



# **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

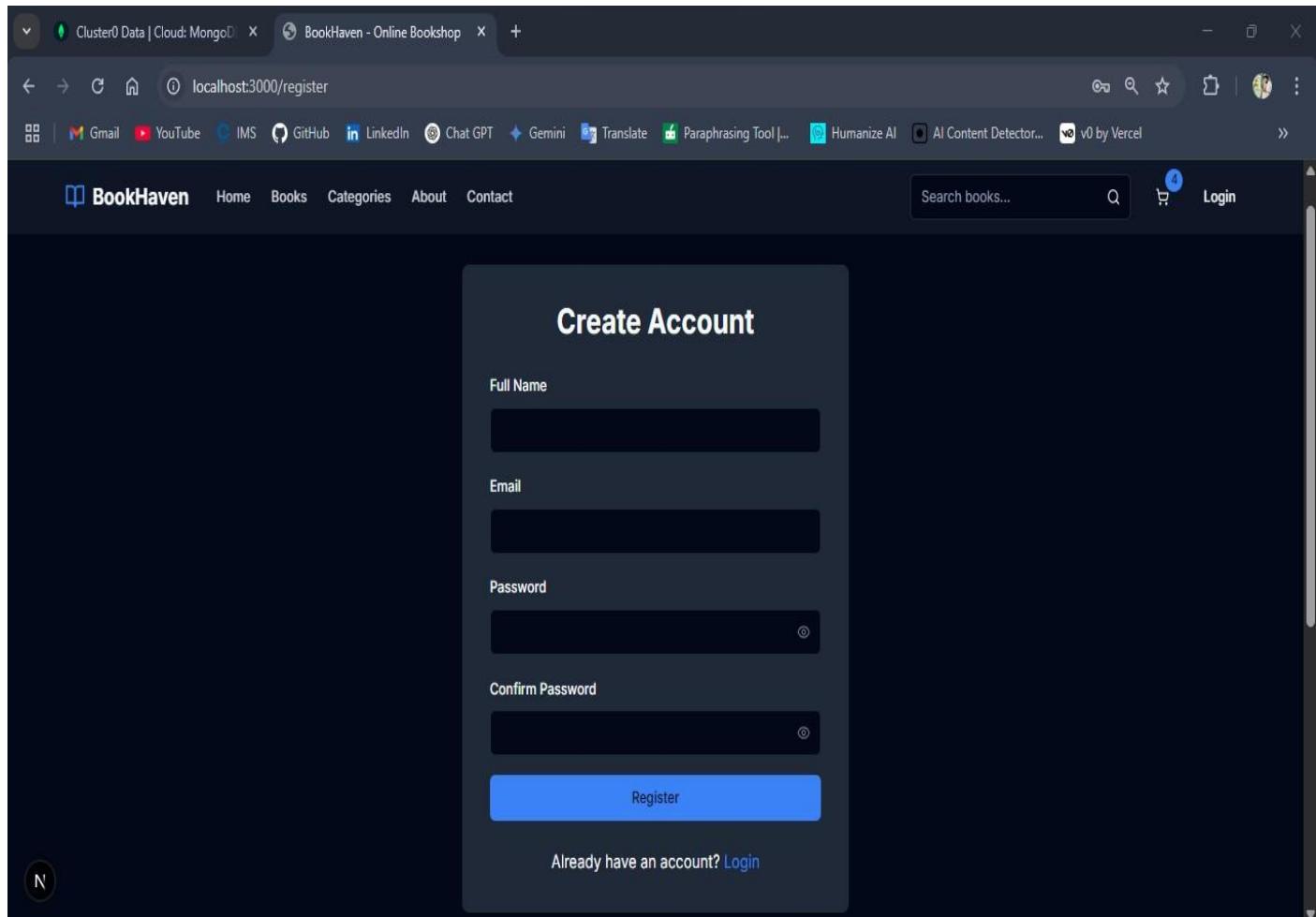
## **Table of Contents**

<b>1. User Interface Documentation Requirements .....</b>	<b>3</b>
<b>    1.1. User Authentication Screens .....</b>	<b>3</b>
<b>    1.2. Book Listing and Search.....</b>	<b>11</b>
<b>    1.3. Shopping Cart .....</b>	<b>16</b>
<b>    1.4. Checkout Process .....</b>	<b>21</b>
<b>    1.5. User Profile.....</b>	<b>29</b>

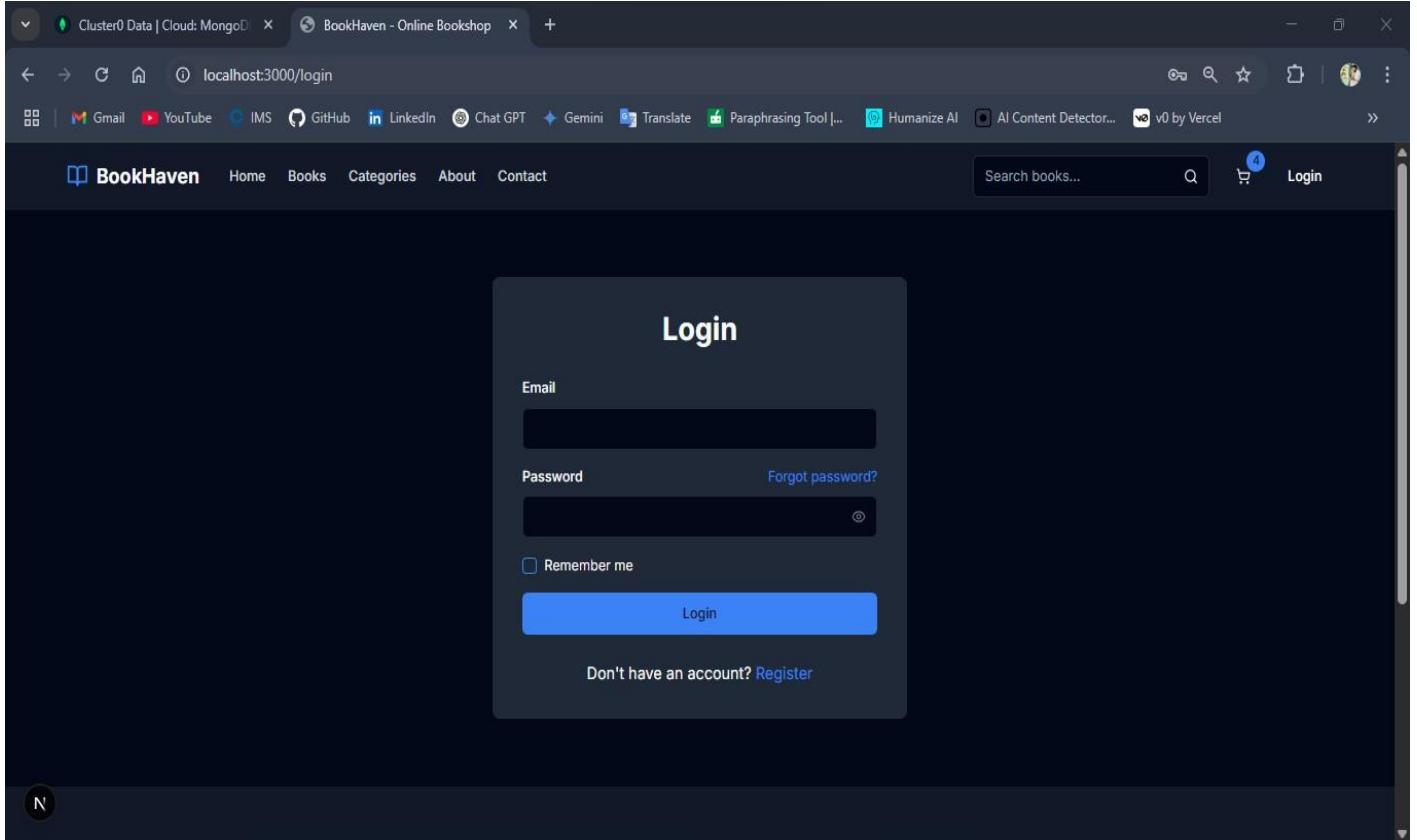
# 1. User Interface Documentation Requirements

## 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 project. 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 uses useToasts for displaying success or error messages. The code editor interface shows the file structure on the left and the code content on the right, with various tools like Windurf and Prettier visible 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

MAIN 0 0 0

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

MAIN 0 0 0

```

app > register > page.tsx > RegisterPage > handleSubmit
16 export default function RegisterPage() {
  ...
  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    const { name, value } = e.target
    setFormData((prev) => ({ ...prev, [name]: value }))
  }

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault()

    if (formData.password !== formData.confirmPassword) {
      toast({
        title: "Passwords don't match",
        description: "Please make sure your passwords match.",
        variant: "destructive",
      })
      return
    }

    try {
      setIsLoading(true)

      const userData = await registerUser({
        name: formData.name,
        email: formData.email,
        password: formData.password,
      })

      setUser({
        ...userData,
        profileImageUrl: userData.profileImageUrl || "", // Provide a default value if missing
      })
    } finally {
      setIsLoading(false)
    }
  }
}

return (
  <div className="container mx-auto px-4 py-16 flex justify-center">
    <div className="w-full max-w-md">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-8">
        <h1 className="text-3xl font-bold mb-6 text-center">Create Account</h1>
        <form onSubmit={handleSubmit} className="space-y-4">
          <div className="space-y-2">
            <Label htmlFor="name">Full Name</Label>
            ...
          </div>
        </form>
      </div>
    </div>
  </div>
)

```

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 page.tsx M
ONLINE-BOOKSHOP
> .next
> app
  > about
  > api
  > books
  > cart
  > categories
  > checkout
  > confirmati...
  > contact
  > login
  > orders
  > profile
  > register
    > page.tsx M
    > scripts
    > globals.css
    > layout.tsx
    > page.tsx
  > components
  > context
  > hooks
  > lib
  > node_modules
  > public
  > other
  > OUTLINE
  > TIMELINE
File Edit Selection View Go Run ... Online-bookshop
EXPLORER page.tsx M
ONLINE-BOOKSHOP
> .next
> app
  > about
  > api
  > books
  > cart
  > categories
  > checkout
  > confirmati...
  > contact
  > login
  > orders
  > profile
  > register
    > page.tsx M
    > scripts
    > globals.css
    > layout.tsx
    > page.tsx
  > components
  > context
  > hooks
  > lib
  > node_modules
  > public
  > other
  > OUTLINE
  > TIMELINE
```

page.tsx (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>

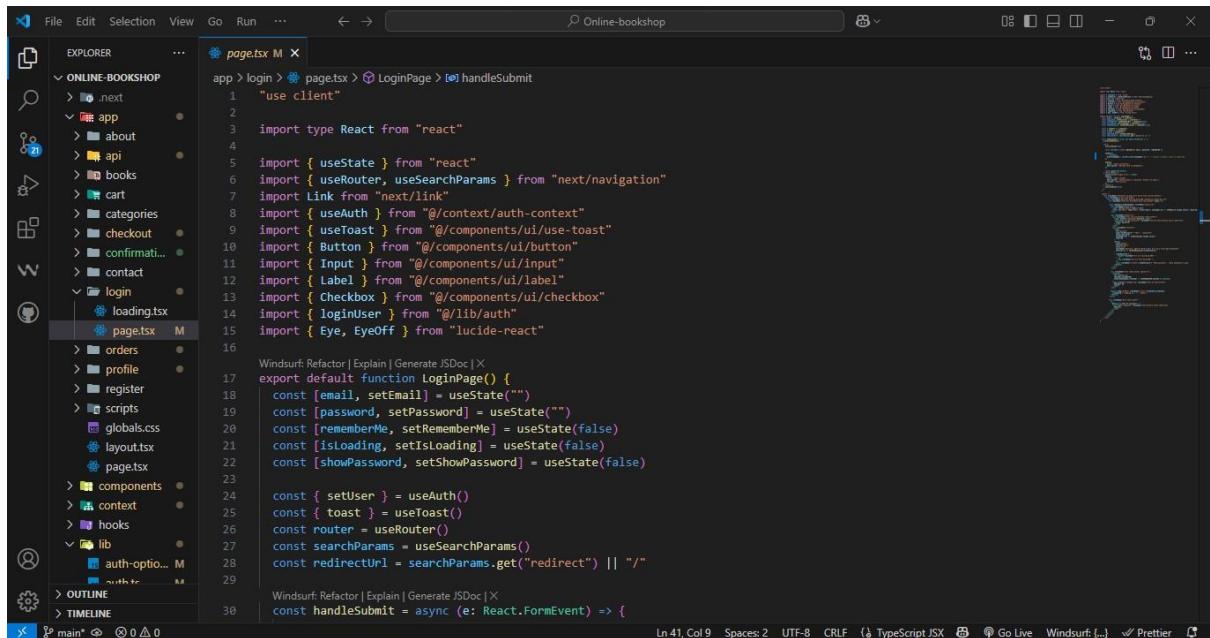
  <div className="space-y-2">
    <Label htmlFor="confirmPassword">Confirm Password</Label>
    <div className="relative">
      <Input
        id="confirmPassword"
        name="confirmPassword"
        type={showConfirmPassword ? "text" : "password"}
        value={formData.confirmPassword}
        onChange={handleChange}
        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 screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under 'ONLINE-BOOKSHOP'. The 'register' folder is expanded, showing files like 'about.tsx', 'api.tsx', 'books.tsx', 'cart.tsx', 'categories.tsx', 'checkout.tsx', 'confirmati... (truncated)', 'contact.tsx', 'login.tsx', 'orders.tsx', 'profile.tsx', and 'page.tsx'.
- Editor:** The 'page.tsx' file is open in the editor. The code is a TypeScript component named 'RegisterPage'. It includes logic for password visibility ('showConfirmPassword'), button states ('disabled' based on 'isLoading'), and account creation ('Creating account...' or 'Register'). It also contains a link to the login page.
- Bottom Status Bar:** Displays information such as 'Not Committed Yet', line numbers (Ln 57, Col 16), spaces (Spaces: 2), encoding (UTF-8), file type (TypeScript JSX), and various developer tools like 'Go Live' and 'Windsurf'.

```
app > register > page.tsx > RegisterPage > handleSubmit
16  export default function RegisterPage() {
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23    ...
24    ...
25    ...
26    ...
27    ...
28    ...
29    ...
30    ...
31    ...
32    ...
33    ...
34    ...
35    ...
36    ...
37    ...
38    ...
39    ...
40    ...
41    ...
42    ...
43    ...
44    ...
45    ...
46    ...
47    ...
48    ...
49    ...
50    ...
51    ...
52    ...
53    ...
54    ...
55    ...
56    ...
57    ...
58    ...
59    ...
60    ...
61    ...
62    ...
63    ...
64    ...
65    ...
66    ...
67    ...
68    ...
69    ...
70    ...
71    ...
72    ...
73    ...
74    ...
75    ...
76    ...
77    ...
78    ...
79    ...
80    ...
81    ...
82    ...
83    ...
84    ...
85    ...
86    ...
87    ...
88    ...
89    ...
90    ...
91    ...
92    ...
93    ...
94    ...
95    ...
96    ...
97    ...
98    ...
99    ...
100   ...
101   ...
102   ...
103   ...
104   ...
105   ...
106   ...
107   ...
108   ...
109   ...
110   ...
111   ...
112   ...
113   ...
114   ...
115   ...
116   ...
117   ...
118   ...
119   ...
120   ...
121   ...
122   ...
123   ...
124   ...
125   ...
126   ...
127   ...
128   ...
129   ...
130   ...
131   ...
132   ...
133   ...
134   ...
135   ...
136   ...
137   ...
138   ...
139   ...
140   ...
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   ...
```

- Login page code screenshots



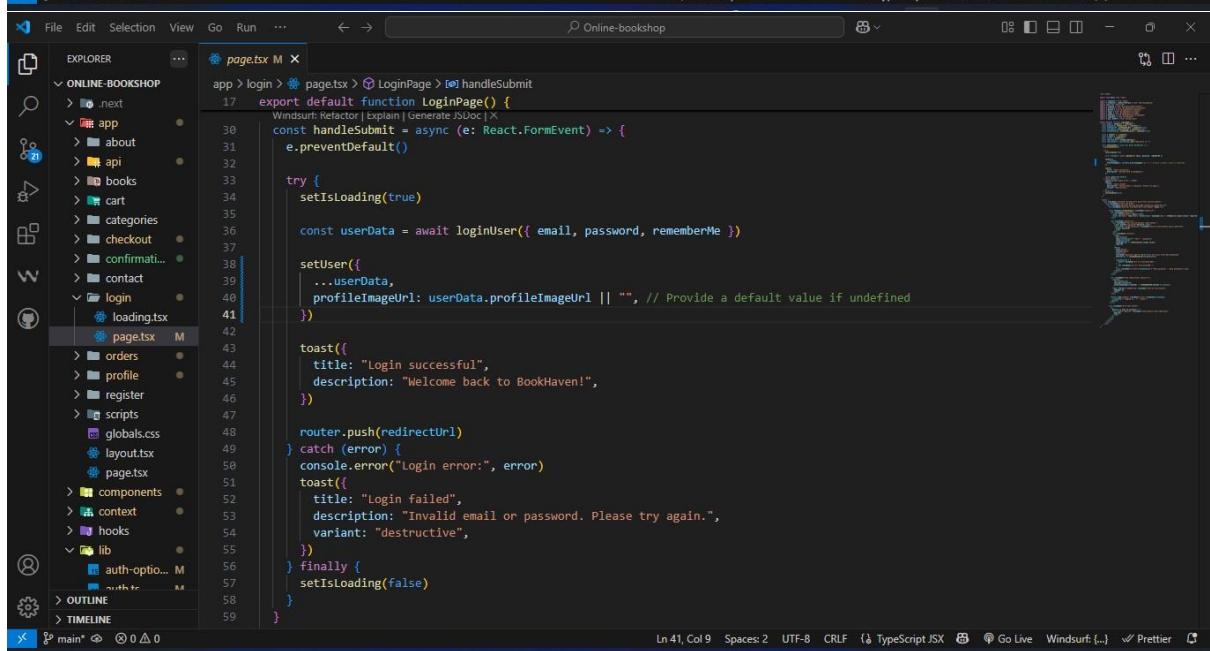
The screenshot shows the VS Code interface with the file `page.tsx` open in the editor. The code is as follows:

```

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 export default function LoginPage() {
18   const [email, setEmail] = useState("")
19   const [password, setPassword] = useState("")
20   const [rememberMe, setRememberMe] = useState(false)
21   const [isLoading, setIsLoading] = useState(false)
22   const [showPassword, setShowPassword] = useState(false)
23
24   const { setUser } = useAuth()
25   const { toast } = useToast()
26   const router = useRouter()
27   const searchParams = useSearchParams()
28   const redirectUrl = searchParams.get("redirect") || "/"
29
30   const handleSubmit = async (e: React.FormEvent) => {
31     e.preventDefault()
32
33     try {
34       setIsLoading(true)
35
36       const userData = await loginUser({ email, password, rememberMe })
37
38       setUser({
39         ...userData,
40         profileImageUrl: userData.profileImageUrl || "", // Provide a default value if undefined
41       })
42
43       toast({
44         title: "Login successful",
45         description: "Welcome back to BookHaven!",
46       })
47
48       router.push(redirectUrl)
49     } catch (error) {
50       console.error("Login error:", error)
51       toast({
52         title: "Login failed",
53         description: "Invalid email or password. Please try again.",
54         variant: "destructive",
55       })
56     } finally {
57       setIsLoading(false)
58     }
59   }
}

```

The code is annotated with several Windurf comments: `Windsurf: Refactor | Explain | Generate JSDoc | X` at line 17, `Windsurf: Refactor | Explain | Generate JSDoc | X` at line 29, and `Windsurf: Refactor | Explain | Generate JSDoc | X` at line 30.

The screenshot shows the VS Code interface with the file `page.tsx` open in the editor. The code has been refactored, and the `handleSubmit` function now includes a `Windsurf: Go Live` annotation. The code is identical to the previous screenshot.

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 ⌂

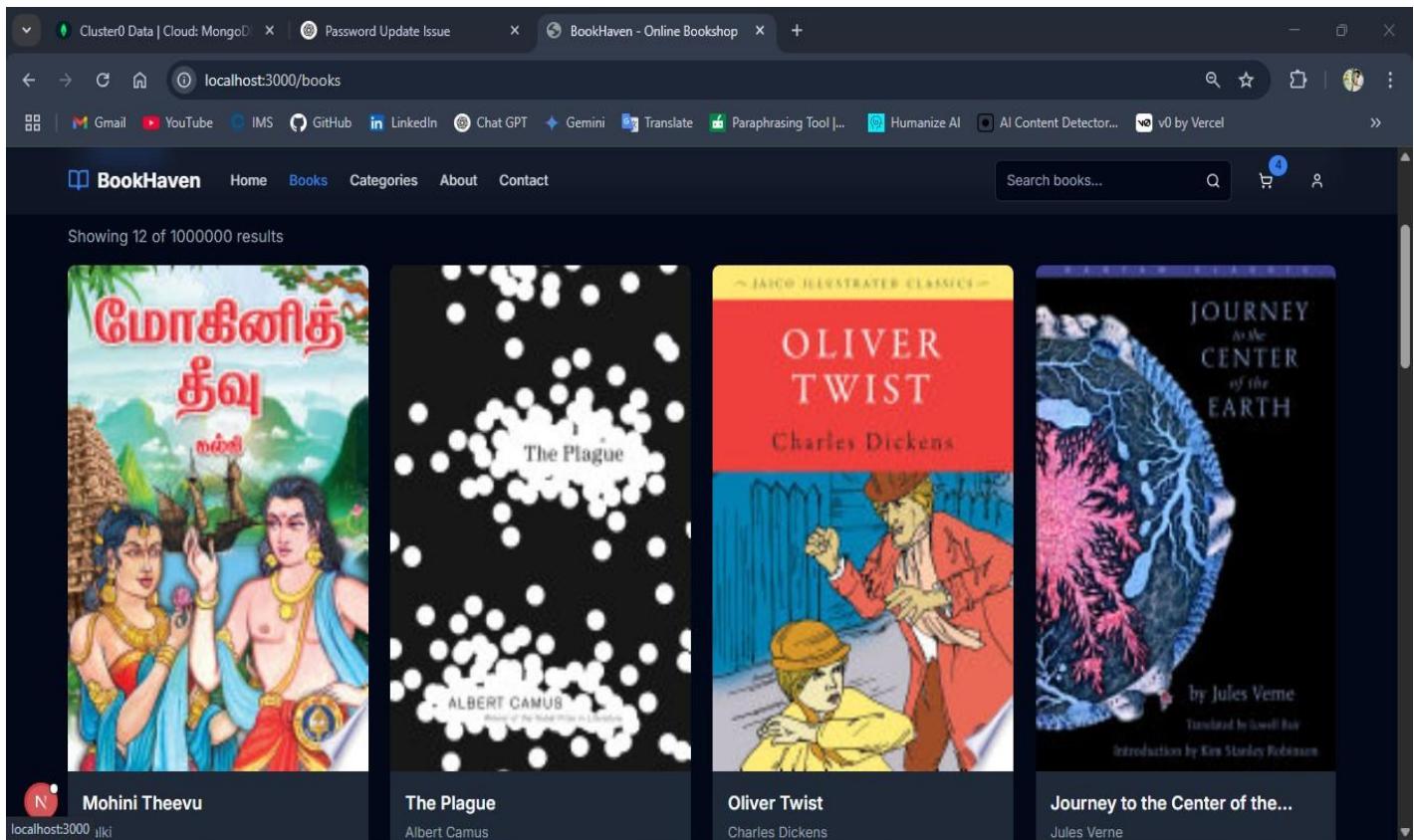
```
app > login > page.tsx > LoginPage > handleSubmit
17  export default function LoginPage() {
116    <Button type="submit" className="w-full" disabled={isLoading}>
117      | {isLoading ? "Logging in..." : "Login"}
118    </Button>
119
120    <div className="mt-6 text-center">
121      <p>
122        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 ⓘ

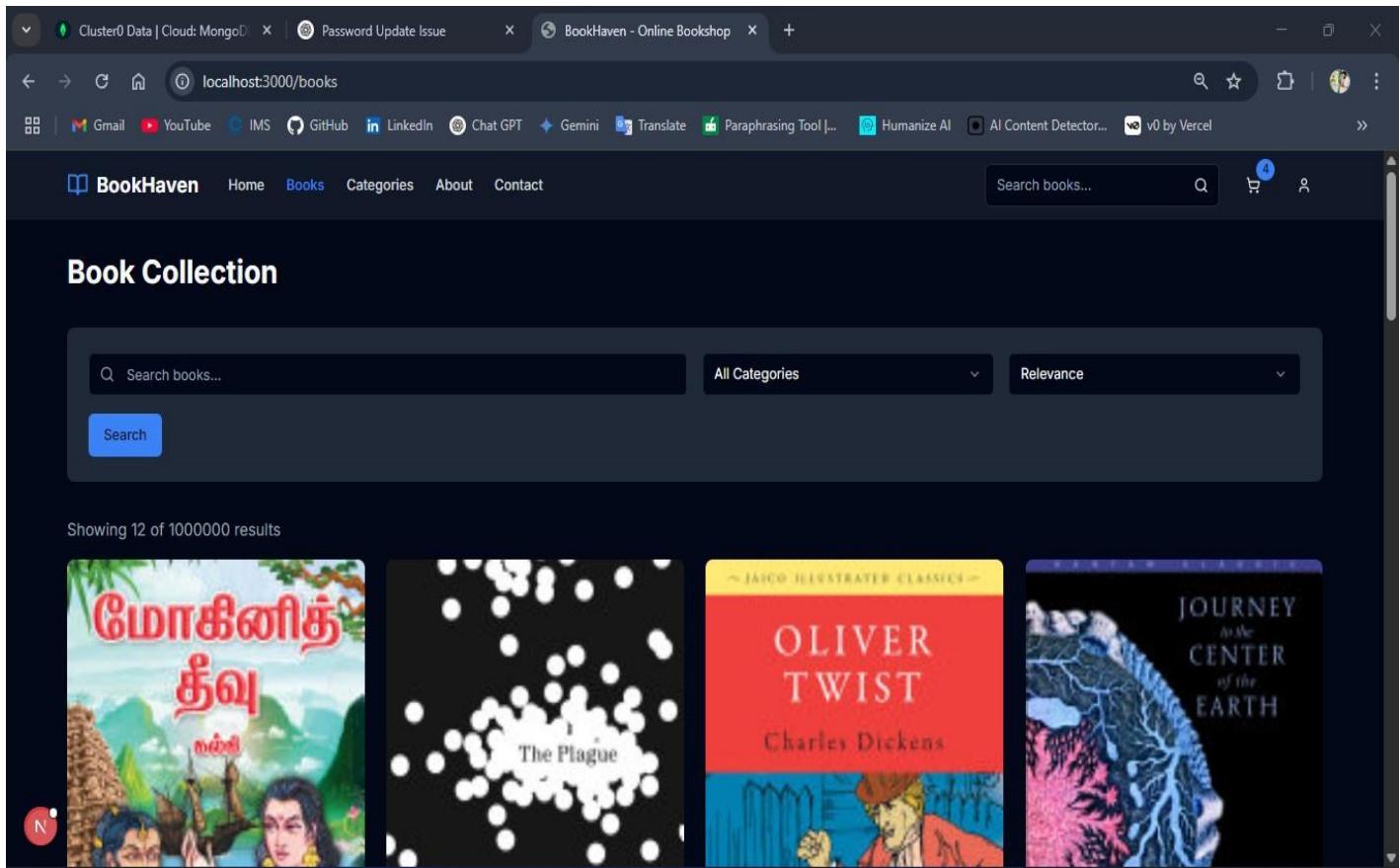
- Authentication success/error messages screenshots

## 1.2. Book Listing and Search

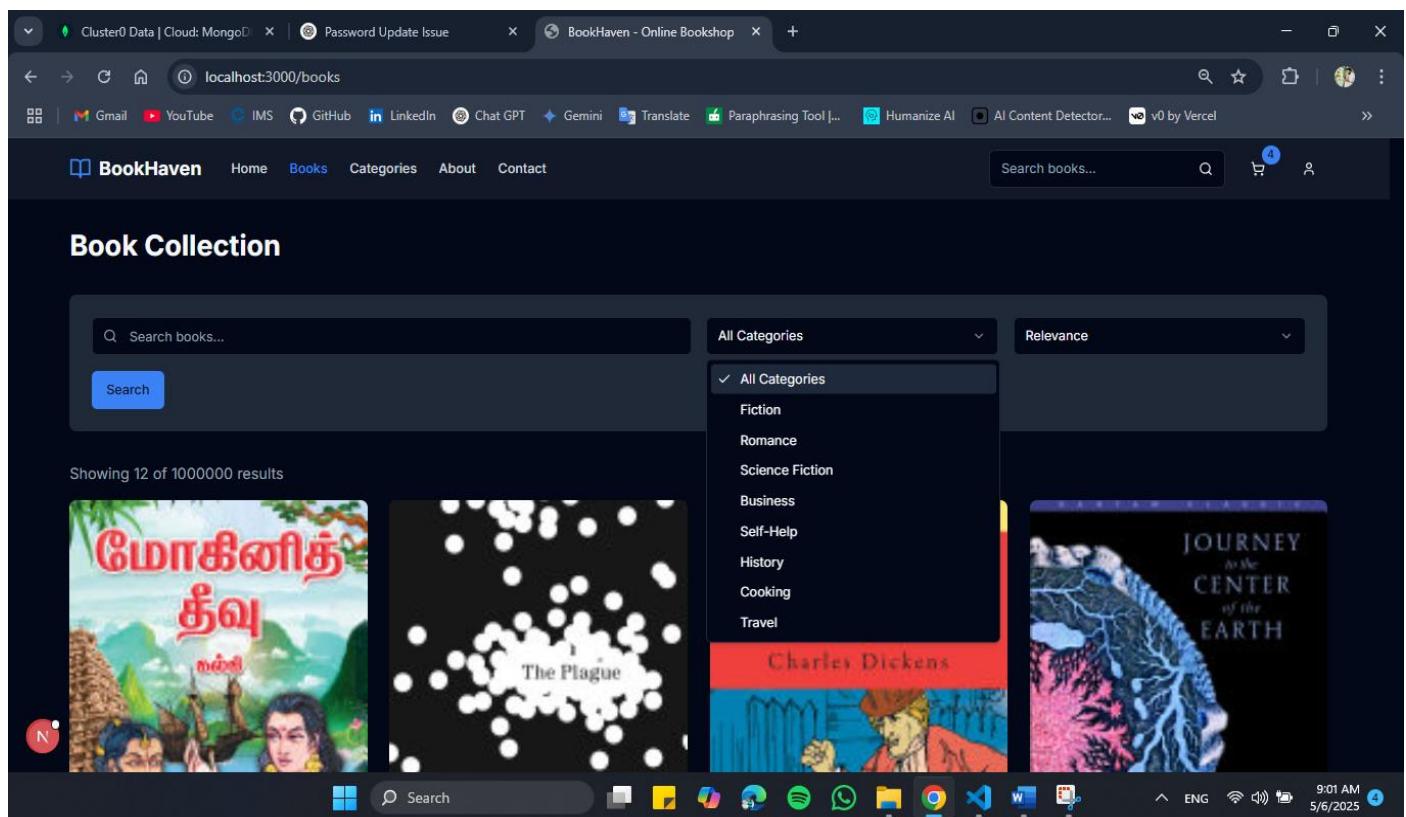
- Main book listing page screenshot



➤ Search functionality interface screenshot



➤ Filter/Sort options screenshot



Screenshot of the BookHaven - Online Bookshop homepage (localhost:3000/books). The page features a search bar, category filters (All Categories, Relevance), and sorting options (Newest, Price: Low to High, Price: High to Low). Below the filters, four book covers are displayed: "மோகினிக் தீவு" (in Tamil), "The Plague" by Albert Camus, "OLIVER TWIST" by Charles Dickens, and "JOURNEY to the CENTER of the EARTH" by Jules Verne. The interface includes a navigation bar with Home, Books, Categories, About, Contact, and a search bar.

➤ Book detail's view screenshot

Screenshot of the BookHaven - Online Bookshop product page for "Oliver Twist" by Charles Dickens (localhost:3000/books/3oSqDwAAQBAJ). The page shows the book cover, title, author, genre (Fiction / Classics), price (\$30.09), and an "Add to Cart" button. Below the main content are tabs for Description, Details, and Reviews. The Details section provides book details such as Title, Author, Publisher, Publication Date, Language, ISBN, Pages, Categories, Format, and Maturity Rating.

Book Details		Additional Information	
Title:	Oliver Twist	ISBN:	9388423038
Author:	Charles Dickens	Pages:	240
Publisher:	Jaico Publishing House	Categories:	Fiction / Classics
Publication Date:	2019-01-01	Format:	BOOK
Language:	en	Maturity Rating:	NOT_MATURE

➤ Code screenshots for API integration

```
File Edit Selection View Go Run ... route.ts ...[id] Online-bookshop
ONLINE-BOOKSHOP
  app > api > books > [id] > route.ts > GET > book
    import { NextResponse } from "next/server"
    import { getGoogleBookById } from "@/lib/google-books"

    export async function GET(request: Request, { params }: { params: { id: string } }) {
      try {
        const bookId = params.id

        if (!bookId) {
          return NextResponse.json({ error: "Book ID is required" }, { status: 400 })
        }

        const book = await getGoogleBookById(bookId)

        if (!book) {
          return NextResponse.json({ error: "Book not found" }, { status: 404 })
        }

        return NextResponse.json(book)
      } catch (error) {
        console.error("Error fetching book:", error)
        return NextResponse.json({ error: "Failed to fetch book" }, { status: 500 })
      }
    }
  
```

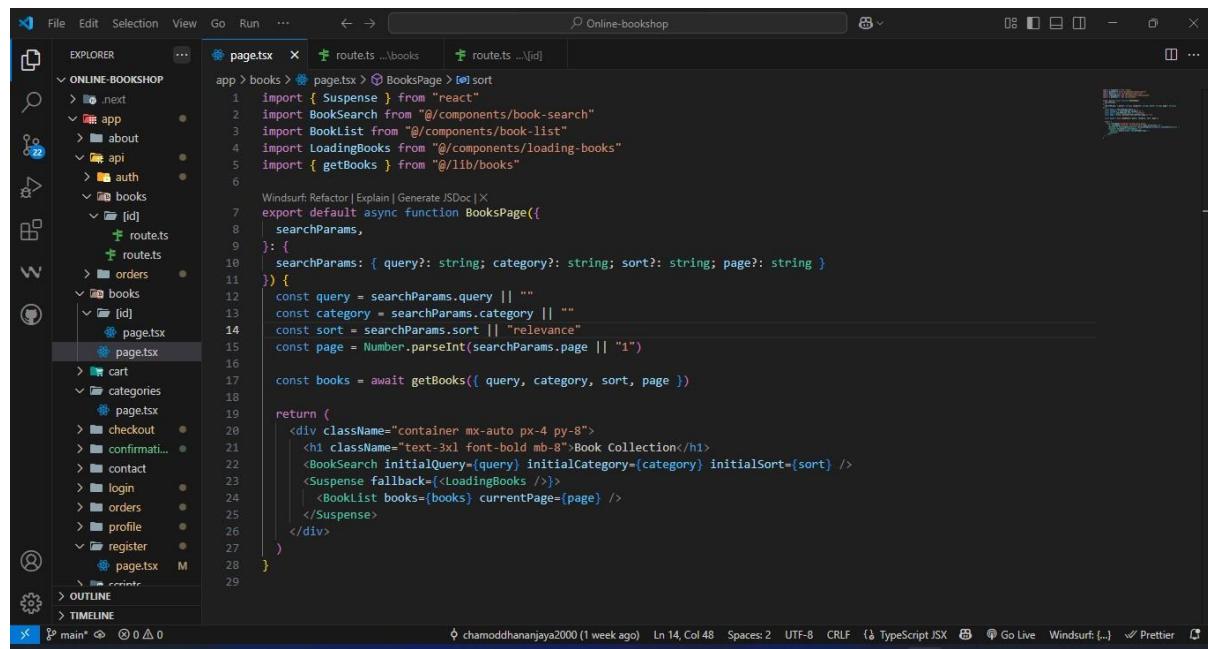
```
File Edit Selection View Go Run ... route.ts ...[id] Online-bookshop
ONLINE-BOOKSHOP
  app > api > books > route.ts > GET > books
    import { NextResponse } from "next/server"
    import { getGoogleBooks } from "@/lib/google-books"

    export async function GET(request: Request) {
      try {
        const { searchParams } = new URL(request.url)
        const query = searchParams.get("query") || ""
        const category = searchParams.get("category") || ""
        const sort = searchParams.get("sort") || "relevance"
        const page = Number.parseInt(searchParams.get("page") || "1")
        const limit = Number.parseInt(searchParams.get("limit") || "12")

        const books = await getGoogleBooks({
          query,
          category,
          sort,
          page,
          limit,
        })

        return NextResponse.json(books)
      } catch (error) {
        console.error("Error fetching books:", error)
        return NextResponse.json({ error: "Failed to fetch books" }, { status: 500 })
      }
    }
  
```

## ➤ Search implementation code screenshots



This screenshot shows the VS Code interface with the code editor open to the `BooksPage` component. The code uses the `useEffect` hook to fetch books from an API based on search parameters. It includes imports for `Suspense`, `BookSearch`, `BookList`, `LoadingBooks`, and `getBooks`. The component returns a `div` containing an `h1` and a `BookSearch` component with specific initial values.

```

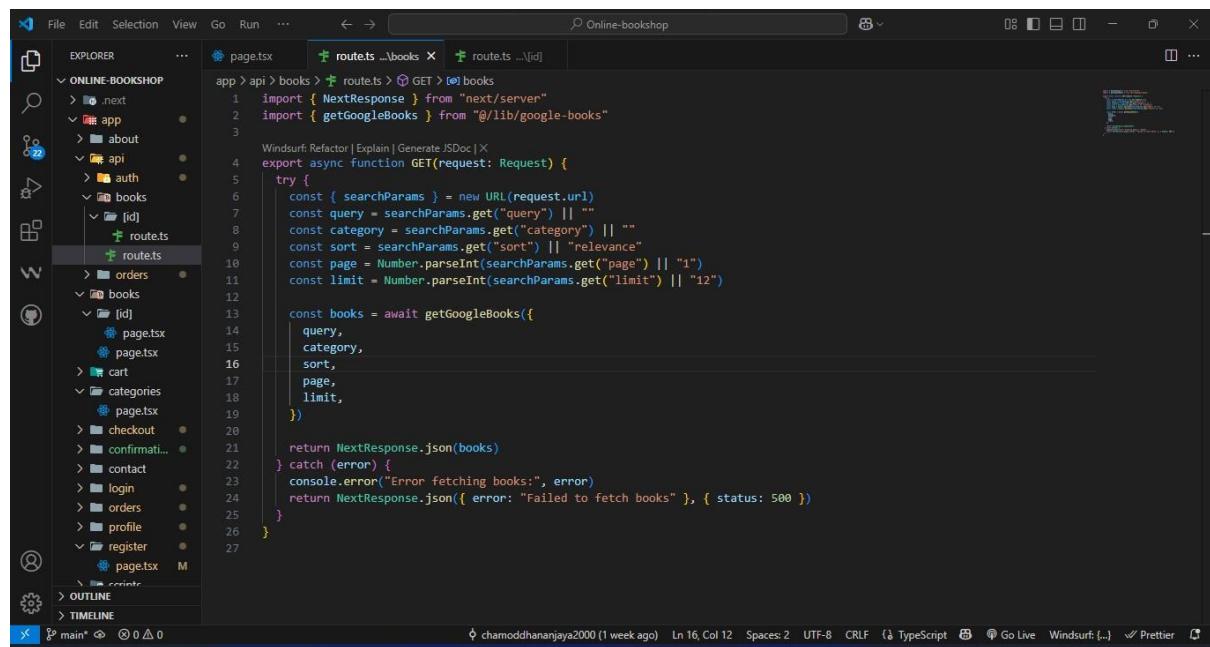
import { Suspense } from "react"
import BookSearch from "@/components/book-search"
import BookList from "@/components/book-list"
import LoadingBooks from "@/components/loading-books"
import { getBooks } from "@/lib/books"

const BooksPage = ({ searchParams }) => {
  const query = searchParams.query || ""
  const category = searchParams.category || ""
  const sort = searchParams.sort || "relevance"
  const page = Number.parseInt(searchParams.page) || 1

  const books = await getBooks({ query, category, sort, page })

  return (
    <div className="container mx-auto px-4 py-8">
      <h1 className="text-3xl font-bold mb-8">Book Collection</h1>
      <BookSearch initialQuery={query} initialCategory={category} initialSort={sort} />
      <Suspense fallback={<LoadingBooks />}>
        <BookList books={books} currentPage={page} />
      </Suspense>
    </div>
  )
}

```



This screenshot shows the VS Code interface with the code editor open to the `GET` handler for the `books` API endpoint. The handler uses `getGoogleBooks` to fetch books and returns them as a JSON response. It handles errors by returning a 500 status code.

```

import { NextResponse } from "next/server"
import { getGoogleBooks } from "@/lib/google-books"

export async function GET(request: Request) {
  try {
    const { searchParams } = new URL(request.url)
    const query = searchParams.get("query") || ""
    const category = searchParams.get("category") || ""
    const sort = searchParams.get("sort") || "relevance"
    const page = Number.parseInt(searchParams.get("page")) || 1
    const limit = Number.parseInt(searchParams.get("limit")) || 12

    const books = await getGoogleBooks({
      query,
      category,
      sort,
      page,
      limit,
    })

    return NextResponse.json(books)
  } catch (error) {
    console.error("Error fetching books:", error)
    return NextResponse.json({ error: "Failed to fetch books" }, { status: 500 })
  }
}

```

## 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
<b>Total</b>	<b>\$47.25</b>

[Proceed to Checkout](#)

➤ Add to cart functionality screenshot

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

Image	Title	Author	Price	Action
	The Sherlock Holmes Book	DK	\$15.88	<a href="#">Add to Cart</a>
	The Best of Sherlock Holmes	Arthur Conan Doyle	\$21.11	<a href="#">Add to Cart</a>
	Adventures of Sherlock Holmes	Arthur Conan Doyle	\$9.13	<a href="#">Add to Cart</a>
	The Case-Book of Sherlock...	Arthur Conan Doyle	\$25.65	<a href="#">Add to Cart</a>

➤ Update quantity interface screenshot

The screenshot shows a dark-themed web browser window with several tabs open at the top. The active tab is 'BookHaven - Online Bookshop'. The main content area displays a shopping cart with four items:

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

To the right of the cart, there is an 'Order Summary' box:

Subtotal	\$63.85
Tax (8%)	\$5.11
Shipping	\$5.99
<b>Total</b>	<b>\$74.95</b>

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

➤ Remove items interface screenshot

The screenshot shows a dark-themed web browser window with several tabs open at the top. The active tab is 'BookHaven - Online Bookshop'. The main content area displays a shopping cart with two items:

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

To the right of the cart, there is an 'Order Summary' box:

Subtotal	\$31.37
Tax (8%)	\$2.51
Shipping	\$5.99
<b>Total</b>	<b>\$39.87</b>

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

➤ Cart total calculation screenshot

The screenshot shows the BookHaven Online Bookshop cart page. The cart contains two items:

Book	Quantity	Price	Actions
Oliver Twist Charles Dickens	1	\$5.08	
Journey to the Center of the Earth Jules Verne	3	\$21.21	

**Order Summary**

Subtotal	\$26.29
Tax (8%)	\$2.10
Shipping	\$5.99
<b>Total</b>	<b>\$34.38</b>

[Proceed to Checkout](#)

At the bottom, there are links for BookHaven (with 1 issue), Quick Links (Home, FAQ), Customer Service, and Contact Us (123 Book Street).

➤ Related code screenshots for cart management

The screenshot shows the code for the `CartPage` component in a code editor. The code handles cart operations like adding items, calculating totals, and managing checkout.

```
use client"
import { useCart } from "@/context/cart-context"
import { useAuth } from "@/context/auth-context"
import { useRouter } from "next/navigation"
import Image from "next/image"
import { Button } from "@components/ui/button"
import { Trash2 } from "lucide-react"
import Link from "next/link"
import { formatPrice } from "@/lib/utils"

export default function CartPage() {
  const { cart, removeFromCart, updateQuantity, clearCart } = useCart()
  const { user } = useAuth()
  const router = useRouter()

  const subtotal = cart.reduce((total, item) => total + item.price * item.quantity, 0)
  const tax = subtotal * 0.08 // 8% tax
  const shipping = subtotal > 0 ? 5.99 : 0
  const total = subtotal + tax + shipping

  const handleCheckout = () => {
    if (user) {
      router.push("/login?redirect=/checkout")
    } else {
      router.push("/checkout")
    }
  }

  if (cart.length === 0) {
```





## 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:

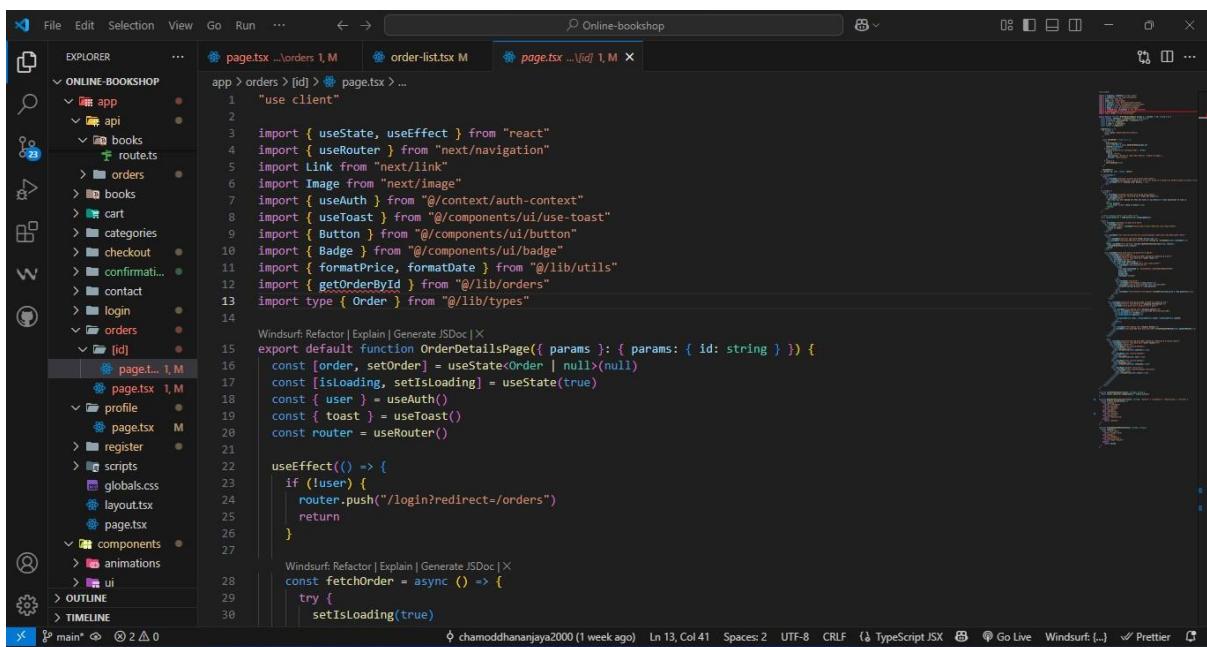
Order Summary	
Oliver Twist	\$5.08
Qty: 1	
Journey to the Center of the Earth	\$21.21
Qty: 3	
Subtotal	\$26.29
Tax (8%)	\$2.10
Shipping	\$5.99
Total	\$34.38

- 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

➤ Implementation code screenshots



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 'orders' folder contains files like 'index.tsx', 'page.tsx', and 'route.tsx'. The right pane displays the content of a file named 'page.tsx' under the 'orders' folder. The code is written in TypeScript and React, handling order details and user authentication. A vertical scrollbar on the right side of the code editor indicates a large amount of 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
15  Windsurf: Refactor | Explain | Generate JSDoc | X
16  export default function OrderDetailsPage({ params }: { params: { id: string } }) {
17    const [order, setOrder] = useState(null)
18    const [isLoading, setIsLoading] = useState(true)
19    const { user } = useAuth()
20    const { toast } = useToast()
21    const router = useRouter()
22
23    useEffect(() => {
24      if (!user) {
25        router.push("/login?redirect=/orders")
26        return
27      }
28
29      const fetchOrder = async () => {
30        try {
31          setIsLoading(true)
```

The image shows two side-by-side code editors, likely from VS Code, displaying TypeScript code for an "ONLINE-BOOKSHOP" application.

**Top Editor:**

```

File Edit Selection View Go Run ... ⏪ ⏹ Online-bookshop
EXPLORER page.tsx ...|orders 1, M | order-list.tsx M page.tsx ...|/id 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>Order Not Found</h1>
60       <p>The order you are looking for does not exist or you don't have permission to view it.</p>
61       <Button asChild>
62         <Link href="/orders">Back to Orders</Link>
63       </Button>
64     </div>
65   )
66 }
67
68 // Parse shipping address from JSON string
69 const shippingAddress = JSON.parse(order.shippingAddress)
70
71 return (
72   <div className="container mx-auto px-4 py-8">
73     <div className="mb-6">
74       <Link href="/orders" className="text-primary hover:underline flex items-center">
75         <span>Back to Orders</span>
76       </Link>
77     </div>
78
79     <div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-6">
80       <div>
81         <h1>Order #<strong>{order.id}</strong></h1>
82         <p>Placed on <small>formatDate(order.createdAt)</small></p>
83       </div>
84     </div>
85   </div>
86 )

```

**Bottom Editor:**

```

File Edit Selection View Go Run ... ⏪ ⏹ Online-bookshop
EXPLORER page.tsx ...|orders 1, M | order-list.tsx M page.tsx ...|/id 1, M
app > orders > [id] > page.tsx > ...
15  export default function OrderDetailsPage({ params }: { params: { id: string } }) {
56
57 if (!order) {
58   return (
59     <div className="container mx-auto px-4 py-16 text-center">
60       <h1>Order Not Found</h1>
61       <p>The order you are looking for does not exist or you don't have permission to view it.</p>
62       <Button asChild>
63         <Link href="/orders">Back to Orders</Link>
64       </Button>
65     </div>
66   )
67 }
68
69 // Parse shipping address from JSON string
70 const shippingAddress = JSON.parse(order.shippingAddress)
71
72 return (
73   <div className="container mx-auto px-4 py-8">
74     <div className="mb-6">
75       <Link href="/orders" className="text-primary hover:underline flex items-center">
76         <span>Back to Orders</span>
77       </Link>
78     </div>
79
80     <div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-6">
81       <div>
82         <h1>Order #<strong>{order.id}</strong></h1>
83         <p>Placed on <small>formatDate(order.createdAt)</small></p>
84       </div>
85     </div>
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)} x {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                         <div>
132                             <h3 className="font-medium mb-2">Payment Method</h3>
133                             <p className="text-gray-600 dark:text-gray-400">{formatPaymentMethod(order.paymentMethod)}</p>
134                         </div>
135                     </div>
136
137                     <div>
138                         <h2 className="text-xl font-bold mb-4">Order Summary</h2>
139                         <div className="space-y-2">
140                             <div>
141                                 <h3>Subtotal</h3>
142                                 <p>{formatPrice(order.total)}</p>
143                             </div>
144                         </div>
145                     </div>
146                 </div>
147             </div>
148         </div>
149     </div>
150     <div>
151         <h2>Order Placed!</h2>
152         <p>Your order has been successfully placed. You will receive an email confirmation shortly.</p>
153     </div>
154 
```
- 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 } }) {
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)} x {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 
```
- components
  - animations
  - ui
- OUTLINE
- TIMELINE

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 ...

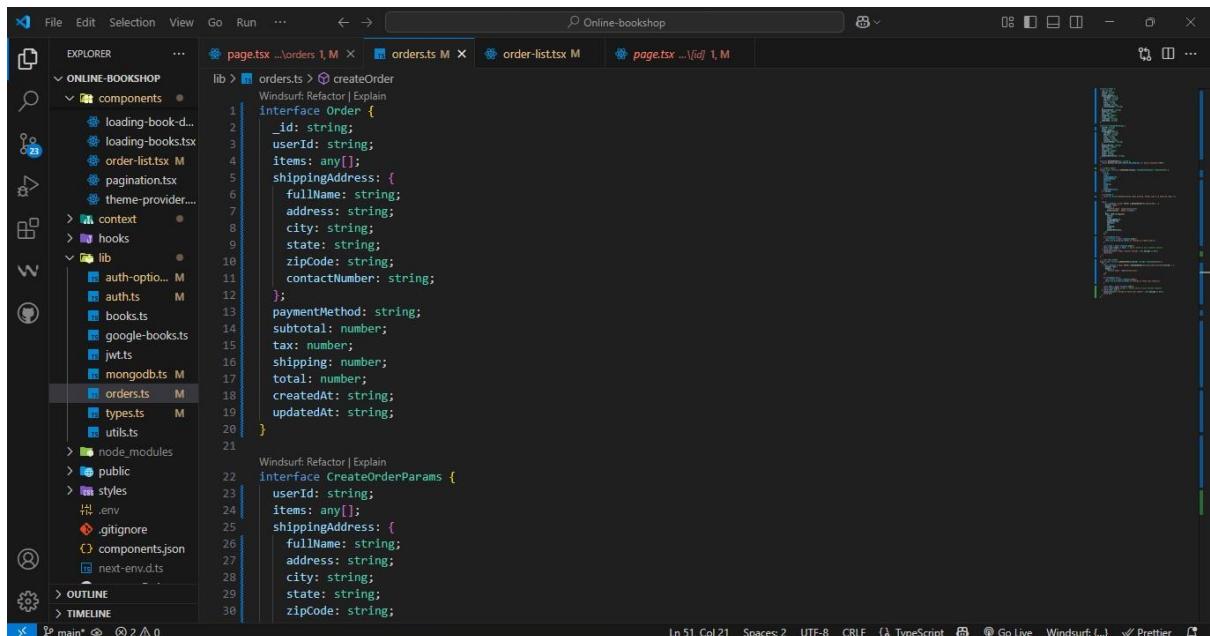
ONLINE-BOOKSHOP

- 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 order = params.id ? await prisma.order.findUnique({ where: { id: params.id } }) : null
  17   if (!order) return null
  18
  19   const subtotal = calculateSubtotal(order.items)
  20   const tax = calculateTax(subtotal)
  21   const shipping = calculateShipping()
  22
  23   const total = subtotal + tax + shipping
  24
  25   const formattedOrder = formatOrder(order)
  26   const formattedAddress = formatAddress(formattedOrder.address)
  27
  28   const formattedOrderStatus = formatOrderStatus(formattedOrder.status)
  29
  30   const formattedPaymentMethod = formatPaymentMethod(formattedOrder.paymentMethod)
  31
  32   const formattedDeliveryTime = formatDeliveryTime(formattedOrder.deliveryTime)
  33
  34   const formattedDeliveryAddress = formatDeliveryAddress(formattedOrder.deliveryAddress)
  35
  36   const formattedDeliveryInstructions = formatDeliveryInstructions(formattedOrder.deliveryInstructions)
  37
  38   const formattedDeliveryNotes = formatDeliveryNotes(formattedOrder.deliveryNotes)
  39
  40   const formattedDeliveryComments = formatDeliveryComments(formattedOrder.deliveryComments)
  41
  42   const formattedDeliverySignature = formatDeliverySignature(formattedOrder.deliverySignature)
  43
  44   const formattedDeliverySignatureImage = formatDeliverySignatureImage(formattedOrder.deliverySignatureImage)
  45
  46   const formattedDeliverySignatureImageBase64 = formatDeliverySignatureImageBase64(formattedOrder.deliverySignatureImage)
  47
  48   const formattedDeliverySignatureImageUrl = formatDeliverySignatureImageUrl(formattedOrder.deliverySignatureImage)
  49
  50   const formattedDeliverySignatureImageWidth = formatDeliverySignatureImageWidth(formattedOrder.deliverySignatureImage)
  51
  52   const formattedDeliverySignatureImageHeight = formatDeliverySignatureImageHeight(formattedOrder.deliverySignatureImage)
  53
  54   const formattedDeliverySignatureImageMime = formatDeliverySignatureImageMime(formattedOrder.deliverySignatureImage)
  55
  56   const formattedDeliverySignatureImageType = formatDeliverySignatureImageType(formattedOrder.deliverySignatureImage)
  57
  58   const formattedDeliverySignatureImageTypeMime = formatDeliverySignatureImageTypeMime(formattedOrder.deliverySignatureImage)
  59
  60   const formattedDeliverySignatureImageTypeMimeBase64 = formatDeliverySignatureImageTypeMimeBase64(formattedOrder.deliverySignatureImage)
  61
  62   const formattedDeliverySignatureImageTypeMimeWidth = formatDeliverySignatureImageTypeMimeWidth(formattedOrder.deliverySignatureImage)
  63
  64   const formattedDeliverySignatureImageTypeMimeHeight = formatDeliverySignatureImageTypeMimeHeight(formattedOrder.deliverySignatureImage)
  65
  66   const formattedDeliverySignatureImageTypeMimeMime = formatDeliverySignatureImageTypeMimeMime(formattedOrder.deliverySignatureImage)
  67
  68   const formattedDeliverySignatureImageTypeMimeType = formatDeliverySignatureImageTypeMimeType(formattedOrder.deliverySignatureImage)
  69
  70   const formattedDeliverySignatureImageTypeMimeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeBase64(formattedOrder.deliverySignatureImage)
  71
  72   const formattedDeliverySignatureImageTypeMimeTypeWidth = formatDeliverySignatureImageTypeMimeTypeWidth(formattedOrder.deliverySignatureImage)
  73
  74   const formattedDeliverySignatureImageTypeMimeTypeHeight = formatDeliverySignatureImageTypeMimeTypeHeight(formattedOrder.deliverySignatureImage)
  75
  76   const formattedDeliverySignatureImageTypeMimeTypeMime = formatDeliverySignatureImageTypeMimeTypeMime(formattedOrder.deliverySignatureImage)
  77
  78   const formattedDeliverySignatureImageTypeMimeTypeType = formatDeliverySignatureImageTypeMimeTypeType(formattedOrder.deliverySignatureImage)
  79
  80   const formattedDeliverySignatureImageTypeMimeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  81
  82   const formattedDeliverySignatureImageTypeMimeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  83
  84   const formattedDeliverySignatureImageTypeMimeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  85
  86   const formattedDeliverySignatureImageTypeMimeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeMime(formattedOrder.deliverySignatureImage)
  87
  88   const formattedDeliverySignatureImageTypeMimeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeType(formattedOrder.deliverySignatureImage)
  89
  90   const formattedDeliverySignatureImageTypeMimeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  91
  92   const formattedDeliverySignatureImageTypeMimeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  93
  94   const formattedDeliverySignatureImageTypeMimeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  95
  96   const formattedDeliverySignatureImageTypeMimeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  97
  98   const formattedDeliverySignatureImageTypeMimeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  99
  100  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  101
  102  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  103
  104  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  105
  106  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  107
  108  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  109
  110  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  111
  112  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  113
  114  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  115
  116  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  117
  118  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  119
  120  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  121
  122  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  123
  124  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  125
  126  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  127
  128  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  129
  130  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  131
  132  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  133
  134  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  135
  136  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  137
  138  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  139
  140  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  141
  142  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  143
  144  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  145
  146  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  147
  148  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  149
  150  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  151
  152  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  153
  154  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  155
  156  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  157
  158  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  159
  160  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  161
  162  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeWidth = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeWidth(formattedOrder.deliverySignatureImage)
  163
  164  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeHeight = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeHeight(formattedOrder.deliverySignatureImage)
  165
  166  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeMime = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeMime(formattedOrder.deliverySignatureImage)
  167
  168  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeType = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeType(formattedOrder.deliverySignatureImage)
  169
  170  const formattedDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64 = formatDeliverySignatureImageTypeMimeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeTypeBase64(formattedOrder.deliverySignatureImage)
  171  function formatOrderStatus(status: string): string {
  172    return status.charAt(0).toUpperCase() + status.slice(1)
  173  }
  174
  175  Windsurf: Refactor | Explain | Generate JSDoc | X
  176  function getOrderStatusVariant(status: string): "default" | "secondary" | "destructive" | "outline" {
  177    switch (status.toLowerCase()) {
  178      case "pending":
  179        return "secondary"
  180      case "processing":
  181        return "default"
  182      case "shipped":
  183        return "outline"
  184      case "delivered":
  185        return "destructive"
  186      case "cancelled":
  187        return "outline"
  188      default:
  189        return "default"
  190    }
  191  Windsurf: Refactor | Explain | Generate JSDoc | X
  192  function formatPaymentMethod(method: string): string {
  193    switch (method) {
  194      case "credit-card":
  195        return "Credit Card"
  196      case "paypal":
  197        return "PayPal"
  198      case "bank-transfer":
  199        return "Bank Transfer"
  200      default:
  201        return method
  202    }
  203  
```

➤ Order processing code screenshots



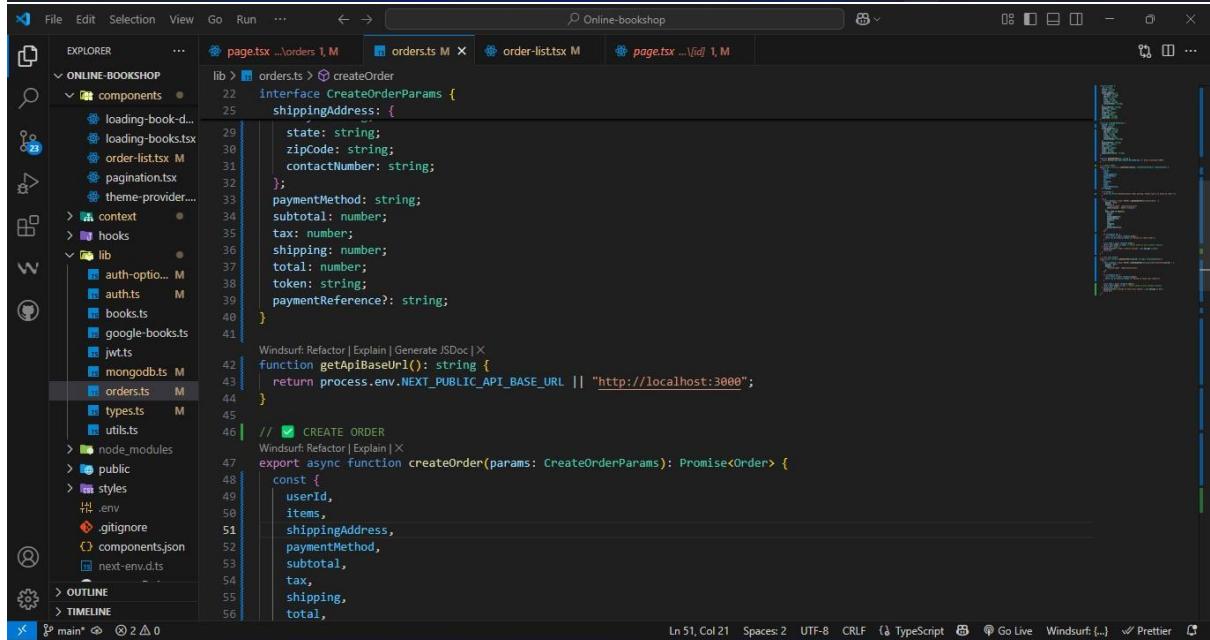
The screenshot shows the VS Code interface with the 'orders.ts' file open in the center editor tab. The file contains TypeScript code defining an 'Order' interface and a 'createOrder' function. The 'orders.ts' file is part of a library named 'lib'. The code includes imports for 'Windsurf' and 'Prettier', and exports the 'createOrder' function.

```
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;
  };
  paymentMethod: string;
  subtotal: number;
  tax: number;
  shipping: number;
  total: number;
  token: string;
  paymentReference?: 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;
  const response = await fetch(`${getApiBaseUrl()}/orders`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      userId,
      items,
      shippingAddress,
      paymentMethod,
      subtotal,
      tax,
      shipping,
      total,
    }),
  });
  const data = await response.json();
  return data;
}
```



The second screenshot shows the same 'orders.ts' file in VS Code, but with code completion suggestions visible in the right margin. The cursor is positioned at the end of the 'params' object in the 'createOrder' function. Suggested properties include '\_id', 'userId', 'items', 'shippingAddress', 'paymentMethod', 'subtotal', 'tax', 'shipping', 'total', 'token', and 'paymentReference'. The 'Windsurf' and 'Prettier' imports at the top of the file are also visible.

The screenshot shows two instances of Visual Studio Code side-by-side, both displaying code for an 'Online-bookshop' project.

**Top Window (orders.ts):**

```

lib > orders.ts > createOrder
47 export async function createOrder(params: CreateOrderParams): Promise<Order> {
55     shipping,
56     total,
57     token,
58     paymentReference,
59 } = params;
60
61 if (!token) {
62     throw new Error("Authentication token missing. Please log in to place an order.");
63 }
64
try {
const response = await fetch(`${getApiBaseUrl()}/api/orders`,
    method: "POST",
    headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify({
        userId,
        items,
        shippingAddress,
        paymentMethod,
        subtotal,
        tax,
        shipping,
        total,
        paymentReference,
    }),
);
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

```

**Bottom Window (page.tsx):**

```

lib > orders.ts > createOrder
47 export async function createOrder(params: CreateOrderParams): Promise<Order> {
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

```

Both windows show the same code content, which is a TypeScript function named 'createOrder' that handles creating an order by sending a POST request to the backend API. The code includes validation for the authentication token and handling of the response.

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

- File Explorer:** Shows the project structure under "ONLINE-BOOKSHOP". The "lib" folder is expanded, showing files like auth-option.ts, auth.ts, books.ts, google-books.ts, jwt.ts, mongodb.ts, orders.ts (which is currently selected), types.ts, and utils.ts.
- Editor:** The main editor area displays the "orders.ts" file. The code is as follows:

```
lib > orders.ts > createOrder
99  export async function getUserOrders(userId: string): Promise<Order[]> {
100    ...
111  )
112  const data = await response.json();
113  return data.orders || [];
114  } catch (err: any) {
115    console.error(`Failed to fetch user orders: ${err.message || err}`);
116    throw err;
117  }
118}
119}
```

- Bottom Status Bar:** Shows the current file path as "orders.ts", line 51, column 21, and other status indicators like "Spaces 2", "CRLF", "TypeScript", "Go Live", "Windsurf", and "Prettier".

## 1.5. User Profile

➤ Profile view screenshot

This screenshot shows the BookHaven user profile page at [localhost:3000/profile](http://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...". On the right, a user menu is open for "Chamod Dhananjaya", showing options for Profile, My Orders, and Logout. A notification badge with the number 4 is visible above the menu. Below the navigation, there are tabs for "Profile" (which is active) and "Order History". 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 provides a "New Password" link.

➤ Edit profile interface screenshot

This screenshot shows the BookHaven user profile edit page at [localhost:3000/profile](http://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 "Full Name" and "Email" fields are identical to the view page. The "Change Password" section is expanded, showing "Current Password" (2000chamod), "New Password" (2001chamod), and "Confirm New Password" fields. A blue "Update Profile" button is located at the bottom. The "2 Issues" notification from the previous screenshot is still present at the bottom left.

## ➤ Order history view screenshot

The screenshot shows the 'My Orders' page of the BookHaven Online Bookshop. The main content area displays a message: 'You haven't placed any orders yet' with a sub-message: '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 links), 'Quick Links' (Home, Books, Categories, About Us, Contact), 'Customer Service' (FAQ, Shipping Policy, Returns & Refunds, Privacy Policy, Terms & Conditions), and 'Contact Us' (Address: 123 Book Street, Reading City, RC 12345, United States, Email: info@bookhaven.com).

## ➤ Related implementation code screenshots

The screenshot shows the implementation of the `OrderList` component in the `page.tsx` file. The code uses TypeScript and Next.js components like `Link` and `Badge`. The component maps over an array of orders to render each one with its ID, a badge indicating the order status, and a link to view details. The code editor interface shows the project structure with components like `loading-book.tsx`, `loading-books.tsx`, and `order-list.tsx`.

```
1 import Link from "next/link"
2 import { formatPrice, formatDate } from "@/lib/utils"
3 import { Badge } from "@components/ui/badge"
4 import { Button } from "@components/ui/button"
5 import type { Order } from "@lib/types"
6
7 Windsurf: Refactor | Explain | Generate JSDoc | X
8 export default function OrderList({ orders }: { orders: Order[] }) {
9   return (
10     <div className="space-y-6">
11       {orders.map((order) => (
12         <div key={order.id} className="border rounded-lg overflow-hidden">
13           <div className="bg-gray-50 dark:bg-gray-800 p-4 flex flex-col sm:flex-row justify-between items-start sm:items-center gap-2">
14             <h3 className="font-medium">Order #{order.id}</h3>
15             <Badge variant={getOrderStatusVariant(order.status)}>{formatOrderStatus(order.status)}</Badge>
16           </div>
17           <p className="text-sm text-gray-500 dark:text-gray-400">Placed on {formatDate(order.createdAt)}</p>
18         </div>
19         <div className="flex items-center gap-2">
20           <span className="font-medium">{formatPrice(order.total)}</span>
21           <Button asChild variant="outline" size="sm">
22             <Link href={`/orders/${order.id}`}>View Details</Link>
23           </Button>
24         </div>
25       </div>
26     )
27   )
28 }
29
30 <div className="p-4">
31   <div className="space-y-2">
32     {order.orderItems.slice(0, 3).map((item) => (
33       <div key={item.id} className="flex justify-between items-center">
34         <span>
```

```
components > order-listtsx > ...
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
```

```
Windsurf: Refactor | Explain | Generate JSDoc | X
function formatOrderStatus(status: string): string {
  return status.charAt(0).toUpperCase() + status.slice(1)
}
```

```
components > order-listtsx > ...
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}
```