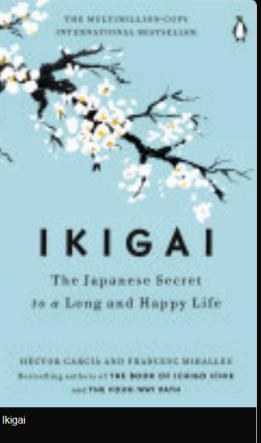
LKR 2000.00



**RICH DAD POOR DAD**  
WHAT THE RICH TEACH THEIR KIDS ABOUT MONEY — THAT THE POOR AND MIDDLE CLASS DO NOT!

Rich Dad Poor Dad  
Robert T Kiyosaki

LKR 2250.00



**IKIGAI**  
The Japanese Secret to a Long and Happy Life

THE MILLION-COPY INTERNATIONAL BESTSELLER

Ikigai  
Héctor García, Francesc Miralles

LKR 2500.00

## External API Integration

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "BOOKSTORE" with files like app, api, auth, current, login, logout, me, profile, signup, books, orders, cart, page.tsx, favicon.ico, globals.css, and layout.tsx.
- Editor:** The main editor window displays a file named "TS route.ts". The code is a Next.js API route for fetching books from Google Books API. It includes imports for NextRequest and NextResponse, defines a constant for the API endpoint, and handles a search query with parameters for limit and sort. It then fetches the data from the API and returns it.
- Terminal:** The terminal shows the output of a build or run command, indicating successful compilation and execution of the application.
- Output:** The output panel shows various logs, including network requests and responses for different routes like /api/auth/me, /api/books/featured, and /api/books/featured with specific parameters.
- Problems:** The problems panel shows no errors or warnings.
- Right Panel:** The sidebar on the right contains several tabs: powershell, node, and a "node" tab which is currently selected.

This screenshot is nearly identical to the one above, showing the same project structure, file content, and terminal output. The main difference is in the terminal output, where the logs show a different sequence of requests and responses, likely due to a different state or configuration of the application.

The screenshot shows a Microsoft Edge browser window with a Next.js project open. The left sidebar displays the project structure:

- BOOKSTORE
  - app
    - login
    - orders
    - profile
    - signup
  - components
  - context
    - auth-context.tsx
    - cart-context.tsx
  - hooks
  - lib
    - api-helpers.ts (selected)
    - auth.ts
    - db.ts
    - next-auth.ts
    - prisms.ts
    - utils.ts
  - node\_modules
  - prisma
  - public
  - styles
    - # globals.css
  - .env
  - .gitignore
  - next-env.d.ts
  - next.config.ts
    - package-lock.json
    - package.json
  - postcss.config.mic
- OUTLINE

The main content area shows the code for `api-helpers.ts`:

```
lib > TS api-helpers.ts > ...
1 import { cookies } from "next/headers"
2 import jwt from "jsonwebtoken"
3
4 // JWT secret key (should be in .env file in production)
5 const JWT_SECRET = process.env.JWT_SECRET || "your-secret-key"
6
7 // Cache API responses
8 Windsurf:Refactor | Explain | X
9 export const cacheResponse = (ttl = 60) => {
10   return {
11     "Cache-Control": `public, s-maxage=${ttl}, stale-while-revalidate=${ttl * 2}`,
12   }
13
14 // Get authenticated user from token
15 Windsurf:Refactor | Explain | X
16 export const getAuthUser = async () => {
17   try {
18     const token = (await cookies()).get("auth_token")?.value
19
20     if (!token) {
21       return null
22     }
23
24     const decoded = jwt.verify(token, JWT_SECRET) as {
25       id: string
26       username: string
27       email: string
28     }
29
30     return {
31       id: decoded.id,
32       username: decoded.username,
33       email: decoded.email,
34     }
35   } catch (error) {
36     console.error(`Error getting auth user: ${error}`)
37   }
38 }
39
40 // Format API error response
```

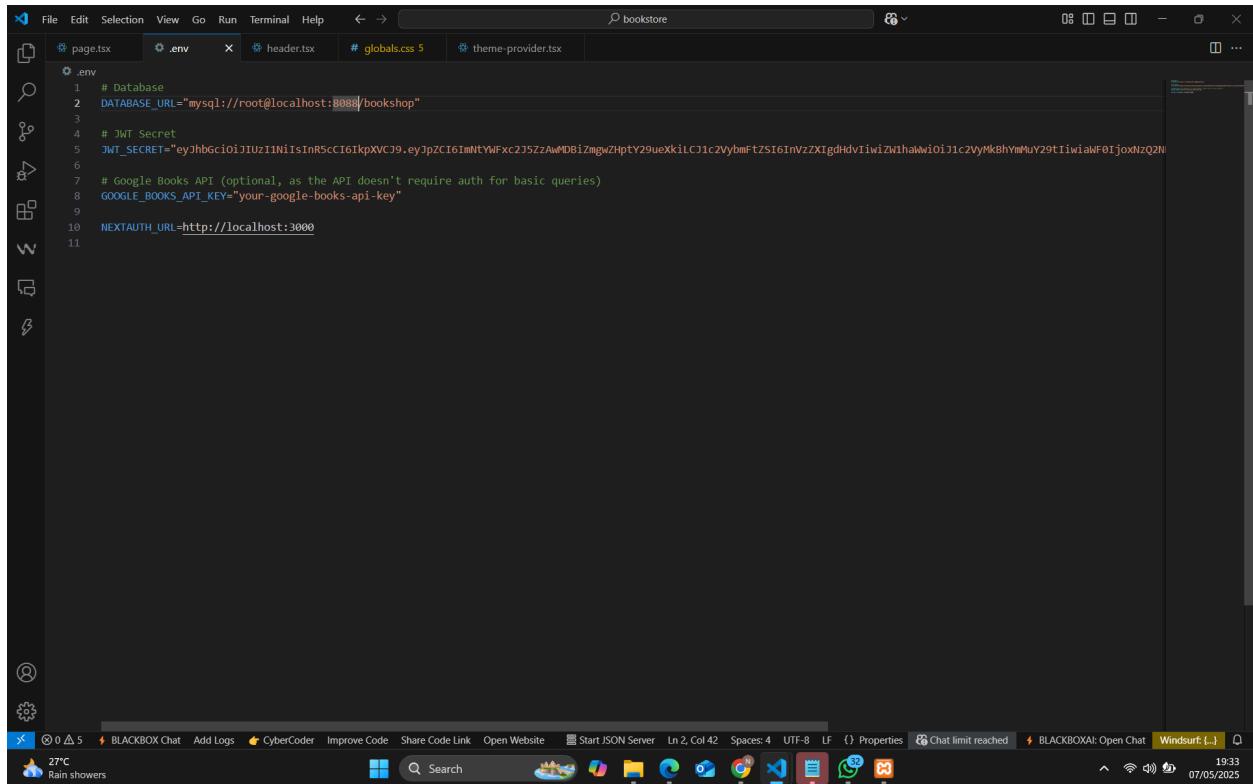
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure. The `api-helpers.ts` file is selected in the tree view.
- Editor (Center):** Displays the code for `api-helpers.ts`. The code includes functions for getting auth users and formatting API error responses.
- Bottom Bar:** Contains various icons for extensions like BLACKBOX Chat, CyberCoder, Improve Code, Share Code Link, Open Website, Start JSON Server, and others.

```
lib > ts api-helpers.ts > ...
15 export const getAuthUser = async () => {
16   try {
17     const user = await getUser();
18     if (!user) {
19       console.error("Error getting auth user:", error);
20       return null;
21     }
22   }
23
24   // Format API error response
25   WIndows:Refactor | Explain | X
26   export const formatError = (message: string, status = 400) => {
27     return {
28       error: {
29         message,
30         status,
31       },
32     };
33   }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48 }
```

## 7. Security Implementation

JWT tokens for session management



The screenshot shows a code editor window with a dark theme. The left sidebar lists files: page.tsx, .env, header.tsx, globals.css, and theme-provider.tsx. The main editor area displays the .env file content:

```
# Database
DATABASE_URL="mysql://root@localhost:8088/bookshop"

# JWT Secret
JWT_SECRET="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16ImNyWFxc2J5ZzAwMDBizmgw2HptY29ueXkICJ1c2VybmltZS16InVzZXlgdHdVliwiZWlhaWwiOjIc2VyMkBhYmMuY29tliwiaWF0IjoxNzQH"

# Google Books API (optional, as the API doesn't require auth for basic queries)
GOOGLE_BOOKS_API_KEY="your-google-books-api-key"

NEXTAUTH_URL=http://localhost:3000
```

The bottom status bar shows system icons like battery level (27°C Rain showers), weather (27°C Rain showers), and system date/time (19:33 07/05/2025).

## 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