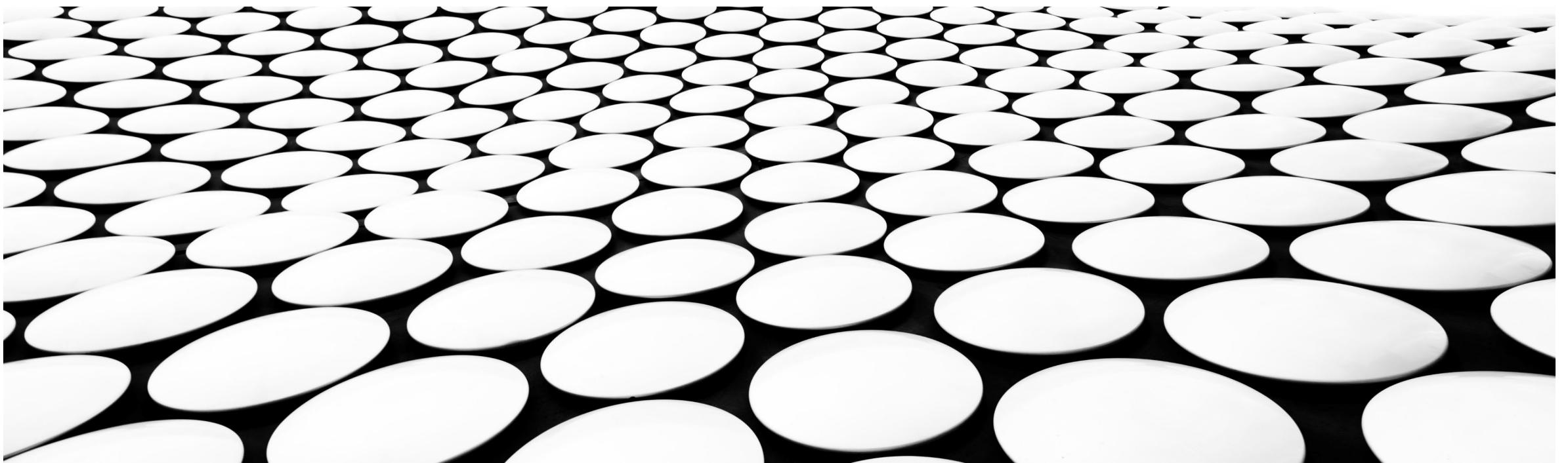




S16 SHELF PAGE



1. REACT- CREATE SHELF PAGE



There are two modules in this page.

On the left, is the image of current borrowed books.

On the right , is the options to manage the borrowed books.

Home Search Books Shelf

Loans Your History

Current Loans:



Crash Course in
Big Data

Judy Luv

Loan Options

Due in 1 days.

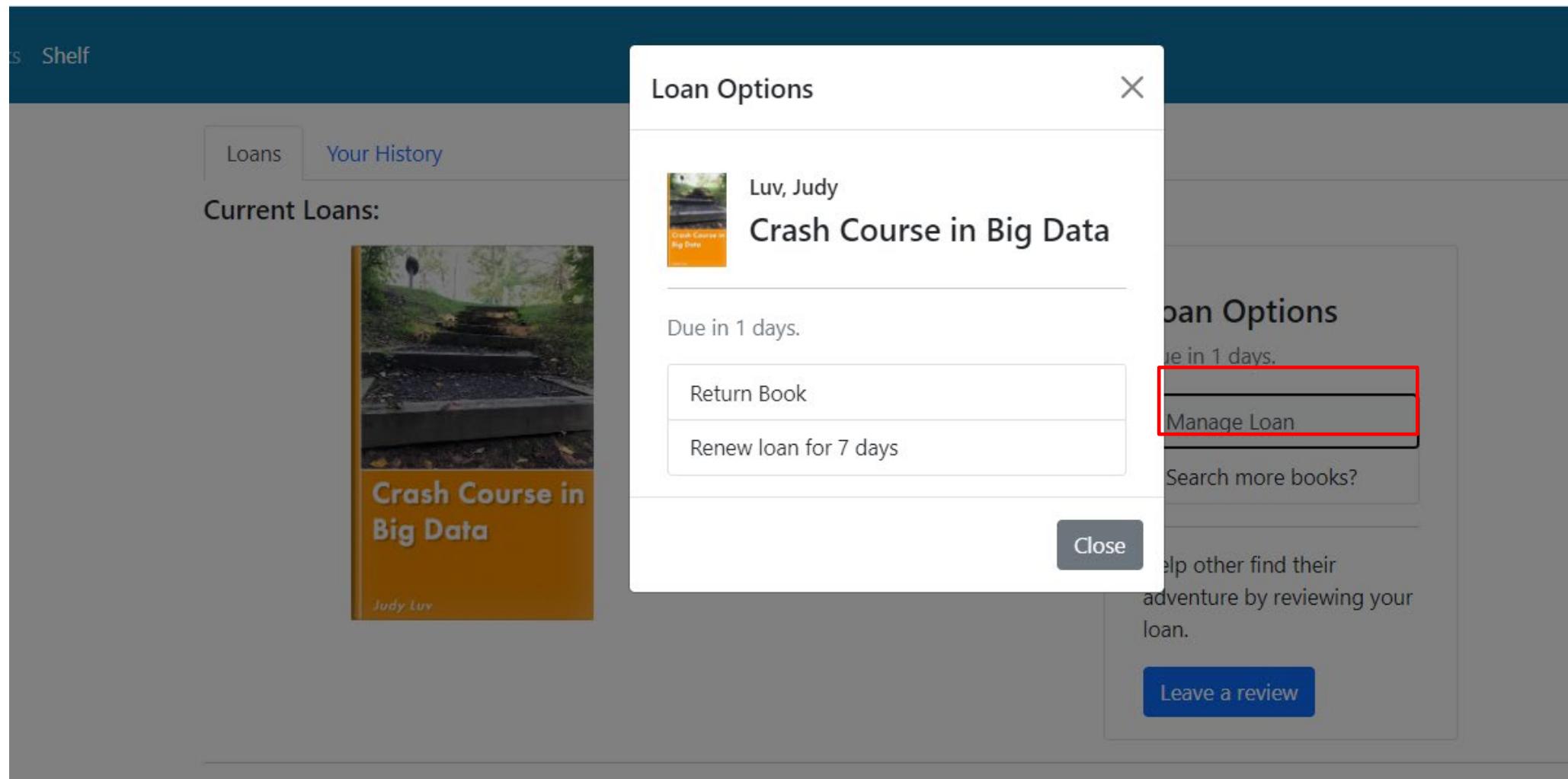
Manage Loan

Search more books?

Help other find their adventure by reviewing your loan.

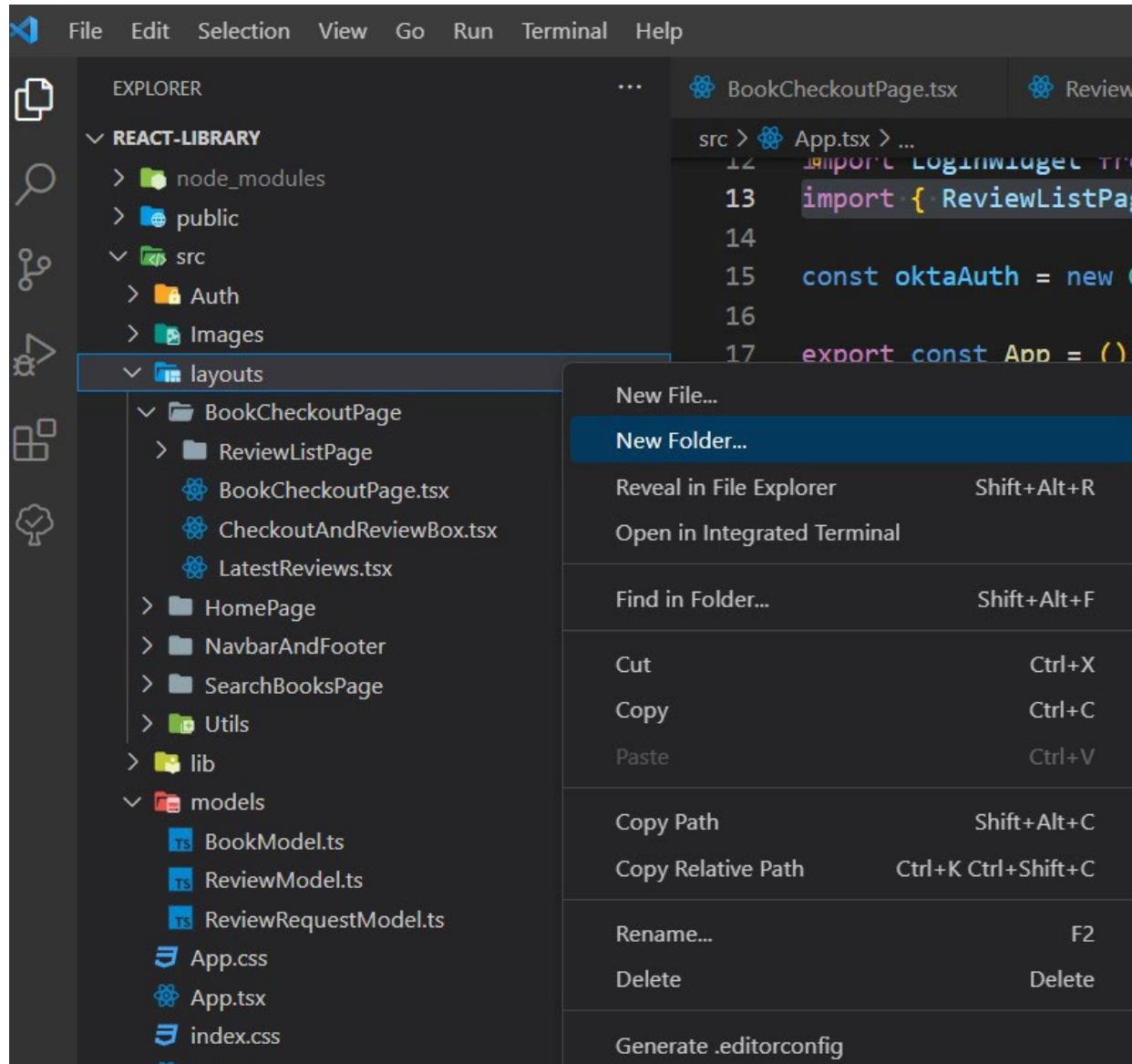
Leave a review

Clicking on “Manage loan”, a dialogue box will pop out, providing the choice of “Return book” and “renew for 7 days”.

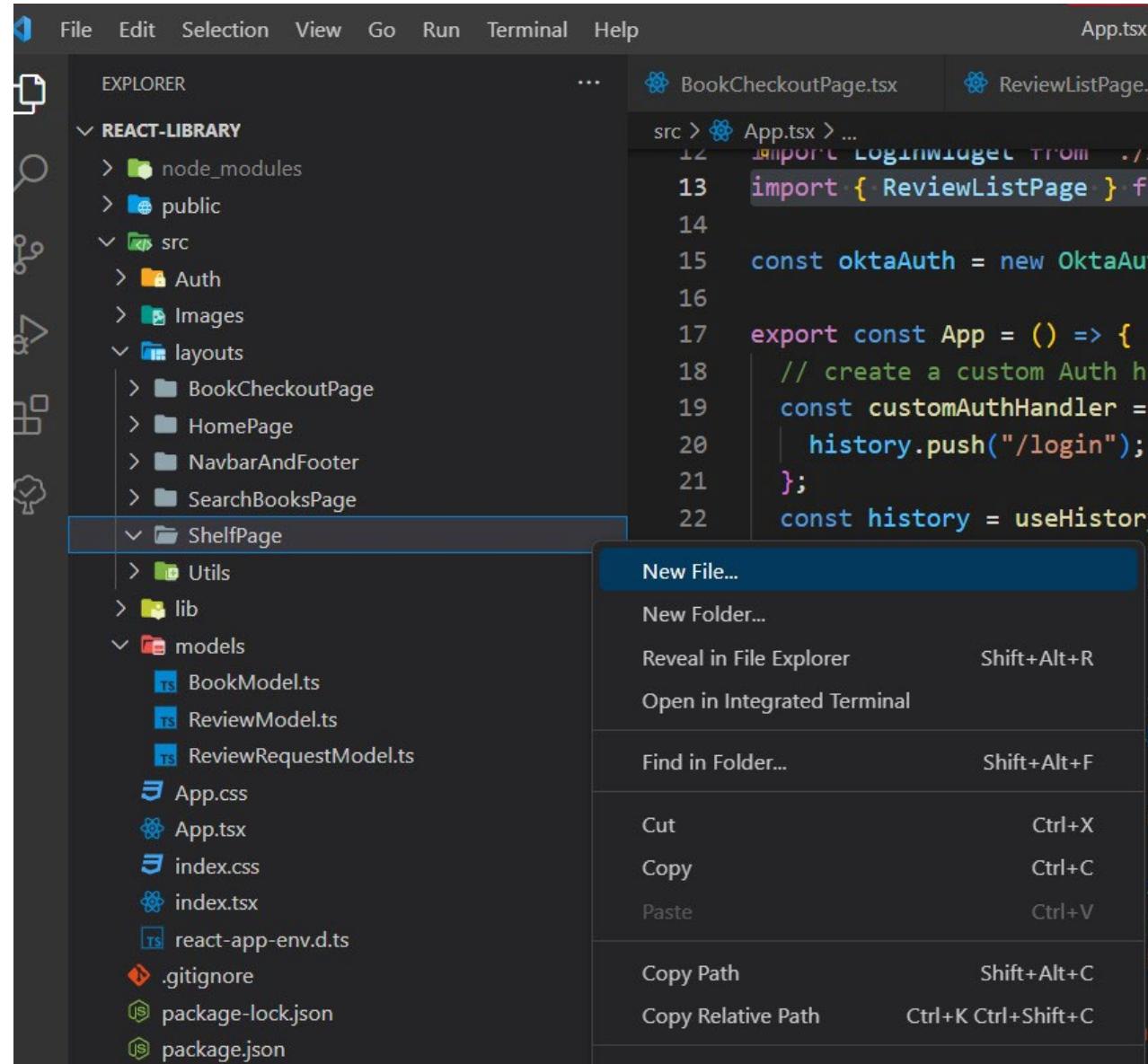


But if you missed the due day, you can only return the book.

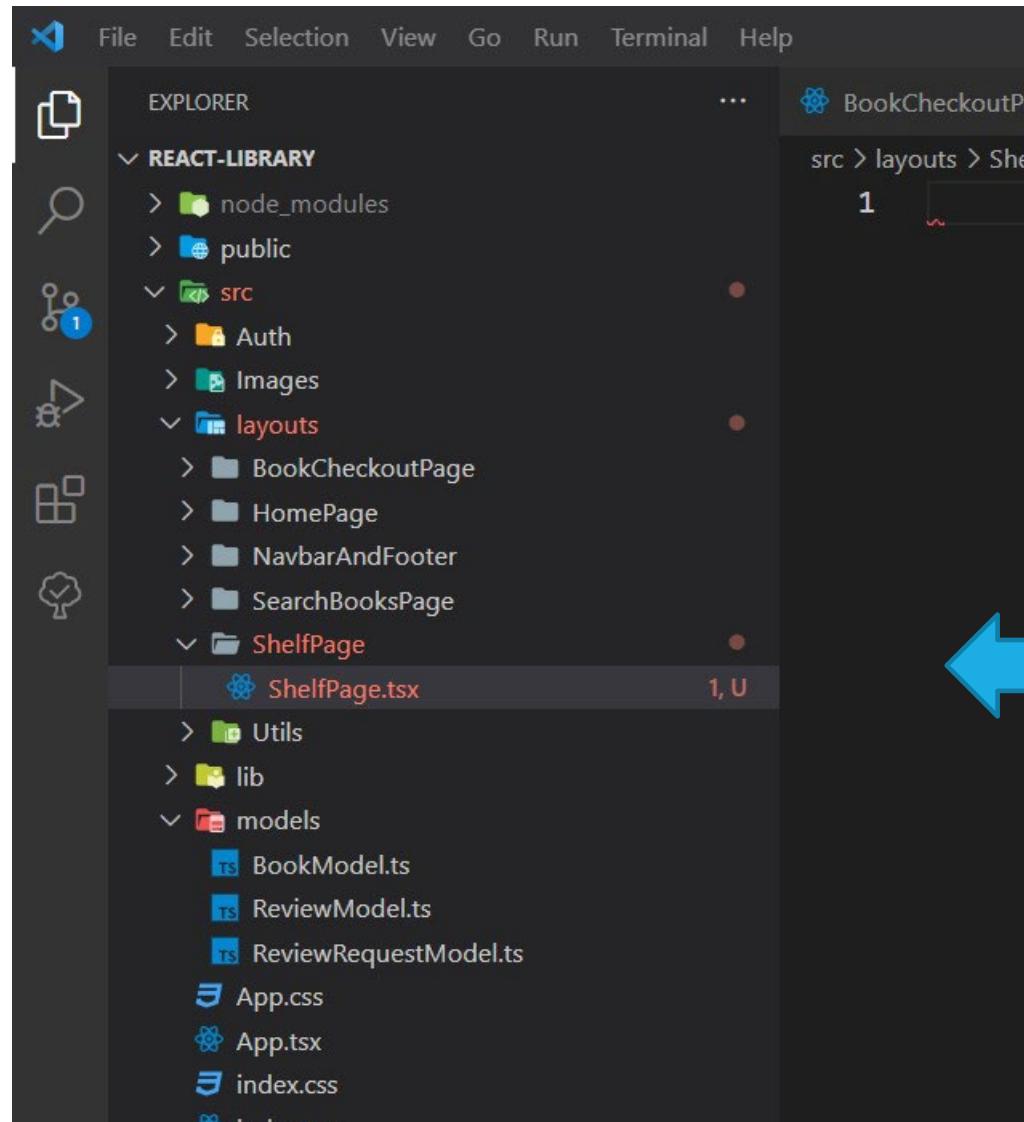
The screenshot shows a library management system interface. At the top, there's a navigation bar with 'Home', 'Search Books', and 'Shelf' buttons. Below this, a 'Loans' tab is selected, and a 'Your History' link is visible. A 'Current Loans:' section lists books. One book, 'Become a Guru in JavaScript' by Lena Luv, is highlighted with a thumbnail showing a brick archway and the book cover. The main content area displays a 'Loan Options' dialog for this book. The dialog title is 'Loan Options' with a close button 'X'. It shows the book cover, the borrower 'Luv, Lena', and the title 'Become a Guru in JavaScript'. It also indicates the book is 'Past due by -19 days.' Below this, there are two buttons: 'Return Book' and 'Late dues cannot be renewed'. At the bottom of the dialog are 'Close' and 'Leave a review' buttons. In the background, another 'Loan Options' dialog is partially visible for a book due in 7 days, with buttons for 'Manage Loan' and 'Search more books?'. The overall theme is a dark blue header and light gray body.



Step 1 Right click on layouts folder and choose “new folder”, name it “shelfPage”.



Step 2 Right click on the ShelfPage folder and choose “new file”



name it ShelfPage.tsx.

Step 3, as usual , create export cons ShelfPage structure.

The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar (EXPLORER) displays a project structure under 'REACT-LIBRARY'. The 'src' folder contains 'Auth', 'Images', and 'layouts'. The 'layouts' folder contains 'BookCheckoutPage', 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', and 'ShelfPage'. The 'ShelfPage' folder is currently selected, and its file 'ShelfPage.tsx' is open in the main editor area. The code in the editor is:

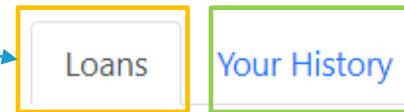
```
src > layouts > ShelfPage > ShelfPage.tsx > ShelfPage
1  export const ShelfPage = () =>{
2      return(
3          ...
4      );
5  }
6 }
```

The status bar at the bottom indicates the file is '2, U' (2 changes, unstaged).

```
:> layouts > ShelfPage >  ShelfPage.tsx > ...  
1  export const ShelfPage = () => {  
2    return (  
3      <div className="container">  
4        <div className="mt-3">  
5          <nav>  
6            <div className="nav nav-tabs" id="nav-tab" role="tablist">  
7              <button  
8                className="nav-link active"  
9                  id="nav-loans-tab"  
10                 data-bs-toggle="tab"  
11                 data-bs-target="#nav-loans"  
12                 type="button"  
13                 role="tab"  
14                 aria-controls="nav-loans"  
15                 aria-selected="true"  
16               >  
17               Loans  
18             </button>  
19              <button  
20                className="nav-link"  
21                  id="nav-history-tab"  
22                  data-bs-toggle="tab"  
23                  data-bs-target="#nav-history"  
24                  type="button"  
25                  role="tab"  
26                  aria-controls="nav-history"  
27                  aria-selected="false"  
28                >  
29                Your History
```

Step 4, fill in the return part with HTML/CSS code.

4.1 create two tab buttons.



4.2 create two tab contents.

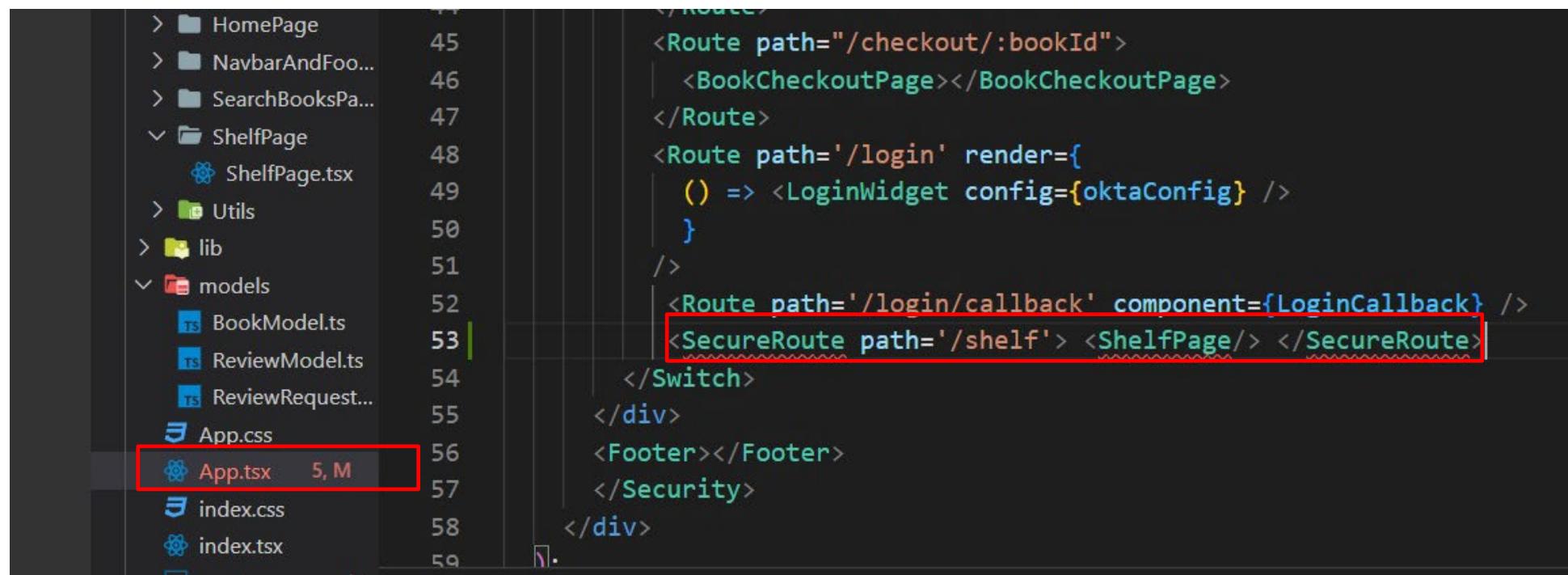
The screenshot shows a user interface with two tabs at the top: 'Loans' (highlighted with a red box) and 'Your History'. The 'Loans' tab is active, showing the text 'Loans'. The 'Your History' tab is inactive, showing the text 'Checkout History'. Below the interface is a code block with line numbers 31 to 40. Arrows point from the code to the corresponding UI elements: one arrow from line 35 to the 'Loans' tab, and another from line 38 to the 'Checkout History' text.

```
31      </div>
32  </nav>
33  <div className="tab-content" id="nav-tabContent">
34    <div className="tab-pane fade show active" id="nav-loans" role="tabpanel" aria-labelledby="nav-loans-tab">
35      <p> Loans</p>
36    </div>
37    <div className="tab-pane fade" id="nav-history" role="tabpanel" aria-labelledby="nav-history-tab">
38      <p> Checkout History</p>
39    </div>
40  </div>
```

2. SECURE ROUTES



Step1 Go to App.tsx, add SecureRoute and specify the path.



The screenshot shows a code editor with a dark theme. On the left is a file tree:

- > HomePage
- > NavbarAndFoo...
- > SearchBooksPa...
- ShelfPage
 - ShelfPage.tsx
- > Utils
- > lib
- models
 - BookModel.ts
 - ReviewModel.ts
 - ReviewRequest...
- App.css
- App.tsx 5, M
- index.css
- index.tsx

The main pane shows the content of `App.tsx`:

```
    </Route>
    <Route path="/checkout/:bookId">
      <BookCheckoutPage></BookCheckoutPage>
    </Route>
    <Route path='/login' render={
      () => <LoginWidget config={oktaConfig} />
    }>
      <Route path='/login/callback' component={LoginCallback} />
      <SecureRoute path='/shelf'> <ShelfPage/> </SecureRoute>
    </Switch>
  </div>
  <Footer></Footer>
</Security>
</div>
```

A red box highlights the file `App.tsx` in the file tree, and another red box highlights the `SecureRoute` component in the code.

Step 2 Hover over the red tilde of SecureRoute, click on “Quick Fix”.

The screenshot shows a code editor interface with a dark theme. On the left is a file tree:

- HomePage
- NavbarAndFoo...
- SearchBooksPa...
- ShelfPage
 - ShelfPage.tsx
- Utils
- lib
- models
 - BookModel.ts
 - ReviewModel.ts
 - ReviewRequest...
- App.css
- App.tsx
- index.css

The main pane displays a portion of `App.tsx`:

```
> 44   </Route>
      45   <Route path="/checkout/:bookId">
      46     <BookCheckoutPage></BookCheckoutPage>
      47   </Route>
      48   <Route path='/login' render={
      49     () => <LoginWidget config={oktaConfig} />
      50   }
      51   >
      52     <Route path='/login/callback' component={Lo
      53       <SecureRoute path='/shelf'> <ShelfPage/> </SecureRoute>
      54     </Switch>
      55   </div>
      56   <Footer></Footer>
      57   <Security>
      58 </div>
```

A tooltip is displayed over the `SecureRoute` component at line 53, column 15:

any
Cannot find name 'SecureRoute'. ts(2304)
View Problem (Alt+F8) Quick Fix... (Ctrl+.)

The word `SecureRoute` is underlined with a red squiggle, indicating a TypeScript error. A red rectangular box highlights the "Quick Fix..." button in the tooltip.

Choose “Update import from @okta(okta-react)”.

```
45     <Route path="/checkout/:bookId">
46       <BookCheckoutPage></BookCheckoutPage>
47     </Route>
48     <Route path='/login' render={
49       () => <LoginWidget config={oktaConfig} />
50     }
51   />
52   <Route path='/login/callback' component={LoginCallback} />
53   <SecureRoute path='/shelf'> <ShelfPage/> </SecureRoute>
54 </Switch>
55 </div>
56 <Footer></Footer>
57 </Security>
58 </div>
59 <div>
```

Quick Fix...

- 💡 Update import from "@okta(okta-react)"
- 💡 Add all missing imports

Step 3 Hover over the red tilde of ShelfPage, click on “Quick Fix”. Choose “Add import from...”.

The screenshot shows a code editor interface with a dark theme. On the left is a file tree:

- > HomePage
- > NavbarAndFoo...
- > SearchBooksPa...
- ✓ ShelfPage
 - ShelfPage.tsx
- > Utils
- > lib
- ✓ models
 - BookModel.ts
 - ReviewModel.ts
 - ReviewRequest...
- ✓ App.css
- ✓ App.tsx 2, M
- ✓ index.css
- ✓ index.tsx

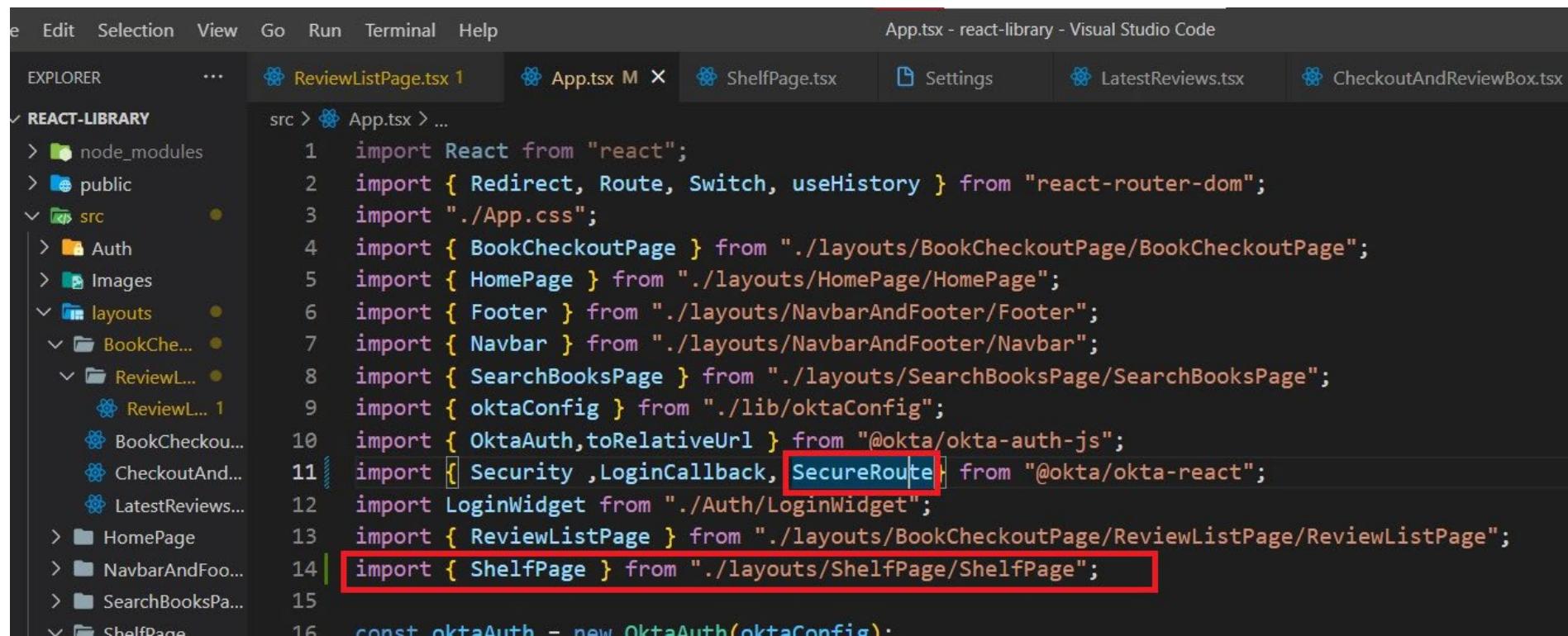
The main pane displays the following code snippet:

```
    </Route>
    <Route path="/checkout/:bookId">
      <BookCheckoutPage></BookCheckoutPage>
    </Route>
    <Route path='/login' render={
      () => <LoginWidget config={oktaConfig} />
    }
    >
      <Route path='/login/callback' component={LoginCallback} />
      <SecureRoute path='/shelf'> <ShelfPage/> </SecureRoute>
    </Switch>
  </div>
  <Footer></Footer>
  <Security>
</div>
```

A red squiggle underline is positioned under the word "ShelfPage" in the line: <SecureRoute path='/shelf'> <ShelfPage/> </SecureRoute>. A "Quick Fix..." menu is open at the bottom right of the squiggle, containing the following options:

- 💡 Add import from "./layouts/ShelfPage/ShelfPage"
- 💡 Disable react/jsx-no-undef for this line
- 💡 Disable react/jsx-no-undef for the entire file
- 💡 Show documentation for react/jsx-no-undef

Then , on the top of the page, we can see both the two are imported.

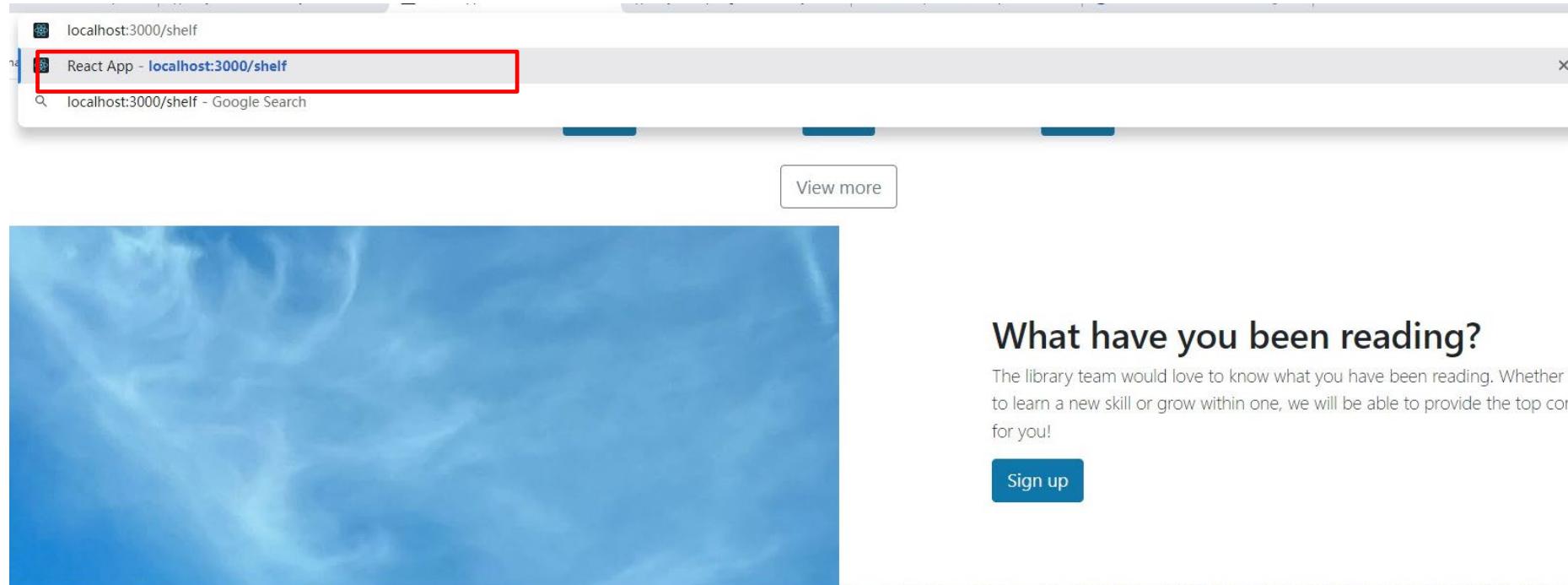


The screenshot shows the Visual Studio Code interface with the title bar "App.tsx - react-library - Visual Studio Code". The left sidebar displays a project structure under "REACT-LIBRARY" with folders like node_modules, public, src, Auth, Images, layouts, BookChe..., ReviewL..., HomePage, NavbarAndFoo..., SearchBooksPa..., and ShelfPage. The "src" folder is expanded. The main editor area shows the code for "App.tsx":

```
src > App.tsx > ...
1 import React from "react";
2 import { Redirect, Route, Switch, useHistory } from "react-router-dom";
3 import "./App.css";
4 import { BookCheckoutPage } from "./layouts/BookCheckoutPage/BookCheckoutPage";
5 import { HomePage } from "./layouts/HomePage/HomePage";
6 import { Footer } from "./layouts/NavbarAndFooter/Footer";
7 import { Navbar } from "./layouts/NavbarAndFooter/Navbar";
8 import { SearchBooksPage } from "./layouts/SearchBooksPage/SearchBooksPage";
9 import { oktaConfig } from "./lib/oktaConfig";
10 import { OktaAuth,toRelativeUrl } from "@okta/okta-auth-js";
11 import { Security ,LoginCallback, SecureRoute } from "@okta/okta-react";
12 import LoginWidget from "./Auth/LoginWidget";
13 import { ReviewListPage } from "./layouts/BookCheckoutPage/ReviewListPage/ReviewListPage";
14 import { ShelfPage } from "./layouts/ShelfPage/ShelfPage";
15
16 const oktaAuth = new OktaAuth(oktaConfig);
```

Two specific imports are highlighted with red boxes: "SecureRoute" at line 11 and "ShelfPage" at line 14.

If we type <http://localhost:3000/shelf>,

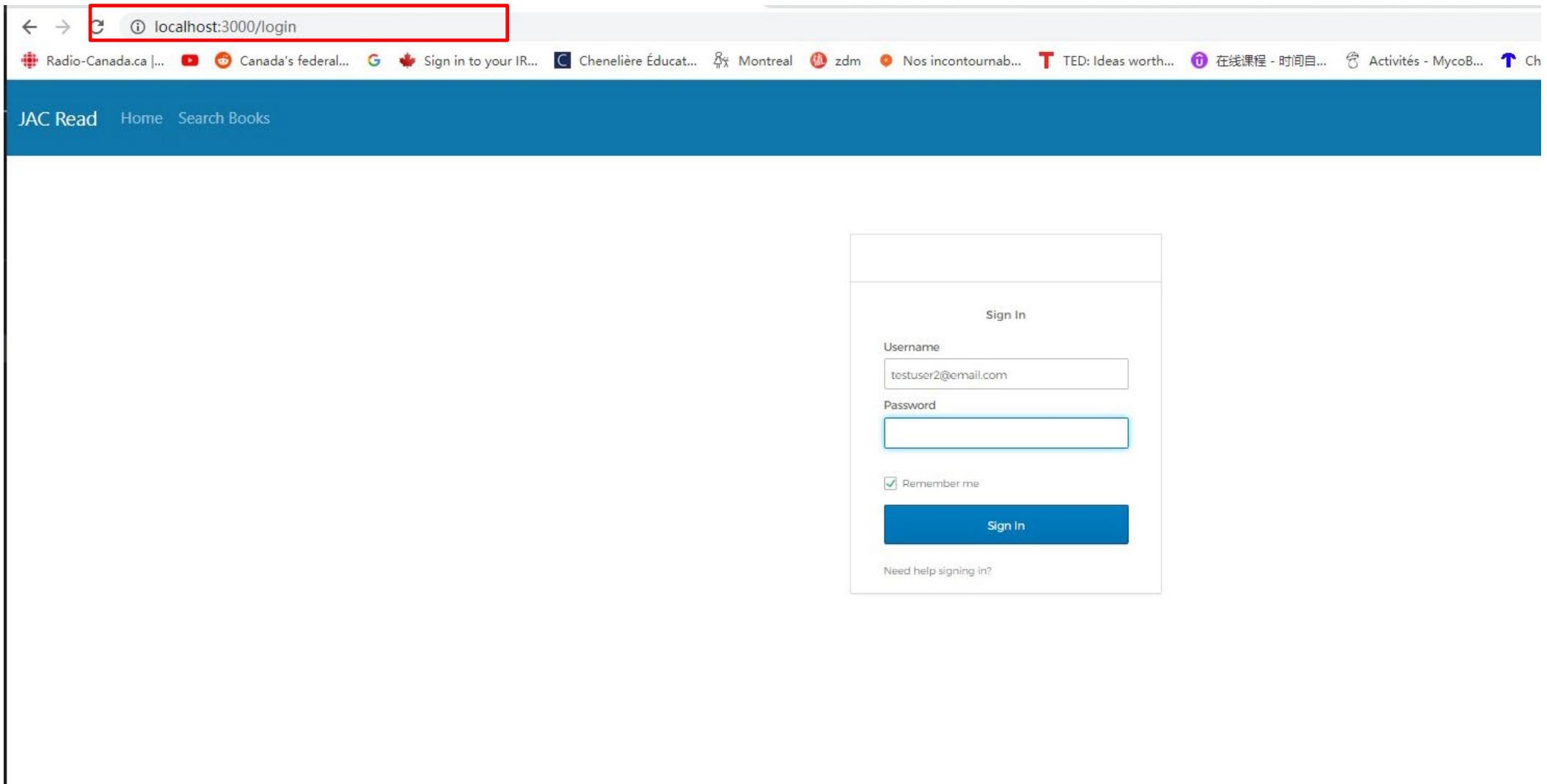


Our collection is always changing!

Try to check in daily as our collection is always changing! We work nonstop to provide the most accurate book selection possible for our JAC Read member! We are diligent about our book selection and our books are always going to be our top priority.



It will redirect us to the log in page.



Input our username and password :

Sign In

Username
testuser2@email.com

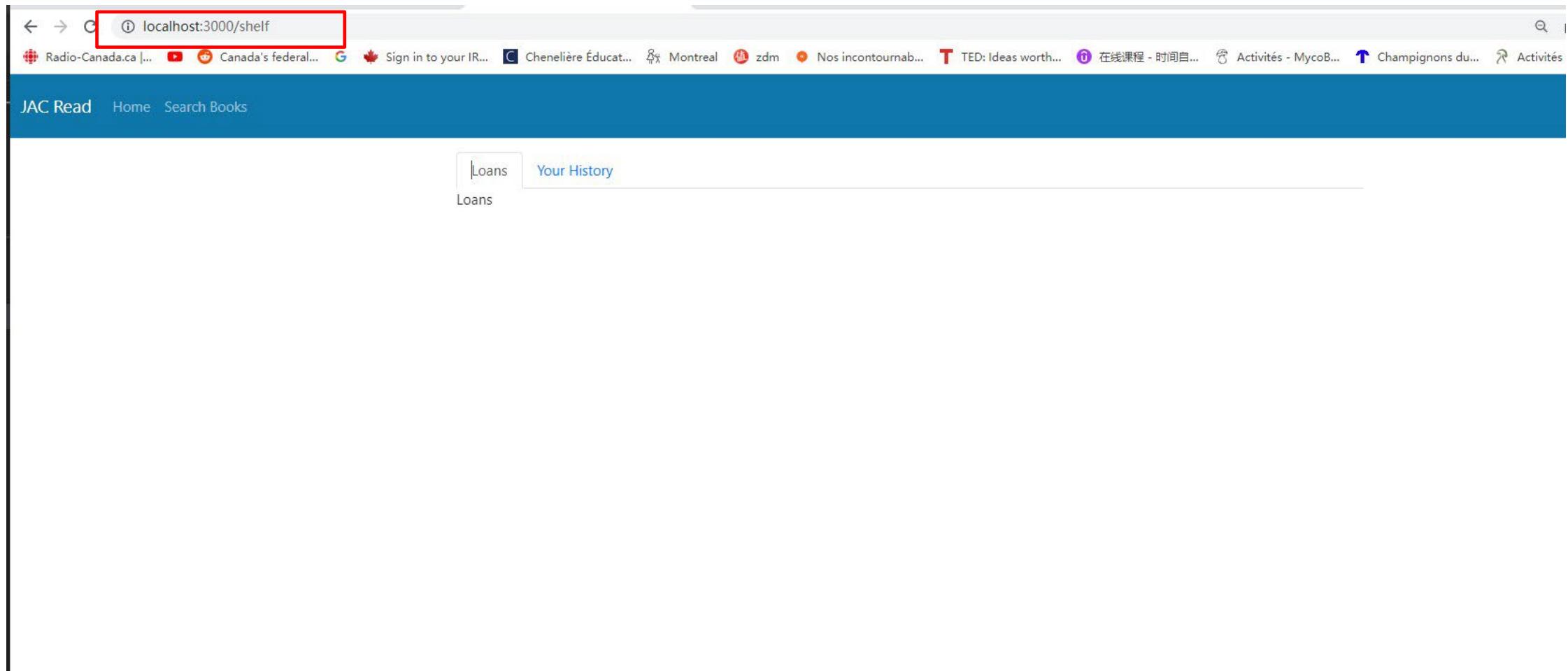
Password

Remember me

Sign In

Need help signing in?

Then, we can visit shelf page.



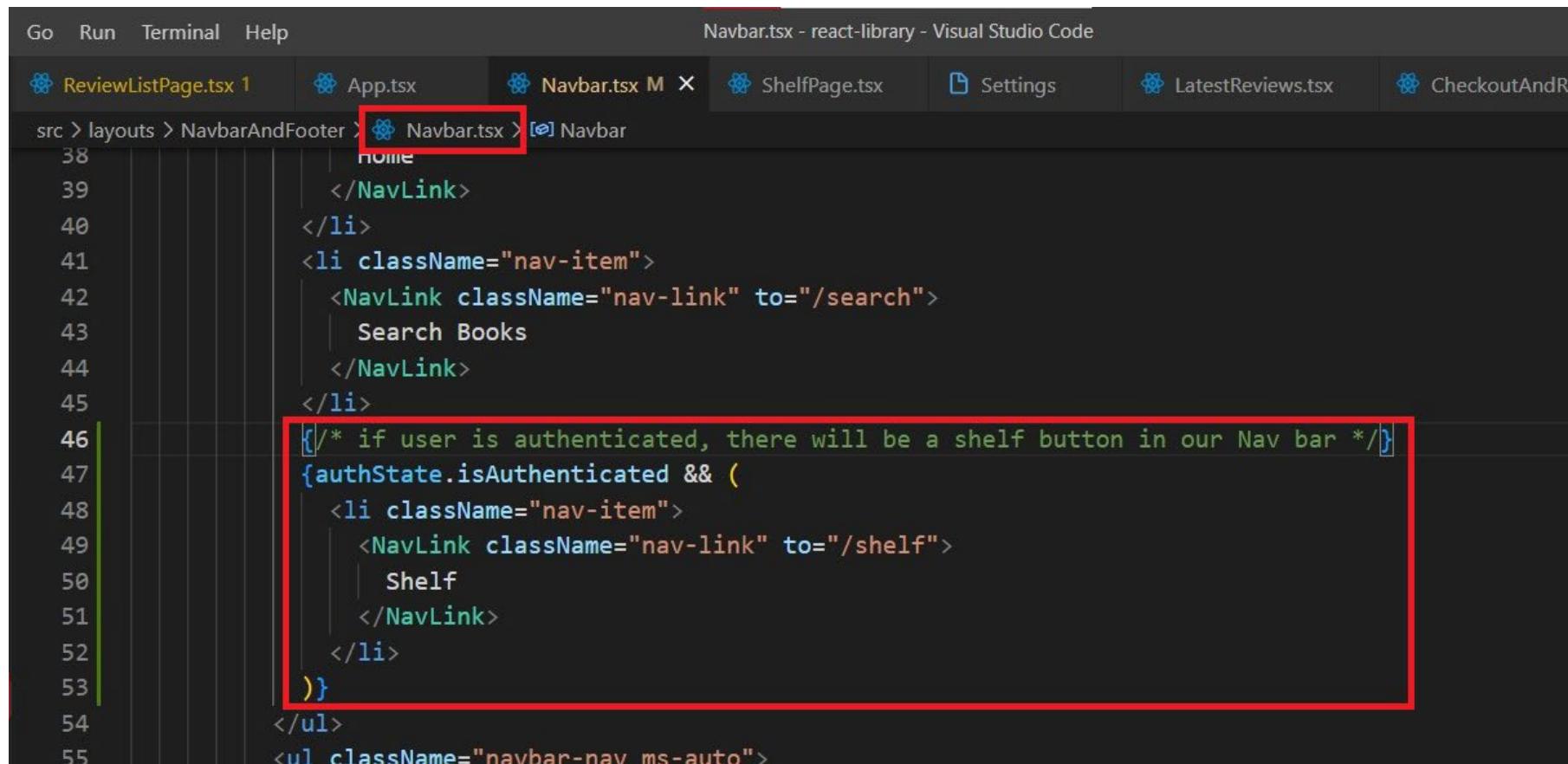


3.NAVIGAGTION BAR ENHANCEMENT



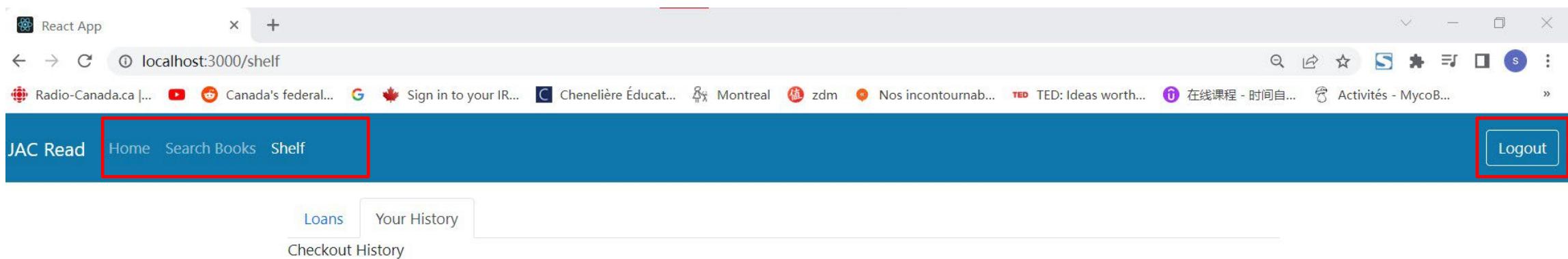
Let's add shelf to our navigation bar.

Step 1 Let 's go to layouts -> NavbarAndFooter, open Navbar.tsx.
Add a NavLink of shelf, and add a authState judgement to it.

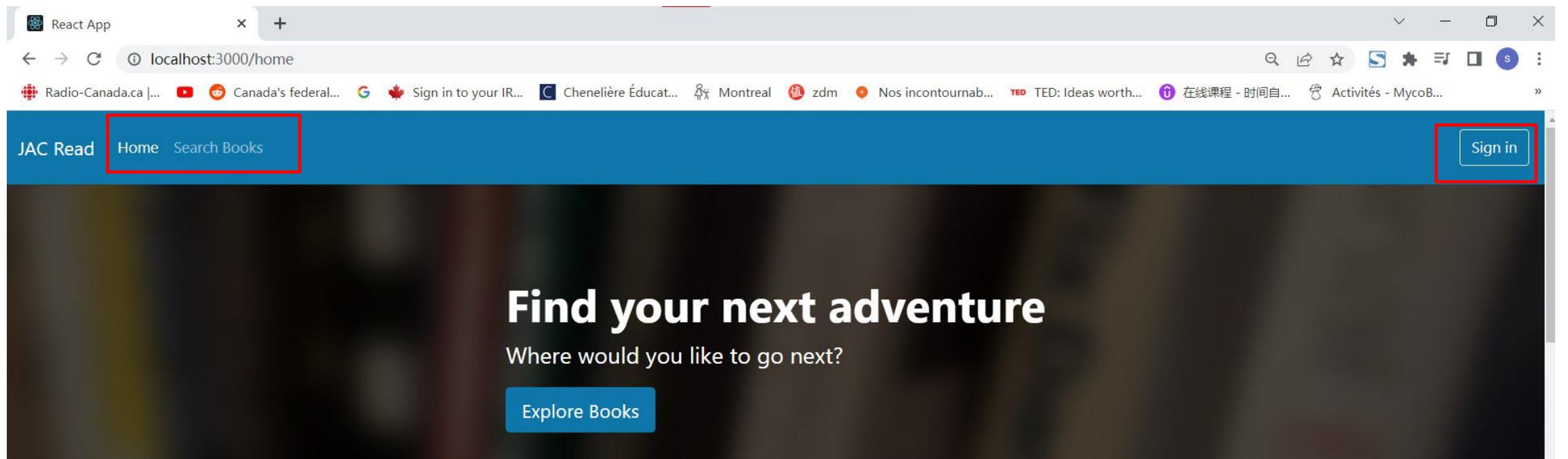


```
Go Run Terminal Help Navbar.tsx - react-library - Visual Studio Code
src > layouts > NavbarAndFooter > Navbar.tsx > Navbar
38     |     home
39     |     </NavLink>
40     |     </li>
41     |     <li className="nav-item">
42     |         <NavLink className="nav-link" to="/search">
43     |             Search Books
44     |         </NavLink>
45     |     </li>
46     |     /* if user is authenticated, there will be a shelf button in our Nav bar */
47     |     {authState.isAuthenticated && (
48     |         <li className="nav-item">
49     |             <NavLink className="nav-link" to="/shelf">
50     |                 Shelf
51     |             </NavLink>
52     |         </li>
53     |     )}
54     </ul>
55     <ul className="navbar-nav ms-auto">
```

When we logged in, in the navigation bar, there is a “ shelf ” link.



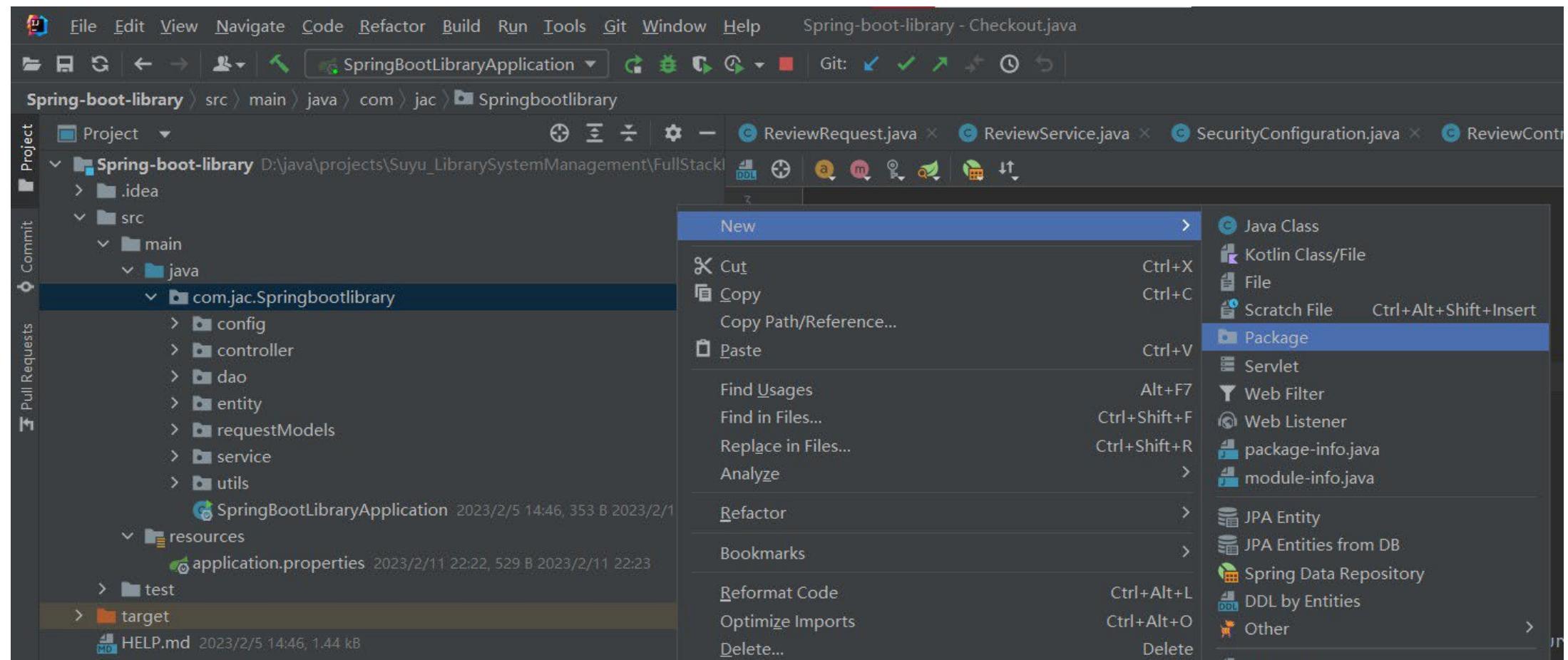
If we log out , in the navigation bar, there will be no “ shelf ” link.

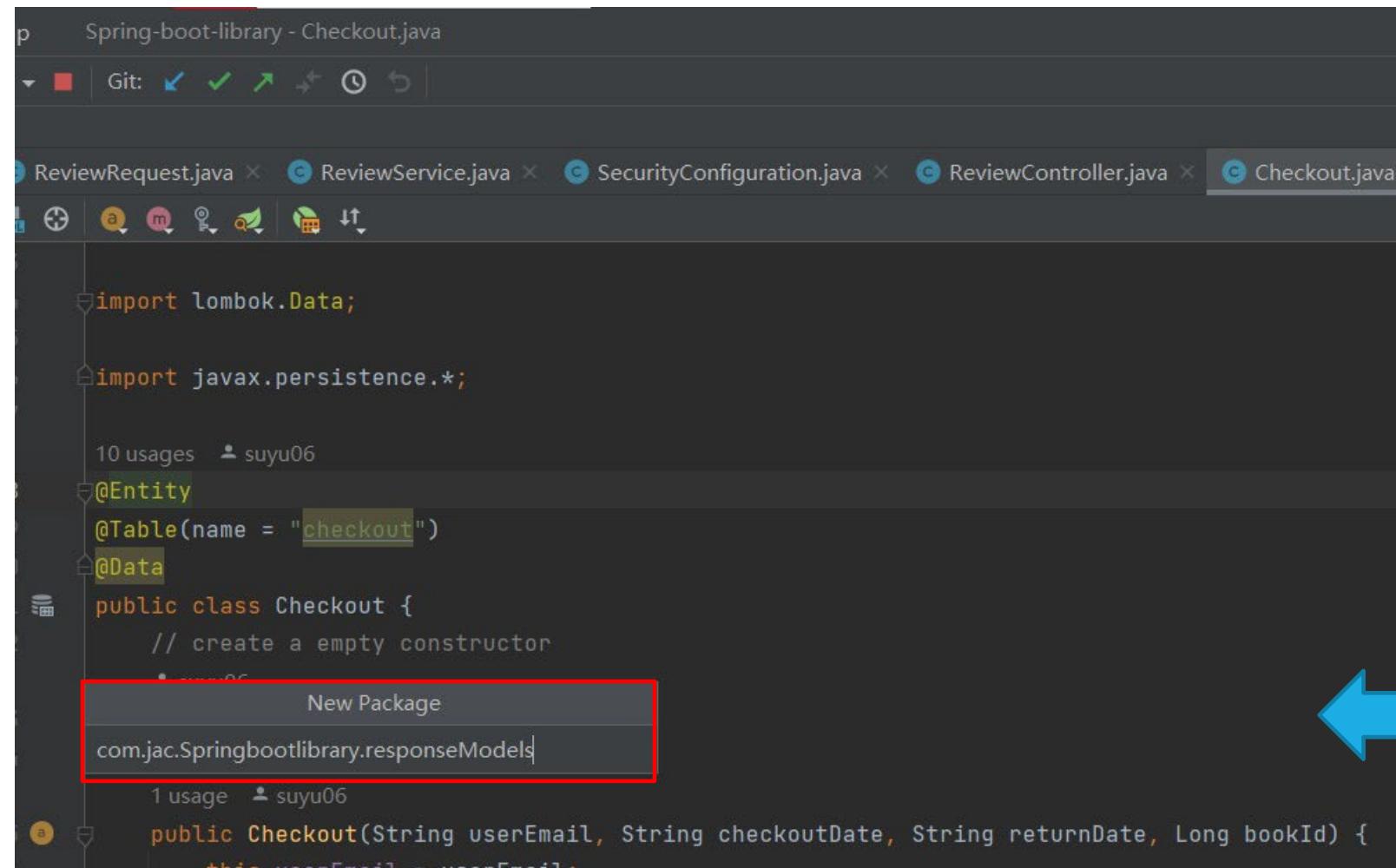


4. SPRINGBOOT CREATE A RESPONSE



Step 1 So let's go ahead and right click on root directory and create a new package.





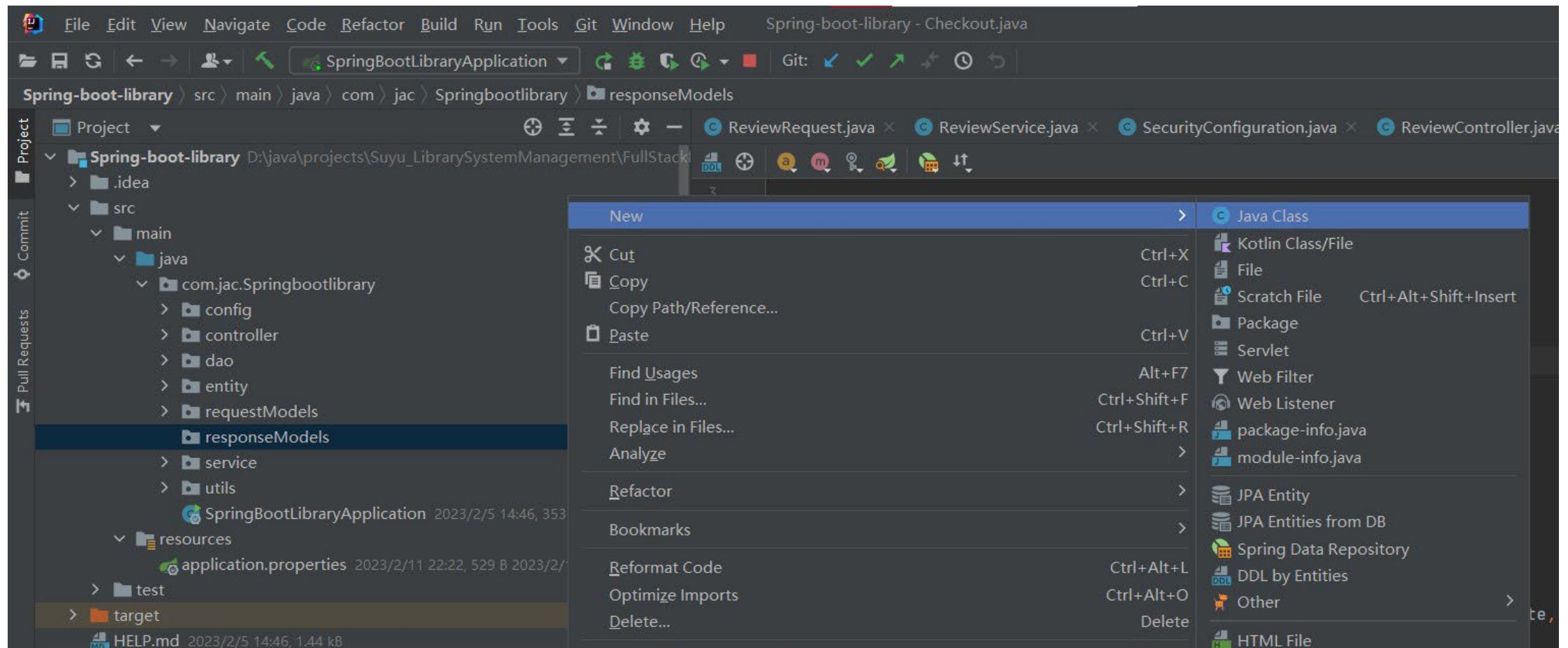
```
p Spring-boot-library - Checkout.java
Git: ↛ ✓ ↞ ⏪ ⏪ ⏪ ⏪ ⏪ ⏪ |
```

ReviewRequest.java ✘ ReviewService.java ✘ SecurityConfiguration.java ✘ ReviewController.java ✘ Checkout.java

```
import lombok.Data;
import javax.persistence.*;
10 usages  suyu06
@Entity
@Table(name = "checkout")
@Data
public class Checkout {
    // create a empty constructor
    ...
    New Package
com.jac.Springbootlibrary.responseModels
1 usage  suyu06
    public Checkout(String userEmail, String checkoutDate, String returnDate, Long bookId) {
        this.userEmail = userEmail;
```

Name it responseModels

Step 2 Inside responseModels, right click and choose “new”, “Java class”

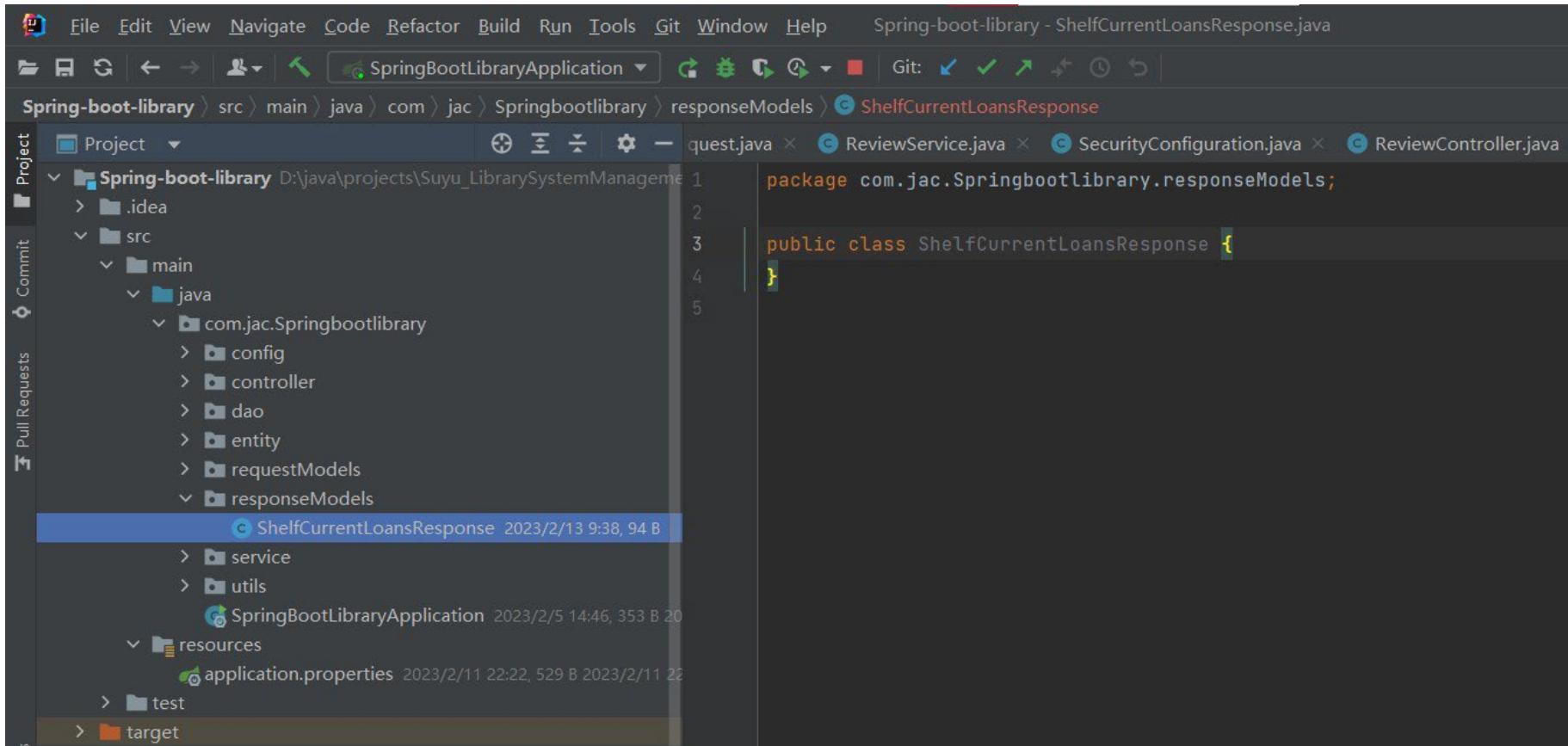


A screenshot of a Java IDE interface, likely IntelliJ IDEA, showing a code editor with Java code. The code is annotated with Lombok annotations like `@Entity`, `@Table`, and `@Data`. A code completion dropdown menu is open at the bottom of the editor, listing several options: "New Java Class", "ShelfCurrentLoansResponse", "Class", "Interface", "Enum", and "@ Annotation". The option "Class" is highlighted with a blue selection bar, and the option "ShelfCurrentLoansResponse" is highlighted with a red selection bar. The background shows other tabs like "ReviewRequest.java", "ReviewService.java", and "SecurityConfiguration.java".

```
3  
4 import lombok.Data;  
5  
6 import javax.persistence.*;  
7  
10 usages  suyu06  
8 @Entity  
9 @Table(name = "checkout")  
10 @Data  
11  
12 New Java Class  
13 ShelfCurrentLoansResponse  
14 Class  
15 Interface  
16 Enum  
17 @ Annotation  
18  
19  
20  
21
```

Choose “Class”, input the class name “ShelfCurrentLoansResponse”.

We will get this class:



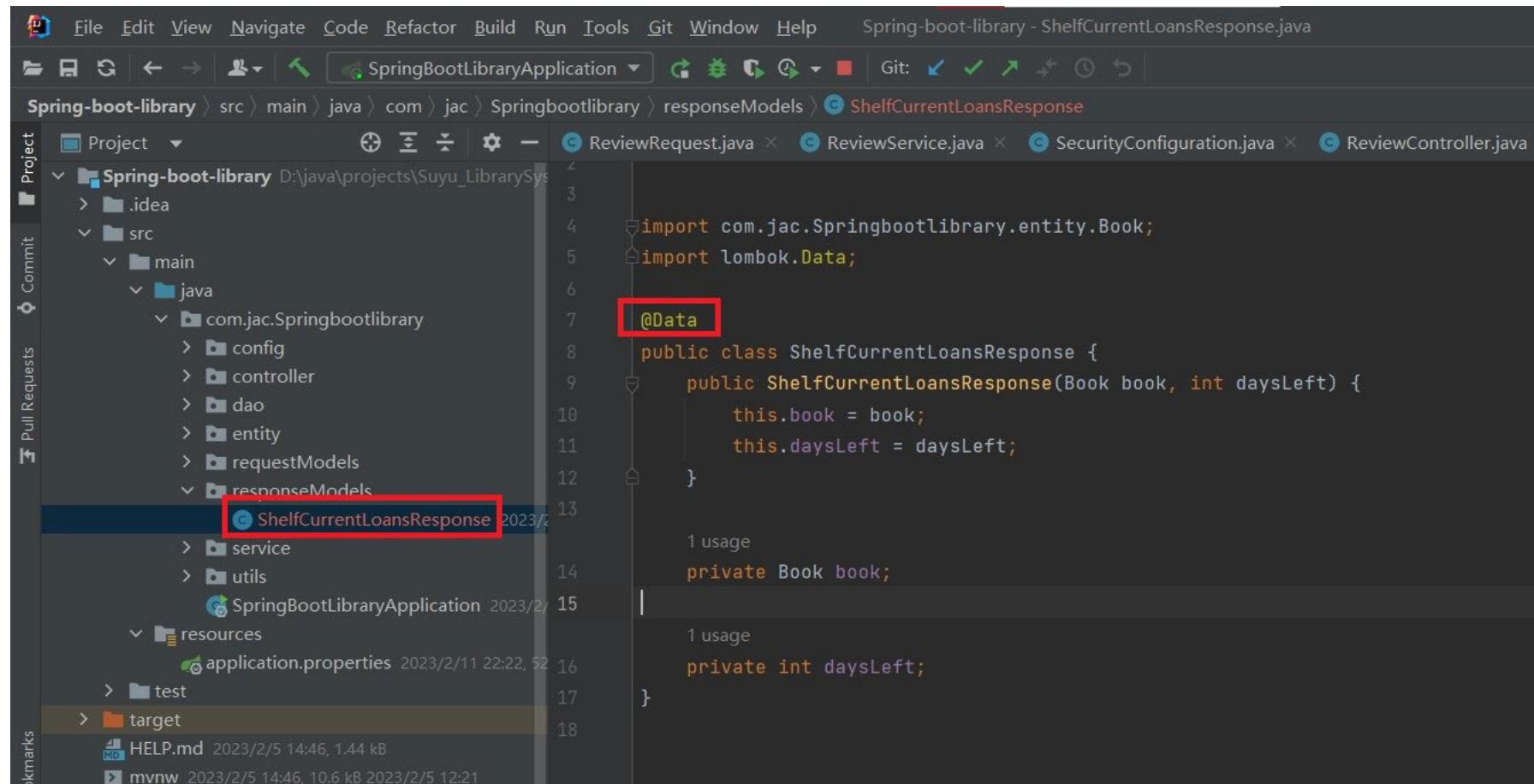
The screenshot shows the IntelliJ IDEA interface with the following details:

- File Path:** Spring-boot-library > src > main > java > com > jac > Springbootlibrary > responseModels > ShelfCurrentLoansResponse
- Code Editor Content:**

```
package com.jac.Springbootlibrary.responseModels;

public class ShelfCurrentLoansResponse {
```
- Project Structure:**
 - Project: Spring-boot-library
 - src
 - main
 - java
 - com.jac.Springbootlibrary
 - config
 - controller
 - dao
 - entity
 - requestModels
 - responseModels
 - ShelfCurrentLoansResponse
 - service
 - utils
 - SpringBootLibraryApplication
 - resources
 - application.properties
 - test
 - target
- Toolbars and Status Bar:** The top bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help, and a Git status bar.

Step 3, add @Data annotation and add properties for this class. And create its two properties : “book” and “daysleft”.



The screenshot shows the IntelliJ IDEA interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project Bar:** SpringBootLibraryApplication.
- Toolbars:** Standard, Git.
- Code Editor:** The file `ShelfCurrentLoansResponse.java` is open. The code contains:

```
import com.jac.Springbootlibrary.entity.Book;
import lombok.Data;

@Data
public class ShelfCurrentLoansResponse {
    public ShelfCurrentLoansResponse(Book book, int daysLeft) {
        this.book = book;
        this.daysLeft = daysLeft;
    }

    private Book book;

    private int daysLeft;
}
```
- Project Tree:** Shows the project structure under `Spring-boot-library`, including `.idea`, `src` (with `main` and `java`), `com.jac.Springbootlibrary` (with `config`, `controller`, `dao`, `entity`, `requestModels`, `responseModels`), and `ShelfCurrentLoansResponse`.
- Annotations:** The `@Data` annotation is highlighted with a red box.
- Status Bar:** Shows `2023/2/5 14:46, 1.44 kB` and `mvnw 2023/2/5 14:46, 10.6 kB 2023/2/5 12:21`.

5. SPRINGBOOT CURRENT LOANS SERVICE



In BookService class, underneath currentLoansCount function, create a new function:
public List<ShelfCurrentLoansResponse> currentLoans(String userEmail)

The screenshot shows a Java IDE interface with the following details:

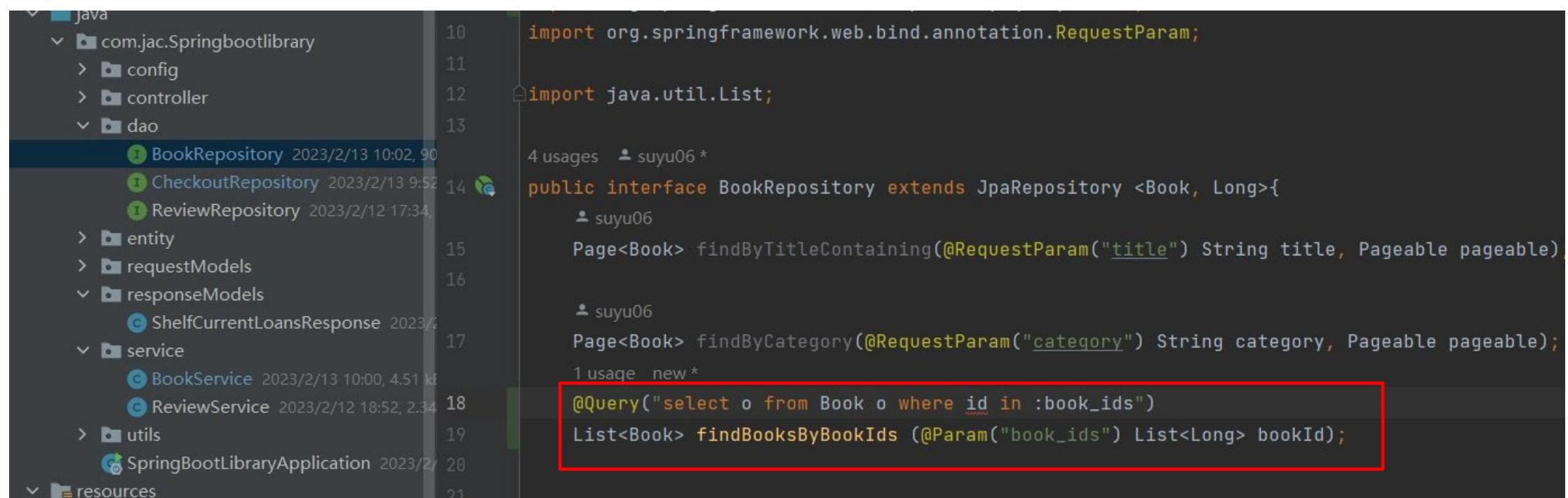
- Menu Bar:** Run, Tools, Git, Window, Help.
- Title Bar:** Spring-boot-library - BookService.java
- Toolbar:** Application, Git, and other development tools.
- Project Explorer:** gbootlibrary > service > BookService > currentLoans
- Code Editor:** The code for BookService.java is displayed. The `public List<ShelfCurrentLoansResponse> currentLoans(String userEmail)` method is highlighted with a red box.

```
72     //  
73     new.*  
74     public List<ShelfCurrentLoansResponse> currentLoans(String userEmail) throws Exception {  
75         List<ShelfCurrentLoansResponse> shelfCurrentLoansResponses = new ArrayList<>();  
76         // extract all the current loans books by user email  
77         List<Checkout> checkoutList = checkoutRepository.findBooksByUserEmail(userEmail);  
78         List<Long> bookIdList = new ArrayList<>();  
79         //loop the checkout list ,extract all of the book IDs  
80         for (Checkout i : checkoutList) {  
81             bookIdList.add(i.getBookId());  
82         }  
83         // Passing bookIdList, find all the books in our bookRepository.  
84         List<Book> books = bookRepository.findBooksByBookIds(bookIdList);  
85     }
```

6. SPRINGBOOT QUERY ANNOTATION



Step 2 So go to dao package and open BookRepository.
Create a interface function findBooksByBookIds and add a @Query annotation to it.



```
import org.springframework.web.bind.annotation.RequestParam;
import java.util.List;

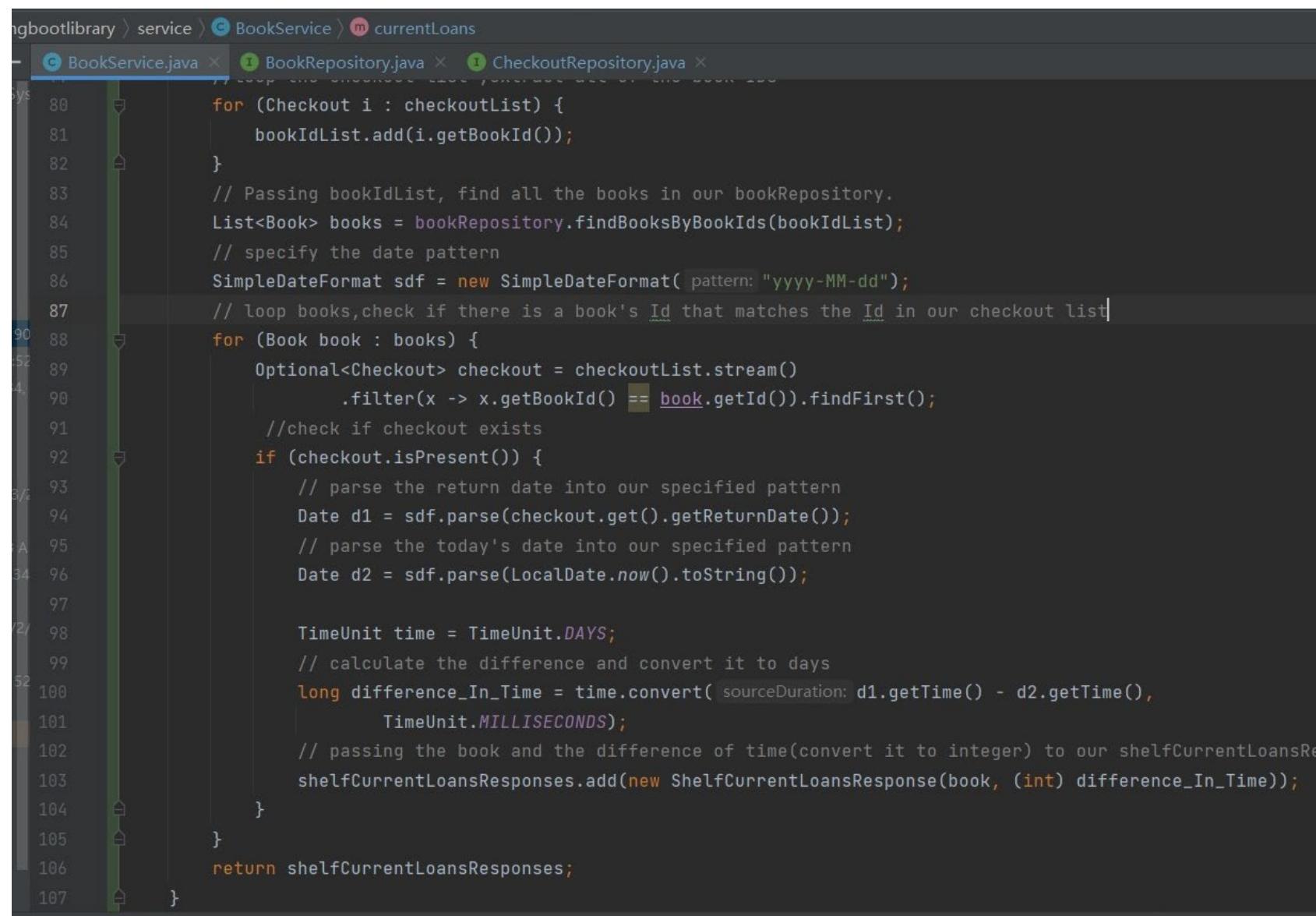
4 usages  suyu06 *
public interface BookRepository extends JpaRepository <Book, Long>{
    suyu06
    Page<Book> findByTitleContaining(@RequestParam("title") String title, Pageable pageable);

    suyu06
    Page<Book> findByCategory(@RequestParam("category") String category, Pageable pageable);
    1 usage  new *

    @Query("select o from Book o where id in :book_ids")
    List<Book> findBooksByBookIds (@Param("book_ids") List<Long> bookId);
```

7. SPRINGBBOT CURRENT LOANS SERVICE IMPLEMENTATION





```
ngbootlibrary > service > BookService > currentLoans
- BookService.java × BookRepository.java × CheckoutRepository.java ×
Sys 80     for (Checkout i : checkoutList) {
81         bookIdList.add(i.getBookId());
82     }
83     // Passing bookIdList, find all the books in our bookRepository.
84     List<Book> books = bookRepository.findBooksByBookIds(bookIdList);
85     // specify the date pattern
86     SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-MM-dd");
87     // loop books, check if there is a book's Id that matches the Id in our checkout list
88     for (Book book : books) {
89         Optional<Checkout> checkout = checkoutList.stream()
90             .filter(x -> x.getBookId() == book.getId()).findFirst();
91         // check if checkout exists
92         if (checkout.isPresent()) {
93             // parse the return date into our specified pattern
94             Date d1 = sdf.parse(checkout.get().getReturnDate());
95             // parse the today's date into our specified pattern
96             Date d2 = sdf.parse(LocalDate.now().toString());
97
98             TimeUnit time = TimeUnit.DAYS;
99             // calculate the difference and convert it to days
100            long difference_In_Time = time.convert( sourceDuration: d1.getTime() - d2.getTime(),
101                TimeUnit.MILLISECONDS);
102            // passing the book and the difference of time(convert it to integer) to our shelfCurrentLoansRes
103            shelfCurrentLoansResponses.add(new ShelfCurrentLoansResponse(book, (int) difference_In_Time));
104        }
105    }
106
107    return shelfCurrentLoansResponses;
}
```

Back to BookService class,
ShelfCurrentLoansResponse(),
We specify the date type, and
find the book, then calculate
the days left

8. SPRINGBOOT CONTROLLER ENDPOINT



The screenshot shows a Java IDE interface with several tabs at the top: BookService.java, BookController.java (which is currently selected), BookRepository.java, and CheckoutRepository.java. The BookController.java code is displayed in the main editor area:

```
import java.util.List;
import suyu06.*;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("api/books")
public class BookController {
    // inject BookService
    5 usages
    private BookService bookService;

    //
    // suyu06
    @Autowired
    public BookController(BookService bookService) { this.bookService = bookService; }

    //Shelf Current Loans Response
    new ^
    @GetMapping("/secure/currentloans")
    public List<ShelfCurrentLoansResponse> currentLoans(@RequestHeader(value = "Authorization") String token)
        throws Exception {
        String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
        return bookService.currentLoans(userEmail);
    }
}
```

A red rectangular box highlights the `@GetMapping` annotated method starting from line 28.

Go to BookController,
Use Get method to
create API for current
loans.

Let's copy the accessToken in the console and test our API in Postman.

The screenshot shows a browser window with the URL `localhost:3000/home`. The main content area displays a dark-themed landing page for a service named "JAC Read". The page features a large call-to-action button with the text "Find your next adventure" and "Where would you like to go next?". Below this is a blue button labeled "Explore Books". In the top right corner of the browser window, the React DevTools are open, specifically the "Console" tab. The console log shows several objects, with one object's `accessToken` field highlighted with a red rectangle. The highlighted value is a long string of characters: `"eyJraWQiOijFeW1GaFZESVk4TWF0cVkJtmt..."`. The rest of the object properties include `authorizeUrl`, `claims`, `expiresAt`, `scopes`, `tokenType`, `userinfoUrl`, `idToken`, `isAuthenticated`, and `refreshToken`.

```
react-dom.development.js:29840
Download the React DevTools for a better development
experience: https://reactjs.org/link/react-devtools
▶ Object
  ▶ Object ⓘ
    ▶ accessToken:
      accessToken: "eyJraWQiOijFeW1GaFZESVk4TWF0cVkJtmt...
        authorizeUrl: "https://dev-65756343.okta.com/oaut...
      > claims: {ver: 1, jti: 'AT.JfQgZslo12MwxSkND5ASH4u...
      > expiresAt: 1676306939
      > scopes: (3) ['email', 'openid', 'profile']
      > tokenType: "Bearer"
      > userinfoUrl: "https://dev-65756343.okta.com/oauth...
      > [[Prototype]]: Object
    > idToken: {idToken: 'eyJraWQiOijFeW1GaFZESVk4TWF0cVkJtmt...
      isAuthenticated: true
      refreshToken: undefined
      > [[Prototype]]: Object
    >
```

Choose “GET”, type <http://localhost:8080/api/books/secure/currentloans>.
Click on “Authorization”, choose “Bearer Token” in type , paste access token in the token box, then click on “Send”.

The screenshot shows the Postman application interface. At the top, there is a header bar with a dark grey background. Below it, the main interface has a light grey background. In the center, there is a large rectangular form for making API requests.

The form contains the following fields:

- Method:** GET (highlighted with a yellow border)
- URL:** http://localhost:8080/api/books/secure/currentloans
- Send:** A blue button with a dropdown arrow.

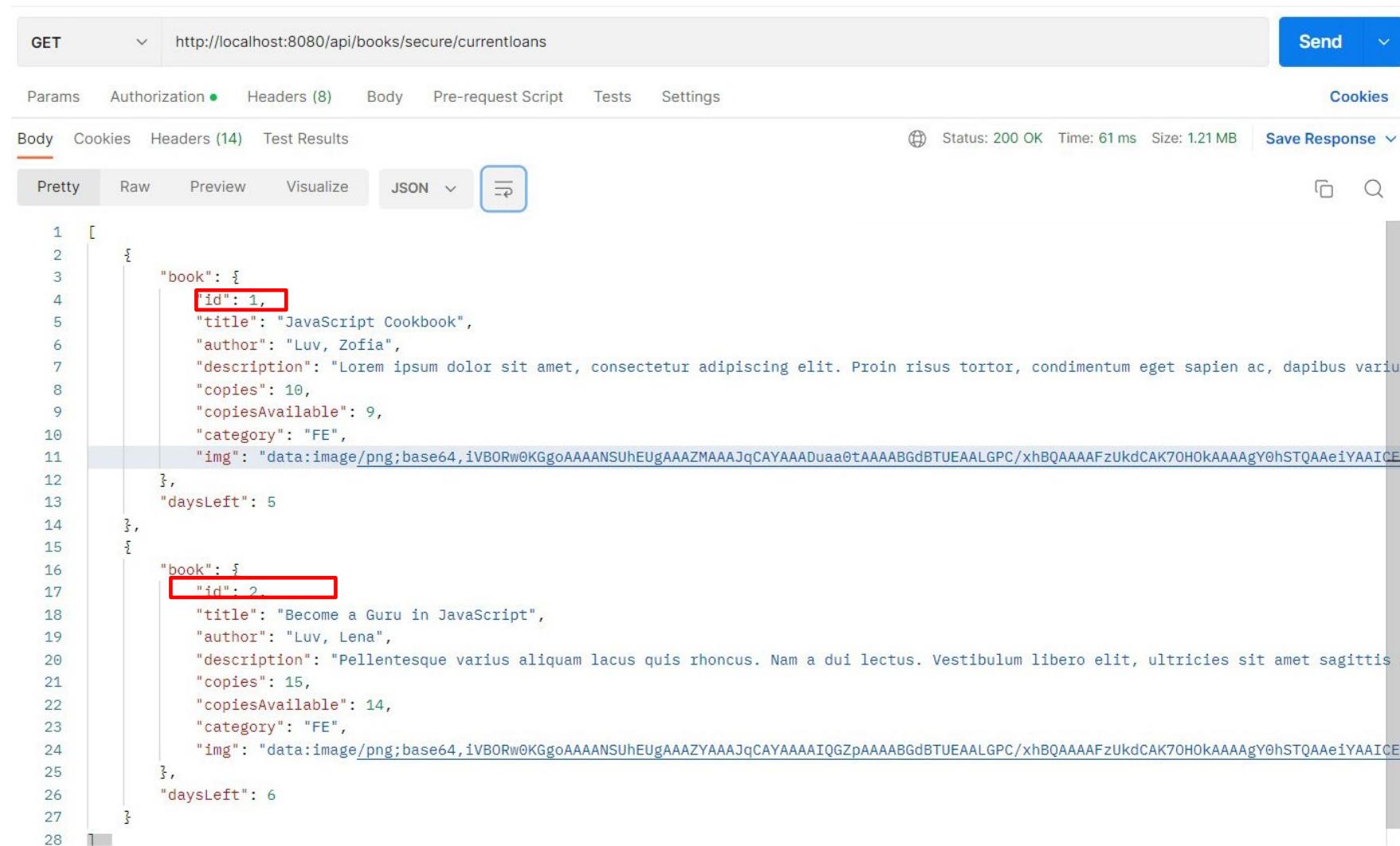
Below the URL field, there are several tabs: Params, **Authorization** (highlighted with a red border), Headers (8), Body, Pre-request Script, Tests, Settings, and Cookies.

The **Authorization** tab is active, showing the following configuration:

- Type:** Bearer Token (highlighted with a red border)
- Token:** eyJraWQiOiJFeW1GaFZESVkJWF0cVkJTmt... (A long string of characters, likely a JWT token, enclosed in a red box for emphasis.)

To the left of the authorization section, there is a note: "The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)".

We will get two results returned.



GET http://localhost:8080/api/books/secure/currentloans

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Status: 200 OK Time: 61 ms Size: 1.21 MB Save Response

Pretty Raw Preview Visualize JSON

```
1 [  
2 {  
3   "book": {  
4     "id": 1,  
5     "title": "JavaScript Cookbook",  
6     "author": "Luv, Zofia",  
7     "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius.",  
8     "copies": 10,  
9     "copiesAvailable": 9,  
10    "category": "FE",  
11    "img": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAZMAAAJqCAYAAADuaa0tAAAABgdBTUEALGPC/xhBQAAAFAzUkdCAK7OHOkAAAAgY0hSTQAAeiYAAICEA  
12  },  
13  "daysLeft": 5  
14 },  
15 {  
16   "book": {  
17     "id": 2,  
18     "title": "Become a Guru in JavaScript",  
19     "author": "Luv, Lena",  
20     "description": "Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis e",  
21     "copies": 15,  
22     "copiesAvailable": 14,  
23     "category": "FE",  
24     "img": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAZYAAAJqCAYAAAIIQGZpAAAABgdBTUEALGPC/xhBQAAAFAzUkdCAK7OHOkAAAAgY0hSTQAAeiYAAICEA  
25  },  
26  "daysLeft": 6  
27 }]  
28 ]
```

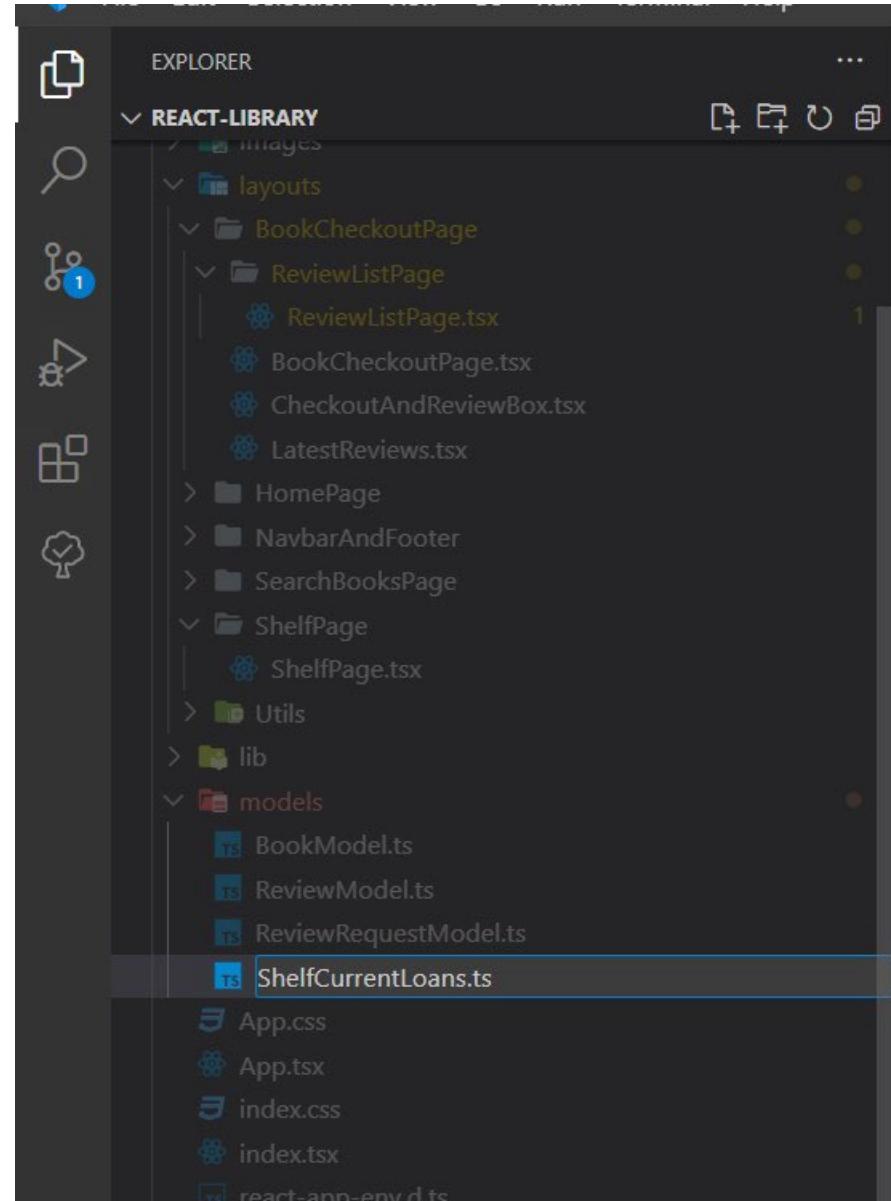
Open checkout table in our database , we will get the two same results.

The screenshot shows the MySQL Workbench interface. The title bar says "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. The toolbar has icons for SQL, Navigator, Schemas, Tables, Views, Stored Procedures, Functions, and a search icon. The Navigator pane shows the database structure: SCHEMAS (b'mydb', customer_tracker, employees, member_tracker, myblog, mydb, passenger_tracker, reactlibrarydatabase), TABLES (book, checkout, history, messages, review), Views, Stored Procedures, and Functions. The reactlibrarydatabase schema is expanded, showing its tables: book, checkout, history, messages, review. The checkout table is selected and highlighted with a red border. The SQL editor pane contains the query: "SELECT * FROM reactlibrarydatabase.checkout;". The Result Grid pane displays the following data:

	id	user_email	checkout_date	return_date	book_id
1	1	testuser2@email.com	2023-02-11	2023-02-18	1
2	2	testuser2@email.com	2023-02-12	2023-02-19	2

9. REACT SHELF CURRENT LOANS MODEL





Step 1 Right click on models and choose “new file”.
Name it ShelfCurrentLoans.ts

Step2 create the properties which are mapping to the ShelfCurrentLoansResponse class in our backend,

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

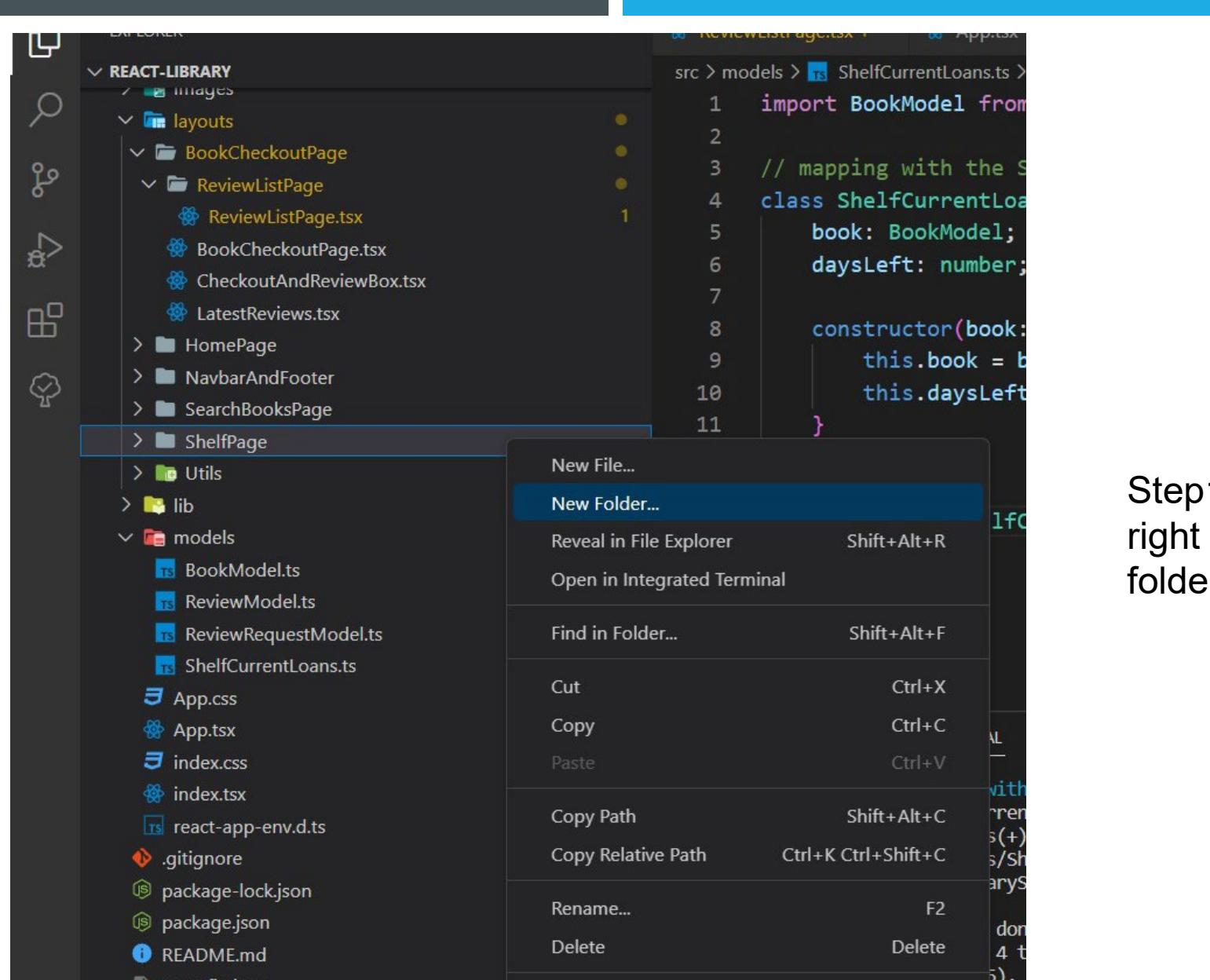
- File Explorer (Left):** Shows the project structure under "REACT-LIBRARY".
 - images
 - layouts
 - BookCheckoutPage
 - ReviewListPage
 - ReviewListPage.tsx
 - BookCheckoutPage.tsx
 - CheckoutAndReviewBox.tsx
 - LatestReviews.tsx
 - HomePage
 - NavbarAndFooter
 - SearchBooksPage
 - ShelfPage
 - ShelfPage.tsx
 - Utils
 - lib
 - models
 - BookModel.ts
- Code Editor (Right):** The active file is "ShelfCurrentLoans.ts" located at "src > models".

```
import BookModel from "./BookModel";
// mapping with the ShelfCurrentLoansResponse class in our backend
class ShelfCurrentLoans {
    book: BookModel;
    daysLeft: number;

    constructor(book: BookModel, daysLeft: number) {
        this.book = book;
        this.daysLeft = daysLeft;
    }
}
export default ShelfCurrentLoans;
```
- Top Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Tab Bar:** ReviewListPage.tsx, App.tsx, Navbar.tsx, ShelfCurrentLoans.ts (highlighted), ShelfPage.tsx.

10. REACT NEW LOANS COMPONENT





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

- Explorer View:** Shows a tree structure of files and folders. The 'REACT-LIBRARY' folder is expanded, showing 'images', 'layouts', 'utils', 'lib', 'models', and other files like 'App.css', 'App.tsx', etc.
- Editor View:** Displays two code files: 'ReviewListPage.tsx' and 'App.tsx'. The 'ReviewListPage.tsx' file is active, showing code related to 'BookModel' and 'ShelfCurrentLoans.ts'.
- Context Menu:** A context menu is open over the 'ShelfPage' folder in the Explorer. The menu items include:
 - New File...
 - New Folder...** (highlighted)
 - Reveal in File Explorer Shift+Alt+R
 - Open in Integrated Terminal
 - Find in Folder... Shift+Alt+F
 - Cut Ctrl+X
 - Copy Ctrl+C
 - Paste Ctrl+V
 - Copy Path Shift+Alt+C
 - Copy Relative Path Ctrl+K Ctrl+Shift+C
 - Rename... F2
 - Delete Delete

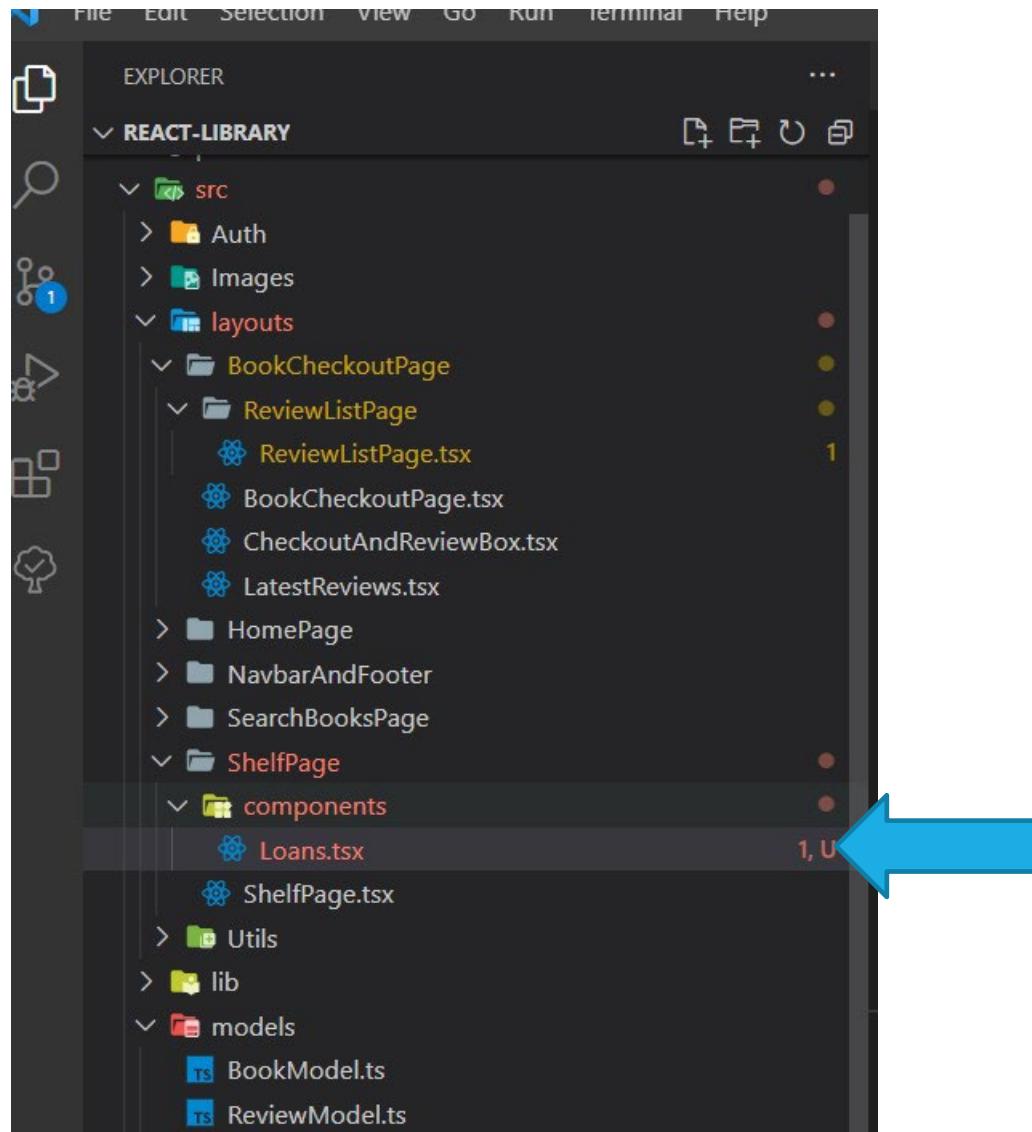
Step1 Go into our layouts folder, right click and create a new folder called components.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for navigation and search. The main area displays a file structure for a 'REACT-LIBRARY'. The 'src' folder contains 'Auth', 'Images', 'layouts', 'ReviewListPage', 'BookCheckoutPage', 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', 'ShelfPage', and 'components'. The 'models' folder contains 'BookModel.ts', 'ReviewModel.ts', 'ReviewRequestModel.ts', and 'ShelfCurrentLoans.ts'. The 'ShelfCurrentLoans.ts' file is open in the editor, showing the following code:

```
1 import BookModel from './BookModel'
2
3 // mapping with the
4 class ShelfCurrentLoans {
5   book: BookModel
6   daysLeft: number
7
8   constructor(book: BookModel, daysLeft: number) {
9     this.book = book
10    this.daysLeft = daysLeft
11  }
12}
13
14 export default ShelfCurrentLoans
```

A context menu is open over the 'components' folder in the sidebar, listing options like 'New File...', 'New Folder...', 'Reveal in File Explorer', 'Open in Integrated Terminal', 'Find in Folder...', 'Cut', 'Copy', and 'Paste'.

Step 2
Right click “components”, choose “New file.”



Name it Loans.tsx

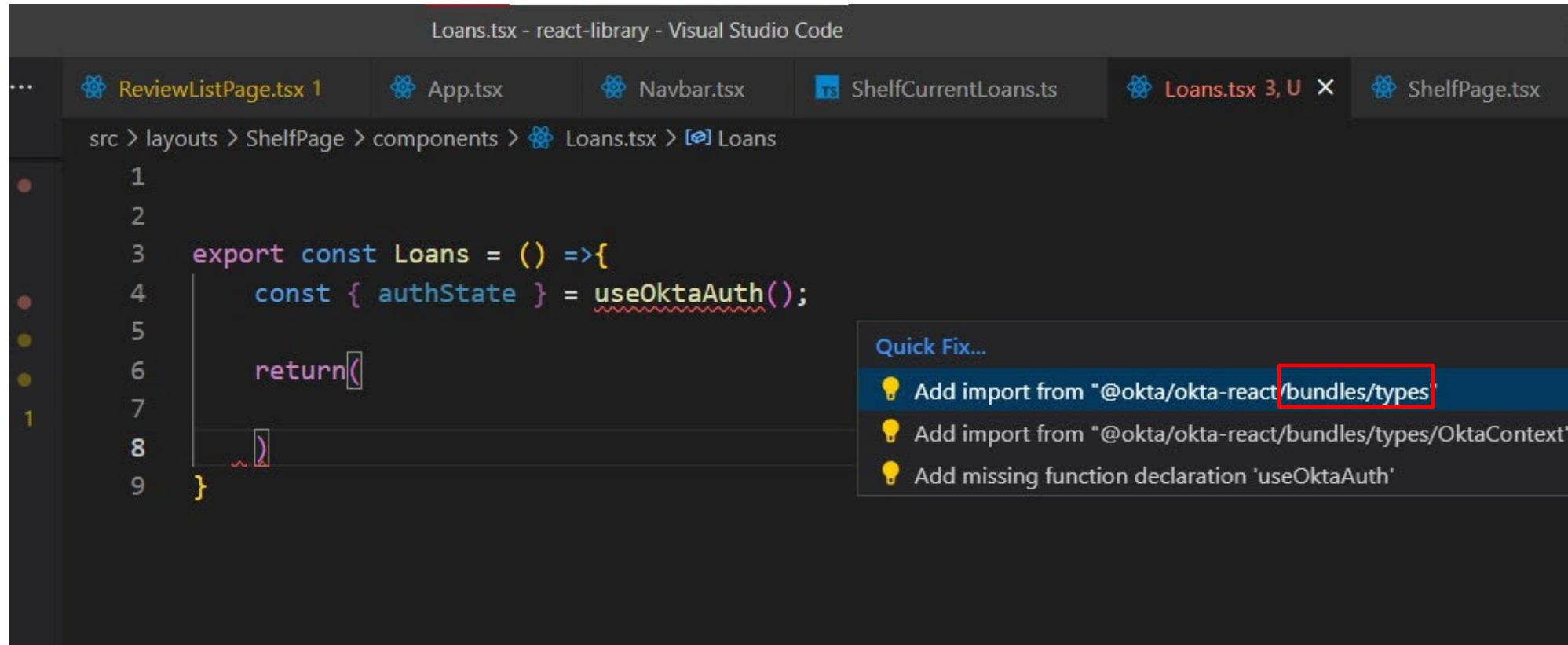
Step 3 create a export const structure as always.

The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar contains a file tree under the heading 'REACT-LIBRARY'. The 'src' folder contains 'Auth', 'Images', 'layouts' (which includes 'BookCheckoutPage' and 'ReviewListPage' sub-folders), 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', 'ShelfPage' (which includes 'components' sub-folders), and 'Utils' and 'lib' folders. The 'components' folder under 'ShelfPage' contains 'Loans.tsx' and 'ShelfPage.tsx'. The right side of the screen shows the code editor with the file 'Loans.tsx' open. The code is:

```
1  export const Loans = () =>[  
2  
3      return(  
4          ...  
5      )  
6 ]
```

Step 4, create authState and import okta.

Remember to remove the “/bundles/types” from the path generated automatically.



The screenshot shows a Visual Studio Code interface with the title bar "Loans.tsx - react-library - Visual Studio Code". Below the title bar, there are tabs for "ReviewListPage.tsx 1", "App.tsx", "Navbar.tsx", "ShelfCurrentLoans.ts", "Loans.tsx 3, U X", and "ShelfPage.tsx". The current file is "Loans.tsx". The code editor displays the following TypeScript code:

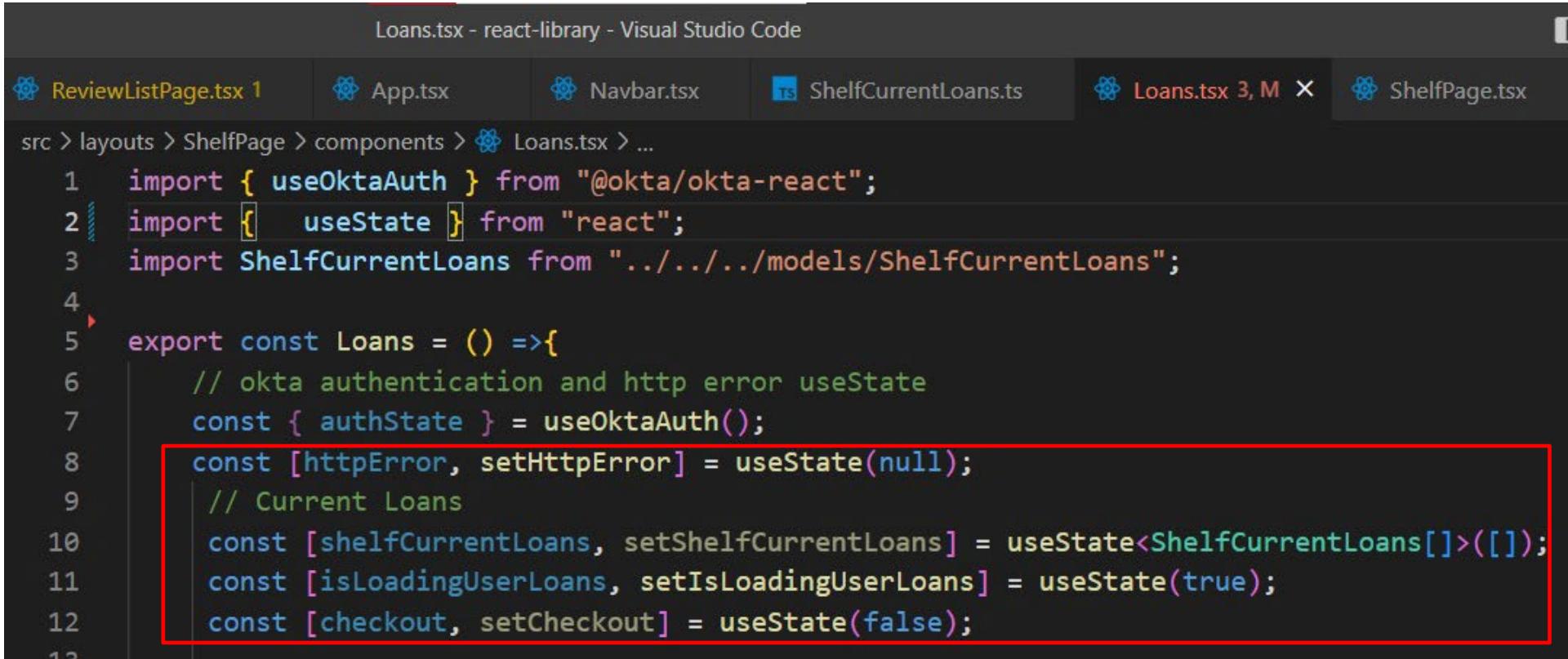
```
1
2
3  export const Loans = () =>{
4      const { authState } = useOktaAuth();
5
6      return(
7          ...
8      )
9 }
```

A "Quick Fix..." tooltip is open over the line "useOktaAuth()", which is underlined with a red squiggle. The tooltip contains three suggestions:

- 💡 Add import from "@okta(okta-react/bundles/types"
- 💡 Add import from "@okta(okta-react/bundles/types/OktaContext"
- 💡 Add missing function declaration 'useOktaAuth'

The third suggestion, "Add missing function declaration 'useOktaAuth'", is highlighted with a red rectangle.

Step 5 create useState.



Loans.tsx - react-library - Visual Studio Code

ReviewListPage.tsx 1 App.tsx Navbar.tsx ShelfCurrentLoans.ts Loans.tsx 3, M X ShelfPage.tsx

src > layouts > ShelfPage > components > Loans.tsx > ...

```
1 import { useOktaAuth } from "@okta(okta-react";
2 import { useState } from "react";
3 import ShelfCurrentLoans from "../../models/ShelfCurrentLoans";
4
5 export const Loans = () =>{
6     // okta authentication and http error useState
7     const { authState } = useOktaAuth();
8     const [httpError, setHttpError] = useState(null);
9     // Current Loans
10    const [shelfCurrentLoans, setShelfCurrentLoans] = useState<ShelfCurrentLoans[]>([]);
11    const [isLoadingUserLoans, setIsLoadingUserLoans] = useState(true);
12    const [checkout, setCheckout] = useState(false);
13
```

Loans.tsx - react-library - Visual Studio Code

ReviewListPage.tsx 1 App.tsx Navbar.tsx ShelfCurrentLoans.ts Loans.tsx 3, M X

src > layouts > ShelfPage > components > Loans.tsx > ...

```
7 // okta authentication and http error useState
8 const { authState } = useOktaAuth();
9 const [httpError, setHttpError] = useState(null);
10 // Current Loans
11 any
12 const [isLoadingUserLoans, setIsLoadingUserLoans] = useState<ShelfCurrentLoans>(true);
13 Cannot find name 'useEffect'. ts(2304) setIsLoadingUserLoans(false);
14 View Problem (Alt+F8) Quick Fix... (Ctrl+.)
15 useEffect(()=>{
16     const fetchUserCurrentLoans = async () => {
17         ...
18     }
19
20     fetchUserCurrentLoans().catch((error: any) => {
21         setIsLoadingUserLoans(false);
22         setHttpError(error.message);
23     })
24 }
25 return(
```

Step 6 create useEffect structure, which contains async function fetchUserCurrentLoans and its error catch function.

Import useEffect :

Hover over the useEffect, choose “Quick Fix”.

Click on “Update import from react”.

Loans.tsx - react-library - Visual Studio Code

ReviewListPage.tsx 1 App.tsx Navbar.tsx ShelfCurrentLoans.ts Loans.tsx 4, M

src > layouts > ShelfPage > components > Loans.tsx > [e] Loans > [D] useEffect() callback

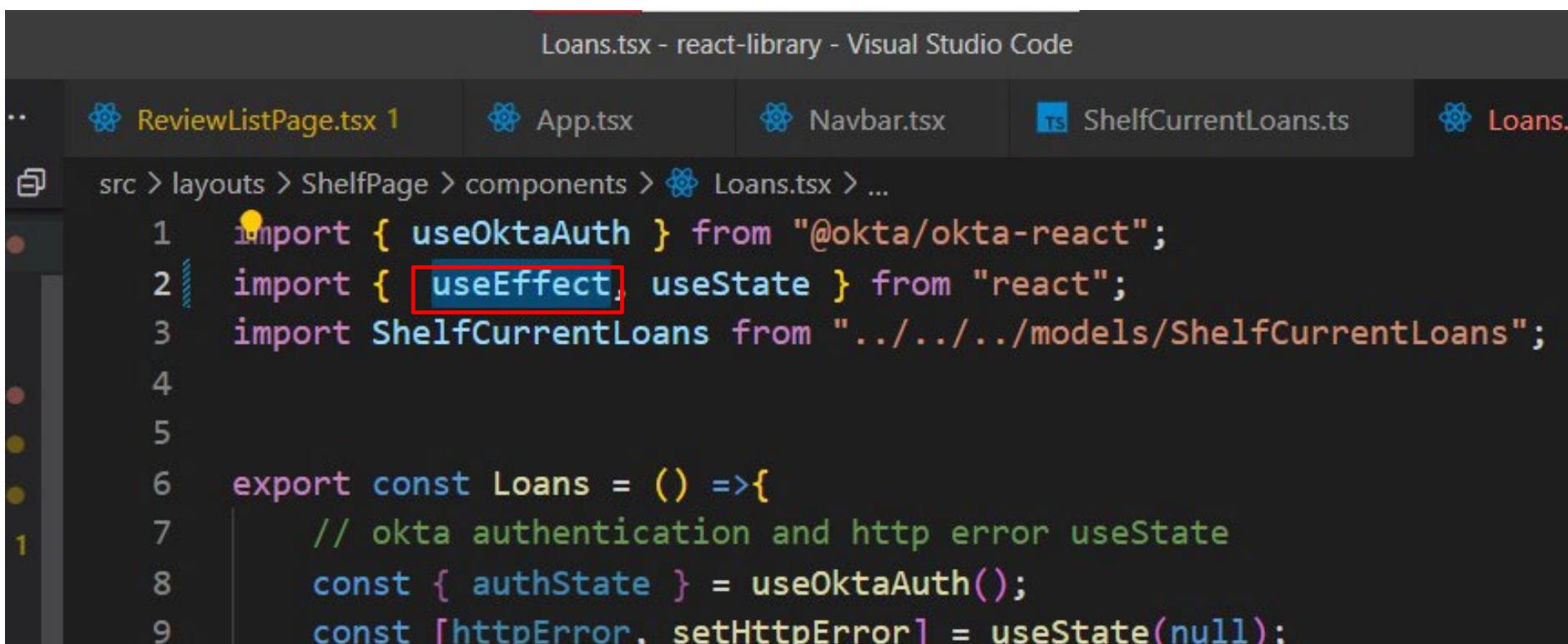
```
7 // okta authentication and http error useState
8 const { authState } = useOktaAuth();
9 const [httpError, setHttpError] = useState(null);
10 // Current Loans
11 const [shelfCurrentLoans, setShelfCurrentLoans] = useState<ShelfCur
12 const [isLoadingUserLoans, setIsLoadingUserLoans] = useState(true);
13 const [checkout, setCheckout] = useState(false);
14
15 useEffect(()=>{
16     const fetchUs
17
18 }
19
20 fetchUserCurrentLoans().catch(error: any) => {
```

Quick Fix...

- Update import from "react"
- Add missing function declaration 'useEffect'

useEffect is imported now.

Loans.tsx - react-library - Visual Studio Code



```
1 import { useOktaAuth } from "@okta(okta-react";
2 import { useEffect, useState } from "react";
3 import ShelfCurrentLoans from "../../../../../models/ShelfCurrentLoans";
4
5
6 export const Loans = () =>{
7     // okta authentication and http error useState
8     const { authState } = useOktaAuth();
9     const [httpError, setHttpError] = useState(null);
```

```
4
5  export const Loans = () =>{
6    // okta authentication and http error useState
7    const { authState } = useOktaAuth();
8    const [httpError, setHttpError] = useState(null);
9    // Current Loans
10   const [shelfCurrentLoans, setShelfCurrentLoans] = useState([]);
11   const [isLoadingUserLoans, setIsLoadingUserLoans] = useState(true);
12   const [checkout, setCheckout] = useState(false);
13
14   useEffect(()=>{
15     const fetchUserCurrentLoans = async () => {
16
17       }
18
19       fetchUserCurrentLoans().catch(error: any) => {
20         setIsLoadingUserLoans(false);
21         setHttpError(error.message);
22       }
23       //each time this use effect is called, scroll to the top
24       window.scrollTo(0, 0);
25   },[authState])
26
27   return()
```

Step 7

7.1 Add “window.scrollTo(0, 0)” to after the error catch code; So each time we open the page, it will jump to the top of the page.

7.2 add “authState” to the end array of useEffect as a change factor.



11. REACT LOANS USEFFECT



src > layouts > ShelfPage > components > Loans.tsx > Loans > useEffect() callback > fetchUserCurrentLoans

```
14  useEffect(()=>{
15      const fetchUserCurrentLoans = async () => [
16          //if user is authenticated,
17          if (authState && authState.isAuthenticated) {
18
19              const url = `http://localhost:8080/api/books/secure/currentloans`;
20              const requestOptions = {
21                  method: 'GET',
22                  headers: {
23                      Authorization: `Bearer ${authState.accessToken?.accessToken}`,
24                      'Content-Type': 'application/json'
25                  }
26              };
27              //fetch data with url and request url and request headers
28              const shelfCurrentLoansResponse = await fetch(url, requestOptions);
29              //if api call fails
30              if (!shelfCurrentLoansResponse.ok) {
31                  throw new Error('Something went wrong!');
32              }
33              //convert to json object
34              const shelfCurrentLoansResponseJson = await shelfCurrentLoansResponse.json();
35              setShelfCurrentLoans(shelfCurrentLoansResponseJson);
36          }
37          // fetching data finished, turn off the loading process
38          setIsLoadingUserLoans(false);
39      }
40  }
```

Step 1 fill in the
fetchUserCurrentLoans
async function to fetch
data.

```
src > layouts > ShelfPage > components > Loans.tsx > Loans
38         setIsLoadingUserLoans(false);
39     }
40
41     fetchUserCurrentLoans().catch((error: any) => {
42         setIsLoadingUserLoans(false);
43         setHttpError(error.message);
44     })
45     //each time this use effect is called, scroll to the top
46     window.scrollTo(0, 0);
47 ,[authState])
48
49 if (isLoadingUserLoans) {
50     return (
51         <SpinnerLoading/>
52     );
53 }
54
55 if (httpError) {
56     return (
57         <div className='container m-5'>
58             <p>
59                 {httpError}
60             </p>
61         </div>
62     );
63 }
```

Step 2 add SpinnerLoading and httpError code.

12. REACT LOANS COMPONENT HTML/CSS

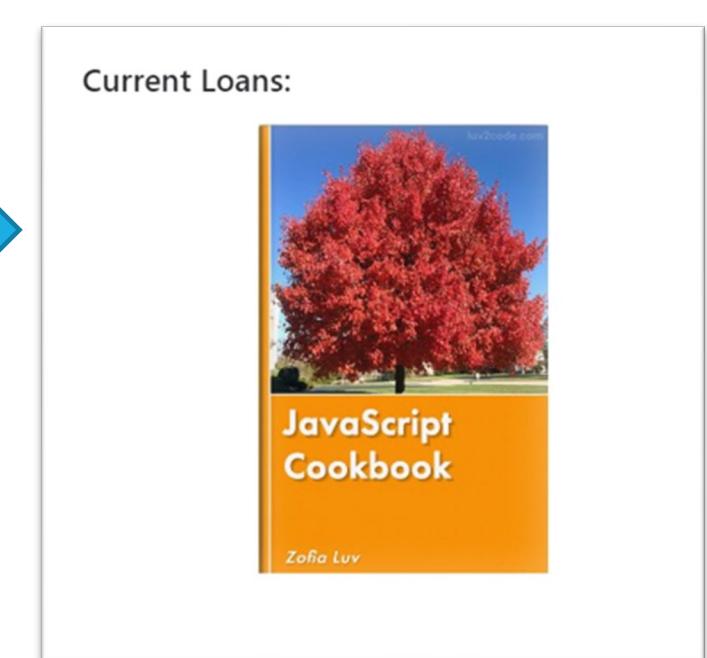


```
//html and css
return (
  <div>
    {/* Desktop */}

    <div className="d-none d-lg-block mt-2">
      {shelfCurrentLoans.length > 0 ? (
        <>
        {/*user has borrowed books*/}
        <h5>Current Loans: </h5>

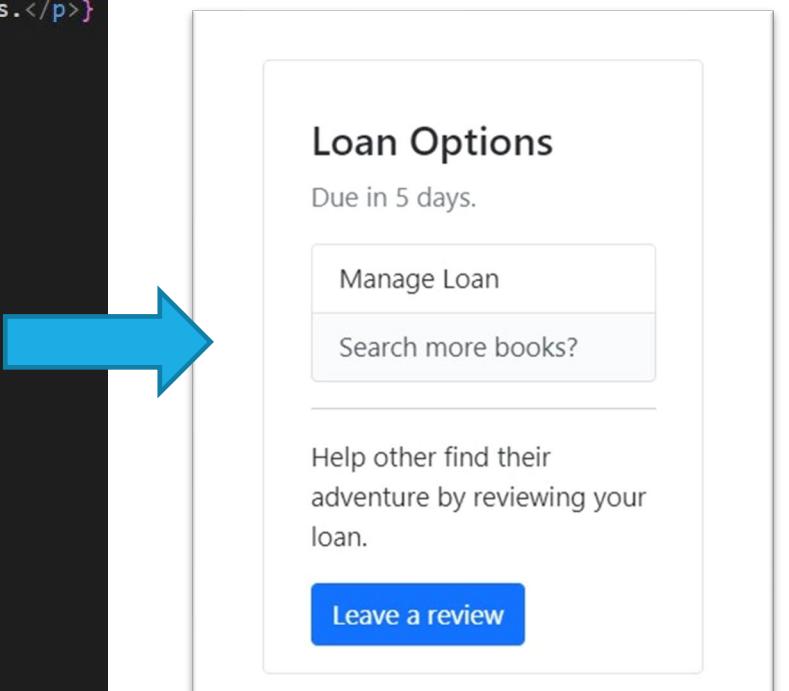
        {shelfCurrentLoans.map((shelfCurrentLoan) => [
          <div key={shelfCurrentLoan.book.id}>
            <div className="row mt-3 mb-3">
              {/* book image part */}
              <div className="col-4 col-md-4 container">
                {shelfCurrentLoan.book?.img ?
                  // image of current borrowed book
                  <img src={shelfCurrentLoan.book?.img} width="226" height="349" alt="Book" />
                ) : (
                  //default image
                  <img src={require("../../../../Images/BooksImages/book-luv2code-1000.png")}>
                  width="226" height="349" alt="Book" />
                )
              </div>
            </div>
            <div className="card col-3 col-md-3 container d-flex">
              <div className="card-body">
                <div className="mt-2">
```

Step 1 image part.



```
</div>
<div className="card col-3 col-md-3 container d-flex">
  <div className="card-body">
    <div className="mt-3">
      <h4>Loan Options</h4>
      {/* present different messages based on how many days are left for that current loan. */}
      {shelfCurrentLoan.daysLeft > 0 && <p className="text-secondary">Due in {shelfCurrentLoan.daysLeft} days.</p>}
      {shelfCurrentLoan.daysLeft === 0 && <p className="text-success">Due Today.</p>}
      {shelfCurrentLoan.daysLeft < 0 && <p className="text-danger">Past due by {shelfCurrentLoan.daysLeft} days.</p>}
      <div className="list-group mt-3">
        <button
          className="list-group-item list-group-item-action"
          aria-current="true"
          data-bs-toggle="modal"
          data-bs-target={`#modal${shelfCurrentLoan.book.id}`}
        >
          Manage Loan
        </button>
        <Link to={"search"} className="list-group-item list-group-item-action">
          Search more books?
        </Link>
      </div>
      <hr />
      <p className="mt-3">Help other find their adventure by reviewing your loan.</p>
      <Link className="btn btn-primary" to={`/checkout/${shelfCurrentLoan.book.id}`}>
        Leave a review
      </Link>
    </div>
  </div>
</div>
```

Step2 card part:
loans option,manage
loan, leave a review



```
118      //>
119    ) : (
120      // user has not borrowed books yet
121      <>
122        <h3 className="mt-3">Currently no loans</h3>
123        <Link className="btn btn-primary" to={`search`}>
124          | Search for a new book
125        </Link>
126      </>
127    )
128  </div>
129</div>
130);
```

Step 3 if current user had no loan yet.

Step 4 , go to ShelfPage.tsx, replace <p>loans with loans component.
And import loans component.

The image shows a code editor interface with three main sections:

- File Tree:** On the left, it shows a project structure with folders like NavbarAndFooter, SearchBooksPage, and ShelfPage. Under ShelfPage, there are components and files Loans.tsx and ShelfPage.tsx. The Loans.tsx file is currently selected.
- Top Editor:** This panel displays the content of the ShelfPage.tsx file. A specific line of code, <p> Loans </p>, is highlighted with a red box. A large blue arrow points downwards from this box towards the second code editor.
- Bottom Editor:** This panel shows the same ShelfPage.tsx file after the update. The line <p> Loans </p> has been replaced by the component <Loans/>, which is also highlighted with a red box.
- Bottom Left:** A status bar at the bottom left indicates the current file path: src > layouts > ShelfPage > ShelfPage.tsx. It also shows the line numbers 1, 2, and 3, corresponding to the code lines.

```
src > layouts > ShelfPage > ShelfPage.tsx > ShelfPage
1 import { Loans } from "./components/Loans";
2
3 export const ShelfPage = () => {
```

Final shelf page :

React App X +

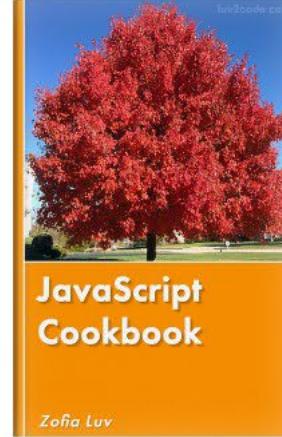
localhost:3000/shelf

Radio-Canada.ca |... Canada's federal... Sign in to your IR... Chenelière Éducat... Montreal zdm Nos incontournab... TED TED: Ideas worth... 在线课程 - 时间自...

JAC Read Home Search Books Shelf

Loans Your History

Current Loans:



JavaScript
Cookbook

Zofia Luv

Loan Options

Due in 5 days.

Manage Loan

Search more books?

Help other find their adventure by reviewing your loan.

Leave a review

13. REACT LOANS COMPONENT MOBILE HTML/CSS



The screenshot shows a code editor with a sidebar containing a project tree for a 'REACT-LIBRARY'. The tree includes 'src' (with 'Auth', 'Images', 'layouts' (containing 'BookCheckoutPage', 'ReviewListPage', 'ReviewListPage.tsx', 'BookCheckoutPage.tsx', 'CheckoutAndReviewBox.tsx', 'LatestReviews.tsx'), 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', 'ShelfPage' (containing 'components', 'Loans.tsx', 'ShelfPage.tsx'), 'Utils', 'lib', and 'models' (containing 'BookModel.ts', 'ReviewModel.ts', 'ReviewRequestModel.ts', 'ShelfCurrentLoans.ts'))), 'App.css', 'App.tsx', 'index.css', 'index.tsx', and 'react-app-env.d.ts'. The main panel displays the content of 'Loans.tsx'.

```
src > layouts > ShelfPage > components > Loans.tsx > [Loans]
```

```
58     </div>
59   );
60 }
61 //html and css
62 return (
63   <div>
64     /* Desktop */
65     <div className="d-none d-lg-block mt-2">
66       {shelfCurrentLoans.length > 0 ? (
67         <>
68           {/*user has borrowed books*/}
69           <h5>Current Loans:</h5>
70           {shelfCurrentLoans.map((shelfCurrentLoan) => (
71             <div key={shelfCurrentLoan.book.id}>...
72             </div>
73           )));
74         </>
75       ) : (
76         // user has not borrowed books yet
77         <>
78           <h3>Currently no loans</h3>
79           <Link className="btn btn-primary" to={`search`}>
80             Search for a new book
81           </Link>
82         </>
83       )
84     );
85   </div>
86   /* mobile version */
87 </div>
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 );
```

Step 1 copy desktop version code

```
layouts > ShelfPage > components > Loans.tsx > Loans
  ...
  </div>
  /* mobile version */
  <div className="d-none d-lg-block mt-2">
    {shelfCurrentLoans.length > 0 ? (
      <>
      {/*user has borrowed books*/}
      <h5>Current Loans: </h5>

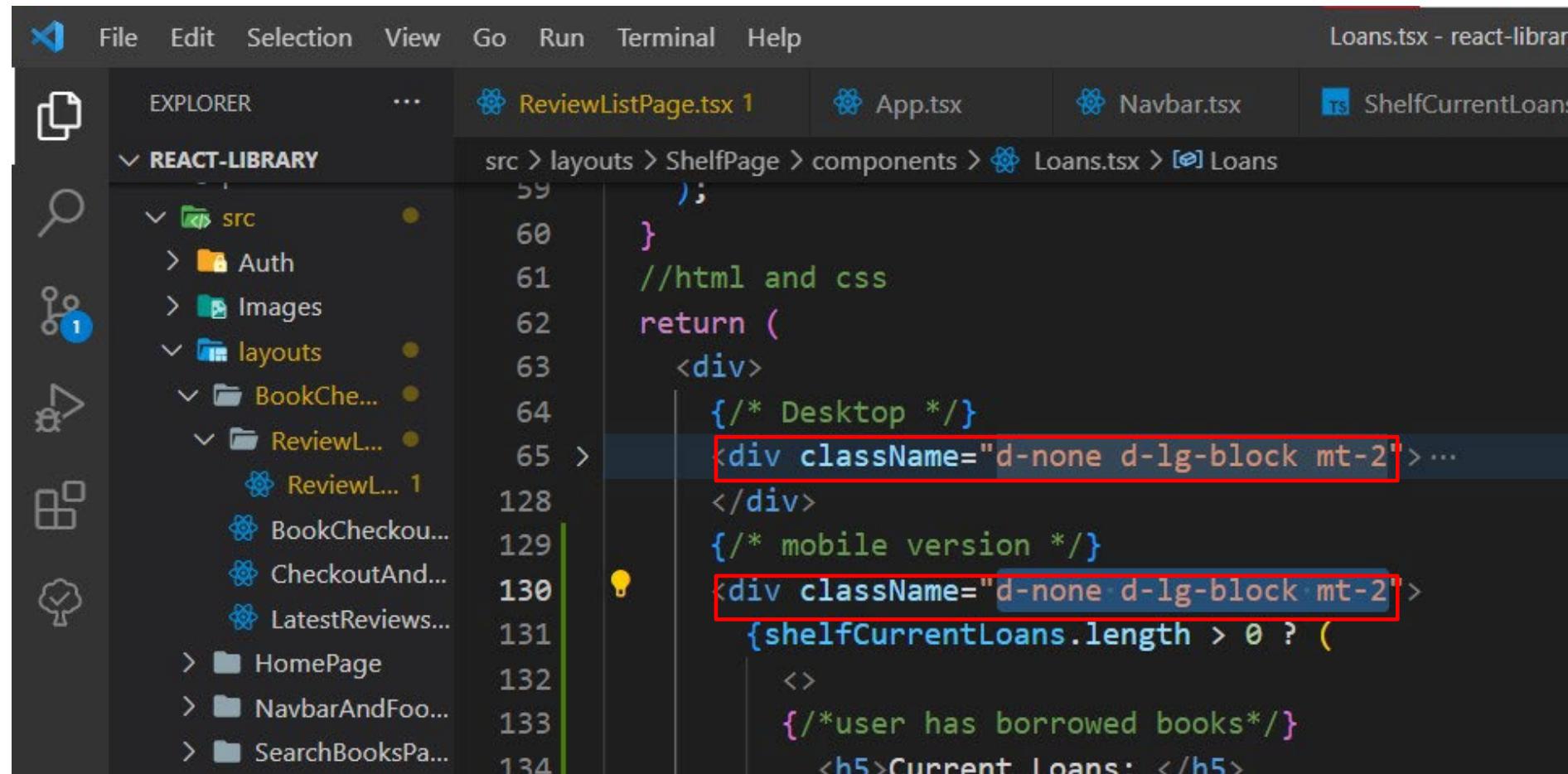
      {shelfCurrentLoans.map(shelfCurrentLoan) =>
        <div key={shelfCurrentLoan.book.id}>
          <div className="row mt-3 mb-3">
            {/* book image part */}
            <div className="col-4 col-md-4 container">
              {shelfCurrentLoan.book?.img ?
                // image of current borrowed book
                <img src={shelfCurrentLoan.book?.img} width="226" height="349" alt="Book" />
              ) : (
                //default image
                <img src={require("./../../../../Images/BooksImages/book-luv2code-1000.png")}
                  width="226" height="349" alt="Book" />
              )}
            </div>
            <div className="card col-3 col-md-3 container d-flex">
              <div className="card-body">
                <div className="mt-3">
                  <h4>Loan Options</h4>
                  {/* present different messages based on how many days are left for that cu
                  {shelfCurrentLoan.daysLeft > 0 && <p className="text-secondary">Due in {sh
                  {shelfCurrentLoan.daysLeft === 0 && <p className="text-success">Due Today.
                  {shelfCurrentLoan.daysLeft < 0 && <p className="text-danger">Past due by {
                  <div className="list-group mt-3">
                    <button
```

Step 2 paste it into our mobile section.

So as of right now, our desktop and our mobile section will be showing the exact same data.

Step 3, Change the CSS style in the className of mobile version.

3.1 the first parent div

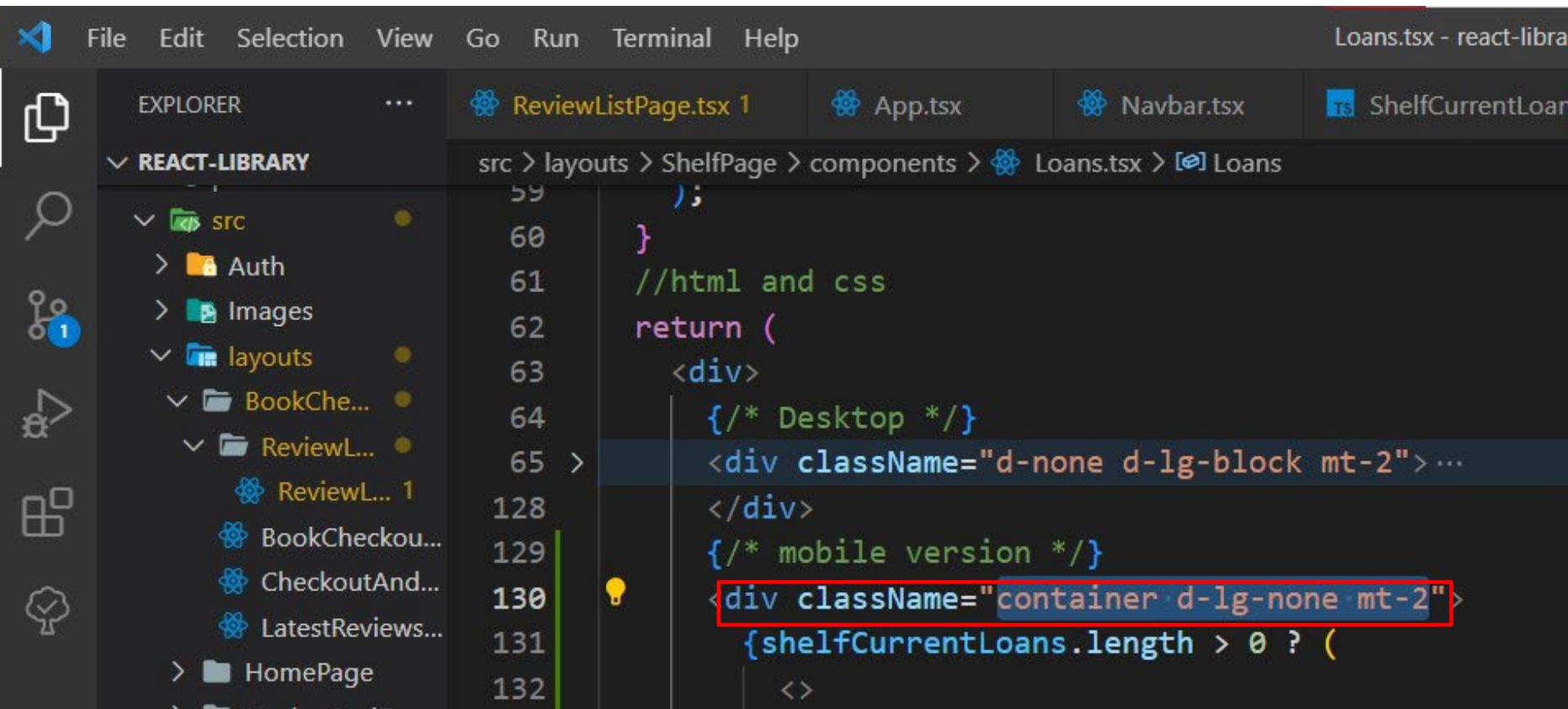


The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Loans.tsx - react-library.
- Explorer:** Shows the project structure under REACT-LIBRARY, including src, layouts, and various components like Auth, Images, BookCheck..., ReviewL..., and HomePage.
- Code Editor:** The Loans.tsx file is open. The code is a conditional rendering block:

```
      );
    }
    //html and css
    return (
      <div>
        /* Desktop */
        <div className="d-none d-lg-block mt-2">...
      </div>
      /* mobile version */
      <div className="d-none d-lg-block mt-2">
        {shelfCurrentLoans.length > 0 ? (
          <>
          /*user has borrowed books*/
          <h5>Current Loans: </h5>
        ) : (
          <h5>No Current Loans</h5>
        )}
      </div>
    )
  }
}
```
- Decorations:** Two specific lines of code are highlighted with red boxes: the desktop class at line 65 and the mobile class at line 130.

Change it to “container d-lg-none mt-2”



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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Loans.tsx - react-libra
- Explorer:** Shows the project structure under REACT-LIBRARY:
 - src
 - Auth
 - Images
 - layouts
 - BookChe...
 - ReviewL...
 - ReviewListPage.tsx
 - BookCheckou...
 - CheckoutAnd...
 - LatestReviews...
 - HomePage
 - Model
- Code Editor:** The Loans.tsx file is open. The code is as follows:

```
59      );
60    }
61    //html and css
62    return (
63      <div>
64        /* Desktop */
65        <div className="d-none d-lg-block mt-2">...
66      </div>
67      /* mobile version */
68      <div className="container d-lg-none mt-2">
69        {shelfCurrentLoans.length > 0 ? (
70          <>
```
- Status Bar:** Shows 1 error.

3.2 add className “mb-3” for the <h5> Current Loans</h5>

```
129     /* mobile version */
130     <div className="container d-lg-none mt-2">
131       {shelfCurrentLoans.length > 0 ? (
132         <>
133         /*user has borrowed books*/
134         <h5>Current Loans:</h5>
135       )
136       {shelfCurrentLoans.map((shelfCurrentLoan) =>
```

```
128   </div>
129   /* mobile version */
130   <div className="container d-lg-none mt-2">
131     {shelfCurrentLoans.length > 0 ? (
132       <>
133       /*user has borrowed books*/
134       <h5 className='mb-3'>Current Loans:</h5>
135     )
```

3.3 . delete <div className="row mt-3 mb-3">

The image shows a code editor with two code snippets. The top snippet is the original code, and the bottom snippet is the code after the deletion. A large blue arrow points from the deleted code in the top snippet down to the bottom snippet.

Original Code (Top Snippet):

```
136     {shelfCurrentLoans.map(shelfCurrentLoan) => [
137       <div key={shelfCurrentLoan.book.id}>
138         <div className="row mt-3 mb-3"> ← Delete
139           /* book image part */
140         </div>
141       <div className="card col-3 col-md-3 container d-flex" ...
142         </div>
143       </div>
144     ]}
```

Modified Code (Bottom Snippet):

```
135
136     {shelfCurrentLoans.map(shelfCurrentLoan) => [
137       <div key={shelfCurrentLoan.book.id}>
138         /* book image part */
139       <div className="col-4 col-md-4 container" ...
```

3.4. change image div className to "d-flex justify-content-center align-items-center"

```
138     /* book image part */
139     <div className="col-4 col-md-4 container">
140       {shelfCurrentLoan.book?.img ? (
141         // image of current borrowed book
142         <img src={shelfCurrentLoan.book?.img} width="2
143       ) : (
```



```
138     /* book image part */
139     <div className="d-flex justify-content-center align-items-center">
140       {shelfCurrentLoan.book?.img ? (
141         // image of current borrowed book
142         <img src={shelfCurrentLoan.book?.img} width="226" height="349"
```

3.5.change card div's className to "card d-flex mt-5 mb-3"

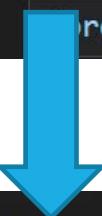
```
137 |         <div key={book.currentBook.id}>
138 |             /* book image part */
139 |         > <div className="d-flex justify-content-center align-items-cent
148 |             </div>
149 |         > <div className="card col-3 col-md-3 container d-flex">...
177 |
178 |             </div>
179 |         </div>
```



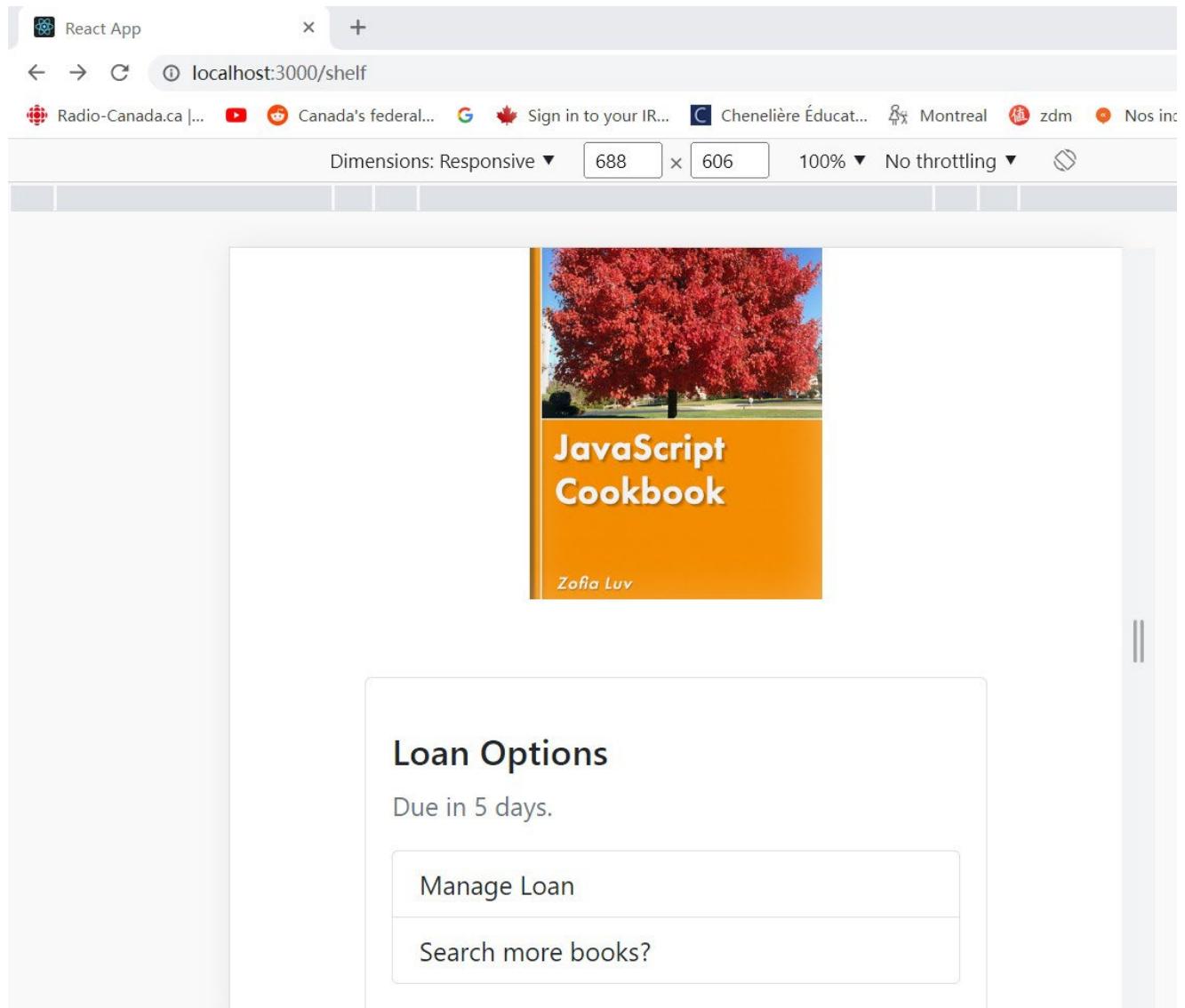
```
138 |             /* book image part */
139 |         > <div className="d-flex justify-content-center align-items-cen
148 |             </div>
149 |         > <div className="card d-flex mt-5 mb-3">...
177 |
178 |             </div>
179 |         </div>
```

3.6.change the “Manage Loan” button’s className from “modal” to “mobilemodal”

```
158 |           <button
159 |             className="list-group-item list-group-item-action"
160 |             aria-current="true"
161 |             data-bs-toggle="modal"
162 |             data-bs-target={`#modal${shelfCurrentLoan.book.id}`}}
163 |           >
164 |             Manage Loan
165 |           </button>
```



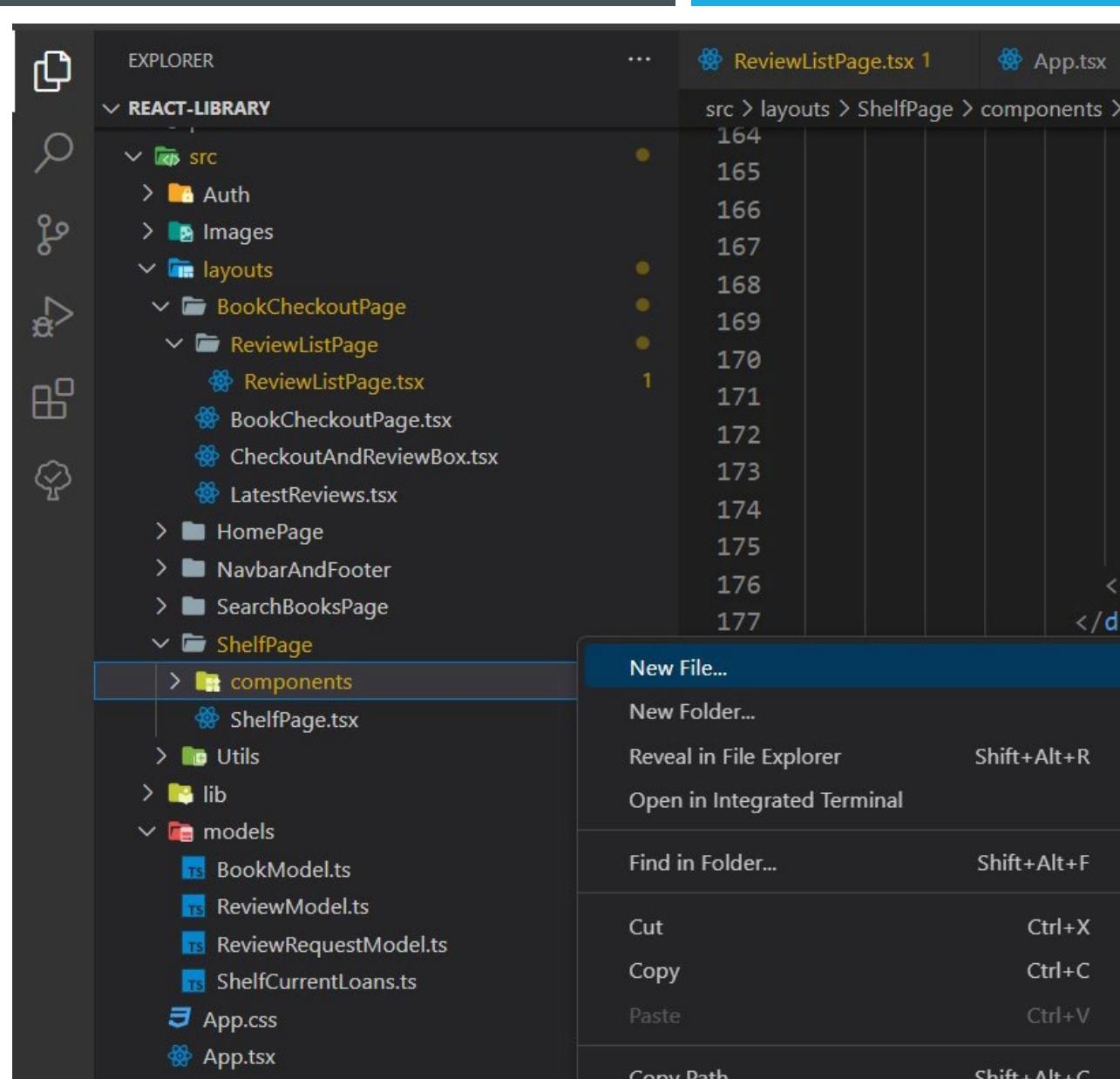
```
158 |           <button
159 |             className="list-group-item list-group-item-action"
160 |             aria-current="true"
161 |             data-bs-toggle="modal"
162 |             data-bs-target={`#mobilemodal${shelfCurrentLoan.book.id}`}}
163 |           >
164 |             Manage Loan
165 |           </button>
```



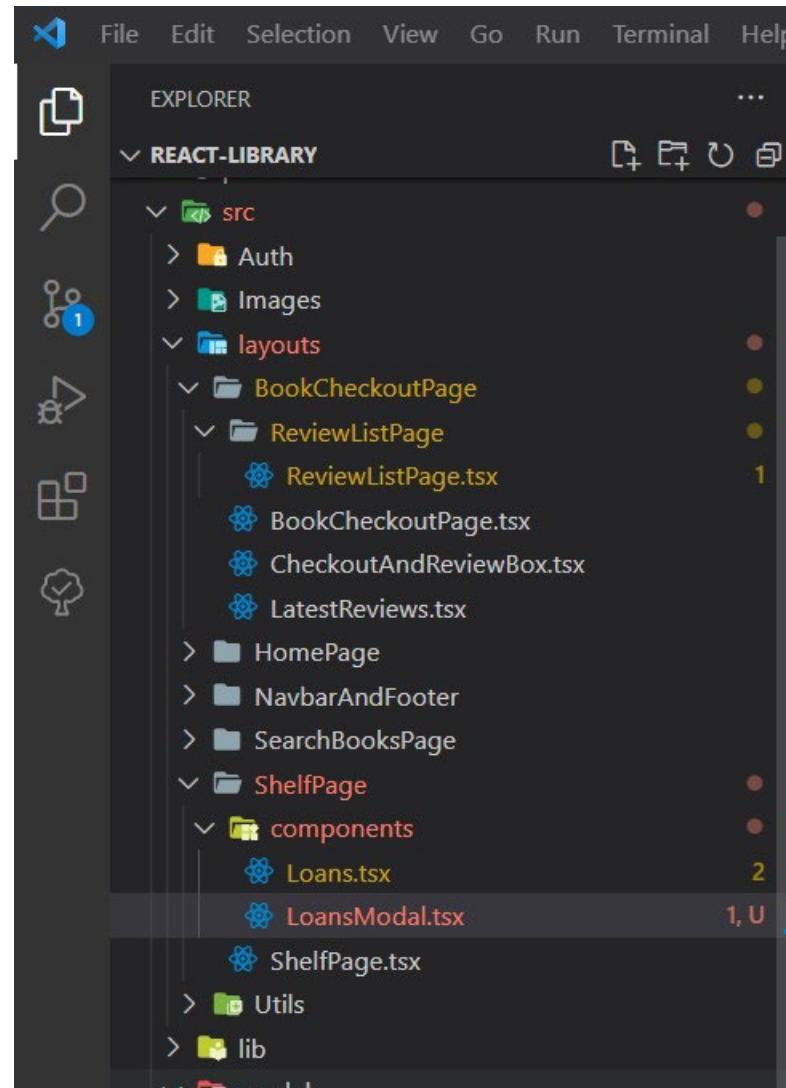
Refresh the page, and zoom out it, we can get a mobile version page.

14. REACT LOANS MODAL HTML/CSS



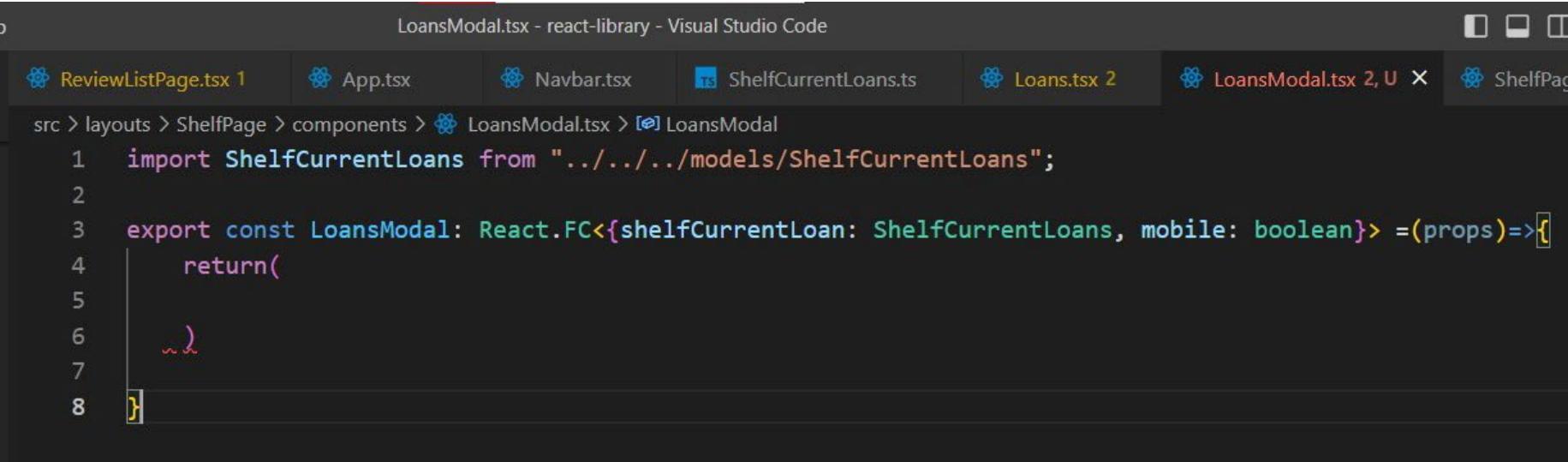


Step 1 Go to ShelfPage folder,
then components folder, right click
and choose “new file”.



name it `LoansModal.tsx`.

Step2 , create export const structure.



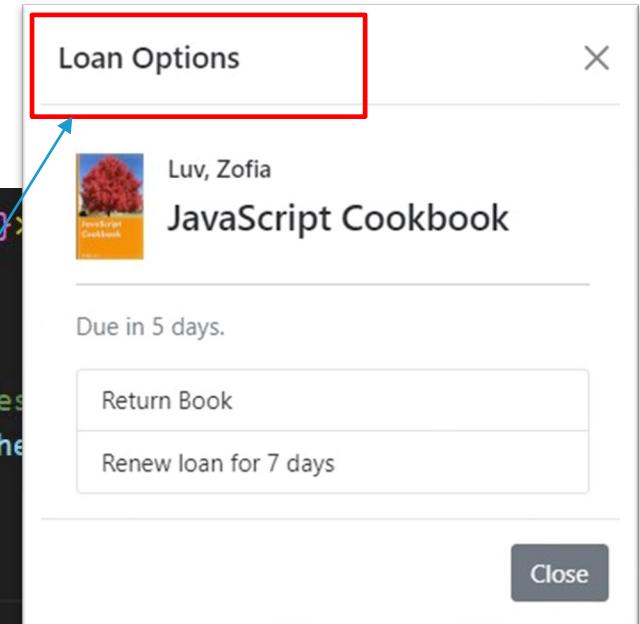
The screenshot shows a Visual Studio Code window with the title "LoansModal.tsx - react-library - Visual Studio Code". The tab bar includes "ReviewListPage.tsx 1", "App.tsx", "Navbar.tsx", "ShelfCurrentLoans.ts", "Loans.tsx 2", "LoansModal.tsx 2, U X", and "ShelfPag". The code editor displays the following TypeScript code:

```
1 import ShelfCurrentLoans from "../../../../../models/ShelfCurrentLoans";
2
3 export const LoansModal: React.FC<{shelfCurrentLoan: ShelfCurrentLoans, mobile: boolean}> =(props)=>{
4     return(
5
6     )
7
8 }
```

Step 3, create html and css in return part.

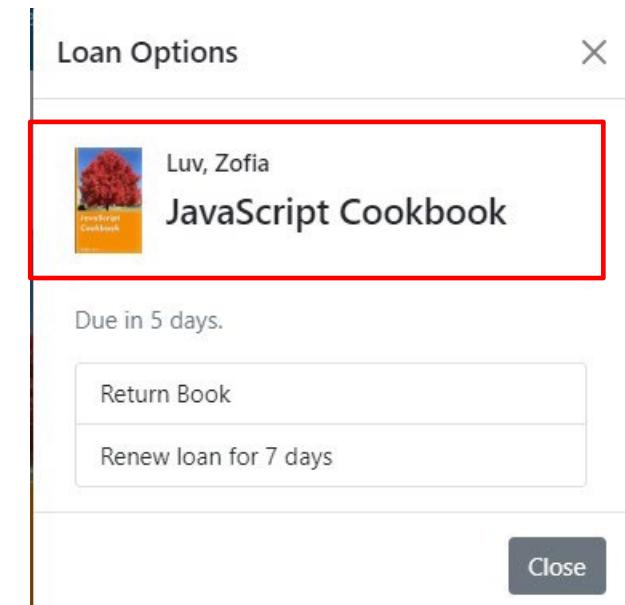
3.1 title part.

```
export const LoansModal: React.FC<{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean }> = () => {
  return (
    <div
      className="modal fade"
      // give this div a id, if it equals to the mobilemodal, show mobile version, else desktop
      id={props.mobile ? `mobilemodal${props.shelfCurrentLoan.book.id}` : `modal${props.shelfCurrentLoan.book.id}`}
      data-bs-backdrop="static"
      data-bs-keyboard="false"
      aria-labelledby="staticBackdropLabel"
      aria-hidden="true"
      key={props.shelfCurrentLoan.book.id}
    >
      <div className="modal-dialog">
        <div className="modal-content">
          <div className="modal-header">
            <h5 className="modal-title" id="staticBackdropLabel">
              Loan Options
            </h5>
            <button type="button" className="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
        </div>
      </div>
    </div>
  )
}
```



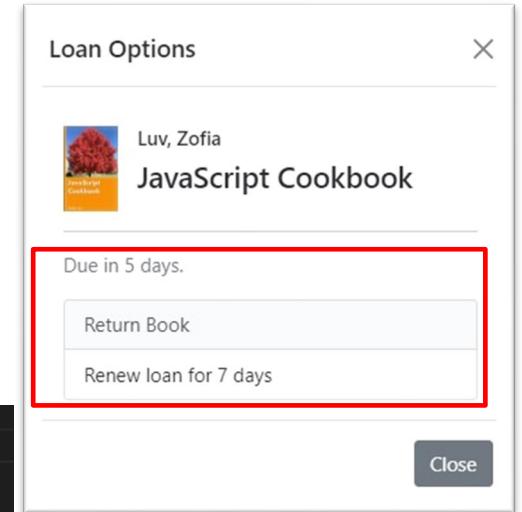
3.2 book info part.

```
</div>
<div className="modal-body">
  <div className="container">
    <div className="mt-3">
      <div className="row">
        {/* image part */}
        <div className="col-2">
          {props.shelfCurrentLoan.book?.img ? (
            //current borrowed book's image
            <img src={props.shelfCurrentLoan.book?.img} width="56" height="84" alt="Book cover image" />
          ) : (
            // default image
            <img src={require("./../../../../Images/BooksImages/book-luv2code.jpg")} width="56" height="84" alt="Default book cover image" />
          )}
        </div>
        <div className="col-10">
          {/* author and title part */}
          <h6>{props.shelfCurrentLoan.book.author}</h6>
          <h4>{props.shelfCurrentLoan.book.title}</h4>
        </div>
      </div>
    </div>
    <hr />
```

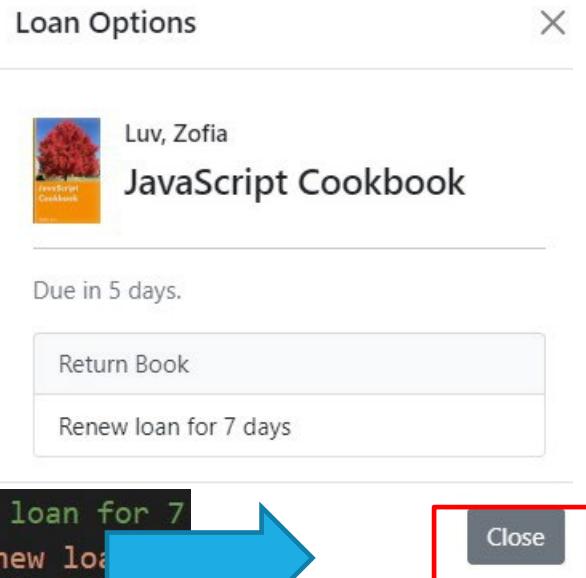


3.3 days reminder, return book, renew book part.

```
<hr />
/* present different messages based on how many days are left for that current loan. */
{props.shelfCurrentLoan.daysLeft > 0 && <p className="text-secondary">Due in {props.shelfCurrentLoan.daysLeft} days.</p>}
{props.shelfCurrentLoan.daysLeft === 0 && <p className="text-success">Due Today.</p>}
{props.shelfCurrentLoan.daysLeft < 0 && <p className="text-danger">Past due by {props.shelfCurrentLoan.daysLeft} days.</p>}
<div className="list-group mt-3">
  /* return book btn */
  <button data-bs-dismiss="modal" className="list-group-item list-group-item-action" aria-current="true">
    Return Book
  </button>
  /* return book btn */
  <button
    data-bs-dismiss="modal"
    className={
      //if book already due , cannot click on renew button
      props.shelfCurrentLoan.daysLeft < 0 ? "list-group-item list-group-item-action inactiveLink" : "list-group-item list-gr
    }
  >
    /* if book is already due, shows"Late dues cannot be renewed", else{Renew loan for 7 days}*/
    {props.shelfCurrentLoan.daysLeft < 0 ? "Late dues cannot be renewed" : "Renew loan for 7 days"}
  </button>
</div>
```



3.4 “close” button part.



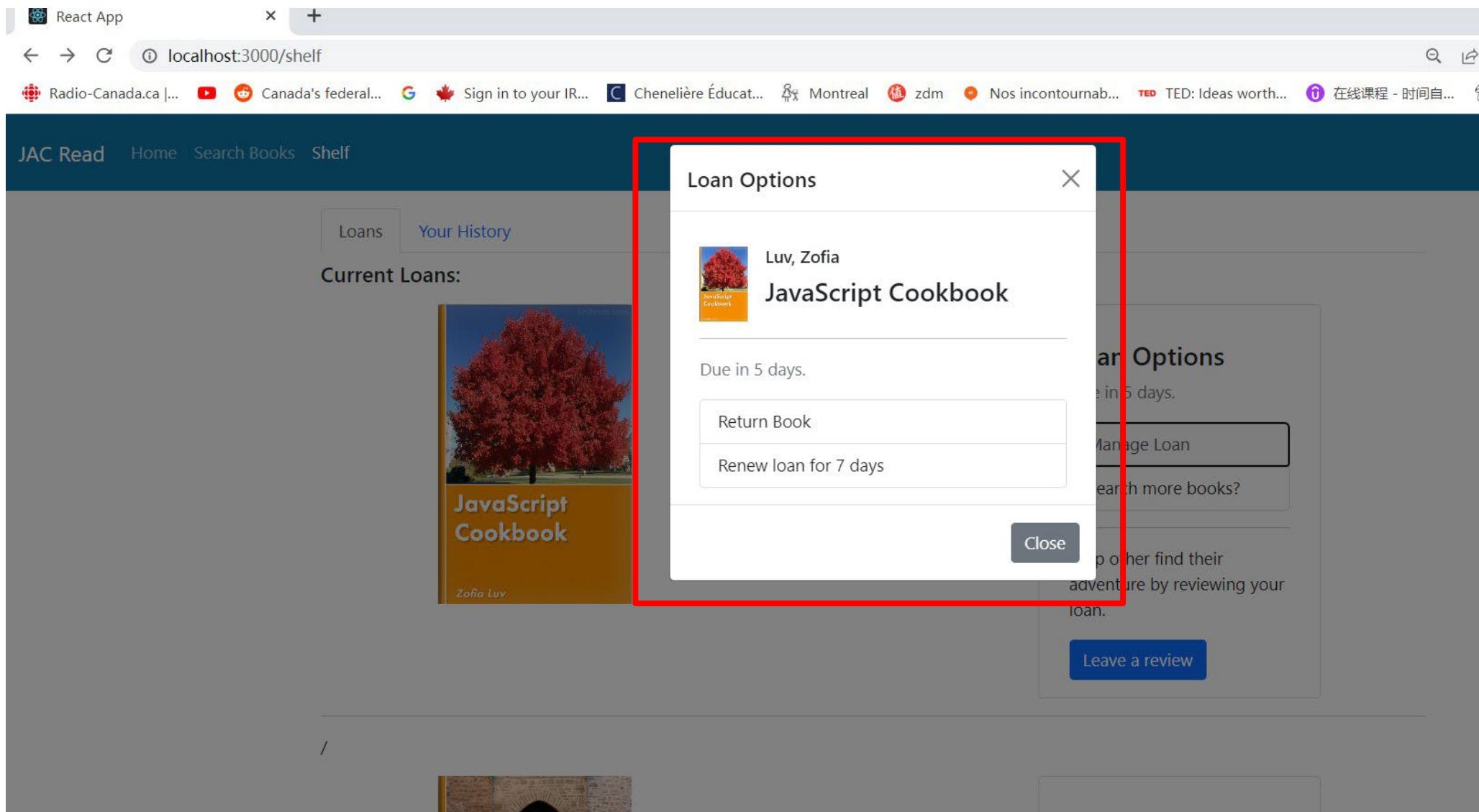
```
    /* if book is already due, shows "Late dues cannot be renewed", else {Renew loan for 7
     * daysLeft < 0 ? "Late dues cannot be renewed" : "Renew loan for " + daysLeft} */
    {props.shelfCurrentLoan.daysLeft < 0 ? "Late dues cannot be renewed" : "Renew loan for " + daysLeft}
  </button>
</div>
</div>
</div>
</div>
<div className="modal-footer">
  <button type="button" className="btn btn-secondary" data-bs-dismiss="modal">
    Close
  </button>
</div>
...

```

```
>           {shelfCurrentLoans.map((shelfCurrentLoan) => (
>             <div key={shelfCurrentLoan.book.id}>
>               <div className="row mt-3 mb-3">...
>                 </div>
>                 <hr />
>                 <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} //>
>               </div>
>           )));
>         ) : ( ...
>           )
>       </div>
>     /* mobile version */
>   <div className="container d-lg-none mt-2">
>     {shelfCurrentLoans.length > 0 ? (
>       <>
>       /*user has borrowed books*/
>       <h5 className='mb-3'>Current Loans: </h5>
>
>       {shelfCurrentLoans.map((shelfCurrentLoan) => (
>         <div key={shelfCurrentLoan.book.id}>
>           /* book image part */
>           <div className="d-flex justify-content-center align-items-center">...
>             </div>
>             <div className="card d-flex mt-5 mb-3">...
>               </div>
>               <hr />
>               <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={true} //>
>             </div>
>           );
>         ));
>       );
>     );
>   );
> 
```

Step 4 Go to Loans.tsx, add “`<LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} />`” to both version, but for the mobile version, change the value of mobile to “true”.

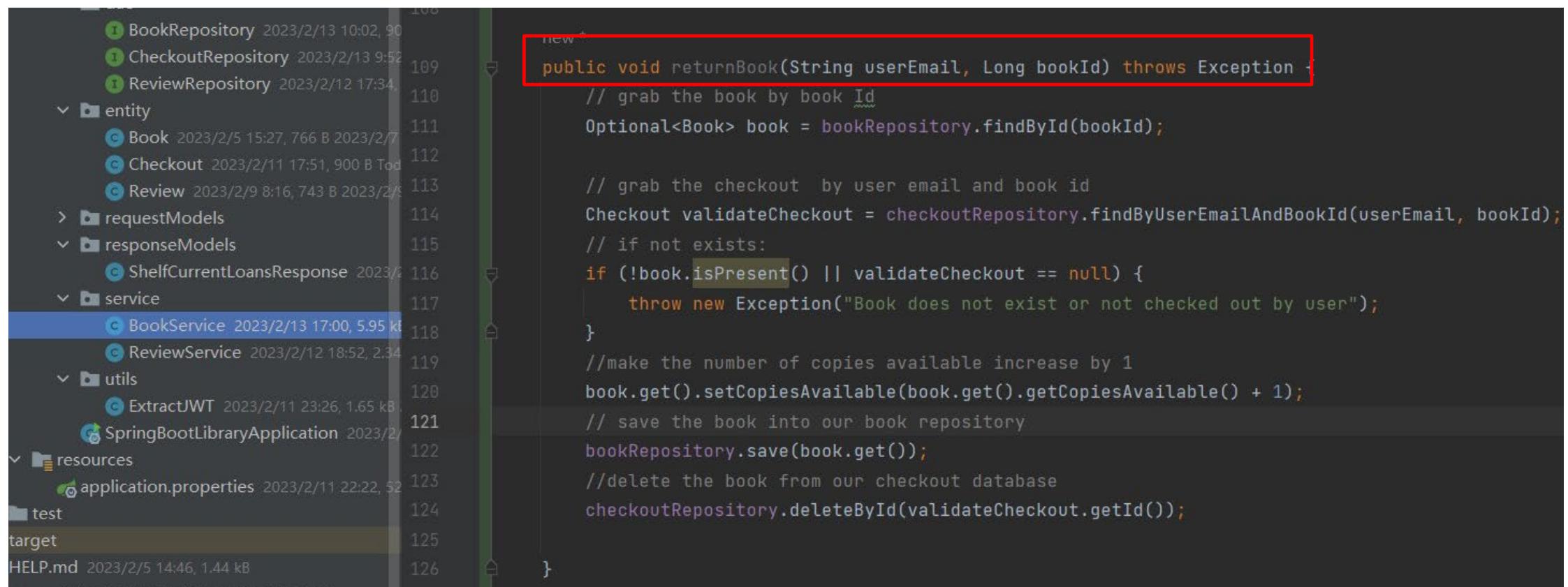
The final pop out dialogue box is as below:



15. SPRINGBOOT RETURN BOOK SERVICE AND ENDPOINT LAYER

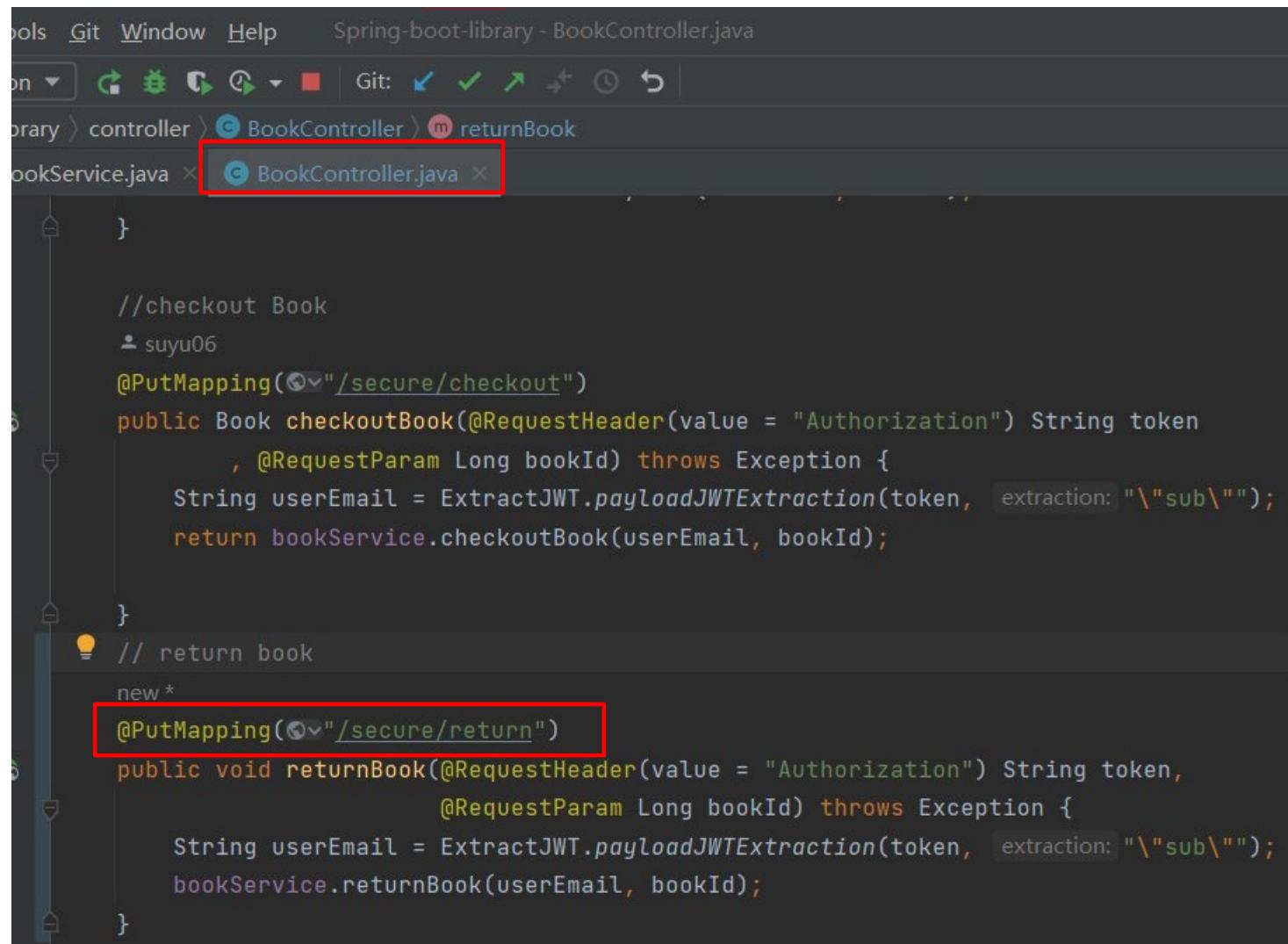


Step 1 Go into our service directory, Open up our BookService. create returnBook()



```
public void returnBook(String userEmail, Long bookId) throws Exception {
    // grab the book by book Id
    Optional<Book> book = bookRepository.findById(bookId);

    // grab the checkout by user email and book id
    Checkout validateCheckout = checkoutRepository.findByUserEmailAndBookId(userEmail, bookId);
    // if not exists:
    if (!book.isPresent() || validateCheckout == null) {
        throw new Exception("Book does not exist or not checked out by user");
    }
    //make the number of copies available increase by 1
    book.get().setCopiesAvailable(book.get().getCopiesAvailable() + 1);
    // save the book into our book repository
    bookRepository.save(book.get());
    //delete the book from our checkout database
    checkoutRepository.deleteById(validateCheckout.getId());
}
```



The screenshot shows a Java IDE interface with the following details:

- Toolbar:** Tools, Git, Window, Help.
- Git Status:** Green checkmarks indicating successful operations.
- Project Structure:** Spring-boot-library > controller > BookController > returnBook.
- Code Editor:** BookController.java
- Code Content:**

```
    }

    //checkout Book
    @PostMapping("/secure/checkout")
    public Book checkoutBook(@RequestHeader(value = "Authorization") String token
        , @RequestParam Long bookId) throws Exception {
        String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
        return bookService.checkoutBook(userEmail, bookId);
    }

    // return book
    new *
    @PostMapping("/secure/return")
    public void returnBook(@RequestHeader(value = "Authorization") String token,
        @RequestParam Long bookId) throws Exception {
        String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
        bookService.returnBook(userEmail, bookId);
    }
}
```

Two specific code snippets are highlighted with red boxes:

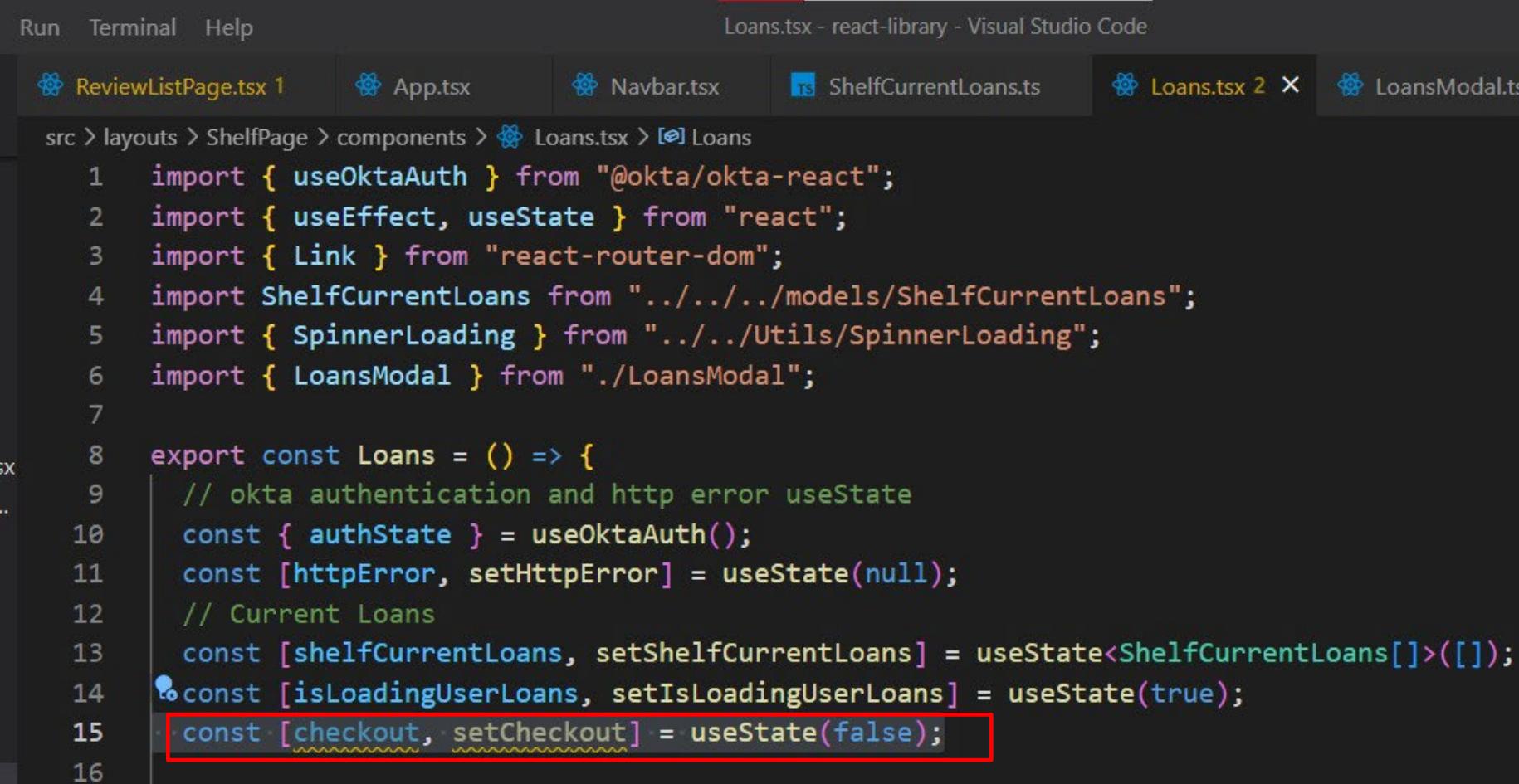
- `@PostMapping("/secure/checkout")`
- `@PostMapping("/secure/return")`

Step 2 Go into our controller package, then BookController, use “Put” method to create the endpoint for book return.

16. REACT RETURN BOOK



Step 1 Go to Loans.tsx, add useState.

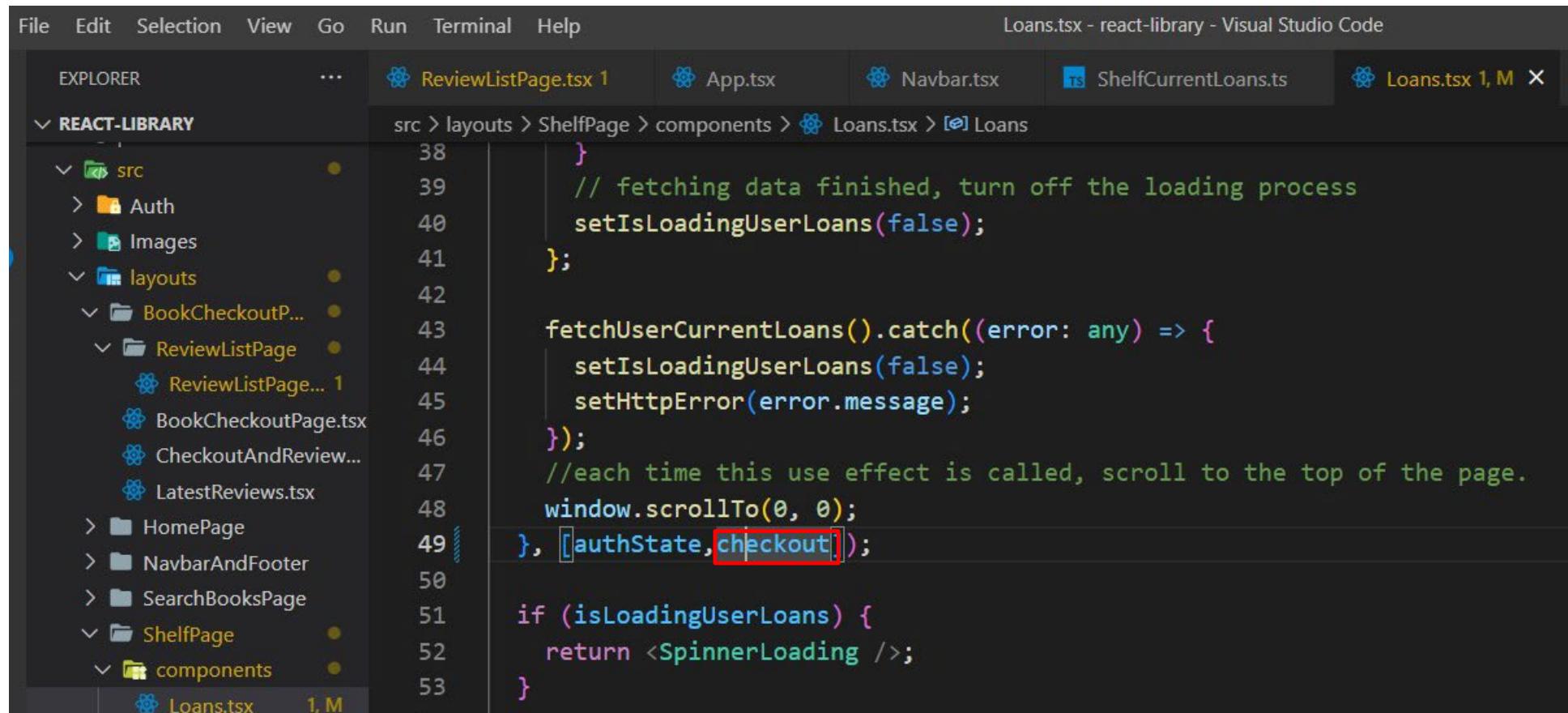


The screenshot shows a Visual Studio Code interface with the title bar "Loans.tsx - react-library - Visual Studio Code". The menu bar includes "Run", "Terminal", and "Help". The tab bar shows "ReviewListPage.tsx 1", "App.tsx", "Navbar.tsx", "ShelfCurrentLoans.ts", "Loans.tsx 2 X", and "LoansModal.ts". The code editor displays the "Loans.tsx" file with the following content:

```
src > layouts > ShelfPage > components > Loans.tsx > Loans
1 import { useOktaAuth } from "@okta(okta-react";
2 import { useEffect, useState } from "react";
3 import { Link } from "react-router-dom";
4 import ShelfCurrentLoans from "../../models/ShelfCurrentLoans";
5 import { SpinnerLoading } from "../../Utils/SpinnerLoading";
6 import { LoansModal } from "./LoansModal";
7
8 export const Loans = () => {
9   // okta authentication and http error useState
10  const { authState } = useOktaAuth();
11  const [httpError, setHttpError] = useState(null);
12  // Current Loans
13  const [shelfCurrentLoans, setShelfCurrentLoans] = useState<ShelfCurrentLoans[]>([]);
14  const [isLoadingUserLoans, setIsLoadingUserLoans] = useState(true);
15  const [checkout, setCheckout] = useState(false);
16
```

The line "const [checkout, setCheckout] = useState(false);" is highlighted with a red border.

Step2 , add checkout useState to the end of the array of useEffect.
So authState and checkout will be part of the state changer for this use effect.



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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Title Bar:** Loans.tsx - react-library - Visual Studio Code
- Explorer:** Shows the project structure under 'REACT-LIBRARY'. The 'src' folder contains 'Auth', 'Images', 'layouts', 'BookCheckoutP...', 'ReviewListPage' (which contains 'ReviewListPage... 1', 'BookCheckoutPage.tsx', 'CheckoutAndReview...', and 'LatestReviews.tsx'), 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', 'ShelfPage' (which contains 'components' and 'Loans.tsx'), and 'SearchBooksPage'.
- Code Editor:** The 'Loans.tsx' file is open. The code is as follows:

```
38     }
39     // fetching data finished, turn off the loading process
40     setIsLoadingUserLoans(false);
41   };

42   fetchUserCurrentLoans().catch(error: any) => {
43     setIsLoadingUserLoans(false);
44     setHttpError(error.message);
45   };
46   //each time this use effect is called, scroll to the top of the page.
47   window.scrollTo(0, 0);
48 }, [authState, checkout]);

49 if (isLoadingUserLoans) {
50   return <SpinnerLoading />;
51 }
52
53 }
```

The line '}, [authState, checkout]);' is highlighted with a red rectangle.

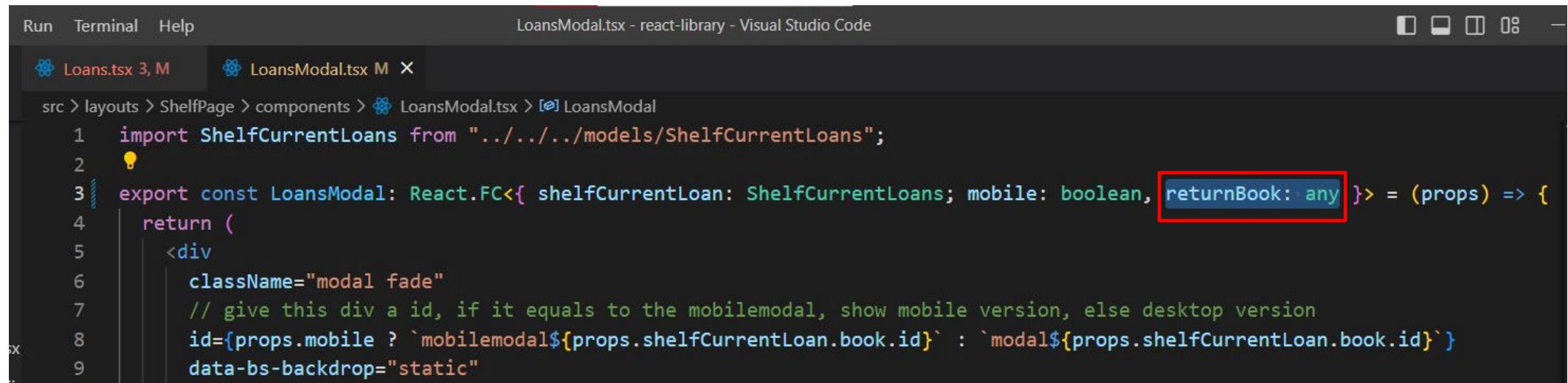
Step 3 create async function returnBook to fetch data from backend.

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

- File Explorer (Left):** Shows the project structure under "REACT-LIBRARY".
 - src
 - Auth
 - Images
 - layouts
 - BookCheckoutPage
 - ReviewListPage
 - ReviewListPage... 1
 - BookCheckoutPage.tsx
 - CheckoutAndReview...
 - LatestReviews.tsx
 - HomePage
 - NavbarAndFooter
 - SearchBooksPage
 - ShelfPage
 - components
 - Loans.tsx 1, M
 - LoansModal.tsx
 - ShelfPage.tsx
 - Utils
 - lib
 - models- Code Editor (Right):** The file Loans.tsx is open, showing the following code:

```
59     </div>
60   );
61 }
62 async function returnBook(bookId: number) {
63   const url = `http://localhost:8080/api/books/secure/return/?bookId=${bookId}`;
64   const requestOptions = {
65     method: "PUT",
66     headers: {
67       Authorization: `Bearer ${authState?.accessToken?.accessToken}`,
68       "Content-Type": "application/json",
69     },
70   };
71   //fetch the data using url and request url and headers
72   const returnResponse = await fetch(url, requestOptions);
73   //if failure
74   if (!returnResponse.ok) {
75     throw new Error("Something went wrong!");
76   }
77   // change the checkout state
78   setCheckout(!checkout);
79 }
```
- Status Bar (Bottom):** Shows the date "2023/2/20".

Step 4 Go to LoansModal.tsx, add return book a new props.



```
Run Terminal Help
LoansModal.tsx - react-library - Visual Studio Code
Loans.tsx 3, M LoansModal.tsx M X
src > layouts > ShelfPage > components > LoansModal.tsx > LoansModal
1 import ShelfCurrentLoans from "../../models/ShelfCurrentLoans";
2
3 export const LoansModal: React.FC<{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean, returnBook: any }> = (props) => {
4   return (
5     <div
6       className="modal fade"
7       // give this div a id, if it equals to the mobilemodal, show mobile version, else desktop version
8       id={props.mobile ? `mobilemodal${props.shelfCurrentLoan.book.id}` : `modal${props.shelfCurrentLoan.book.id}`}
9       data-bs-backdrop="static"
```

Step 5 Go to Loans.tsx, fix LoansModal import error.

```
</div>
<hr />
<LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} />
</div>
))>
</>
) : ( ...
)
</div>
/* mobile version */
<div className="container d-lg-none mt-2">
{shelfCurrentLoans.length > 0 ? (
<>
  /*us (alias) const LoansModal: React.FC<{
    shelfCurrentLoan: ShelfCurrentLoans;
    mobile: boolean;
    [shel returnBook: any;
    <di >
      import LoansModal
    <
      Property 'returnBook' is missing in type '{ shelfCurrentLoan: ShelfCurrentLoans; mobile: true; }' but required in type '{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean; returnBook: any; }'. ts(2741)
      <
        LoansModal.tsx(3, 91): 'returnBook' is declared here.
      <
        <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={true} />
      <

```

Add returnBook={returnBook} to LoansModal properties.

```
133      </div>
134      <hr />
135      <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} returnBook={returnBook} />
136    </div>
137  ))}
138  </>
139 >  ) : ( ...
140   )
141   </div>
142   /* mobile version */
143   <div className="container d-lg-none mt-2">
144     {shelfCurrentLoans.length > 0 ? (
145       <>
146         /*user has borrowed books*/
147         <h5 className="mb-3">Current Loans: </h5>
148
149         {shelfCurrentLoans.map((shelfCurrentLoan) => (
150           <div key={shelfCurrentLoan.book.id}>
151             /* book image part */
152             <div className="d-flex justify-content-center align-items-center">...
153             </div>
154             <div className="card d-flex mt-5 mb-3">...
155             </div>
156             <hr />
157             <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={true} returnBook={returnBook} />
158           </div>
159         )})
160       ) : (
161         <div>
162           <h1>No Loans</h1>
163           <p>You have no loans at the moment.</p>
164           <hr />
165           <button onClick={handleLogout} type="button" style={buttonStyle}>Logout</button>
166         </div>
167       )
168     )
169   )
170 >  )}
```

(local function) return
Promise<void>

Step 6 Go to LoansModal.tsx, add onClick listener to “return book” btn.

```
/* present different messages based on how many days are left for that current loan. */
{props.shelfCurrentLoan.daysLeft > 0 && <p className="text-secondary">Due in {props.shelfCurrentLoan.daysLeft} days</p>}
{props.shelfCurrentLoan.daysLeft === 0 && <p className="text-success">Due Today.</p>}
{props.shelfCurrentLoan.daysLeft < 0 && <p className="text-danger">Past due by {props.shelfCurrentLoan.daysLeft}</p>}
<div className="list-group mt-3">
  /* return book btn */
  <button onClick={() => props.returnBook(props.shelfCurrentLoan.book.id)}>
    data-bs-dismiss="modal" className="list-group-item list-group-item-action" aria-current="true">
      Return Book
    </button>
  </div>
```

With the status of logged in, open our shelf page, try to manage our shelf.

Screenshot of a web browser window showing a library application interface at localhost:3000/shelf.

The browser header shows:

- React App
- x +
- localhost:3000/shelf

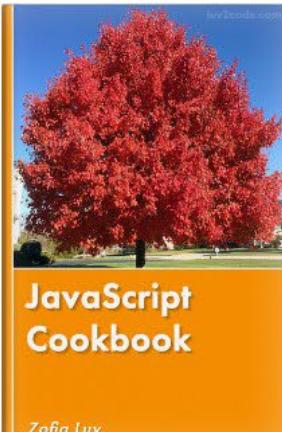
The page header includes:

- Radios (CBC, Canada's federal...)
- Sign in to your IR...
- Chenelière Éducat...
- Montreal
- zdm
- Nos incontournab...
- TED TED: Ideas worth...
- Unsubscribe

The main navigation bar has links for JAC Read, Home, Search Books, and Shelf. The Shelf link is currently active.

The current Loans section shows one item:

Current Loans:



Loan Options

Due in 5 days.

Manage Loan

Search more books?

Help other find their adventure by reviewing your loan.

Leave a review

The user has now borrowed 2 books: bookId 1 and bookId 2.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the `reactlibrarydatabase` database expanded, revealing tables like `book`, `checkout`, `history`, `messages`, and `review`.
- SQL Editor:** Contains the SQL query `SELECT * FROM reactlibrarydatabase.checkout;`
- Result Grid:** Displays the results of the query in a table format.

	id	user_email	checkout_date	return_date	book_id
▶	1	testuser2@email.com	2023-02-11	2023-02-18	1
▶	2	testuser2@email.com	2023-02-12	2023-02-19	2
*	NULL	NULL	NULL	NULL	NULL

- Status Bar:** Shows the tab `checkout 1` and the output pane.

Now, if we click on “Return book” in the pop out window,

The screenshot shows a web browser window with the title "React App" and the URL "localhost:3000/shelf". The main page displays a library interface with tabs for "Home", "Search Books", and "Shelf". The "Shelf" tab is active, showing a "Current Loans:" section with a book titled "JavaScript Cookbook" by Zofia Luv. A "Loans" tab is also visible. A "Loan Options" dialog box is overlaid on the page. The dialog box contains the following information:

- Loaner: Luv, Zofia
- Book Title: JavaScript Cookbook
- Status: Due in 5 days.
- Action Buttons:
 - Return Book (highlighted with a red border)
 - Renew loan for 7 days
- Close button

Then in our shelf page, the first book that we returned disappears.

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/shelf". The browser's address bar also displays "localhost:3000/shelf". Below the address bar is a horizontal navigation bar with various links: Radio-Canada.ca |..., Canada's federal..., Sign in to your IR..., Chenelière Éducat..., Montreal, zdm, Nos incontournab..., TED: Ideas worth..., and 在线课程 - 时间自...". The main content area has a blue header bar with the text "JAC Read" and navigation links "Home", "Search Books", and "Shelf". Below this is a navigation bar with tabs "Loans" and "Your History", where "Loans" is selected. The main content area is divided into two sections: "Current Loans:" and "Loan Options". The "Current Loans:" section shows a single book cover for "Become a Guru in JavaScript" by Lena Luv. The "Loan Options" section displays the text "Due in 6 days.", a "Manage Loan" button, a "Search more books?" button, and a call-to-action "Leave a review". At the bottom of the page, there is a footer bar with the text "© JAC Library App, Inc." and navigation links "Home" and "Search Books".

Go to our database, checkout table, only the second book is there.

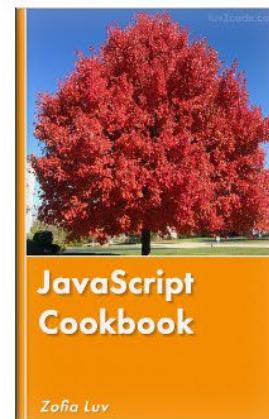
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes databases like b'mydb', customer_tracker, employees, member_tracker, myblog, mydb, passenger_tracker, and reactlibrarydatabase. Under reactlibrarydatabase, there are tables such as book, checkout, history, messages, review, Views, Stored Procedures, and Functions. The 'checkout' table is selected in the tabs at the top of the main window. A SQL query is run in the editor: `SELECT * FROM reactlibrarydatabase.checkout;`. The result grid shows a single row of data:

	id	user_email	checkout_date	return_date	book_id
*	2	testuser2@email.com	2023-02-12	2023-02-19	2

The entire row is highlighted with a red box.

If we go to the detailed page of book 1, we can see it can be checked out again.

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/checkout/1". The page displays a book cover for "JavaScript Cookbook" by Zofia Luv, featuring a red tree against a blue sky. The book's title and author are also visible on the cover. To the right of the book image, the title "JavaScript Cookbook" and author "Luv, Zofia" are listed. A large amount of placeholder text (Lorem ipsum) follows. On the far right, a sidebar provides information about the book's availability: "1/5 books checked out", "Available", "10 copies 10 available", and a prominent green "Checkout" button. Below the sidebar, a note states, "This number can change until placing order has been complete." At the bottom of the sidebar, there is a "Leave a review? ▾" link and a five-star rating icon. The browser's header bar includes various links like "Radio-Canada.ca", "Canada's federal...", "Sign in to your IR...", "Chenelière Éducat...", "Montreal", "zdm", "Nos incontournab...", "TED: Ideas worth...", and "在线课程 -". The navigation bar at the top of the page includes "JAC Read", "Home", "Search Books", and "Shelf".



JavaScript Cookbook

Luv, Zofia

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris,

1/5 books checked out

Available

10 copies 10
available

Checkout

This number can change
until placing order has been
complete.

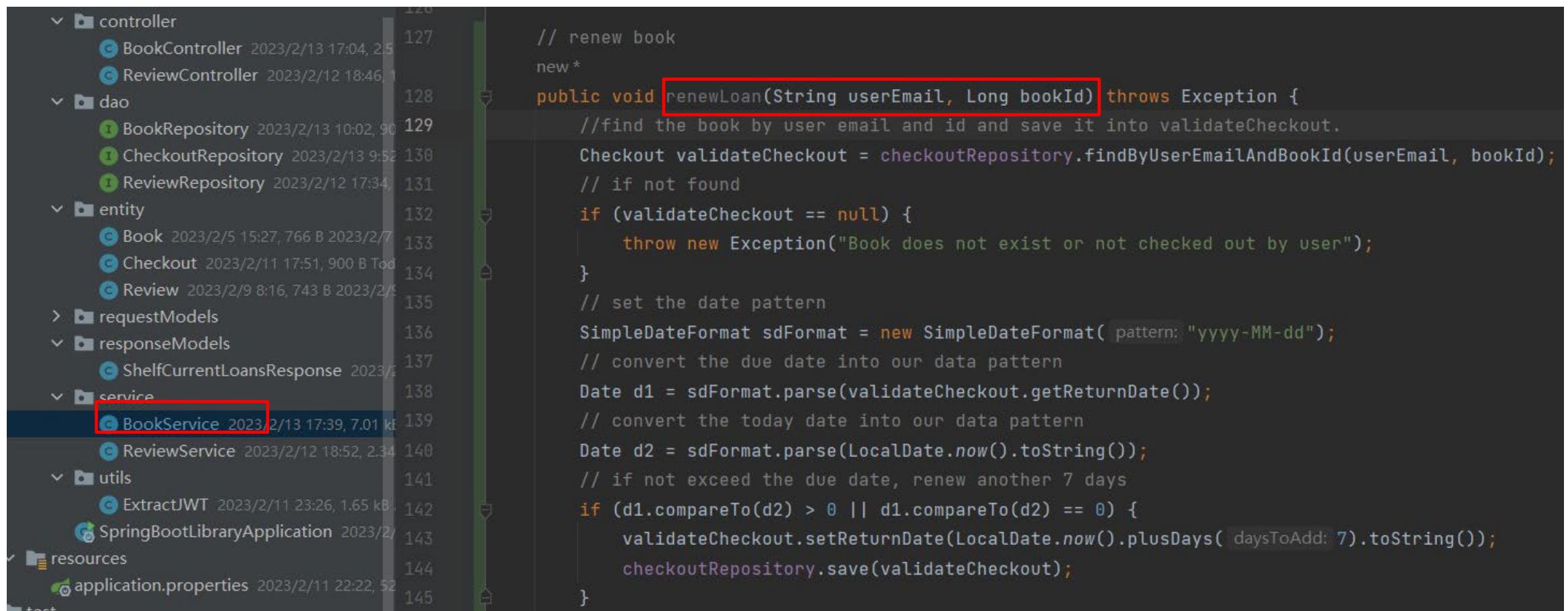
Leave a review? ▾



17. SPRINGBOOT RENEW LOANS SERVICE AND ENDPOINT LAYER



Step1 go into our service directory, Open up our BookService, create renewLoan().



```
// renew book
new*
public void renewLoan(String userEmail, Long bookId) throws Exception {
    //find the book by user email and id and save it into validateCheckout.
    Checkout validateCheckout = checkoutRepository.findByUserEmailAndBookId(userEmail, bookId);
    // if not found
    if (validateCheckout == null) {
        throw new Exception("Book does not exist or not checked out by user");
    }
    // set the date pattern
    SimpleDateFormat sdFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    // convert the due date into our data pattern
    Date d1 = sdFormat.parse(validateCheckout.getReturnDate());
    // convert the today date into our data pattern
    Date d2 = sdFormat.parse(LocalDate.now().toString());
    // if not exceed the due date, renew another 7 days
    if (d1.compareTo(d2) > 0 || d1.compareTo(d2) == 0) {
        validateCheckout.setReturnDate(LocalDate.now().plusDays( daysToAdd: 7).toString());
        checkoutRepository.save(validateCheckout);
    }
}
```

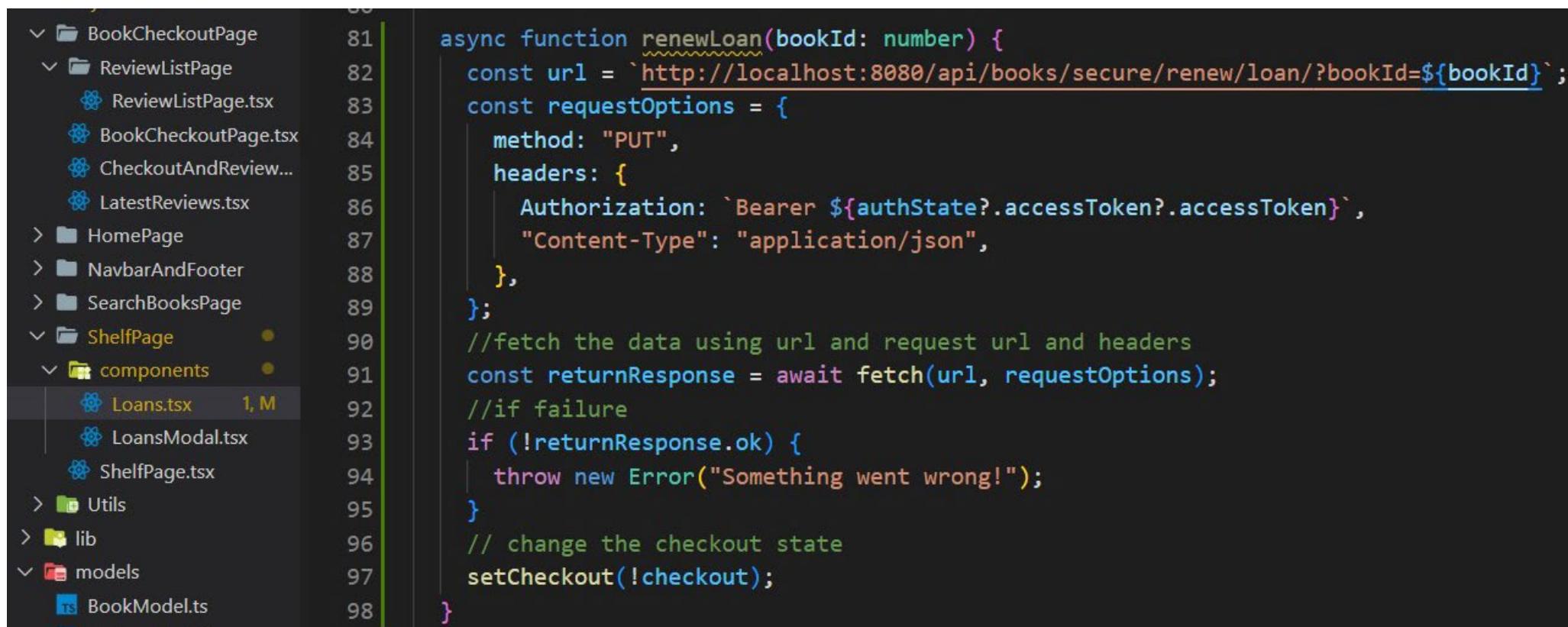
```
BookService.java x BookController.java x
51 @PutMapping("/secure/checkout")
52     public Book checkoutBook(@RequestHeader(value = "Authorization") String token
53             , @RequestParam Long bookId) throws Exception {
54         String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
55         return bookService.checkoutBook(userEmail, bookId);
56     }
57 }
58 // return book
59 // suyu06
60 @PutMapping("/secure/return")
61     public void returnBook(@RequestHeader(value = "Authorization") String token,
62             @RequestParam Long bookId) throws Exception {
63         String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
64         bookService.returnBook(userEmail, bookId);
65     }
66 // renew another 7 days for the book
67 // suyu06
68 @PutMapping("/secure/renew/loan")
69     public void renewLoan(@RequestHeader(value = "Authorization") String token,
70             @RequestParam Long bookId) throws Exception {
71         String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
72         bookService.renewLoan(userEmail, bookId);
73 }
```

Step 2 Go into BookController class, use “Put” method to create an endpoint for renew the book.

18. REACT RENEW LOAN



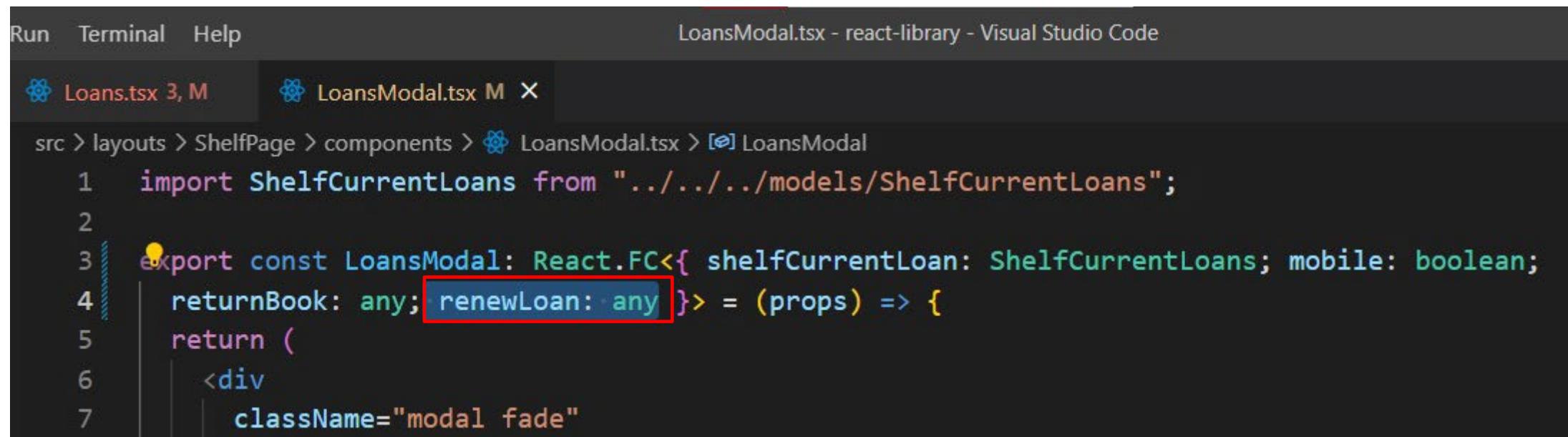
Step 1, In Loans.tsx, scroll until we can find the async function returnBook.
Right under this function, create a new async function renewLoan().



The screenshot shows a code editor with a sidebar on the left displaying a file tree. The tree includes 'BookCheckoutPage', 'ReviewListPage' (with 'ReviewListPage.tsx'), 'BookCheckoutPage.tsx', 'CheckoutAndReview...', 'LatestReviews.tsx', 'HomePage', 'NavbarAndFooter', 'SearchBooksPage', 'ShelfPage' (with 'components' folder containing 'Loans.tsx' and 'LoansModal.tsx', and 'ShelfPage.tsx'), 'Utils', 'lib', and 'models' (with 'BookModel.ts'). The main pane shows the 'Loans.tsx' file content:

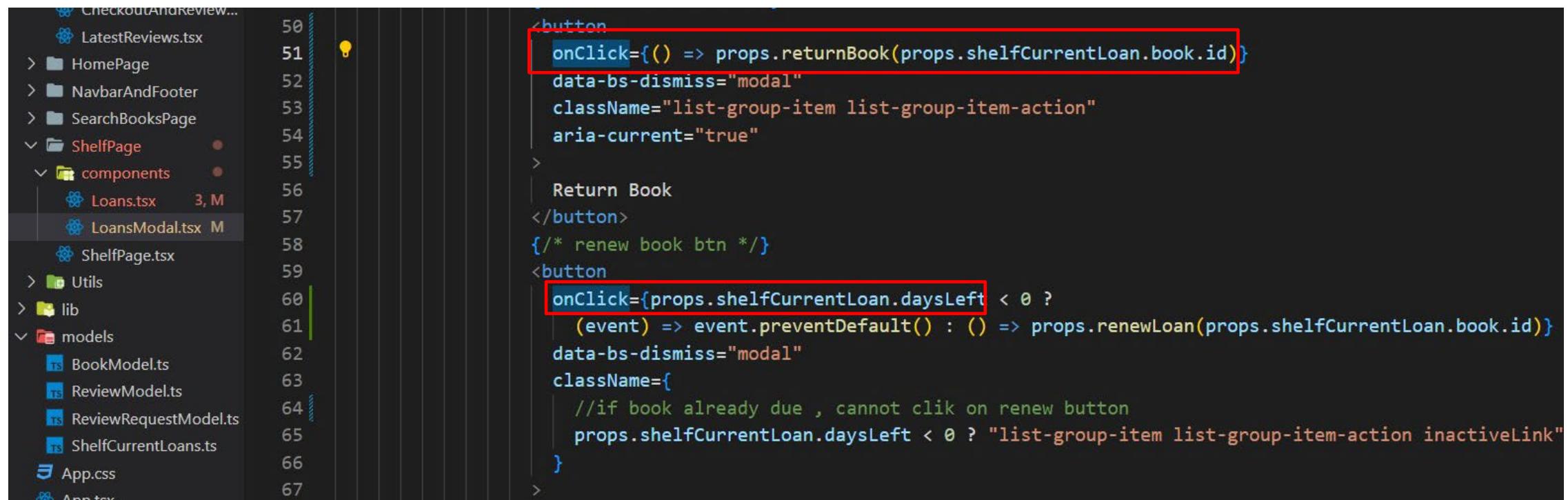
```
80
81 |     async function renewLoan(bookId: number) {
82 |         const url = `http://localhost:8080/api/books/secure/renew/loan/?bookId=${bookId}`;
83 |         const requestOptions = {
84 |             method: "PUT",
85 |             headers: {
86 |                 Authorization: `Bearer ${authState?.accessToken?.accessToken}`,
87 |                 "Content-Type": "application/json",
88 |             },
89 |         };
90 |         //fetch the data using url and request url and headers
91 |         const returnResponse = await fetch(url, requestOptions);
92 |         //if failure
93 |         if (!returnResponse.ok) {
94 |             throw new Error("Something went wrong!");
95 |         }
96 |         // change the checkout state
97 |         setCheckout(!checkout);
98 |     }
```

Step 2 go to LoansModal.tsx, add “renewLoan: any” as a new props



```
Run Terminal Help LoansModal.tsx - react-library - Visual Studio Code
Loans.tsx M LoansModal.tsx M X
src > layouts > ShelfPage > components > LoansModal.tsx > LoansModal
1 import ShelfCurrentLoans from "../../models/ShelfCurrentLoans";
2
3 export const LoansModal: React.FC<{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean;
4   returnBook: any; renewLoan: any }> = (props) => {
5   return (
6     <div
7       className="modal fade"
```

Step 3 add onClick listener to the two buttons: “return book” and “renew for 7 days”.



The screenshot shows a code editor with a dark theme. On the left is a file tree:

- CheckoutAndReview...
- LatestReviews.tsx
- > HomePage
- > NavbarAndFooter
- > SearchBooksPage
- ShelfPage (selected)
- <-- components
- Loans.tsx (3, M)
- LoansModal.tsx (M)
- ShelfPage.tsx
- > Utils
- > lib
- models
- BookModel.ts
- ReviewModel.ts
- ReviewRequestModel.ts
- ShelfCurrentLoans.ts
- App.css
- App.tsx

The main pane displays the following code:

```
50
51 <button
52   onClick={() => props.returnBook(props.shelfCurrentLoan.book.id)}
53   data-bs-dismiss="modal"
54   className="list-group-item list-group-item-action"
55   aria-current="true"
56 >
57   Return Book
58 </button>
59 /* renew book btn */
60 <button
61   onClick={props.shelfCurrentLoan.daysLeft < 0 ?
62     (event) => event.preventDefault() : () => props.renewLoan(props.shelfCurrentLoan.book.id)}
63   data-bs-dismiss="modal"
64   className={
65     //if book already due , cannot click on renew button
66     props.shelfCurrentLoan.daysLeft < 0 ? "list-group-item list-group-item-action inactiveLink"
67   }
68 >
```

Two specific lines of code are highlighted with red boxes:

- Line 51: `onClick={() => props.returnBook(props.shelfCurrentLoan.book.id)}`
- Line 61: `onClick={props.shelfCurrentLoan.daysLeft < 0 ?`

Step 4 Go to Loans.tsx, fix the error of LoansModal component.

The screenshot shows a code editor with a dark theme. On the left is a file tree with several project folders like BookCheckoutPage, ReviewListPage, HomePage, etc., and files like Loans.tsx, LoansModal.tsx, and ShelfPage.tsx. The Loans.tsx file is open in the editor. A red box highlights a TypeScript error message in line 185:

```
Property 'renewLoan' is missing in type '{ shelfCurrentLoan: ShelfCurrentLoans; mobile: true; returnBook: (bookId: number) => Promise<void>; }' but required in type '{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean; returnBook: any; renewLoan: any; }'. ts(2741)
```

The code in Loans.tsx includes imports for React and LoansModal, and defines a component that uses the LoansModal component with props shelfCurrentLoan, mobile (set to false), and returnBook. The error occurs because the LoansModal component expects a 'renewLoan' prop, which is not present in the current component's prop type definition.

Add renewLoan to LoansModal component's property.

```
        <hr />
        <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} returnBook={returnBook} renewLoan={renewLoan}/>
    </div>
)
)
)
</div>
/* mobile version */
<div className="container d-lg-none mt-2">
{shelfCurrentLoans.length > 0 ? (
<>
/*user has borrowed books*/
<h5 className="mb-3">Current Loans: </h5>

{shelfCurrentLoans.map((shelfCurrentLoan) => (
<div key={shelfCurrentLoan.book.id}>
/* book image part */
<div className="d-flex justify-content-center align-items-center">...
</div>
<div className="card d-flex mt-5 mb-3">...
</div>
<hr />
<LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={true} returnBook={returnBook} renewLoan={renewLoan}/>
</div>
```

Now, open our database, table checkout, we can see the second book was checked out on 12 Feb 2023, and will due on 19th Feb 2023.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' section with several databases listed, and the 'reactlibrarydatabase' database is expanded to show its tables: book, checkout, history, messages, review, Views, Stored Procedures, and Functions. The 'checkout' table is selected in the central pane. A SQL query is run against this table:

```
1 • SELECT * FROM reactlibrarydatabase.checkout;
```

The resulting 'Result Grid' shows the following data:

	id	user_email	checkout_date	return_date	book_id
▶	2	testuser2@email.com	2023-02-12	2023-02-19	2
*	NULL	NULL	NULL	NULL	NULL

The row for id 2 is highlighted with a red box, indicating the specific record being referred to.

If we click on the “Renew loan for 7 days”,

The screenshot shows a web browser window titled "React App" at "localhost:3000/shelf". The main interface displays a "Current Loans:" section with a book titled "Become a Guru in JavaScript" by Luv, Lena. A modal dialog titled "Loan Options" is open over the main content. The dialog shows the book's cover, author, title, and due date ("Due in 6 days"). It contains two buttons: "Return Book" and "Renew loan for 7 days", with the latter button highlighted by a red rectangle. A "Close" button is located in the bottom right corner of the dialog.

Then, in our shelf page, the due day becomes 7 now.

The screenshot shows a web browser window with the title "React App" and the URL "localhost:3000/shelf". The page has a blue header bar with the text "JAC Read" and navigation links for "Home", "Search Books", and "Shelf". Below the header, there are two tabs: "Loans" (selected) and "Your History". The main content area is titled "Current Loans:" and displays a book cover for "Become a Guru in JavaScript" by Lena Luv. To the right of the book cover is a "Loan Options" section. This section includes a red-bordered box containing the text "Due in 7 days.", a "Manage Loan" button, and a "Search more books?" link. At the bottom of the "Loan Options" section is a call-to-action button labeled "Leave a review".

And If we open our database, table checkout, we can see the second book now was checked out on 12 Feb 2023, and will due on 20th Feb 2023.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' section with several databases listed, including 'reactlibrarydatabase'. Under 'reactlibrarydatabase', there are tables like 'book', 'checkout', 'history', 'messages', and 'review'. The 'checkout' table is selected and shown in the main pane. A SQL query is run against it:

```
1 • SELECT * FROM reactlibrarydatabase.checkout;
```

The result grid shows the following data:

	id	user_email	checkout_date	return_date	book_id
▶	2	testuser2@email.com	2023-02-12	2023-02-20	2
*	HULL	HULL	HULL	HULL	HULL

The row for user_email 'testuser2@email.com' is highlighted with a red box around the 'checkout_date' and 'return_date' columns.