



BSc. (Hons) in Software Engineering

Faculty of Computing



Online Bookshop Assignment

Module: Advanced Web Technologies

Name: H.A.C.D.P. Hettiarachchi

Student Number: M20000210007

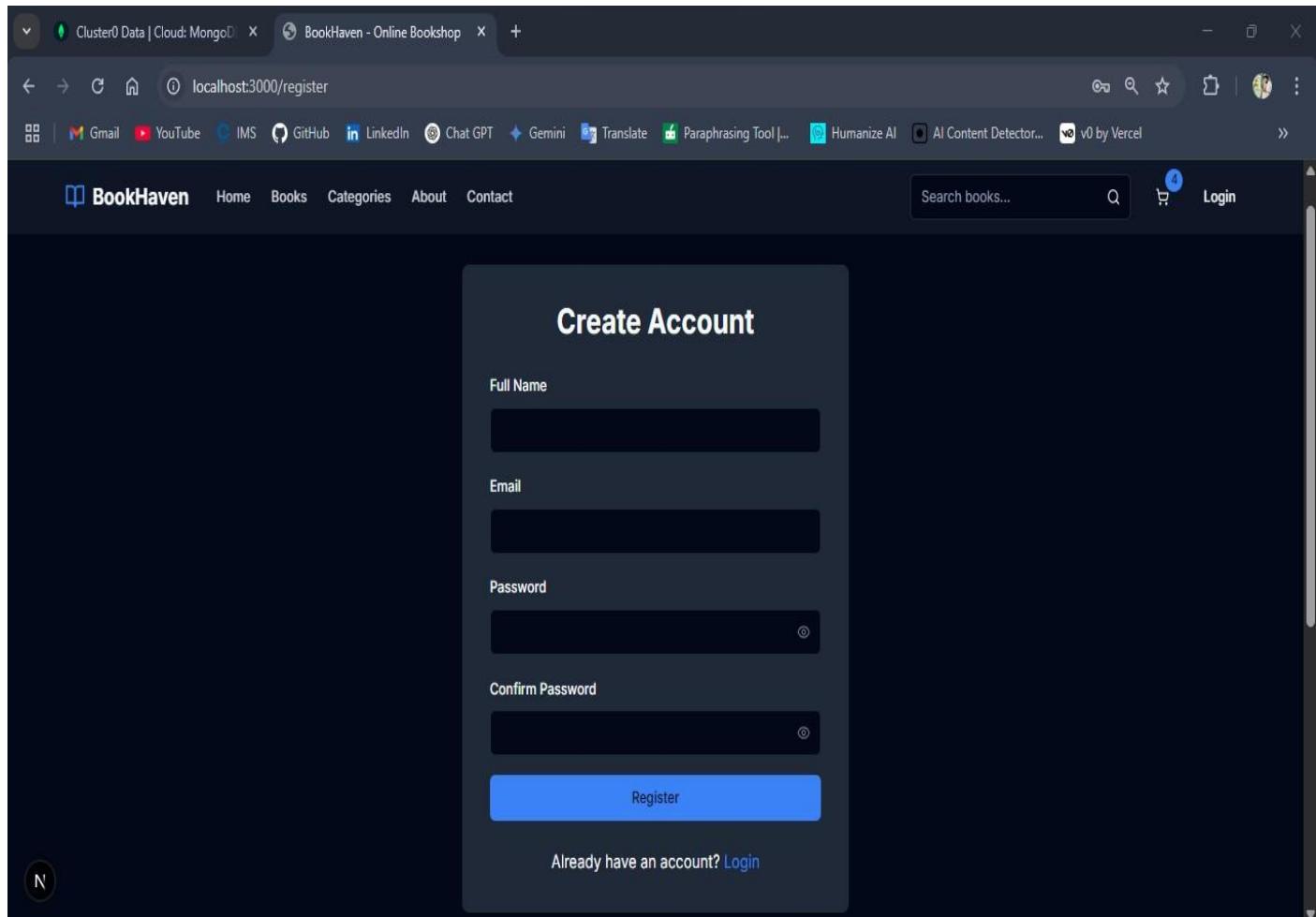
Table of Contents

1. User Interface Documentation Requirements	3
 1.1. User Authentication Screens	3
 1.2. Book Listing and Search.....	11
 1.3. Shopping Cart	18
 1.4. Checkout Process	23
 1.5. User Profile.....	31

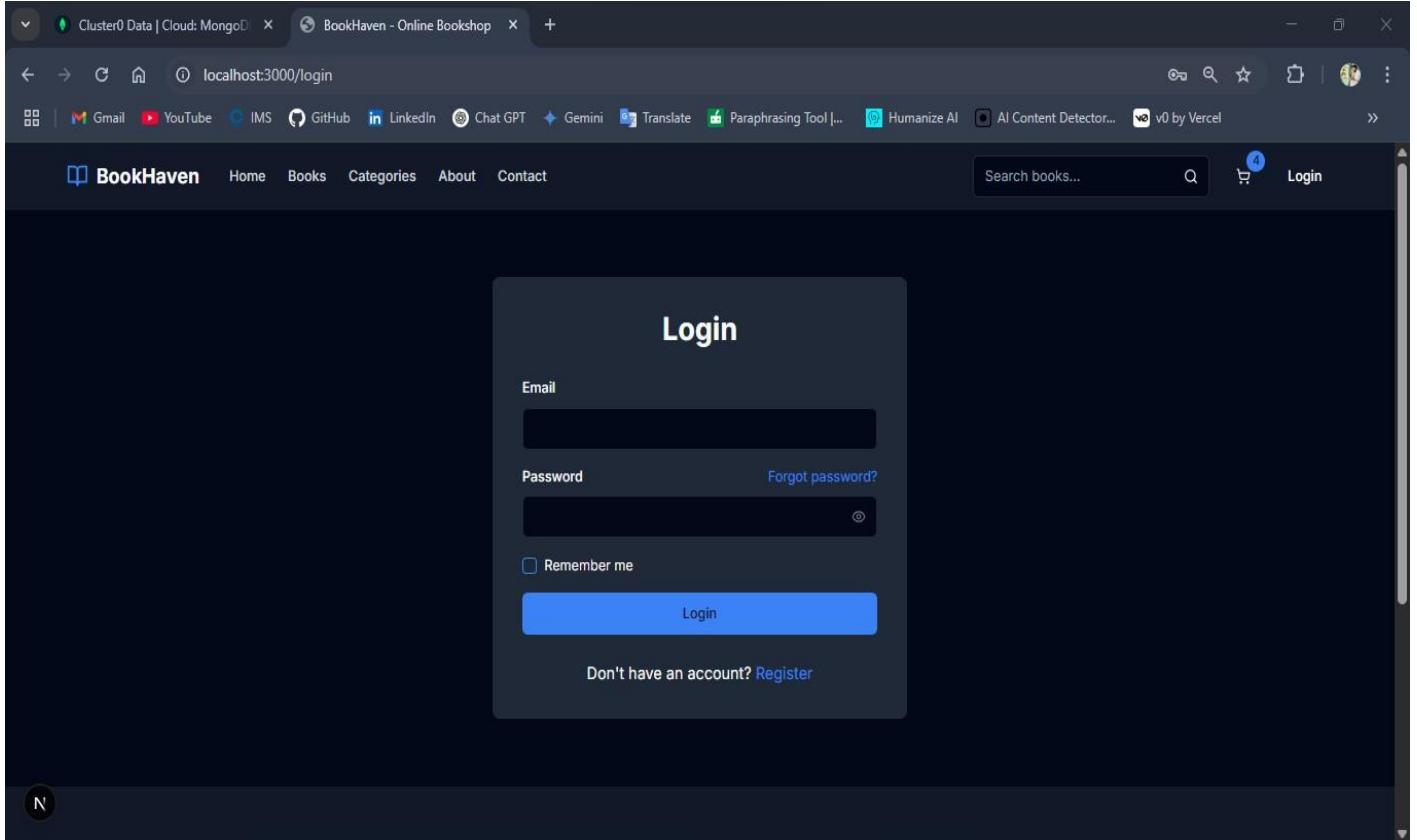
1. User Interface Documentation Requirements

1.1. User Authentication Screens

- Registration page screenshot



➤ Login page screenshot



➤ Associated HTML/CSS/JavaScript code screenshots for each page
1.Registration page code screenshots

A screenshot of a code editor (VS Code) displaying the "page.tsx" file from the "register" directory of a Next.js application. The code is written in TypeScript and React. It defines a functional component "RegisterPage" that handles user registration. The component uses useState to manage form data and useAuth to handle authentication. It also imports various UI components like Link, Button, Input, and Label. The code editor interface shows the file structure on the left and the code content on the right, with various tools and status indicators at the bottom.

File Edit Selection View Go Run ...

Online-bookshop

EXPLORER

ONLINE-BOOKSHOP

- > .next
- > app
 - > about
 - > api
 - > books
 - > cart
 - > categories
 - > checkout
 - > confirmation
 - > contact
 - > login
 - > orders
 - > profile
 - > register
 - page.tsx M
- > components
- > context
- > hooks
- > lib
- > node_modules
- > public
- > scripts
- > globals.css
- > layout.tsx
- > page.tsx
- > timeline

OUTLINE

TIMELINE

page.tsx M

```

app > register > page.tsx > RegisterPage > handleSubmit
16  export default function RegisterPage() {
17
18    Windsurf: Refactor | Explain | Generate JSDoc | X
19    const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
20      const { name, value } = e.target
21      setFormData((prev) => ({ ...prev, [name]: value }))
22    }
23
24    Windsurf: Refactor | Explain | Generate JSDoc | X
25    const handleSubmit = async (e: React.FormEvent) => {
26      e.preventDefault()
27
28      if (formData.password !== formData.confirmPassword) {
29        toast({
30          title: "Passwords don't match",
31          description: "Please make sure your passwords match.",
32          variant: "destructive",
33        })
34        return
35
36      try {
37        setIsLoading(true)
38
39        const userData = await registerUser({
40          name: formData.name,
41          email: formData.email,
42          password: formData.password,
43        })
44
45        setUser({
46          ...userData,
47          profileImageUrl: userData.profileImageUrl || "", // Provide a default value if missing
48        })
49
50      } finally {
51        setIsLoading(false)
52      }
53
54    }
55
56    Windsurf: Refactor | Explain | Generate JSDoc | X
57    handleSubmit(e)
58
59  }

```

Not Committed Yet Ln 57, Col 16 Spaces:2 UTF-8 CRLF TypeScript JSX Go Live Windsurf [...] Prettier

File Edit Selection View Go Run ...

Online-bookshop

EXPLORER

ONLINE-BOOKSHOP

- > .next
- > app
 - > about
 - > api
 - > books
 - > cart
 - > categories
 - > checkout
 - > confirmation
 - > contact
 - > login
 - > orders
 - > profile
 - > register
 - page.tsx M
- > components
- > context
- > hooks
- > lib
- > node_modules
- > public
- > scripts
- > globals.css
- > layout.tsx
- > page.tsx
- > timeline

OUTLINE

TIMELINE

page.tsx M

```

app > register > page.tsx > RegisterPage > handleSubmit
16  export default function RegisterPage() {
17
18    const handleSubmit = async (e: React.FormEvent) => {
19
20      Windsurf: Refactor | Explain | Generate JSDoc | X
21      const { name, value } = e.target
22      setFormData((prev) => ({ ...prev, [name]: value }))
23
24      Windsurf: Refactor | Explain | Generate JSDoc | X
25      const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
26        const { name, value } = e.target
27        setFormData((prev) => ({ ...prev, [name]: value }))
28
29      }
30
31      Windsurf: Refactor | Explain | Generate JSDoc | X
32      const toast = (options) => {
33        const toastContainer = document.querySelector('.toast')
34        if (!toastContainer) {
35          const toastDiv = document.createElement('div')
36          toastDiv.classList.add('toast')
37          document.body.appendChild(toastDiv)
38        }
39        const toastMessage = document.createElement('div')
40        toastMessage.classList.add('toast-message')
41        toastMessage.textContent = options.message
42        toastDiv.appendChild(toastMessage)
43
44        setTimeout(() => {
45          toastDiv.removeChild(toastMessage)
46        }, options.duration)
47
48      }
49
50      Windsurf: Refactor | Explain | Generate JSDoc | X
51      const registerUser = async (userData) => {
52        const response = await fetch('/api/register', {
53          method: 'POST',
54          headers: {
55            'Content-Type': 'application/json',
56          },
57          body: JSON.stringify(userData),
58        })
59
60        if (response.ok) {
61          const data = await response.json()
62          return data.user
63        }
64
65        throw new Error('Registration failed')
66
67      }
68
69      Windsurf: Refactor | Explain | Generate JSDoc | X
70      const toastOptions = {
71        title: 'Registration successful',
72        description: 'Welcome to BookHaven!',
73        variant: 'success',
74      }
75
76      Windsurf: Refactor | Explain | Generate JSDoc | X
77      const errorToastOptions = {
78        title: 'Registration failed',
79        description: 'There was an error creating your account. Please try again.',
80        variant: 'destructive',
81      }
82
83      Windsurf: Refactor | Explain | Generate JSDoc | X
84      const router = useRouter()
85
86      router.push('/')
87
88      try {
89        const userData = await registerUser({
90          name: formData.name,
91          email: formData.email,
92          password: formData.password,
93        })
94
95        setUser(userData)
96
97        toast(toastOptions)
98
99      } catch (error) {
100        console.error('Registration error:', error)
101        toast(errorToastOptions)
102
103      } finally {
104        setIsLoading(false)
105      }
106
107
108      return (
109        <div className="container mx-auto px-4 py-16 flex justify-center">
110          <div className="w-full max-w-md">
111            <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-8">
112              <h1 className="text-3xl font-bold mb-6 text-center">Create Account</h1>
113
114              <form onSubmit={handleSubmit} className="space-y-4">
115                <div className="space-y-2">
116                  <Label htmlFor="name">Full Name</Label>
117
118
119                </div>
120              </form>
121            </div>
122          </div>
123        </div>
124      )
125
126    }
127
128    Windsurf: Refactor | Explain | Generate JSDoc | X
129    handleSubmit(e)
130
131  }

```

Not Committed Yet Ln 57, Col 16 Spaces:2 UTF-8 CRLF TypeScript JSX Go Live Windsurf [...] Prettier

The screenshot displays two instances of a code editor, likely VS Code, showing the same file: `page.tsx` from a project named `ONLINE-BOOKSHOP`.

Initial State (Top Editor):

```
16  export default function RegisterPage() {
    ...
    <div className="space-y-2">
        <Label htmlFor="name">Full Name</Label>
        <Input id="name" name="name" value={formData.name} onChange={handleChange} required />
    </div>

    <div className="space-y-2">
        <Label htmlFor="email">Email</Label>
        <Input id="email" name="email" type="email" value={formData.email} onChange={handleChange} required />
    </div>

    <div className="space-y-2">
        <Label htmlFor="password">Password</Label>
        <div className="relative">
            <Input
                id="password"
                name="password"
                type={showPassword ? "text" : "password"}
                value={formData.password}
                onChange={handleChange}
                required
                minLength={8}
            />
            <Button
                type="button"
                variant="ghost"
                size="icon"
                className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
                onClick={() => setShowPassword(!showPassword)}
            >
                {showPassword ? (
                    <EyeOff className="h-4 w-4 text-gray-500" />
                ) : (
                    <Eye className="h-4 w-4 text-gray-500" />
                )}
            </Button>
        </div>
    </div>
```

State After Click (Bottom Editor):

```
16  export default function RegisterPage() {
    ...
    <div className="space-y-2">
        <Label htmlFor="name">Full Name</Label>
        <Input id="name" name="name" value={formData.name} onChange={handleChange} required />
    </div>

    <div className="space-y-2">
        <Label htmlFor="email">Email</Label>
        <Input id="email" name="email" type="email" value={formData.email} onChange={handleChange} required />
    </div>

    <div className="space-y-2">
        <Label htmlFor="confirmPassword">Confirm Password</Label>
        <div className="relative">
            <Input
                id="confirmPassword"
                name="confirmPassword"
                type={showConfirmPassword ? "text" : "password"}
                value={formData.confirmPassword}
                onChange={handleChange}
                required
                minLength={8}
            />
            <Button
                type="button"
                variant="ghost"
                size="icon"
                className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
                onClick={() => setShowConfirmPassword(!showConfirmPassword)}
            >
                {showConfirmPassword ? (
                    <EyeOff className="h-4 w-4 text-gray-500" />
                ) : (
                    <Eye className="h-4 w-4 text-gray-500" />
                )}
            </Button>
        </div>
    </div>
```

The code editor interface includes a sidebar with icons for file operations like Open, Save, and Delete, and a status bar at the bottom showing file details and toolbars for Go Live, Windsurf, and Prettier.

The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "ONLINE-BOOKSHOP". The "register" folder is expanded, showing "page.tsx" as the active file.
- Code Editor (Center):** Displays the content of "page.tsx". The code is a TypeScript component named "RegisterPage" that handles user registration. It includes logic for password visibility, form submission, and account creation.
- Search Bar (Top):** Contains the text "Online-bookshop".
- Activity Bar (Bottom):** Includes icons for "main*", "Not Committed Yet", "Ln 57, Col 16", "Spaces: 2", "UTF-8", "CRLF", "TypeScript JSX", "Go Live", "Windsurf: {}", "Prettier", and "File History".

- Login page code screenshots

The screenshot shows two instances of the VS Code interface, each displaying the same file: `loading.tsx`. The file contains TypeScript code for a login page, including imports from React, next, and auth-context, and logic for handling user authentication and displaying success/failure messages.

```
File Edit Selection View Go Run ... Online-bookshop Explorer Explorer ONLINE-BOOKSHOP app > login > page.tsx > LoginPage > handleSubmit
1 "use client"
2
3 import type React from "react"
4
5 import { useState } from "react"
6 import { useRouter, useSearchParams } from "next/navigation"
7 import Link from "next/link"
8 import { useAuth } from "@context/auth-context"
9 import { toast } from "@components/ui/use-toast"
10 import { Button } from "@components/ui/button"
11 import { Input } from "@components/ui/input"
12 import { Label } from "@components/ui/label"
13 import { Checkbox } from "@components/ui/checkbox"
14 import { loginUser } from "@lib/auth"
15 import { Eye, EyeOff } from "lucide-react"
16
17 Windsurf: Refactor | Explain | Generate JSDoc | X
18 export default function LoginPage() {
19   const [email, setEmail] = useState("")
20   const [password, setPassword] = useState("")
21   const [rememberMe, setRememberMe] = useState(false)
22   const [isLoading, setIsLoading] = useState(false)
23   const [showPassword, setShowPassword] = useState(false)
24
25   const { user } = useAuth()
26   const toast = useToast()
27   const router = useRouter()
28   const searchParams = useSearchParams()
29   const redirectUrl = searchParams.get("redirect") || "/"
30
31   Windsurf: Refactor | Explain | Generate JSDoc | X
32   const handleSubmit = async (e: React.FormEvent) => {
33     e.preventDefault()
34
35     try {
36       setIsLoading(true)
37
38       const userData = await loginUser({ email, password, rememberMe })
39
40       setUser({
41         ...userData,
42         profileImageUrl: userData.profileImageUrl || "", // Provide a default value if undefined
43       }
44
45       toast({
46         title: "Login successful",
47         description: "Welcome back to BookHaven!",
48       }
49
50       router.push(redirectUrl)
51     } catch (error) {
52       console.error("Login error:", error)
53       toast({
54         title: "Login failed",
55         description: "Invalid email or password. Please try again.",
56         variant: "destructive",
57       }
58     } finally {
59       setIsLoading(false)
60     }
61   }
62 }
```

File Edit Selection View Go Run ...

ONLINE-BOOKSHOP

- > .next
- > app
 - > about
 - > api
 - > books
 - > cart
 - > categories
 - > checkout
 - > confirmation
 - > contact
 - > login
 - loading.tsx
 - page.tsx M
 - > orders
 - > profile
 - > register
 - > scripts
 - globals.css
 - layout.tsx
 - page.tsx
 - > components
 - > context
 - > hooks
 - > lib
 - auth-optio... M
 - auth.ts
- > OUTLINE
- > TIMELINE

page.tsx M

```

app > login > page.tsx > LoginPage > handleSubmit
17 export default function LoginPage() {
18   const handleSubmit = async (e: React.FormEvent) => {
19     ...
20     return (
21       <div className="container mx-auto px-4 py-16 flex justify-center">
22         <div className="w-full max-w-md">
23           <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-8">
24             <h1 className="text-3xl font-bold mb-6 text-center">Login</h1>
25
26             <form onSubmit={handleSubmit} className="space-y-4">
27               <div className="space-y-2">
28                 <div className="flex justify-between items-center">
29                   <Label htmlFor="email">Email</Label>
30                   <Input id="email" type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />
31                 </div>
32
33                 <div className="space-y-2">
34                   <div className="flex justify-between items-center">
35                     <Label htmlFor="password">Password</Label>
36                     <a href="/forgot-password" className="text-sm text-primary hover:underline">
37                       Forgot password?
38                     </a>
39                   </div>
40
41                   <div className="relative">
42                     <Input
43                       id="password"
44                       type={showPassword ? "text" : "password"}
45                       value={password}
46                       onChange={(e) => setPassword(e.target.value)}
47                       required
48                     />
49                     <Button
50                       type="button"
51                       variant="ghost"
52                       size="icon"
53                       className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
54                       onClick={() => setShowPassword(!showPassword)}
55                     >
56                       {showPassword ? (
57                         <EyeOff className="h-4 w-4 text-gray-500" />
58                       ) : (
59                         <Eye className="h-4 w-4 text-gray-500" />
60                       )}
61                       <span className="sr-only">{showPassword ? "Hide password" : "Show password"}</span>
62                     </Button>
63                   </div>
64                 </div>
65
66                 <div className="flex items-center space-x-2">
67                   <Checkbox
68                     id="remember-me"
69                     checked={rememberMe}
70                     onChange={(checked) => setRememberMe(checked as boolean)}
71                   />
72                   <Label htmlFor="remember-me" className="text-sm font-normal">
73                     Remember me
74                   </Label>
75                 </div>
76
77                 <Button type="submit" className="w-full" disabled={isLoading}>
78                   {isLoading ? "Logging in..." : "Login"}
79                 </Button>
80               </div>
81             </form>
82           </div>
83         </div>
84       </div>
85     )
86   )
87 }
88 
```

Ln 41, Col 9 Spaces: 2 UTF-8 CRLF ⓘ TypeScript JSX ⌂ Go Live Windsurf: [...] ⌂ Prettier ⌂

File Edit Selection View Go Run ...

ONLINE-BOOKSHOP

- > .next
- > app
 - > about
 - > api
 - > books
 - > cart
 - > categories
 - > checkout
 - > confirmation
 - > contact
 - > login
 - loading.tsx
 - page.tsx M
 - > orders
 - > profile
 - > register
 - > scripts
 - globals.css
 - layout.tsx
 - page.tsx
 - > components
 - > context
 - > hooks
 - > lib
 - auth-optio... M
 - auth.ts
- > OUTLINE
- > TIMELINE

page.tsx M

```

app > login > page.tsx > LoginPage > handleSubmit
17 export default function LoginPage() {
18   const handleSubmit = async (e: React.FormEvent) => {
19     ...
20     return (
21       <div className="container mx-auto px-4 py-16 flex justify-center">
22         <div className="w-full max-w-md">
23           <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-8">
24             <h1 className="text-3xl font-bold mb-6 text-center">Login</h1>
25
26             <form onSubmit={handleSubmit} className="space-y-4">
27               <div className="space-y-2">
28                 <div className="flex justify-between items-center">
29                   <Label htmlFor="email">Email</Label>
30                   <Input id="email" type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />
31                 </div>
32
33                 <div className="space-y-2">
34                   <div className="flex justify-between items-center">
35                     <Label htmlFor="password">Password</Label>
36                     <a href="/forgot-password" className="text-sm text-primary hover:underline">
37                       Forgot password?
38                     </a>
39                   </div>
40
41                   <div className="relative">
42                     <Input
43                       id="password"
44                       type={showPassword ? "text" : "password"}
45                       value={password}
46                       onChange={(e) => setPassword(e.target.value)}
47                       required
48                     />
49                     <Button
50                       type="button"
51                       variant="ghost"
52                       size="icon"
53                       className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
54                       onClick={() => setShowPassword(!showPassword)}
55                     >
56                       {showPassword ? (
57                         <EyeOff className="h-4 w-4 text-gray-500" />
58                       ) : (
59                         <Eye className="h-4 w-4 text-gray-500" />
60                       )}
61                       <span className="sr-only">{showPassword ? "Hide password" : "Show password"}</span>
62                     </Button>
63                   </div>
64                 </div>
65
66                 <div className="flex items-center space-x-2">
67                   <Checkbox
68                     id="remember-me"
69                     checked={rememberMe}
70                     onChange={(checked) => setRememberMe(checked as boolean)}
71                   />
72                   <Label htmlFor="remember-me" className="text-sm font-normal">
73                     Remember me
74                   </Label>
75                 </div>
76
77                 <Button type="submit" className="w-full" disabled={isLoading}>
78                   {isLoading ? "Logging in..." : "Login"}
79                 </Button>
80               </div>
81             </form>
82           </div>
83         </div>
84       </div>
85     )
86   )
87 }
88 
```

Ln 41, Col 9 Spaces: 2 UTF-8 CRLF ⓘ TypeScript JSX ⌂ Go Live Windsurf: [...] ⌂ Prettier ⌂

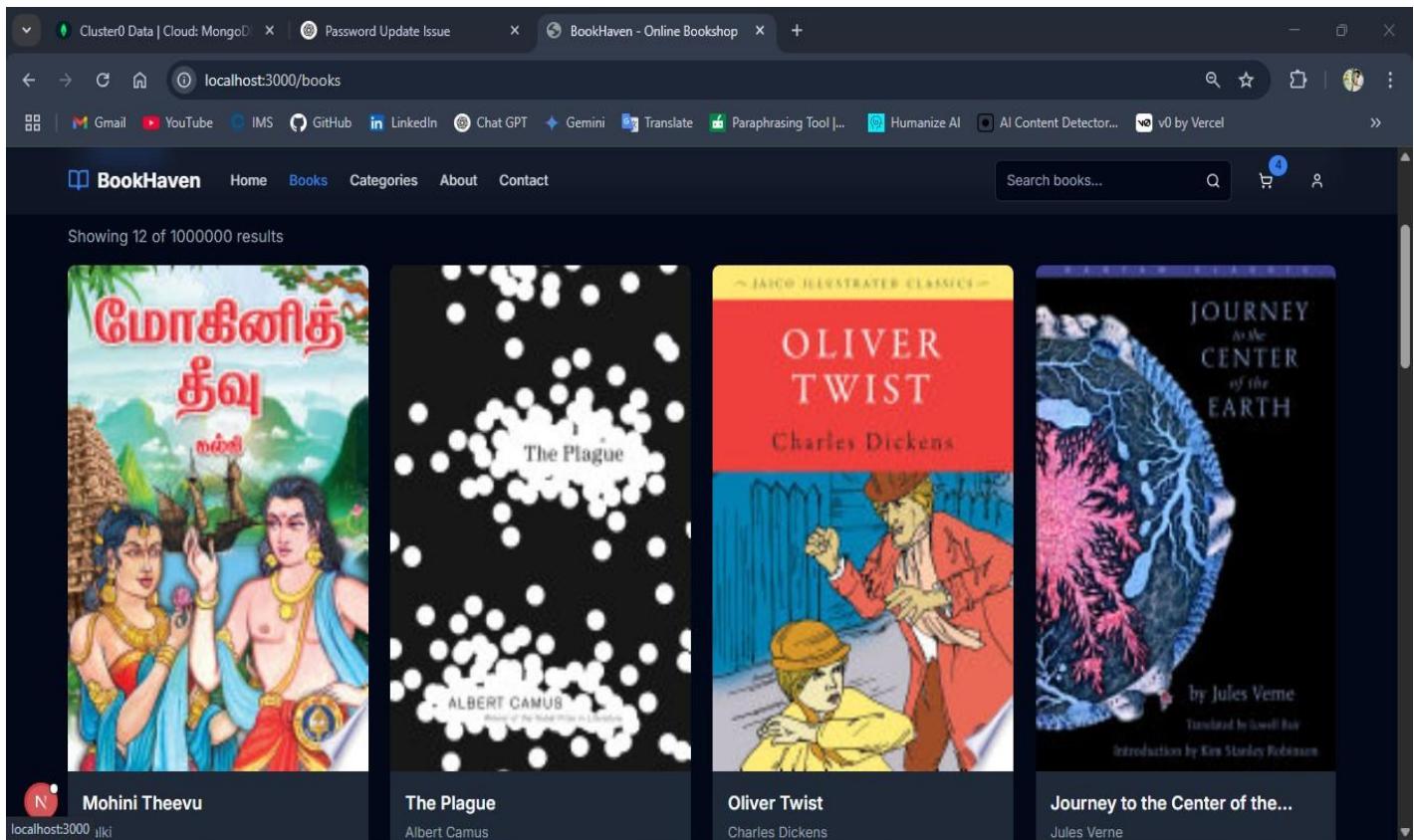
```
app > login > page.tsx > LoginPage > handleSubmit
17  export default function LoginPage() {
116    <Button type="submit" className="w-full" disabled={isLoading}>
117      | {isLoading ? "Logging in..." : "Login"}
118    </Button>
119
120    <div className="mt-6 text-center">
121      <p>
122        Don't have an account? " "
123        <Link href="/register" className="text-primary hover:underline">
124          Register
125        </Link>
126      </p>
127    </div>
128  </div>
129  </div>
130
131  </div>
132  )
133}
```

LN 41, Col 9 Spaces: 2 UTF-8 CRLF ⓘ TypeScript JSX ⓘ Go Live Windsurf: [...] ⓘ Prettier ⓘ

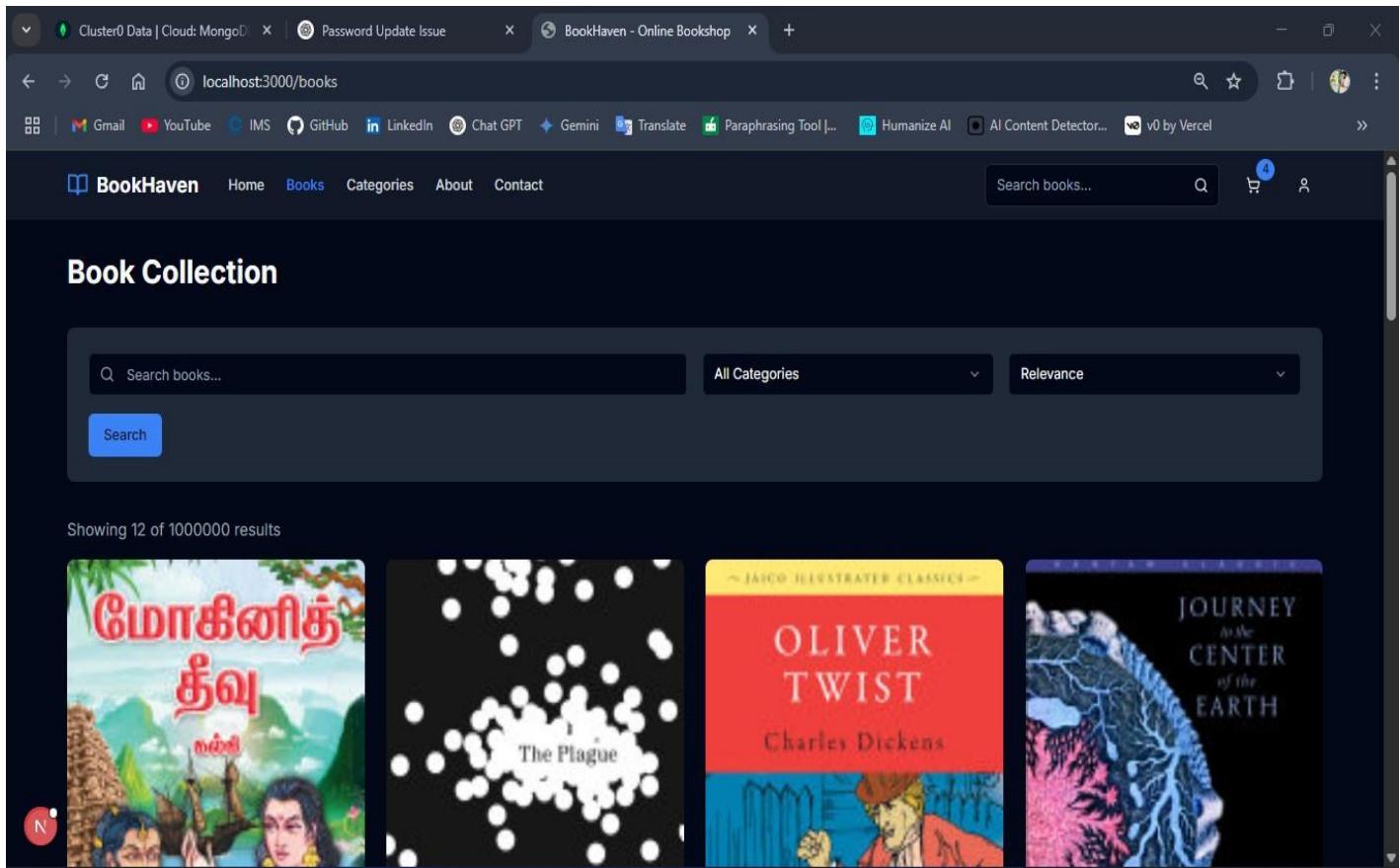
- Authentication success/error messages screenshots

1.2. Book Listing and Search

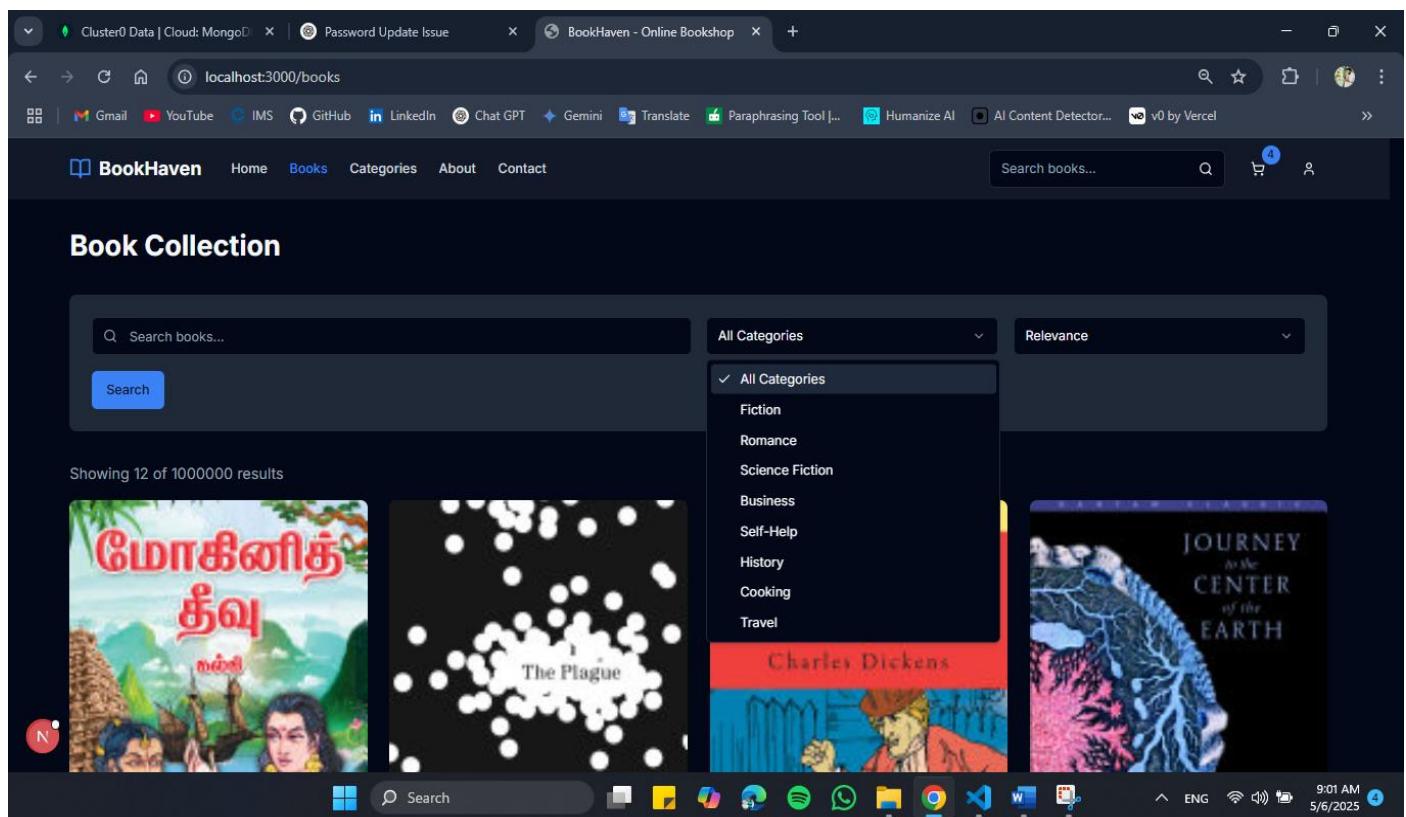
- Main book listing page screenshot



➤ Search functionality interface screenshot



➤ Filter/Sort options screenshot



Screenshot of the BookHaven - Online Bookshop homepage showing the book collection search interface.

The page title is "BookHaven - Online Bookshop". The URL is "localhost:3000/books".

Search bar: "Search books..."

Category dropdown: "All Categories"

Sort dropdown: "Relevance" (selected), "Newest", "Price: Low to High", "Price: High to Low".

Showing 12 of 1000000 results.

Books displayed:

- Meo Kainikku Thivu (Tamil)
- The Plague
- OLIVER TWIST by Charles Dickens
- JOURNEY to the CENTER of the EARTH

System tray icons: N, Search, File, Folder, Spotify, WhatsApp, Chrome, Microsoft Edge, etc.

➤ Book detail's view screenshot

Screenshot of the BookHaven - Online Bookshop showing the detailed view of the book "Oliver Twist".

The URL is "localhost:3000/books/3oSqDwAAQBAJ".

Book cover: "OLIVER TWIST" by Charles Dickens, Jaico Publishing House.

Book details:
Title: Oliver Twist
Author: Charles Dickens
Publisher: Jaico Publishing House
Publication Date: 2019-01-01
Language: en

Additional Information:
ISBN: 9388423038
Pages: 240
Categories: Fiction / Classics
Format: BOOK
Maturity Rating: NOT_MATURE

Buttons: Add to Cart, Description, Details, Reviews.

➤ Code screenshots for API integration

The image shows two side-by-side screenshots of a code editor interface, likely Visual Studio Code, displaying TypeScript code for a Google Books API client. The left screenshot shows the code in light mode, and the right screenshot shows the same code in dark mode. Both screenshots include the following details:

- File Structure:** The left sidebar shows a project structure under "ONLINE-BOOKSHOP" with files like app, scripts, lib, and node_modules.
- Code Editor:** The main area displays a file named "google-books.ts".
- Code Content (Left Screenshot):**

```
lib > google-books.ts > GoogleBooksParams
1 import type { Book } from "@lib/types"
2
3 interface GoogleBooksParams {
4   query?: string
5   category?: string
6   sort?: string
7   page?: number
8   limit?: number
9 }

Windsurf: Refactor | Explain | Generate JSDoc | X
export async function getGoogleBooks(params: GoogleBooksParams = {}) {
  const { query = "", category = "", sort = "relevance", page = 1, limit = 12 } = params

  // Calculate start index for pagination
  const startIndex = (page - 1) * limit

  // Build search query
  let searchQuery = query
  if (category && category !== "All Categories") {
    searchQuery = searchQuery ? `${searchQuery}+subject:${category}` : `subject:${category}`
  }

  // If no search query or category, use a default query to get popular books
  if (!searchQuery) {
    searchQuery = "subject:fiction"
  }

  // Map sort options to Google Books API parameters
  const sortMapping: Record<string, string> = {
    relevance: "relevance",
  }
```
- Code Content (Right Screenshot):**

```
lib > google-books.ts > getGoogleBooks
11 export async function getGoogleBooks(params: GoogleBooksParams = {}) {
28   // Map sort options to Google Books API parameters
29   const sortMapping: Record<string, string> = {
30     relevance: "relevance",
31     newest: "newest",
32     // Google Books API doesn't support price sorting, we'll handle it after fetching
33   }
34
35   const orderBy = sortMapping[sort] || "relevance"
36
37   try {
38     // Fetch books from Google Books API
39     const response = await fetch(
40       `https://www.googleapis.com/books/v1/volumes?q=${encodeURIComponent(searchQuery)}&startIndex=${startIndex}&maxResults=12`
41     )
42
43     if (!response.ok) {
44       throw new Error(`Google Books API error: ${response.status}`)
45     }
46
47     const data = await response.json()
48
49     // Transform Google Books API response to our Book type
50     const items: Book[] = (data.items || []).map((item: any) => {
51       const volumeInfo = item.volumeInfo || {}
52       const saleInfo = item.saleInfo || {}
53
54       // Generate a random price if not available
55       const price = saleInfo.retailPrice?.amount || (Math.floor(Math.random() * 30) + 5 + Math.random()).toFixed(2)
56
57       return {
58         id: item.id,
```
- Status Bar:** The bottom status bar shows the file path ("chamoddhananjaya2000"), line numbers (e.g., "Ln 3, Col 30" or "Ln 28, Col 53"), and other editor settings like "Spaces 2", "UTF-8", "CRLF", "TypeScript", "Go Live", "Windsurf: [...]", and "Prettier".

```
File Edit Selection View Go Run ... ⏪ Online-bookshop

EXPLORER
ONLINE-BOOKSHOP
  app
    orders
    profile
    register
    scripts
      globals.css
      layout.tsx
      page.tsx
    components
    context
    hooks
    IT3113 Final...
    lib
      auth-options.ts
      auth.ts
      books.ts
      google-books.ts
      jwt.ts
      mongodb.ts
      orders.ts
      types.ts
      utils.ts
    node_modules
    public
    styles
    env
  OUTLINE
  TIMELINE
  main ⚡ 0 0 0

google-books.ts x
lib > google-books.ts > ⚡ getGoogleBooks
11 export async function getGoogleBooks(params: GoogleBooksParams = {}) {
  const items: Book[] = (data.items || []).map(item: any) => {
    id: item.id,
    title: volumeInfo.title || "Unknown Title",
    authors: volumeInfo.authors || ["Unknown Author"],
    description: volumeInfo.description || "No description available.",
    categories: volumeInfo.categories || ["Uncategorized"],
    publisher: volumeInfo.publisher,
    publishedDate: volumeInfo.publishedDate,
    language: volumeInfo.language,
    pageCount: volumeInfo.pageCount,
    printType: volumeInfo.printType,
    maturityRating: volumeInfo.maturityRating,
    isbn: volumeInfo.industryIdentifiers?.[0]?.identifier,
    price: Number.parseFloat(price),
    coverImage: volumeInfo.imageLinks.thumbnail || "/placeholder.svg?height=200&width=150",
  }
}

// Handle price sorting if needed
if (sort === "price_low") {
  items.sort((a, b) => a.price - b.price)
} else if (sort === "price_high") {
  items.sort((a, b) => b.price - a.price)
}

return {
  items,
  totalItems: data.totalItems || items.length,
  itemsPerPage: limit,
}
} catch (error) {
  console.error(`Error fetching from Google Books API: ${error.message}`);
  return {
    items: [],
    totalItems: 0,
    itemsPerPage: limit,
  };
}

chamodhhananjaya2000 (1 week ago) Ln 28, Col 53 Spaces: 2 UTF-8 CRLF ⓘ TypeScript 🛡 Go Live Windsurf: {} ✎ Prettier

File Edit Selection View Go Run ... ⏪ Online-bookshop

EXPLORER
ONLINE-BOOKSHOP
  app
    orders
    profile
    register
    scripts
      globals.css
      layout.tsx
      page.tsx
    components
    context
    hooks
    IT3113 Final...
    lib
      auth-options.ts
      auth.ts
      books.ts
      google-books.ts
      jwt.ts
      mongodb.ts
      orders.ts
      types.ts
      utils.ts
    node_modules
    public
    styles
    env
  OUTLINE
  TIMELINE
  main ⚡ 0 0 0

google-books.ts x
lib > google-books.ts > ⚡ getGoogleBooks
11 export async function getGoogleBooks(params: GoogleBooksParams = {}) {
  ...
} catch (error) {
  console.error(`Error fetching from Google Books API: ${error.message}`);
  return {
    items: [],
    totalItems: 0,
    itemsPerPage: limit,
  };
}

Windsurf: Refactor | Explain | Generate JSDoc | X
export async function getGoogleBookById(id: string): Promise<Book | null> {
  try {
    const response = await fetch(
      `https://www.googleapis.com/books/v1/volumes/${id}?key=${process.env.GOOGLE_BOOKS_API_KEY || ""}`,
    );
    ...
  } catch (error) {
    throw new Error(`Google Books API error: ${error.message}`);
  }
  const item = await response.json();
  ...
  const volumeInfo = item.volumeInfo || {};
  const saleInfo = item.saleInfo || {};
}

chamodhhananjaya2000 (1 week ago) Ln 28, Col 53 Spaces: 2 UTF-8 CRLF ⓘ TypeScript 🛡 Go Live Windsurf: {} ✎ Prettier
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "ONLINE-BOOKSHOP". The "google-books.ts" file is selected.
- Code Editor:** Displays the implementation of the `getGoogleBookById` function. The code uses async/await to fetch data from Google Books and return a book object with various properties like title, authors, and price.
- Bottom Status Bar:** Shows the file path as "chamoddhananjaya2000 (1 week ago)", line count as "Ln 28, Col 53", and other status indicators.

```

lib > google-books.ts > getGoogleBooks
97  export async function getGoogleBookById(id: string): Promise<Book | null> {
98    const saleInfo = item.saleInfo || {}
99
100   // Generate a random price if not available
101   const price = saleInfo.retailPrice?.amount || (Math.floor(Math.random() * 30) + 5 + Math.random()).toFixed(2)
102
103   return {
104     id: item.id,
105     title: volumeInfo.title || "Unknown Title",
106     authors: volumeInfo.authors || ["Unknown Author"],
107     description: volumeInfo.description || "No description available.",
108     categories: volumeInfo.categories || ["Uncategorized"],
109     publisher: volumeInfo.publisher,
110     publishedDate: volumeInfo.publishedDate,
111     language: volumeInfo.language,
112     pageCount: volumeInfo.pageCount,
113     printType: volumeInfo.printType,
114     maturityRating: volumeInfo.maturityRating,
115     isbn: volumeInfo.industryIdentifiers?.[0]?.identifier,
116     price: Number.parseFloat(price),
117     coverImage: volumeInfo.imageLinks?.thumbnail || "/placeholder.svg?height=200&width=150",
118   }
119
120 } catch (error) {
121   console.error(`Error fetching book with ID ${id}:`, error)
122   return null
123 }
124
125 }
126
127 }
128
129
130
131
132
133
134
135
136
137
138
139
140

```

➤ Search implementation code screenshots

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "ONLINE-BOOKSHOP". The "page.tsx" file is selected.
- Code Editor:** Displays the implementation of the `BooksPage` component. It handles search parameters, queries the database, and renders a list of books.
- Bottom Status Bar:** Shows the file path as "chamoddhananjaya2000 (1 week ago)", line count as "Ln 14, Col 48", and other status indicators.

```

app > books > page.tsx > BooksPage > [id]
1 import { Suspense } from "react"
2 import BookSearch from "@/components/book-search"
3 import BookList from "@/components/book-list"
4 import LoadingBooks from "@/components/loading-books"
5 import { getBooks } from "@/lib/books"
6
7 Windurf Refactor | Explain | Generate JSDoc | X
8 export default async function BooksPage({
9   searchParams,
10  }: {
11    searchParams: { query?: string; category?: string; sort?: string; page?: string }
12  }) {
13    const query = searchParams.query || ""
14    const category = searchParams.category || ""
15    const sort = searchParams.sort || "relevance"
16    const page = Number.parseInt(searchParams.page || "1")
17
18    const books = await getBooks({ query, category, sort, page })
19
20    return (
21      <div className="container mx-auto px-4 py-8">
22        <h1 className="text-3xl font-bold mb-8">Book Collection</h1>
23        <BookSearch initialQuery={query} initialCategory={category} initialSort={sort} />
24        <Suspense fallback={<LoadingBooks />}>
25          <BookList books={books} currentPage={page} />
26        </Suspense>
27      </div>
28    )
29

```

The screenshot shows a code editor interface with the following details:

- File Path:** route.ts
- Content:** A TypeScript function for handling a GET request to fetch books.
- Code Snippet:**

```
1 import { NextResponse } from "next/server"
2 import { getGoogleBooks } from "@/lib/google-books"
3
4 export async function GET(request: Request) {
5   try {
6     const { searchParams } = new URL(request.url)
7     const query = searchParams.get("query") || ""
8     const category = searchParams.get("category") || ""
9     const sort = searchParams.get("sort") || "relevance"
10    const page = Number.parseInt(searchParams.get("page") || "1")
11    const limit = Number.parseInt(searchParams.get("limit") || "12")
12
13    const books = await getGoogleBooks({
14      query,
15      category,
16      sort,
17      page,
18      limit,
19    })
20
21    return NextResponse.json(books)
22  } catch (error) {
23    console.error("Error fetching books:", error)
24    return NextResponse.json({ error: "Failed to fetch books" }, { status: 500 })
25  }
26}
27
```

- Editor Features:** The editor includes tabs for route.ts, route.ts (books), route.ts (id), and page.tsx. It also shows a preview pane on the right side.
- Status Bar:** The status bar at the bottom displays: chamoddhananjaya2000 (1 week ago) Ln 16, Col 12 Spaces: 2 UTF-8 CRLF (TypeScript) Go Live Windsurf: (...) Prettier

1.3. Shopping Cart

➤ Cart view screenshot

The screenshot shows the BookHaven shopping cart page. The cart contains three items:

Book	Quantity	Price	Actions
The Plague Albert Camus	1	\$6.83	
Oliver Twist Charles Dickens	2	\$10.16	
Journey to the Center of the Earth Jules Verne	3	\$21.21	

Order Summary

Subtotal	\$38.20
Tax (8%)	\$3.06
Shipping	\$5.99
Total	\$47.25

[Proceed to Checkout](#)

➤ Add to cart functionality screenshot

The screenshot shows the BookHaven search results for "sherlock%20holmes". Four book cards are displayed:

- The Sherlock Holmes Book** by DK, \$15.88, Add to Cart button.
- The Best of Sherlock Holmes** by Arthur Conan Doyle, \$21.11, Add to Cart button.
- Adventures of Sherlock Holmes** by Arthur Conan Doyle, \$9.13, Add to Cart button.
- The Case-Book of Sherlock Holmes** by Arthur Conan Doyle, \$25.65, Add to Cart button.

➤ Update quantity interface screenshot

The screenshot shows a dark-themed web browser window with several tabs open. The active tab is 'BookHaven - Online Bookshop' at 'localhost:3000/cart'. The page displays a table of books in the cart:

Book	Quantity	Price	Actions
The Plague Albert Camus	1	\$6.83	Remove
Oliver Twist Charles Dickens	2	\$10.16	Remove
Journey to the Center of the Earth Jules Verne	3	\$21.21	Remove
The Case-Book of Sherlock Holmes Arthur Conan Doyle	1	\$25.65	Remove

A red notification bar at the bottom left says 'N 1 Issue X'. To the right of the cart table is an 'Order Summary' box:

Subtotal	\$63.85
Tax (8%)	\$5.11
Shipping	\$5.99
Total	\$74.95

A blue 'Proceed to Checkout' button is located below the summary.

➤ Remove items interface screenshot

The screenshot shows the same browser window after an item has been removed. The cart table now only contains two items:

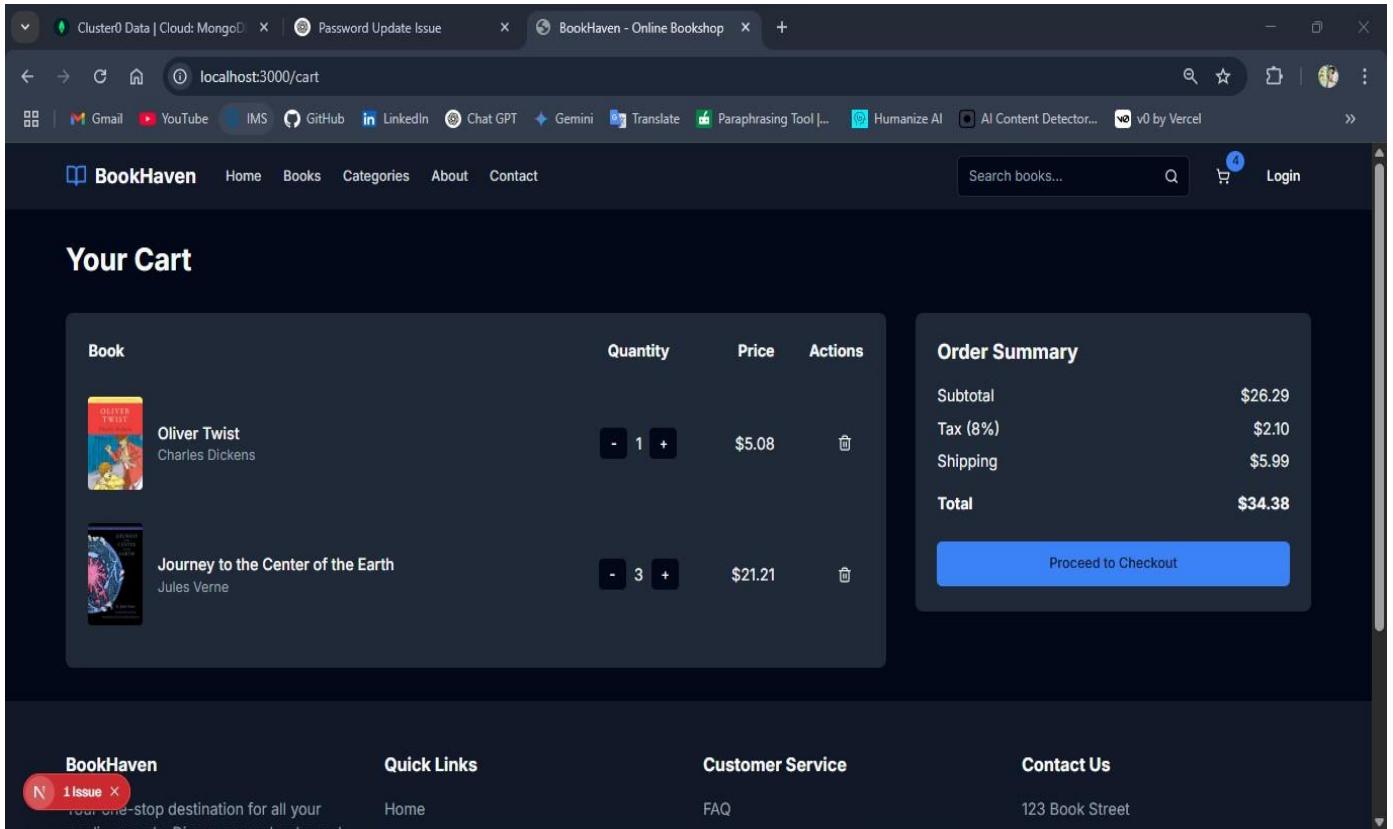
Book	Quantity	Price	Actions
Oliver Twist Charles Dickens	2	\$10.16	Remove
Journey to the Center of the Earth Jules Verne	3	\$21.21	Remove

The 'Order Summary' box remains the same:

Subtotal	\$31.37
Tax (8%)	\$2.51
Shipping	\$5.99
Total	\$39.87

A blue 'Proceed to Checkout' button is located below the summary.

➤ Cart total calculation screenshot



- Related code screenshots for cart management

```
File Edit Selection View Go Run ... ← → 🔍 Online-bookshop 🌐
```

EXPLORER

ONLINE-BOOKSHOP

- .next
- app
 - about
 - api
 - auth
 - cart
 - books
 - orders
 - books
 - cart
- page.tsx
- categories
- checkout
- confirmation
- contact
- login
- orders
- profile
- register
- scripts
- globals.css
- layout.tsx
- page.tsx
- components
- context
- hooks
- lib
- OUTLINE
- TIMELINE

page.tsx ...books route.ts ...books page.tsx ...cart route.ts ...[id]

```
1 "use client"
2
3 import { useCart } from "@/context/cart-context"
4 import { useAuth } from "@/context/auth-context"
5 import { useRouter } from "next/navigation"
6 import Image from "next/image"
7 import { Button } from "@components/ui/button"
8 import { Trash2 } from "lucide-react"
9 import Link from "next/link"
10 import { formatPrice } from "@/lib/utils"
11
12 Windsurf: Refactor | Explain | Generate JSDoc | X
13 export default function CartPage() {
14   const { cart, removeFromCart, updateQuantity, clearCart } = useCart()
15   const { user } = useAuth()
16   const router = useRouter()
17
18   const subtotal = cart.reduce((total, item) => total + item.price * item.quantity, 0)
19   const tax = subtotal * 0.08 // 8% tax
20   const shipping = subtotal > 0 ? 5.99 : 0
21   const total = subtotal + tax + shipping
22
23   Windsurf: Refactor | Explain | Generate JSDoc | X
24   const handleCheckout = () => {
25     if (!user) {
26       router.push("/login?redirect=/checkout")
27     } else {
28       router.push("/checkout")
29     }
30   }
31
32   if (cart.length === 0) {
33     return (
34       <div>
35         <h1>Your Cart</h1>
36         <p>Your cart is currently empty. Add some items to get started!</p>
37       </div>
38     )
39   }
40
41   return (
42     <div>
43       <h2>Your Cart</h2>
44       <table>
45         <thead>
46           <tr>
47             <th>Item</th>
48             <th>Quantity</th>
49             <th>Price</th>
50             <th>Actions</th>
51           </tr>
52         </thead>
53         <tbody>
54           {cart.map(item => (
55             <tr key={item.id}>
56               <td>{item.name}</td>
57               <td>{item.quantity}</td>
58               <td>${formatPrice(item.price)}</td>
59               <td>
60                 <button onClick={()=>removeFromCart(item.id)}>Remove</button>
61                 <button onClick={()=>updateQuantity(item.id, item.quantity + 1)}>+</button>
62                 <button onClick={()=>updateQuantity(item.id, item.quantity - 1)}>-</button>
63               </td>
64             </tr>
65           ))}
66         </tbody>
67       </table>
68       <div>
69         <strong>Subtotal:</strong> ${formatPrice(subtotal)}
70         <strong>Tax:</strong> ${formatPrice(tax)}
71         <strong>Shipping:</strong> ${formatPrice(shipping)}
72         <strong>Total:</strong> ${formatPrice(total)}
73       </div>
74       <button onClick={handleCheckout}>Proceed to Checkout</button>
75     </div>
76   )
77 
```

The image shows two side-by-side screenshots of a code editor, likely VS Code, displaying TypeScript code for a shopping cart feature in a Next.js application.

Left Screenshot:

- Explorer:** Shows the project structure under `ONLINE-BOOKSHOP`, including `app`, `api`, `components`, `context`, `hooks`, `lib`, and `scripts`.
- Code Editor:** The active file is `page.tsx`. The code defines a `CartPage` component that checks if the cart is empty and displays a message or a table of items. It uses a `cart.map()` callback to generate rows for each item.

```
12 export default function CartPage() {
13     if (cart.length === 0) {
14         return (
15             <div className="container mx-auto px-4 py-16 text-center">
16                 <h1 className="text-3xl font-bold mb-8">Your Cart</h1>
17                 <p className="text-xxl mb-8">Your cart is empty</p>
18                 <Link href="/books">
19                     <Button size="lg">Browse Books</Button>
20                 </Link>
21             </div>
22         )
23     }
24
25     return (
26         <div className="container mx-auto px-4 py-8">
27             <h1 className="text-3xl font-bold mb-8">Your Cart</h1>
28
29             <div className="grid grid-cols-1 lg:grid-cols-3 gap-8">
30                 <div className="lg:col-span-2">
31                     <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-6">
32                         <table className="w-full">
33                             <thead>
34                                 <tr className="border-b">
35                                     <th className="text-left pb-4">Book</th>
36                                     <th className="text-center pb-4">Quantity</th>
37                                     <th className="text-right pb-4">Price</th>
38                                     <th className="text-right pb-4">Actions</th>
39                                 </tr>
40                             </thead>
41                             <tbody>
42                                 {cart.map((item) => (
43
44             
```

Right Screenshot:

- Explorer:** Shows the same project structure as the left screenshot.
- Code Editor:** The active file is `page.tsx`. The code is identical to the left screenshot, defining the `CartPage` component.

```
12 export default function CartPage() {
13     <tbody>
14         {cart.map((item) => (
15             <tr key={item.id} className="border-b">
16                 <td className="py-4">
17                     <div className="flex items-center">
18                         <Image
19                             src={item.coverImage || "/placeholder.svg?height=80&width=60"}
20                             alt={item.title}
21                             width={60}
22                             height={80}
23                             className="rounded mr-4"
24                         />
25                         <div>
26                             <h3 className="font-medium">{item.title}</h3>
27                             <p className="text-sm text-gray-500 dark:text-gray-400">{item.authors.join(", ")}/p>
28                         </div>
29                     </div>
30                 <td className="py-4">
31                     <div className="flex justify-center items-center">
32                         <Button
33                             variant="outline"
34                             size="icon"
35                             className="h-8 w-8"
36                             onClick={() => updateQuantity(item.id, Math.max(1, item.quantity - 1))}
37                         >
38                             -
39                         </Button>
40                         <span className="mx-2">{item.quantity}</span>
41                         <Button
42                             variant="outline"
43                         >
```

The image shows two side-by-side screenshots of a code editor, likely VS Code, displaying TypeScript code for a shopping cart feature in a Next.js application.

Top Screenshot:

- File Explorer:** Shows the project structure under `ONLINE-BOOKSHOP`, including `app`, `api`, `components`, `context`, `hooks`, `outline`, and `timeline`.
- Code Editor:** The current file is `page.tsx`. The code defines a `CartPage` component that maps items from the cart map and displays them in a table. It includes a subtotal calculation and a summary section at the bottom.

```
12 export default function CartPage() {
13   const [cartMap, setCartMap] = useState<Record<string, { id: string; quantity: number; price: number }>|null>(null)
14
15   const updateQuantity = (id: string, quantity: number) => {
16     setCartMap((prev) => {
17       if (!prev) return null
18
19       const updatedCartMap = { ...prev }
20       updatedCartMap[id] = { id, quantity, price: prev[id].price }
21
22       return updatedCartMap
23     })
24   }
25
26   const removeFromCart = (id: string) => {
27     setCartMap((prev) => {
28       if (!prev) return null
29
30       const updatedCartMap = { ...prev }
31       delete updatedCartMap[id]
32
33       return updatedCartMap
34     })
35   }
36
37   const formatPrice = (value: number) => {
38     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$<span>$')
39   }
40
41   const formatQuantity = (value: number) => {
42     return value === 1 ? '1 item' : value + ' items'
43   }
44
45   const formatTotal = (value: number) => {
46     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
47   }
48
49   const formatTax = (value: number) => {
50     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
51   }
52
53   const formatShipping = (value: number) => {
54     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
55   }
56
57   const formatSubtotal = (value: number) => {
58     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
59   }
60
61   const [orderSummary, setOrderSummary] = useState<{ subtotal: number; tax: number; shipping: number; total: number }>({ subtotal: 0, tax: 0, shipping: 0, total: 0 })
62
63   const handleCheckout = () => {
64     // Handle checkout logic here
65   }
66
67   const [isCartEmpty, setIsCartEmpty] = useState(true)
68
69   const cartItems = cartMap?.map((item) => {
70     const { id, quantity, price } = item
71
72     return (
73       <tr>
74         <td>
75           <img alt={item.name} src={item.image}>
76           <span>{formatQuantity(quantity)}</span>
77           <span>${formatPrice(price)}</span>
78         </td>
79         <td>
80           <span>${formatPrice(item.price * item.quantity)}</span>
81         </td>
82         <td>
83           <button variant="ghost" size="icon" onClick={() => updateQuantity(id, item.quantity + 1)}>
84             <Icon>
85               <IconImage icon="plus" />
86             </Icon>
87           </button>
88           <button variant="ghost" size="icon" onClick={() => removeFromCart(id)}>
89             <Icon>
90               <IconImage icon="trash" />
91             </Icon>
92           </button>
93         </td>
94       </tr>
95     )
96   )
97
98   const cartTable = cartItems.length ? (
99     <table>
100       <thead>
101         <tr>
102           <th>Image</th>
103           <th>Quantity</th>
104           <th>Price</th>
105         </tr>
106       </thead>
107       <tbody>
108         {cartItems}
109       </tbody>
110     </table>
111   ) : (
112     <div>
113       <h2>Order Summary</h2>
114       <div>
115         <span>Subtotal</span>
116         <span>${formatPrice(subtotal)}</span>
117       </div>
118     </div>
119   )
120
121   const [isCheckoutVisible, setIsCheckoutVisible] = useState(false)
122
123   const handleCheckoutClick = () => {
124     setIsCheckoutVisible(true)
125   }
126
127   const [checkoutForm, setCheckoutForm] = useState<{ name: string; address: string; city: string; zip: string }>({ name: '', address: '', city: '', zip: '' })
128
129   const handleCheckoutSubmit = (e: React.FormEvent) => {
130     e.preventDefault()
131
132     const { name, address, city, zip } = checkoutForm
133
134     if (!name || !address || !city || !zip) {
135       alert('Please fill in all fields')
136       return
137     }
138
139     handleCheckout()
140   }
141
142 }
```

Bottom Screenshot:

- File Explorer:** Shows the project structure under `ONLINE-BOOKSHOP`, including `app`, `api`, `components`, `context`, `hooks`, `outline`, and `timeline`.
- Code Editor:** The current file is `page.tsx`. The code defines a `CartPage` component that maps items from the cart map and displays them in a table. It includes a subtotal calculation and a summary section at the bottom.

```
12 export default function CartPage() {
13   const [cartMap, setCartMap] = useState<Record<string, { id: string; quantity: number; price: number }>|null>(null)
14
15   const updateQuantity = (id: string, quantity: number) => {
16     setCartMap((prev) => {
17       if (!prev) return null
18
19       const updatedCartMap = { ...prev }
20       updatedCartMap[id] = { id, quantity, price: prev[id].price }
21
22       return updatedCartMap
23     })
24   }
25
26   const removeFromCart = (id: string) => {
27     setCartMap((prev) => {
28       if (!prev) return null
29
30       const updatedCartMap = { ...prev }
31       delete updatedCartMap[id]
32
33       return updatedCartMap
34     })
35   }
36
37   const formatPrice = (value: number) => {
38     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
39   }
40
41   const formatQuantity = (value: number) => {
42     return value === 1 ? '1 item' : value + ' items'
43   }
44
45   const formatTotal = (value: number) => {
46     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
47   }
48
49   const formatTax = (value: number) => {
50     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
51   }
52
53   const formatShipping = (value: number) => {
54     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
55   }
56
57   const formatSubtotal = (value: number) => {
58     return value.toFixed(2).replace(/\d{1,3}(?=\.\d{1,2})/, '$,$')
59   }
60
61   const [orderSummary, setOrderSummary] = useState<{ subtotal: number; tax: number; shipping: number; total: number }>({ subtotal: 0, tax: 0, shipping: 0, total: 0 })
62
63   const handleCheckout = () => {
64     // Handle checkout logic here
65   }
66
67   const [isCartEmpty, setIsCartEmpty] = useState(true)
68
69   const cartItems = cartMap?.map((item) => {
70     const { id, quantity, price } = item
71
72     return (
73       <tr>
74         <td>
75           <img alt={item.name} src={item.image}>
76           <span>{formatQuantity(quantity)}</span>
77           <span>${formatPrice(price)}</span>
78         </td>
79         <td>
80           <span>${formatPrice(item.price * item.quantity)}</span>
81         </td>
82         <td>
83           <button variant="ghost" size="icon" onClick={() => updateQuantity(id, item.quantity + 1)}>
84             <Icon>
85               <IconImage icon="plus" />
86             </Icon>
87           </button>
88           <button variant="ghost" size="icon" onClick={() => removeFromCart(id)}>
89             <Icon>
90               <IconImage icon="trash" />
91             </Icon>
92           </button>
93         </td>
94       </tr>
95     )
96   )
97
98   const cartTable = cartItems.length ? (
99     <table>
100       <thead>
101         <tr>
102           <th>Image</th>
103           <th>Quantity</th>
104           <th>Price</th>
105         </tr>
106       </thead>
107       <tbody>
108         {cartItems}
109       </tbody>
110     </table>
111   ) : (
112     <div>
113       <h2>Order Summary</h2>
114       <div>
115         <span>Subtotal</span>
116         <span>${formatPrice(subtotal)}</span>
117       </div>
118     </div>
119   )
120
121   const [isCheckoutVisible, setIsCheckoutVisible] = useState(false)
122
123   const handleCheckoutClick = () => {
124     setIsCheckoutVisible(true)
125   }
126
127   const [checkoutForm, setCheckoutForm] = useState<{ name: string; address: string; city: string; zip: string }>({ name: '', address: '', city: '', zip: '' })
128
129   const handleCheckoutSubmit = (e: React.FormEvent) => {
130     e.preventDefault()
131
132     const { name, address, city, zip } = checkoutForm
133
134     if (!name || !address || !city || !zip) {
135       alert('Please fill in all fields')
136       return
137     }
138
139     handleCheckout()
140   }
141
142 }
```

1.4. Checkout Process

- Checkout form screenshot

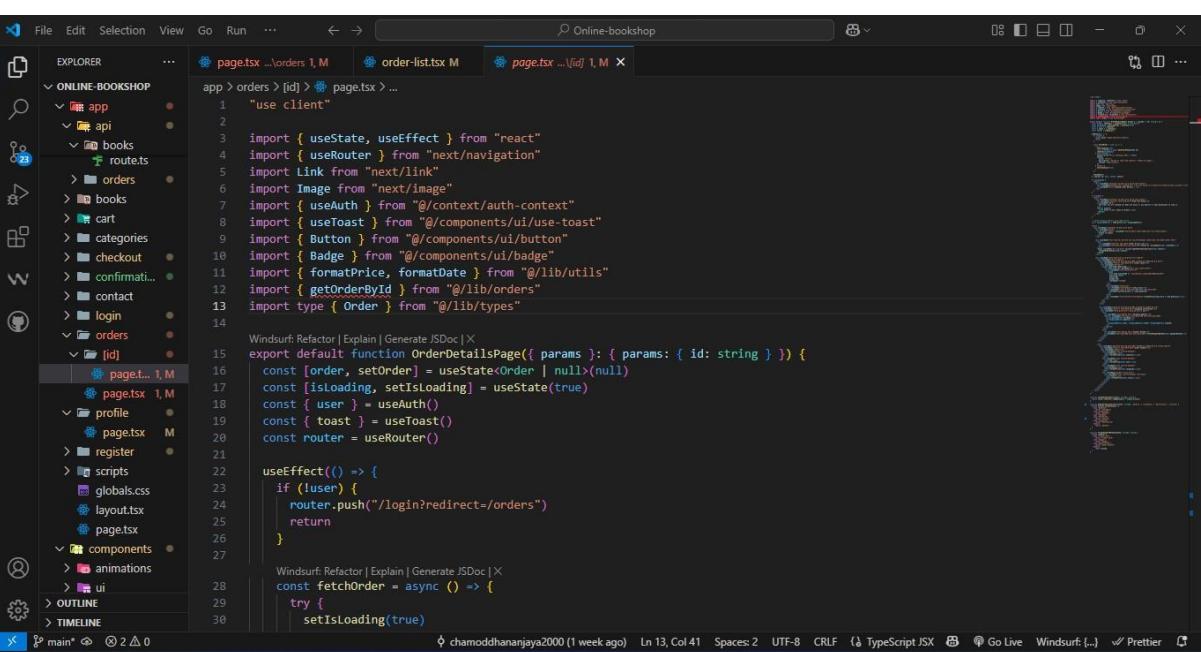
The screenshot shows the BookHaven checkout page. On the left, there's a "Shipping Information" section with fields for Full Name (Chamod Dhananjaya), Email (chamoddhananjaya2000@gmail.com), Address (Kekulugoda, Yakdehiwatta, Nivithigala), City (Ratnapura), State (Sri Lanka), ZIP Code (70400), and Contact Number (0713686622). On the right, an "Order Summary" table details the purchase:

Order Summary	
Oliver Twist	\$5.08
Qty: 1	
Journey to the Center of the Earth	\$21.21
Qty: 3	
Subtotal	\$26.29
Tax (8%)	\$2.10
Shipping	\$5.99
Total	\$34.38

- Payment interface screenshot

The screenshot shows the BookHaven payment interface. On the left, a "Payment Method" section includes radio buttons for Credit Card, PayPal, and Bank Transfer, with Credit Card selected. Below it is a large blue "Place Order" button. On the right, the same "Order Summary" table is displayed as in the previous screenshot.

➤ Order confirmation page screenshot



The screenshot shows a code editor interface with a dark theme. The left sidebar is labeled 'EXPLORER' and lists several project files and folders, including 'ONLINE-BOOKSHOP', 'app', 'api', 'books', 'cart', 'categories', 'checkout', 'confirmation', 'contact', 'login', 'orders', and 'profile'. The main editor area has three tabs open: 'page.tsx ...\\orders 1, M', 'order-list.tsx M', and 'page.tsx ...\\[id] 1, M'. The third tab is currently active and displays the following TypeScript code:

```
app > orders > [id] > page.tsx > ...
1   "use client"
2
3   import { useState, useEffect } from "react"
4   import { useRouter } from "next/navigation"
5   import Link from "next/link"
6   import Image from "next/image"
7   import { useAuth } from "@/context/auth-context"
8   import { useToast } from "@/components/ui/use-toast"
9   import { Button } from "@/components/ui/button"
10  import { Badge } from "@/components/ui/badge"
11  import { formatPrice, formatDate } from "@lib/utils"
12  import { getOrderById } from "@lib/orders"
13  import type { Order } from "@lib/types"
14
Windsurf: Refactor | Explain | Generate JSDoc | X
export default function OrderDetailsPage({ params }: { params: { id: string } }) {
15    const [order, setOrder] = useState(null)
16    const [isLoading, setIsLoading] = useState(true)
17    const { user } = useAuth()
18    const { toast } = useToast()
19    const router = useRouter()
20
21    useEffect(() => {
22      if (!user) {
23        router.push("/login?redirect=/orders")
24        return
25      }
26
27      useEffect(() => {
28        const fetchOrder = async () => {
29          try {
30            setIsLoading(true)
```

```

File Edit Selection View Go Run ... ⏪ ⏩ Online-bookshop
EXPLORER page.tsx ...orders 1, M | order-list.tsx M | page.tsx ...[id] 1, M
ONLINE-BOOKSHOP
  app
    api
      books
        routes.ts
          orders
          books
          cart
          categories
          checkout
          confirmati...
          contact
          login
        orders
          [id]
            page.tsx 1, M
            page.tsx 1, M
          profile
            page.tsx M
          register
          scripts
            globals.css
            layout.tsx
            page.tsx
          components
            animations
            ui
        OUTLINE
        TIMELINE
main* ⏪ 2 △ 0
chamoddhananjaya2000 (1 week ago) Ln 13, Col 41 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Windsurf [...] Prettier

```

```

page.tsx ...orders 1, M
app > orders > [id] > page.tsx > ...
15  export default function OrderDetailsPage({ params }: { params: { id: string } }) {
22    useEffect(() => {
28      const fetchOrder = async () => {
29        try {
30          setIsLoading(true)
31          const orderData = await getOrderById(params.id)
32          setOrder(orderData)
33        } catch (error) {
34          console.error("Error fetching order:", error)
35          toast({
36            title: "Error",
37            description: "Failed to load order details. Please try again.",
38            variant: "destructive",
39          })
40        } finally {
41          setIsLoading(false)
42        }
43      }
44      fetchOrder()
45    }, [params.id, user, router, toast])
46
47      if (isLoading) {
48        return (
49          <div className="container mx-auto px-4 py-16 text-center">
50            <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-primary mx-auto"></div>
51            <p className="mt-4">Loading order details...</p>
52          </div>
53        )
54      }
55
56      if (!order) {
57        return (
58          <div className="container mx-auto px-4 py-16 text-center">
59            <h1 className="text-3xl font-bold mb-4">Order Not Found</h1>
60            <p className="mb-8">
61              The order you are looking for does not exist or you don't have permission to view it.
62            </p>
63            <Button asChild>
64              <Link href="/orders">Back to Orders</Link>
65            </Button>
66          </div>
67        )
68
69      }
70
71      // Parse shipping address from JSON string
72      const shippingAddress = JSON.parse(order.shippingAddress)
73
74      return (
75        <div className="container mx-auto px-4 py-8">
76          <div className="mb-6">
77            <Link href="/orders" className="text-primary hover:underline flex items-center">
78              <Back to Orders>
79            </Link>
80          </div>
81
82          <div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-6">
83            <div>
84              <h1 className="text-3xl font-bold">Order #<code>{order.id}</code></h1>
85              <p className="text-gray-500 dark:text-gray-400">Placed on <code>{formatDate(order.createdAt)}</code></p>
86            </div>

```

File Edit Selection View Go Run ...

ONLINE-BOOKSHOP

- app
 - api
 - books
 - routes.ts
 - orders
 - books
 - cart
 - categories
 - checkout
 - confirmations
 - contact
 - login
 - orders
 - [id]
 - page.tsx 1, M
 - page.tsx 1, M
 - profile
 - register
 - scripts
 - globals.css
 - layout.tsx
 - page.tsx
- components
- animations
- ui
- OUTLINE
- TIMELINE

File Edit Selection View Go Run ...

ONLINE-BOOKSHOP

- app
 - orders > [id] > page.tsx > ...


```

15  export default function OrderDetailsPage({ params }: { params: { id: string } }) {
85      <p className="text-gray-500 dark:text-gray-400">Placed on {formatDate(order.createdAt)}</p>
86      </div>
87      <Badge className="mt-2 md:mt-0" variant={getOrderStatusVariant(order.status)}>
88          {formatOrderStatus(order.status)}
89      </Badge>
90      </div>
91
92      <div className="grid grid-cols-1 lg:grid-cols-3 gap-8">
93          <div className="lg:col-span-2">
94              <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-6 mb-6">
95                  <h2 className="text-xl font-bold mb-4">Order Items</h2>
96                  <div className="divide-y">
97                      {order.orderItems.map(item) => (
98                          <div key={item.id} className="py-4 flex items-center">
99                              <div className="flex-shrink-0 mr-4">
100                                  <Image
101                                      src={item.coverImage || "/placeholder.svg?height=80&width=60"}
102                                      alt={item.title}
103                                      width={60}
104                                      height={80}
105                                      className="rounded"
106                                      />
107                                  </div>
108                                  <div className="flex-grow">
109                                      <h3 className="font-medium">{item.title}</h3>
110                                      <p className="text-sm text-gray-500 dark:text-gray-400">
111                                          {formatPrice(item.price)} × {item.quantity}
112                                      </p>
113                                  </div>
114                                  <div className="flex-shrink-0 font-medium">{formatPrice(item.price * item.quantity)}</div>
115                              </div>
116                          ))}
117                      </div>
118                  </div>
119
120                  <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-6">
121                      <h2 className="text-xl font-bold mb-4">Shipping Information</h2>
122                      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
123                          <div>
124                              <h3 className="font-medium mb-2">Shipping Address</h3>
125                              <address className="not-italic text-gray-600 dark:text-gray-400">
126                                  <p>{shippingAddress.fullName}</p>
127                                  <p>{shippingAddress.address}</p>
128                                  <p>{shippingAddress.city}, {shippingAddress.state} {shippingAddress.zipCode}</p>
129                              </address>
130                          </div>
131
132                          <div>
133                              <h3 className="font-medium mb-2">Payment Method</h3>
134                              <p className="text-gray-600 dark:text-gray-400">{formatPaymentMethod(order.paymentMethod)}</p>
135                          </div>
136
137                          <div>
138                              <h2 className="text-xl font-bold mb-4">Order Summary</h2>
139
140                              <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-6 sticky top-24">
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
848
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
868
869
869
870
871
872
873
874
875
876
877
877
878
878
879
879
880
881
882
883
884
885
886
887
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
928
929
929
930
931
932
933
934
935
936
937
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
948
949
949
950
951
952
953
954
955
956
957
957
958
958
959
959
960
961
962
963
964
965
966
966
967
967
968
968
969
969
970
971
972
973
974
975
976
976
977
977
978
978
979
979
980
981
982
983
984
985
986
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1
```

File Edit Selection View Go Run ...

ONLINE-BOOKSHOP

- app
 - api
 - books
 - routes.ts
 - orders
 - books
 - cart
 - categories
 - checkout
 - confirmations
 - contact
 - login
 - orders
 - [id]
 - profile
 - register
 - scripts
 - globals.css
 - layout.tsx
 - page.tsx
- components
- animations
- ui

OUTLINE

TIMELINE

File Edit Selection View Go Run ...

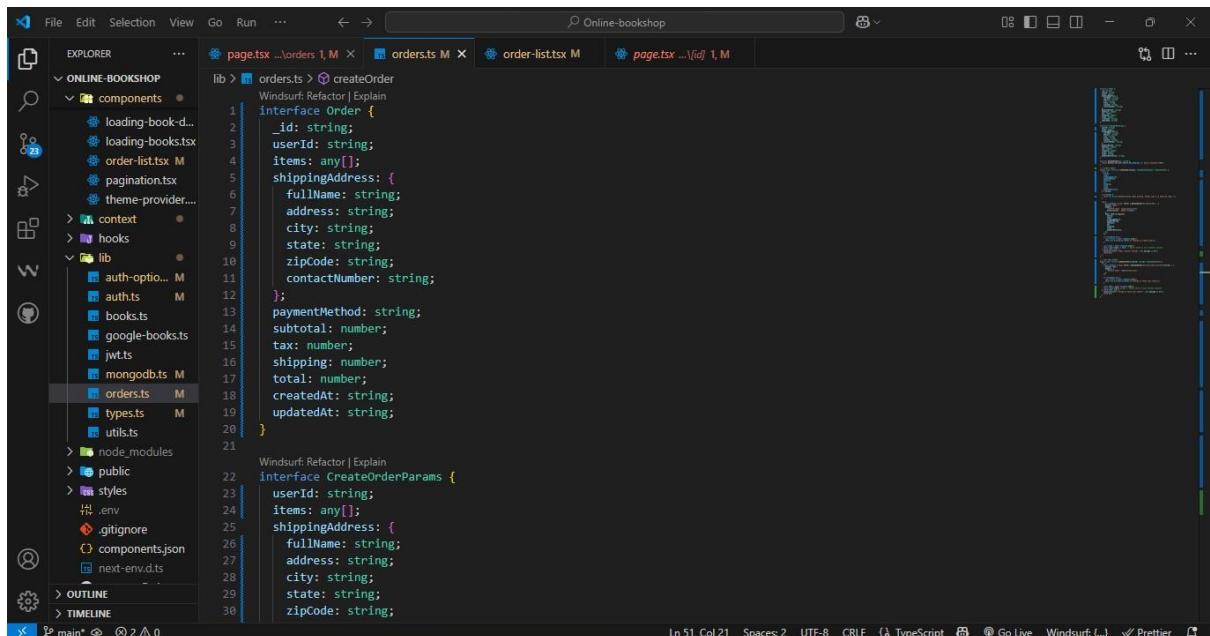
ONLINE-BOOKSHOP

- app
 - orders > [id] > page.tsx > ...
- page.tsx ...[id] 1, M
- order-list.tsx M
- page.tsx ...[id] 1, M X

```

15 export default function OrderDetailsPage({ params }: { params: { id: string } }) {
16   const { id } = params;
17
18   const order = await prisma.order.findUnique({
19     where: { id },
20     include: {
21       books: true,
22       user: true,
23       address: true,
24       paymentMethod: true,
25       status: true,
26       subtotal: true,
27       tax: true,
28       shipping: true,
29       total: true,
30     },
31   });
32
33   const formattedOrder = formatOrder(order);
34
35   return (
36     <div>
37       <h1>Order Details</h1>
38       <p>Order ID: {id}</p>
39       <table>
40         <thead>
41           <tr>
42             <th>Item</th>
43             <th>Quantity</th>
44             <th>Price</th>
45           </tr>
46         </thead>
47         <tbody>
48           {order.books.map(book => {
49             const { title, quantity, price } = book;
50             return <tr>
51               <td>{title}</td>
52               <td>{quantity}</td>
53               <td>${formatPrice(price)}</td>
54             </tr>
55           })}
56         </tbody>
57       </table>
58       <div>
59         <div>Subtotal:</div>
60         <div>${formatPrice(order.subtotal)}</div>
61       </div>
62       <div>Tax:</div>
63       <div>${formatPrice(order.tax)}</div>
64     </div>
65     <div>Shipping:</div>
66     <div>${formatPrice(order.shipping)}</div>
67   </div>
68   <div>Total:</div>
69   <div>${formatPrice(order.total)}</div>
70 }
71
72 function formatOrderStatus(status: string): string {
73   return status.charAt(0).toUpperCase() + status.slice(1)
74 }
75
76 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
77   switch (status.toLowerCase()) {
78     case "pending":
79       return "secondary"
80     case "processing":
81       return "default"
82     case "shipped":
83       return "outline"
84     case "delivered":
85       return "outline"
86     case "cancelled":
87       return "destructive"
88     default:
89       return "default"
90   }
91 }
92
93 function formatPaymentMethod(method: string): string {
94   switch (method) {
95     case "credit-card":
96       return "Credit Card"
97     case "paypal":
98       return "PayPal"
99     case "bank-transfer":
100    return "Bank Transfer"
101  default:
102    return method
103  }
104 }
105
106 function formatPrice(price: number): string {
107   return `₹ ${price.toFixed(2)}`
108 }
109
110 interface Order {
111   id: string;
112   books: Book[];
113   user: User;
114   address: Address;
115   paymentMethod: PaymentMethod;
116   status: Status;
117   subtotal: number;
118   tax: number;
119   shipping: number;
120   total: number;
121 }
122
123 interface Book {
124   title: string;
125   quantity: number;
126   price: number;
127 }
128
129 interface User {
130   name: string;
131   email: string;
132 }
133
134 interface Address {
135   street: string;
136   city: string;
137   state: string;
138   pincode: number;
139 }
140
141 interface PaymentMethod {
142   type: string;
143 }
144
145 interface Status {
146   type: string;
147 }
148
149 interface OrderDetailsPageProps {
150   params: { id: string };
151 }
152
153 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
154   const { id } = params;
155
156   const order = await prisma.order.findUnique({
157     where: { id },
158     include: {
159       books: true,
160       user: true,
161       address: true,
162       paymentMethod: true,
163       status: true,
164       subtotal: true,
165       tax: true,
166       shipping: true,
167       total: true,
168     },
169   });
170
171   const formattedOrder = formatOrder(order);
172
173   return (
174     <div>
175       <h1>Order Details</h1>
176       <p>Order ID: {id}</p>
177       <table>
178         <thead>
179           <tr>
180             <th>Item</th>
181             <th>Quantity</th>
182             <th>Price</th>
183           </tr>
184         </thead>
185         <tbody>
186           {order.books.map(book => {
187             const { title, quantity, price } = book;
188             return <tr>
189               <td>{title}</td>
190               <td>{quantity}</td>
191               <td>${formatPrice(price)}</td>
192             </tr>
193           })}
194         </tbody>
195       </table>
196       <div>
197         <div>Subtotal:</div>
198         <div>${formatPrice(order.subtotal)}</div>
199       </div>
200       <div>Tax:</div>
201       <div>${formatPrice(order.tax)}</div>
202     </div>
203   </div>
204   <div>Shipping:</div>
205   <div>${formatPrice(order.shipping)}</div>
206
207   <div>Total:</div>
208   <div>${formatPrice(order.total)}</div>
209 }
210
211 function formatOrderStatus(status: string): string {
212   return status.charAt(0).toUpperCase() + status.slice(1)
213 }
214
215 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
216   switch (status.toLowerCase()) {
217     case "pending":
218       return "secondary"
219     case "processing":
220       return "default"
221     case "shipped":
222       return "outline"
223     case "delivered":
224       return "outline"
225     case "cancelled":
226       return "destructive"
227     default:
228       return "default"
229   }
230 }
231
232 function formatPaymentMethod(method: string): string {
233   switch (method) {
234     case "credit-card":
235       return "Credit Card"
236     case "paypal":
237       return "PayPal"
238     case "bank-transfer":
239       return "Bank Transfer"
240     default:
241       return method
242   }
243 }
244
245 function formatPrice(price: number): string {
246   return `₹ ${price.toFixed(2)}`
247 }
248
249 interface Order {
250   id: string;
251   books: Book[];
252   user: User;
253   address: Address;
254   paymentMethod: PaymentMethod;
255   status: Status;
256   subtotal: number;
257   tax: number;
258   shipping: number;
259   total: number;
260 }
261
262 interface Book {
263   title: string;
264   quantity: number;
265   price: number;
266 }
267
268 interface User {
269   name: string;
270   email: string;
271 }
272
273 interface Address {
274   street: string;
275   city: string;
276   state: string;
277   pincode: number;
278 }
279
280 interface PaymentMethod {
281   type: string;
282 }
283
284 interface Status {
285   type: string;
286 }
287
288 interface OrderDetailsPageProps {
289   params: { id: string };
290 }
291
292 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
293   const { id } = params;
294
295   const order = await prisma.order.findUnique({
296     where: { id },
297     include: {
298       books: true,
299       user: true,
300       address: true,
301       paymentMethod: true,
302       status: true,
303       subtotal: true,
304       tax: true,
305       shipping: true,
306       total: true,
307     },
308   });
309
310   const formattedOrder = formatOrder(order);
311
312   return (
313     <div>
314       <h1>Order Details</h1>
315       <p>Order ID: {id}</p>
316       <table>
317         <thead>
318           <tr>
319             <th>Item</th>
320             <th>Quantity</th>
321             <th>Price</th>
322           </tr>
323         </thead>
324         <tbody>
325           {order.books.map(book => {
326             const { title, quantity, price } = book;
327             return <tr>
328               <td>{title}</td>
329               <td>{quantity}</td>
330               <td>${formatPrice(price)}</td>
331             </tr>
332           })}
333         </tbody>
334       </table>
335       <div>
336         <div>Subtotal:</div>
337         <div>${formatPrice(order.subtotal)}</div>
338       </div>
339       <div>Tax:</div>
340       <div>${formatPrice(order.tax)}</div>
341     </div>
342   </div>
343   <div>Shipping:</div>
344   <div>${formatPrice(order.shipping)}</div>
345
346   <div>Total:</div>
347   <div>${formatPrice(order.total)}</div>
348 }
349
350 function formatOrderStatus(status: string): string {
351   return status.charAt(0).toUpperCase() + status.slice(1)
352 }
353
354 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
355   switch (status.toLowerCase()) {
356     case "pending":
357       return "secondary"
358     case "processing":
359       return "default"
360     case "shipped":
361       return "outline"
362     case "delivered":
363       return "outline"
364     case "cancelled":
365       return "destructive"
366     default:
367       return "default"
368   }
369 }
370
371 function formatPaymentMethod(method: string): string {
372   switch (method) {
373     case "credit-card":
374       return "Credit Card"
375     case "paypal":
376       return "PayPal"
377     case "bank-transfer":
378       return "Bank Transfer"
379     default:
380       return method
381   }
382 }
383
384 function formatPrice(price: number): string {
385   return `₹ ${price.toFixed(2)}`
386 }
387
388 interface Order {
389   id: string;
390   books: Book[];
391   user: User;
392   address: Address;
393   paymentMethod: PaymentMethod;
394   status: Status;
395   subtotal: number;
396   tax: number;
397   shipping: number;
398   total: number;
399 }
400
401 interface Book {
402   title: string;
403   quantity: number;
404   price: number;
405 }
406
407 interface User {
408   name: string;
409   email: string;
410 }
411
412 interface Address {
413   street: string;
414   city: string;
415   state: string;
416   pincode: number;
417 }
418
419 interface PaymentMethod {
420   type: string;
421 }
422
423 interface Status {
424   type: string;
425 }
426
427 interface OrderDetailsPageProps {
428   params: { id: string };
429 }
430
431 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
432   const { id } = params;
433
434   const order = await prisma.order.findUnique({
435     where: { id },
436     include: {
437       books: true,
438       user: true,
439       address: true,
440       paymentMethod: true,
441       status: true,
442       subtotal: true,
443       tax: true,
444       shipping: true,
445       total: true,
446     },
447   });
448
449   const formattedOrder = formatOrder(order);
450
451   return (
452     <div>
453       <h1>Order Details</h1>
454       <p>Order ID: {id}</p>
455       <table>
456         <thead>
457           <tr>
458             <th>Item</th>
459             <th>Quantity</th>
460             <th>Price</th>
461           </tr>
462         </thead>
463         <tbody>
464           {order.books.map(book => {
465             const { title, quantity, price } = book;
466             return <tr>
467               <td>{title}</td>
468               <td>{quantity}</td>
469               <td>${formatPrice(price)}</td>
470             </tr>
471           })}
472         </tbody>
473       </table>
474       <div>
475         <div>Subtotal:</div>
476         <div>${formatPrice(order.subtotal)}</div>
477       </div>
478       <div>Tax:</div>
479       <div>${formatPrice(order.tax)}</div>
480     </div>
481   </div>
482   <div>Shipping:</div>
483   <div>${formatPrice(order.shipping)}</div>
484
485   <div>Total:</div>
486   <div>${formatPrice(order.total)}</div>
487 }
488
489 function formatOrderStatus(status: string): string {
490   return status.charAt(0).toUpperCase() + status.slice(1)
491 }
492
493 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
494   switch (status.toLowerCase()) {
495     case "pending":
496       return "secondary"
497     case "processing":
498       return "default"
499     case "shipped":
500       return "outline"
501     case "delivered":
502       return "outline"
503     case "cancelled":
504       return "destructive"
505     default:
506       return "default"
507   }
508 }
509
510 function formatPaymentMethod(method: string): string {
511   switch (method) {
512     case "credit-card":
513       return "Credit Card"
514     case "paypal":
515       return "PayPal"
516     case "bank-transfer":
517       return "Bank Transfer"
518     default:
519       return method
520   }
521 }
522
523 function formatPrice(price: number): string {
524   return `₹ ${price.toFixed(2)}`
525 }
526
527 interface Order {
528   id: string;
529   books: Book[];
530   user: User;
531   address: Address;
532   paymentMethod: PaymentMethod;
533   status: Status;
534   subtotal: number;
535   tax: number;
536   shipping: number;
537   total: number;
538 }
539
540 interface Book {
541   title: string;
542   quantity: number;
543   price: number;
544 }
545
546 interface User {
547   name: string;
548   email: string;
549 }
550
551 interface Address {
552   street: string;
553   city: string;
554   state: string;
555   pincode: number;
556 }
557
558 interface PaymentMethod {
559   type: string;
560 }
561
562 interface Status {
563   type: string;
564 }
565
566 interface OrderDetailsPageProps {
567   params: { id: string };
568 }
569
570 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
571   const { id } = params;
572
573   const order = await prisma.order.findUnique({
574     where: { id },
575     include: {
576       books: true,
577       user: true,
578       address: true,
579       paymentMethod: true,
580       status: true,
581       subtotal: true,
582       tax: true,
583       shipping: true,
584       total: true,
585     },
586   });
587
588   const formattedOrder = formatOrder(order);
589
590   return (
591     <div>
592       <h1>Order Details</h1>
593       <p>Order ID: {id}</p>
594       <table>
595         <thead>
596           <tr>
597             <th>Item</th>
598             <th>Quantity</th>
599             <th>Price</th>
600           </tr>
601         </thead>
602         <tbody>
603           {order.books.map(book => {
604             const { title, quantity, price } = book;
605             return <tr>
606               <td>{title}</td>
607               <td>{quantity}</td>
608               <td>${formatPrice(price)}</td>
609             </tr>
610           })}
611         </tbody>
612       </table>
613       <div>
614         <div>Subtotal:</div>
615         <div>${formatPrice(order.subtotal)}</div>
616       </div>
617       <div>Tax:</div>
618       <div>${formatPrice(order.tax)}</div>
619     </div>
620   </div>
621   <div>Shipping:</div>
622   <div>${formatPrice(order.shipping)}</div>
623
624   <div>Total:</div>
625   <div>${formatPrice(order.total)}</div>
626 }
627
628 function formatOrderStatus(status: string): string {
629   return status.charAt(0).toUpperCase() + status.slice(1)
630 }
631
632 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
633   switch (status.toLowerCase()) {
634     case "pending":
635       return "secondary"
636     case "processing":
637       return "default"
638     case "shipped":
639       return "outline"
640     case "delivered":
641       return "outline"
642     case "cancelled":
643       return "destructive"
644     default:
645       return "default"
646   }
647 }
648
649 function formatPaymentMethod(method: string): string {
650   switch (method) {
651     case "credit-card":
652       return "Credit Card"
653     case "paypal":
654       return "PayPal"
655     case "bank-transfer":
656       return "Bank Transfer"
657     default:
658       return method
659   }
660 }
661
662 function formatPrice(price: number): string {
663   return `₹ ${price.toFixed(2)}`
664 }
665
666 interface Order {
667   id: string;
668   books: Book[];
669   user: User;
670   address: Address;
671   paymentMethod: PaymentMethod;
672   status: Status;
673   subtotal: number;
674   tax: number;
675   shipping: number;
676   total: number;
677 }
678
679 interface Book {
680   title: string;
681   quantity: number;
682   price: number;
683 }
684
685 interface User {
686   name: string;
687   email: string;
688 }
689
690 interface Address {
691   street: string;
692   city: string;
693   state: string;
694   pincode: number;
695 }
696
697 interface PaymentMethod {
698   type: string;
699 }
700
701 interface Status {
702   type: string;
703 }
704
705 interface OrderDetailsPageProps {
706   params: { id: string };
707 }
708
709 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
710   const { id } = params;
711
712   const order = await prisma.order.findUnique({
713     where: { id },
714     include: {
715       books: true,
716       user: true,
717       address: true,
718       paymentMethod: true,
719       status: true,
720       subtotal: true,
721       tax: true,
722       shipping: true,
723       total: true,
724     },
725   });
726
727   const formattedOrder = formatOrder(order);
728
729   return (
730     <div>
731       <h1>Order Details</h1>
732       <p>Order ID: {id}</p>
733       <table>
734         <thead>
735           <tr>
736             <th>Item</th>
737             <th>Quantity</th>
738             <th>Price</th>
739           </tr>
740         </thead>
741         <tbody>
742           {order.books.map(book => {
743             const { title, quantity, price } = book;
744             return <tr>
745               <td>{title}</td>
746               <td>{quantity}</td>
747               <td>${formatPrice(price)}</td>
748             </tr>
749           })}
750         </tbody>
751       </table>
752       <div>
753         <div>Subtotal:</div>
754         <div>${formatPrice(order.subtotal)}</div>
755       </div>
756       <div>Tax:</div>
757       <div>${formatPrice(order.tax)}</div>
758     </div>
759   </div>
760   <div>Shipping:</div>
761   <div>${formatPrice(order.shipping)}</div>
762
763   <div>Total:</div>
764   <div>${formatPrice(order.total)}</div>
765 }
766
767 function formatOrderStatus(status: string): string {
768   return status.charAt(0).toUpperCase() + status.slice(1)
769 }
770
771 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
772   switch (status.toLowerCase()) {
773     case "pending":
774       return "secondary"
775     case "processing":
776       return "default"
777     case "shipped":
778       return "outline"
779     case "delivered":
780       return "outline"
781     case "cancelled":
782       return "destructive"
783     default:
784       return "default"
785   }
786 }
787
788 function formatPaymentMethod(method: string): string {
789   switch (method) {
790     case "credit-card":
791       return "Credit Card"
792     case "paypal":
793       return "PayPal"
794     case "bank-transfer":
795       return "Bank Transfer"
796     default:
797       return method
798   }
799 }
800
801 function formatPrice(price: number): string {
802   return `₹ ${price.toFixed(2)}`
803 }
804
805 interface Order {
806   id: string;
807   books: Book[];
808   user: User;
809   address: Address;
810   paymentMethod: PaymentMethod;
811   status: Status;
812   subtotal: number;
813   tax: number;
814   shipping: number;
815   total: number;
816 }
817
818 interface Book {
819   title: string;
820   quantity: number;
821   price: number;
822 }
823
824 interface User {
825   name: string;
826   email: string;
827 }
828
829 interface Address {
830   street: string;
831   city: string;
832   state: string;
833   pincode: number;
834 }
835
836 interface PaymentMethod {
837   type: string;
838 }
839
840 interface Status {
841   type: string;
842 }
843
844 interface OrderDetailsPageProps {
845   params: { id: string };
846 }
847
848 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
849   const { id } = params;
850
851   const order = await prisma.order.findUnique({
852     where: { id },
853     include: {
854       books: true,
855       user: true,
856       address: true,
857       paymentMethod: true,
858       status: true,
859       subtotal: true,
860       tax: true,
861       shipping: true,
862       total: true,
863     },
864   });
865
866   const formattedOrder = formatOrder(order);
867
868   return (
869     <div>
870       <h1>Order Details</h1>
871       <p>Order ID: {id}</p>
872       <table>
873         <thead>
874           <tr>
875             <th>Item</th>
876             <th>Quantity</th>
877             <th>Price</th>
878           </tr>
879         </thead>
880         <tbody>
881           {order.books.map(book => {
882             const { title, quantity, price } = book;
883             return <tr>
884               <td>{title}</td>
885               <td>{quantity}</td>
886               <td>${formatPrice(price)}</td>
887             </tr>
888           })}
889         </tbody>
890       </table>
891       <div>
892         <div>Subtotal:</div>
893         <div>${formatPrice(order.subtotal)}</div>
894       </div>
895       <div>Tax:</div>
896       <div>${formatPrice(order.tax)}</div>
897     </div>
898   </div>
899   <div>Shipping:</div>
900   <div>${formatPrice(order.shipping)}</div>
901
902   <div>Total:</div>
903   <div>${formatPrice(order.total)}</div>
904 }
905
906 function formatOrderStatus(status: string): string {
907   return status.charAt(0).toUpperCase() + status.slice(1)
908 }
909
910 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
911   switch (status.toLowerCase()) {
912     case "pending":
913       return "secondary"
914     case "processing":
915       return "default"
916     case "shipped":
917       return "outline"
918     case "delivered":
919       return "outline"
920     case "cancelled":
921       return "destructive"
922     default:
923       return "default"
924   }
925 }
926
927 function formatPaymentMethod(method: string): string {
928   switch (method) {
929     case "credit-card":
930       return "Credit Card"
931     case "paypal":
932       return "PayPal"
933     case "bank-transfer":
934       return "Bank Transfer"
935     default:
936       return method
937   }
938 }
939
940 function formatPrice(price: number): string {
941   return `₹ ${price.toFixed(2)}`
942 }
943
944 interface Order {
945   id: string;
946   books: Book[];
947   user: User;
948   address: Address;
949   paymentMethod: PaymentMethod;
950   status: Status;
951   subtotal: number;
952   tax: number;
953   shipping: number;
954   total: number;
955 }
956
957 interface Book {
958   title: string;
959   quantity: number;
960   price: number;
961 }
962
963 interface User {
964   name: string;
965   email: string;
966 }
967
968 interface Address {
969   street: string;
970   city: string;
971   state: string;
972   pincode: number;
973 }
974
975 interface PaymentMethod {
976   type: string;
977 }
978
979 interface Status {
980   type: string;
981 }
982
983 interface OrderDetailsPageProps {
984   params: { id: string };
985 }
986
987 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
988   const { id } = params;
989
990   const order = await prisma.order.findUnique({
991     where: { id },
992     include: {
993       books: true,
994       user: true,
995       address: true,
996       paymentMethod: true,
997       status: true,
998       subtotal: true,
999       tax: true,
1000      shipping: true,
1001      total: true,
1002    },
1003  });
1004
1005  const formattedOrder = formatOrder(order);
1006
1007  return (
1008    <div>
1009      <h1>Order Details</h1>
1010      <p>Order ID: {id}</p>
1011      <table>
1012        <thead>
1013          <tr>
1014            <th>Item</th>
1015            <th>Quantity</th>
1016            <th>Price</th>
1017          </tr>
1018        </thead>
1019        <tbody>
1020          {order.books.map(book => {
1021            const { title, quantity, price } = book;
1022            return <tr>
1023              <td>{title}</td>
1024              <td>{quantity}</td>
1025              <td>${formatPrice(price)}</td>
1026            </tr>
1027          })}
1028        </tbody>
1029      </table>
1030      <div>
1031        <div>Subtotal:</div>
1032        <div>${formatPrice(order.subtotal)}</div>
1033      </div>
1034      <div>Tax:</div>
1035      <div>${formatPrice(order.tax)}</div>
1036    </div>
1037  </div>
1038  <div>Shipping:</div>
1039  <div>${formatPrice(order.shipping)}</div>
1040
1041  <div>Total:</div>
1042  <div>${formatPrice(order.total)}</div>
1043 }
1044
1045 function formatOrderStatus(status: string): string {
1046   return status.charAt(0).toUpperCase() + status.slice(1)
1047 }
1048
1049 function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
1050   switch (status.toLowerCase()) {
1051     case "pending":
1052       return "secondary"
1053     case "processing":
1054       return "default"
1055     case "shipped":
1056       return "outline"
1057     case "delivered":
1058       return "outline"
1059     case "cancelled":
1060       return "destructive"
1061     default:
1062       return "default"
1063   }
1064 }
1065
1066 function formatPaymentMethod(method: string): string {
1067   switch (method) {
1068     case "credit-card":
1069       return "Credit Card"
1070     case "paypal":
1071       return "PayPal"
1072     case "bank-transfer":
1073       return "Bank Transfer"
1074     default:
1075       return method
1076   }
1077 }
1078
1079 function formatPrice(price: number): string {
1080   return `₹ ${price.toFixed(2)}`
1081 }
1082
1083 interface Order {
1084   id: string;
1085   books: Book[];
1086   user: User;
1087   address: Address;
1088   paymentMethod: PaymentMethod;
1089   status: Status;
1090   subtotal: number;
1091   tax: number;
1092   shipping: number;
1093   total: number;
1094 }
1095
1096 interface Book {
1097   title: string;
1098   quantity: number;
1099   price: number;
1100 }
1101
1102 interface User {
1103   name: string;
1104   email: string;
1105 }
1106
1107 interface Address {
1108   street: string;
1109   city: string;
1110   state: string;
1111   pincode: number;
1112 }
1113
1114 interface PaymentMethod {
1115   type: string;
1116 }
1117
1118 interface Status {
1119   type: string;
1120 }
1121
1122 interface OrderDetailsPageProps {
1123   params: { id: string };
1124 }
1125
1126 function OrderDetailsPage({ params }: OrderDetailsPageProps) {
1127   const { id } = params;
1128
1129   const order = await prisma.order.findUnique({
1130     where: { id },
1131     include: {
1132       books: true,
1133       user: true,
1134       address: true,
1135       paymentMethod: true,
1136       status: true,
1137       subtotal: true,
1138       tax: true,
1139       shipping: true,
1140       total: true,
1141     },
1142   });
1143
1144   const formattedOrder = formatOrder(order);
1145
1146   return (
1147     <div>
1148       <h1>Order Details</h1>
1149       <p>Order ID: {id}</p>
1150       <table>
1151         <thead>
1152           <tr>
1153             <th>Item</th>
1154             <th>Quantity</th>
1155             <th>Price</th>
1156           </tr>
1157         </thead>
1158         <tbody>
1159           {order.books.map(book => {
1160             const { title, quantity, price } = book;
1161             return <tr>
1162               <td>{title}</td>
1163               <td>{quantity}</td>
1164               <td>${formatPrice(price)}</td>
1165             </tr>
1166           })}
1167         </tbody>
1168       </table>
1169       <div>
1170         <div>Subtotal:</div>
1171         <div>${formatPrice(order.subtotal)}</div>
1172       </div>
1173       <div>Tax:</div>
1174       <div>${formatPrice(order.tax)}
```

➤ Order processing code screenshots



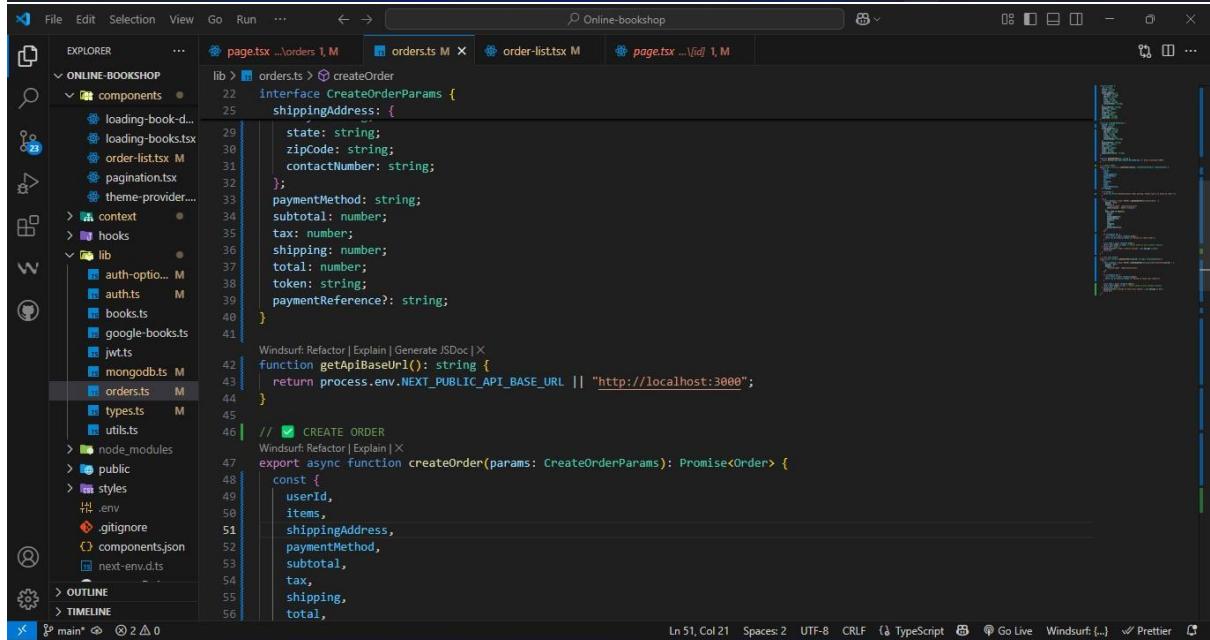
The screenshot shows the VS Code interface with the 'orders.ts' file open in the center editor tab. The file contains TypeScript code defining an 'Order' interface and a 'createOrder' function. The 'orders.ts' file is part of a library named 'lib'. The code includes imports for 'Windsurf', 'Refactor', and 'Explain'. It defines an 'Order' interface with properties like '_id', 'userId', 'items', 'shippingAddress', 'paymentMethod', 'subtotal', 'tax', 'shipping', 'total', 'createdAt', and 'updatedAt'. It also defines a 'CreateOrderParams' interface with 'userId', 'items', 'shippingAddress', 'paymentMethod', 'subtotal', 'tax', 'shipping', and 'total'. The 'createOrder' function uses the 'getApiBaseUrl' function to return a promise for an 'Order' object.

```
lib > orders.ts > createOrder
interface Order {
  _id: string;
  userId: string;
  items: any[];
  shippingAddress: {
    fullName: string;
    address: string;
    city: string;
    state: string;
    zipCode: string;
    contactNumber: string;
  };
  paymentMethod: string;
  subtotal: number;
  tax: number;
  shipping: number;
  total: number;
  createdAt: string;
  updatedAt: string;
}

interface CreateOrderParams {
  userId: string;
  items: any[];
  shippingAddress: {
    fullName: string;
    address: string;
    city: string;
    state: string;
    zipCode: string;
  };
}

function getApiBaseUrl(): string {
  return process.env.NEXT_PUBLIC_API_BASE_URL || "http://localhost:3008";
}

// ✅ CREATE ORDER
export async function createOrder(params: CreateOrderParams): Promise<Order> {
  const {
    userId,
    items,
    shippingAddress,
    paymentMethod,
    subtotal,
    tax,
    shipping,
    total,
  } = params;
}
```



The second screenshot shows the same 'orders.ts' file in VS Code, but with code completion suggestions visible in the editor. A tooltip for the 'shipping' parameter in the 'createOrder' function shows options like 'state', 'zipCode', 'contactNumber', 'paymentMethod', 'subtotal', 'tax', and 'total'. The rest of the code is identical to the first screenshot.

Online-bookshop

```

File Edit Selection View Go Run ... ⏪ ⏩ Online-bookshop
EXPLORER page.tsx ...orders 1, M orders.ts M order-listtsx M page.tsx ...[id] 1, M
ONLINE-BOOKSHOP
  components
    loading-book-d...
    loading-books.tsx
    order-list.tsx M
    pagination.tsx
    theme-provider...
  context
  hooks
  lib
    auth-optio... M
    authts M
    books.ts
    google-books.ts
    jwt.ts
    mongodb.ts M
    orders.ts M
    types.ts M
    utils.ts
  node_modules
  public
  styles
  .env
  .gitignore
  components.json
  next-env.d.ts
  OUTLINE
  TIMELINE
  main* ⏪ 2 △ 0
  Orders lib > orders.ts > createOrder
  47 export async function createOrder(params: CreateOrderParams): Promise<Order> {
  48   const response = await fetch(`${getApiClientUrl()}/api/orders`, {
  49     method: "POST",
  50     headers: {
  51       "Content-Type": "application/json",
  52       Authorization: `Bearer ${token}`,
  53     },
  54     body: JSON.stringify({
  55       userId,
  56       items,
  57       shippingAddress,
  58       paymentMethod,
  59       subtotal,
  60       tax,
  61       shipping,
  62       total,
  63       paymentReference,
  64     }),
  65   );
  66   if (!response.ok) {
  67     throw new Error("Authentication token missing. Please log in to place an order.");
  68   }
  69   try {
  70     const data = await response.json();
  71     return data.order || data; // Adjust based on your backend response
  72   } catch (err: any) {
  73     console.error(`Order creation failed: ${err.message || err}`);
  74     throw err;
  75   }
  76 }
  77 // GET USER ORDERS
  78 // Wardsurf Refactor | Explain X
  79 export async function getUserOrders(userId: string): Promise<Order[]> {
  80   try {
  81     const response = await fetch(`${getApiClientUrl()}/api/orders?userId=${userId}`, {
  82       method: "GET",
  83       headers: {
  84         "Content-Type": "application/json",
  85       },
  86     });
  87     if (!response.ok) {
  88       const error = await response.json();
  89       throw new Error(error.error || "Failed to fetch user orders");
  90     }
  91     const data = await response.json();
  92     return data.orders || data; // Adjust based on your backend response
  93   } catch (err: any) {
  94     console.error(`User order retrieval failed: ${err.message || err}`);
  95     throw err;
  96   }
  97 }
  98
  99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
  
```

Ln 51, Col 21 Spaces 2 UTF-8 CRLF ⓘ TypeScript ⓘ Go Live ⓘ Windsurf: [...] ⓘ Prettier ⓘ

Online-bookshop

```

File Edit Selection View Go Run ... ⏪ ⏩ Online-bookshop
EXPLORER page.tsx ...orders 1, M orders.ts M order-listtsx M page.tsx ...[id] 1, M
ONLINE-BOOKSHOP
  components
    loading-book-d...
    loading-books.tsx
    order-list.tsx M
    pagination.tsx
    theme-provider...
  context
  hooks
  lib
    auth-optio... M
    authts M
    books.ts
    google-books.ts
    jwt.ts
    mongodb.ts M
    orders.ts M
    types.ts M
    utils.ts
  node_modules
  public
  styles
  .env
  .gitignore
  components.json
  next-env.d.ts
  OUTLINE
  TIMELINE
  main* ⏪ 2 △ 0
  Orders lib > orders.ts > createOrder
  47 export async function createOrder(params: CreateOrderParams): Promise<Order> {
  48   const response = await fetch(`${getApiClientUrl()}/api/orders`, {
  49     method: "POST",
  50     headers: {
  51       "Content-Type": "application/json",
  52       Authorization: `Bearer ${token}`,
  53     },
  54     body: JSON.stringify({
  55       userId,
  56       items,
  57       shippingAddress,
  58       paymentMethod,
  59       subtotal,
  60       tax,
  61       shipping,
  62       total,
  63       paymentReference,
  64     }),
  65   );
  66   if (!response.ok) {
  67     const error = await response.json();
  68     throw new Error(error.error || "Failed to create order");
  69   }
  70   const data = await response.json();
  71   return data.order || data; // Adjust based on your backend response
  72 }
  73 // GET USER ORDERS
  74 // Wardsurf Refactor | Explain X
  75 export async function getUserOrders(userId: string): Promise<Order[]> {
  76   try {
  77     const response = await fetch(`${getApiClientUrl()}/api/orders?userId=${userId}`, {
  78       method: "GET",
  79       headers: {
  80         "Content-Type": "application/json",
  81       },
  82     });
  83     if (!response.ok) {
  84       const error = await response.json();
  85       throw new Error(error.error || "Failed to fetch user orders");
  86     }
  87     const data = await response.json();
  88     return data.orders || data; // Adjust based on your backend response
  89   } catch (err: any) {
  90     console.error(`User order retrieval failed: ${err.message || err}`);
  91     throw err;
  92   }
  93 }
  94
  95
  96
  97
  98
  99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
  
```

Ln 51, Col 21 Spaces 2 UTF-8 CRLF ⓘ TypeScript ⓘ Go Live ⓘ Windsurf: [...] ⓘ Prettier ⓘ



```
lib > orders.ts > createOrder
  99  export async function getUserOrders(userId: string): Promise<Order[]> {
100    const response = await fetch(`https://api.example.com/orders?userId=${userId}`);
101    if (!response.ok) {
102      throw new Error(`Failed to fetch user orders: ${response.statusText}`);
103    }
104    const data = await response.json();
105    return data.orders || [];
106  } // Adjust based on your backend response
107  catch (err: any) {
108    console.error(`Failed to fetch user orders: ${err.message || err}`);
109    throw err;
110  }
111}
112
113
114
115
116
117
118
119}
```

1.5. User Profile

➤ Profile view screenshot

This screenshot shows the BookHaven user profile page at localhost:3000/profile. The page has a dark theme. At the top, there's a navigation bar with links for Home, Books, Categories, About, Contact, and a search bar labeled "Search books...". A sidebar on the right shows a user menu with options: Chamod Dhananjaya, Profile, My Orders, and Logout. The main content area is titled "Profile Information" and contains fields for "Full Name" (Chamod Dhananjaya), "Email" (chamoddhananjaya20000@gmail.com), and a "Change Password" section with "Current Password" and "New Password" fields. A red notification bar at the bottom left indicates "2 Issues" and a "New Password" link.

➤ Edit profile interface screenshot

This screenshot shows the BookHaven user profile edit page at localhost:3000/profile. The interface is similar to the view page but includes a "Change Profile Picture" section with a placeholder image and a "Choose File" button. The "Edit Profile" section contains fields for "Full Name" (Chamod Dhananjaya), "Email" (chamoddhananjaya20000@gmail.com), and a "Change Password" section with "Current Password" (2000chamod), "New Password" (2001chamod), and "Confirm New Password" fields. A blue "Update Profile" button is located at the bottom. A red notification bar at the bottom left indicates "2 Issues" and a "New Password" link.

➤ Order history view screenshot

The screenshot shows the 'My Orders' section of the BookHaven website. The main content area features a dark background with white text. It says 'You haven't placed any orders yet' and 'Browse our collection and find your next favorite book!'. Below this is a blue 'Browse Books' button. At the top, there's a navigation bar with links for Home, Books, Categories, About, Contact, and a search bar labeled 'Search books...'. The footer is divided into four sections: BookHaven (with a brief description and social media icons), Quick Links (Home, Books, Categories, About Us, Contact), Customer Service (FAQ, Shipping Policy, Returns & Refunds, Privacy Policy, Terms & Conditions), and Contact Us (with address 123 Book Street, Reading City, RC 12345, United States, and email info@bookhaven.com).

➤ Related implementation code screenshots

The screenshot shows the code editor interface with the file 'page.tsx' open. The code is a functional component named 'OrderList' that takes an array of 'Order' objects as a prop. It maps through the orders to render a list item for each. Each list item contains a badge indicating the order status, a link to view details, and a button to place the order. The code uses various utility functions and types from '@/lib/utils' and '@/components/ui'. The editor has a sidebar showing the project structure, including components like 'loading-book.tsx' and 'order-list.tsx', and files like 'orders.ts' and 'utils.ts'. The bottom status bar shows the file is 1,120 bytes long and was last modified by chamoddhananjay2000.

File Edit Selection View Go Run ... ⏪ ⏩ Online-bookshop

EXPLORER ONLINE-BOOKSHOP

components > order-list.tsx > ...

```
7  export default function OrderList({ orders }: { orders: Order[] }) {
8    orders.map((order) => (
9      order.orderItems.slice(0, 3).map((item) => (
10        <p className="font-medium line-clamp-1">{item.title}</p>
11        <p className="text-sm text-gray-500 dark:text-gray-400">
12          Qty: {item.quantity} <span>{formatPrice(item.price)}</span>
13        </p>
14        </div>
15        <div className="text-right">
16          <p className="font-medium">{formatPrice(item.price * item.quantity)}</p>
17        </div>
18      )));
19
20      if(order.orderItems.length > 3 && (
21        <p className="text-sm text-gray-500 dark:text-gray-400 italic">
22          +{order.orderItems.length - 3} more items
23        </p>
24      ))
25    );
26  );
27}
28
29 Windsurf: Refactor | Explain | Generate JSDoc | X
30 function formatOrderStatus(status: string): string {
31   return status.charAt(0).toUpperCase() + status.slice(1)
32 }
```

File Edit Selection View Go Run ... ⏪ ⏩ Online-bookshop

EXPLORER ONLINE-BOOKSHOP

components > order-list.tsx > ...

```
59 }
60
61 Windsurf: Refactor | Explain | Generate JSDoc | X
62 function getOrderStatusVariant(status: string): "default" | "secondary" | "outline" | "destructive" {
63   switch (status.toLowerCase()) {
64     case "pending":
65       return "secondary"
66     case "processing":
67       return "default"
68     case "shipped":
69       return "outline"
70     case "delivered":
71       return "outline"
72     case "cancelled":
73       return "destructive"
74     default:
75       return "default"
76   }
77 }
```