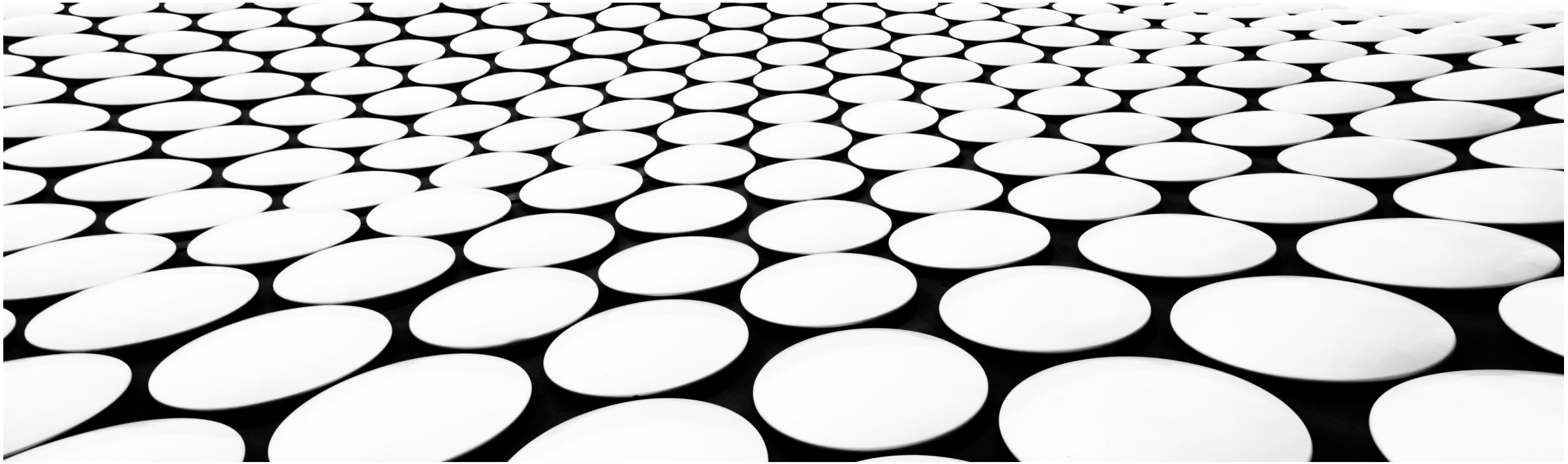
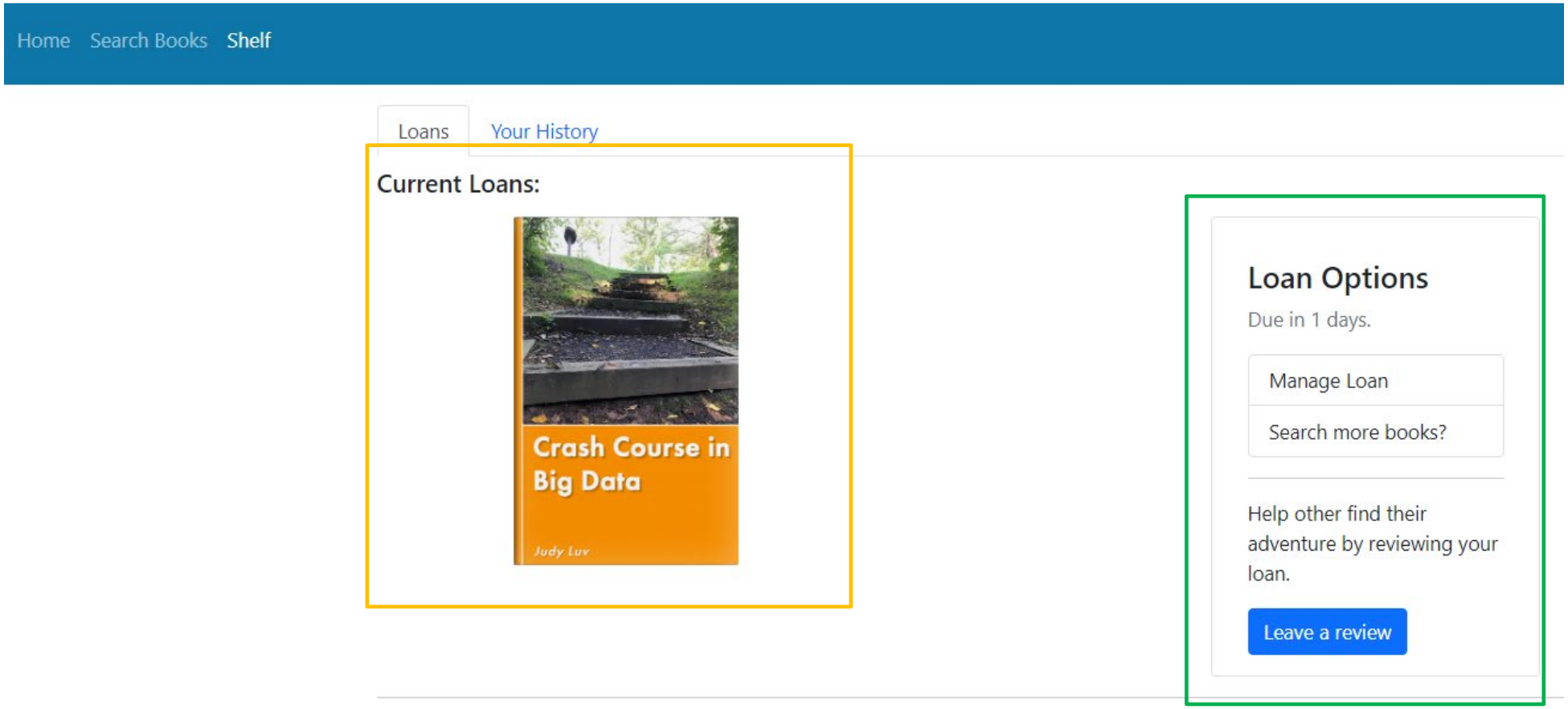
S16 SHELF PAGE

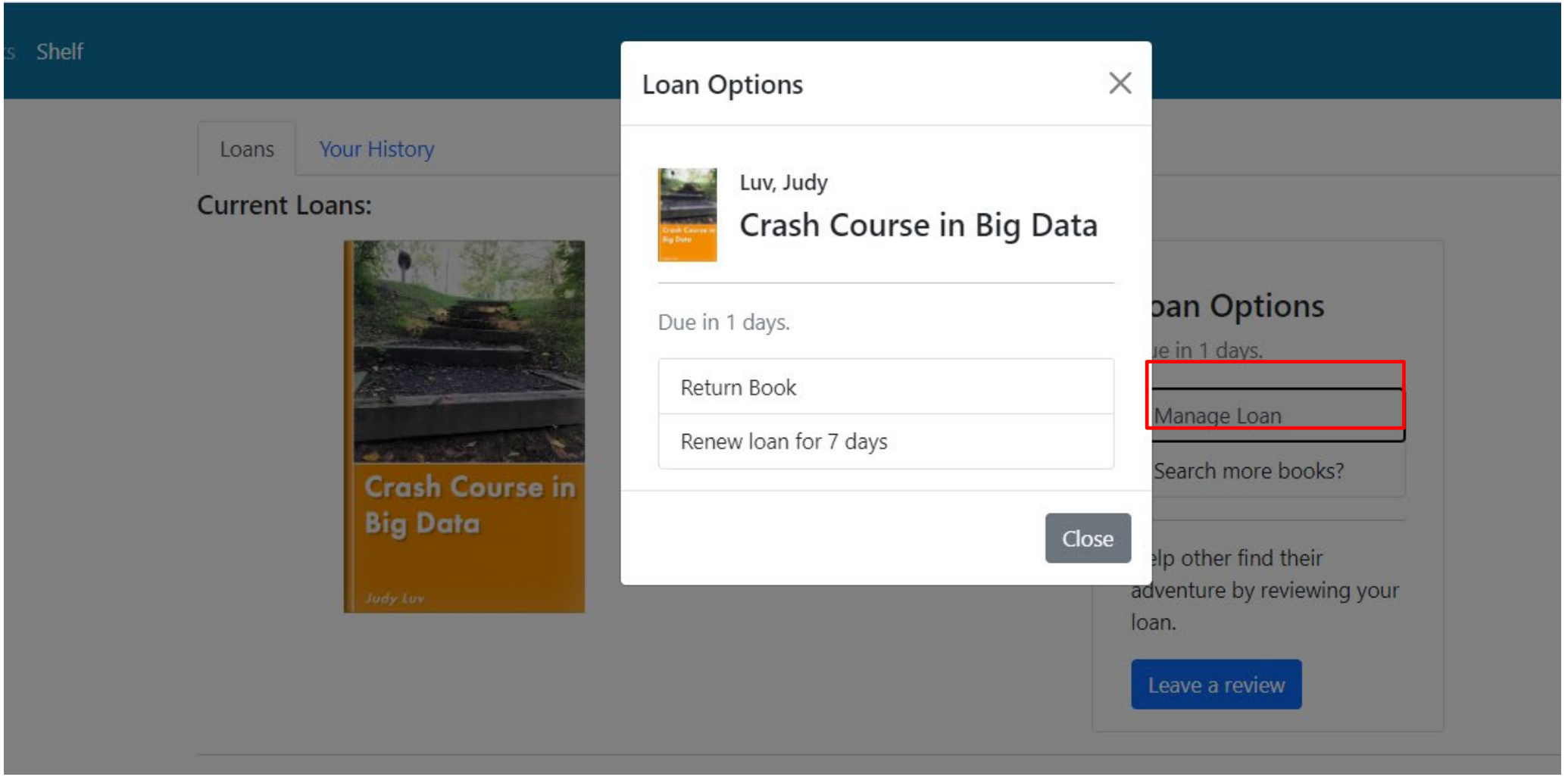


1. REACT- CREATE SHELF PAGE

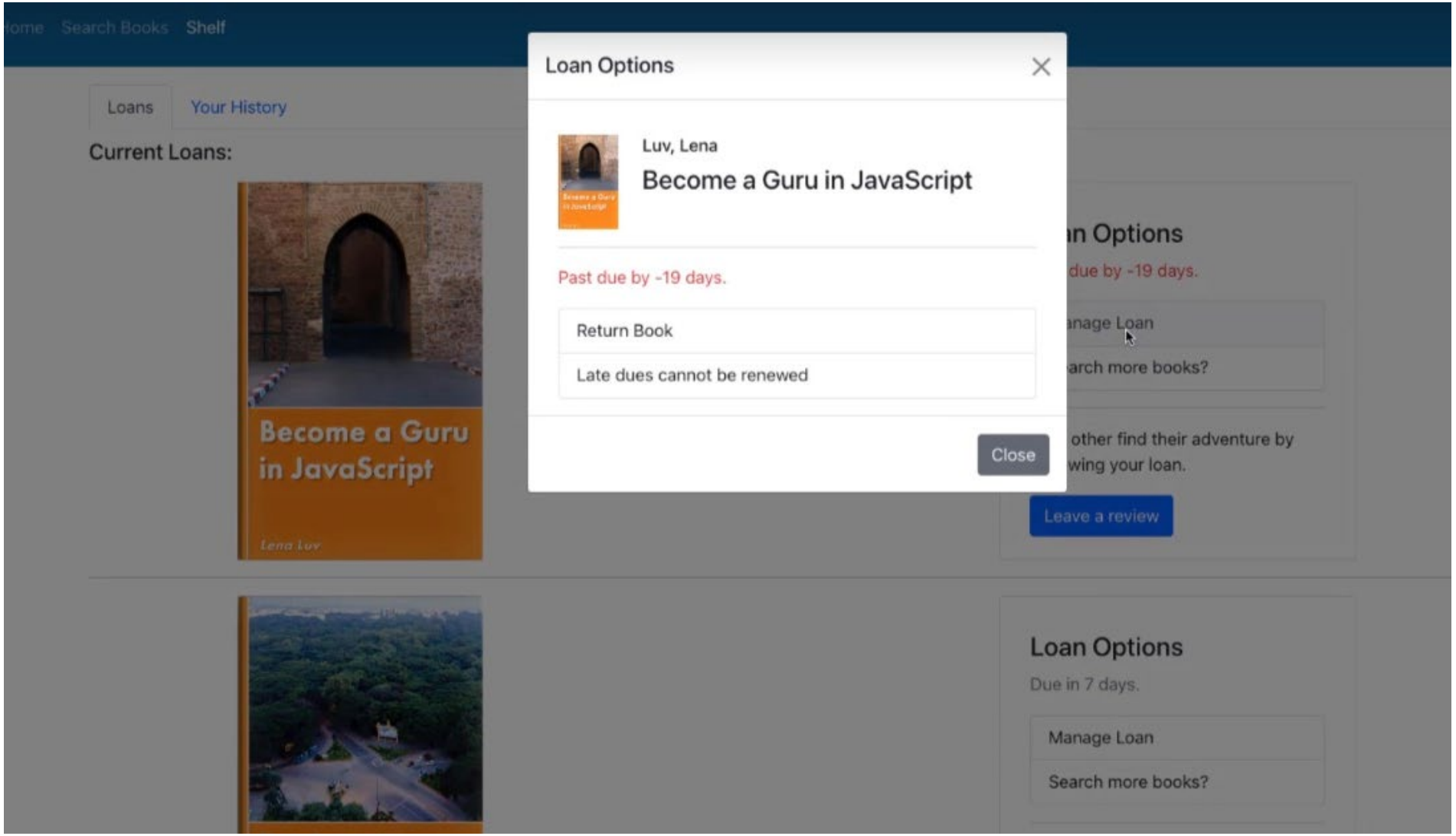
There are two modules in this page.
On the left, is the image of current borrowed books.
On the right , is the options to manage the borrowed books.

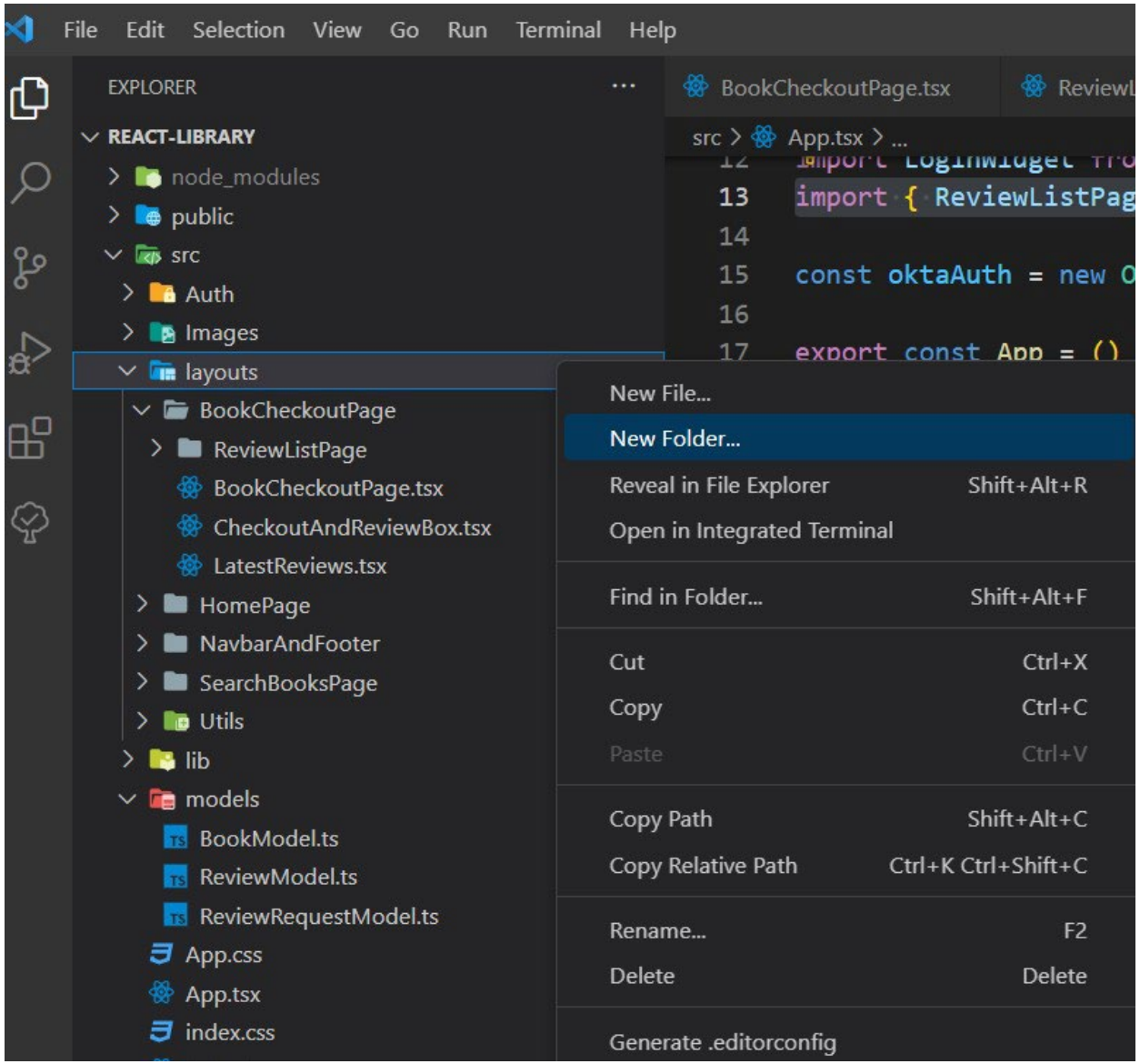


Clicking on “Manage loan”, a dialogue box will pop out, providing the choice of “Return book” and “renew for 7 days”.

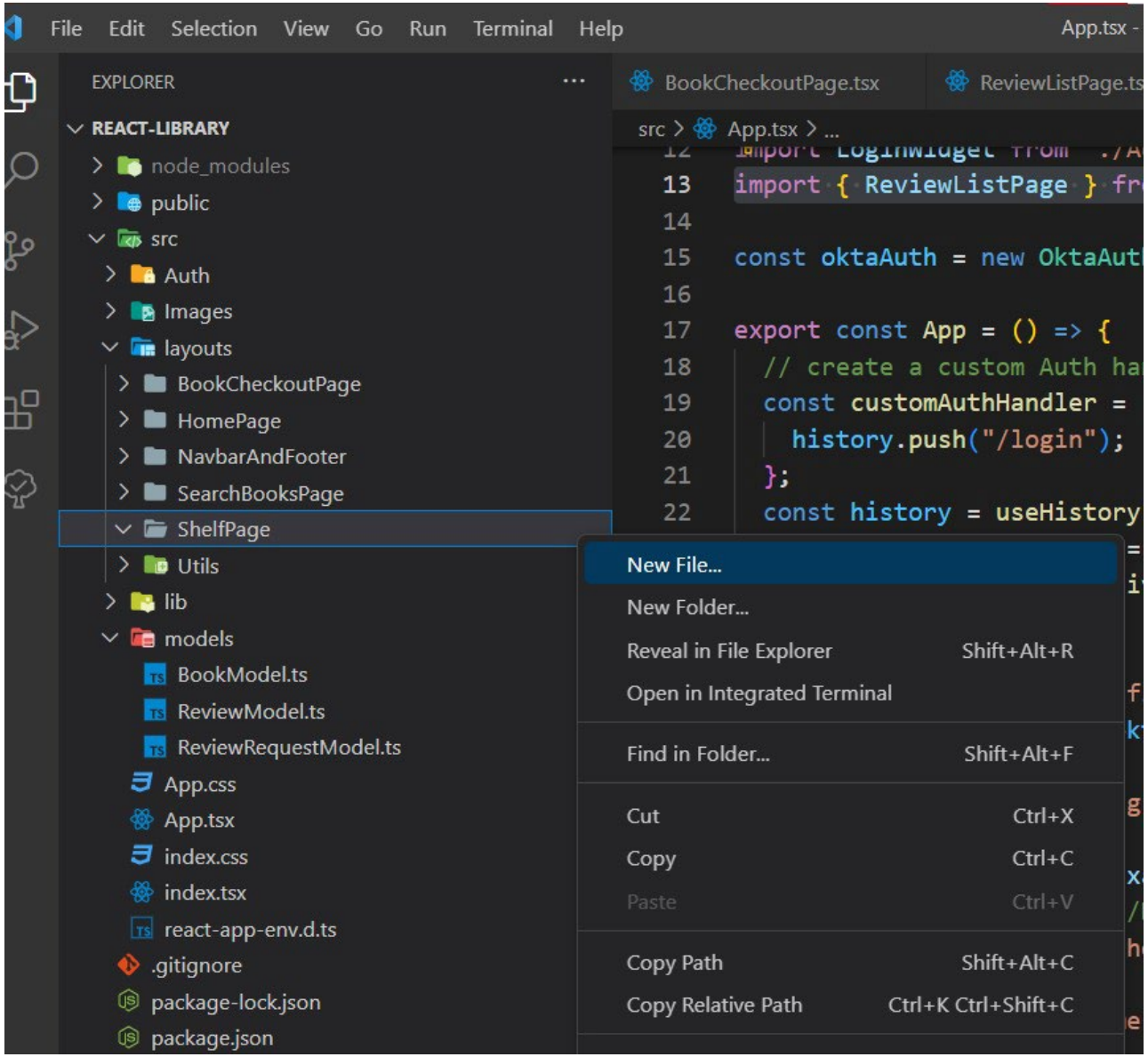


But if you missed the due day, you can only return the book.

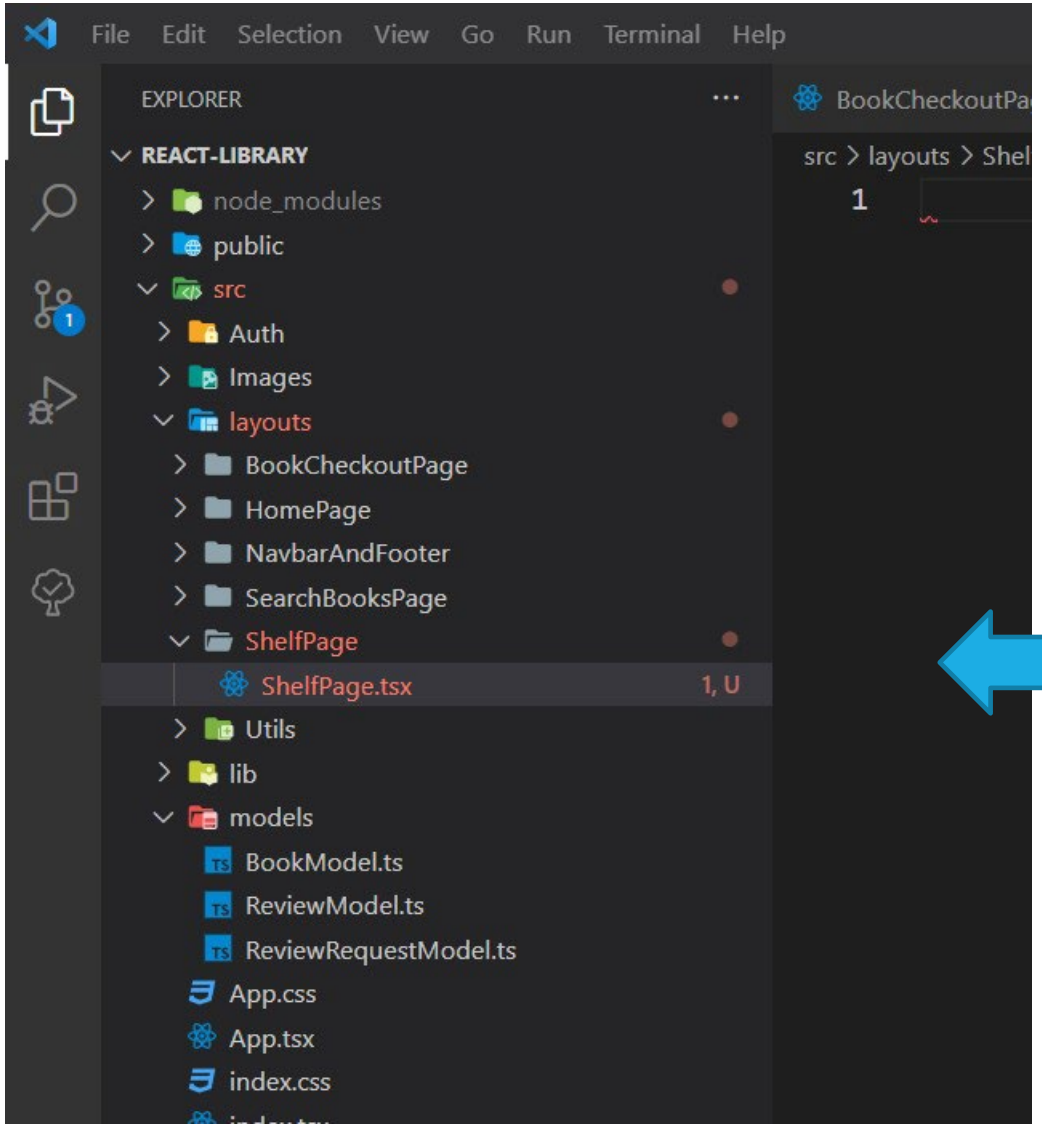




Step 1 Right click on layouts folder and choose “new folder”, name it “shelfPage”.

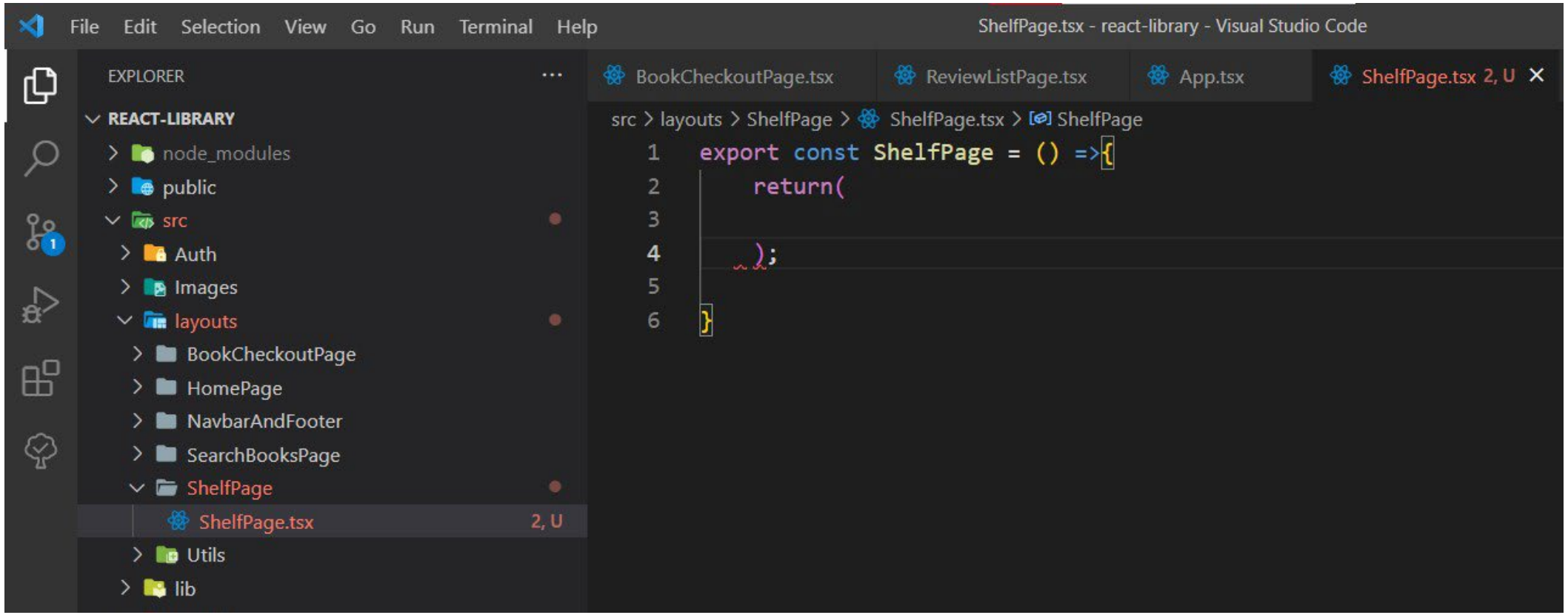


Step 2 Right click on the ShelfPage folder and choose “new file”



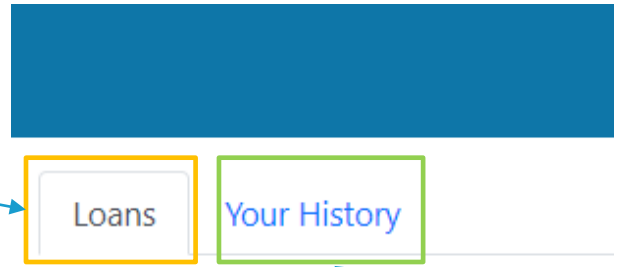
name it ShelfPage.tsx.

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



```
> 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.

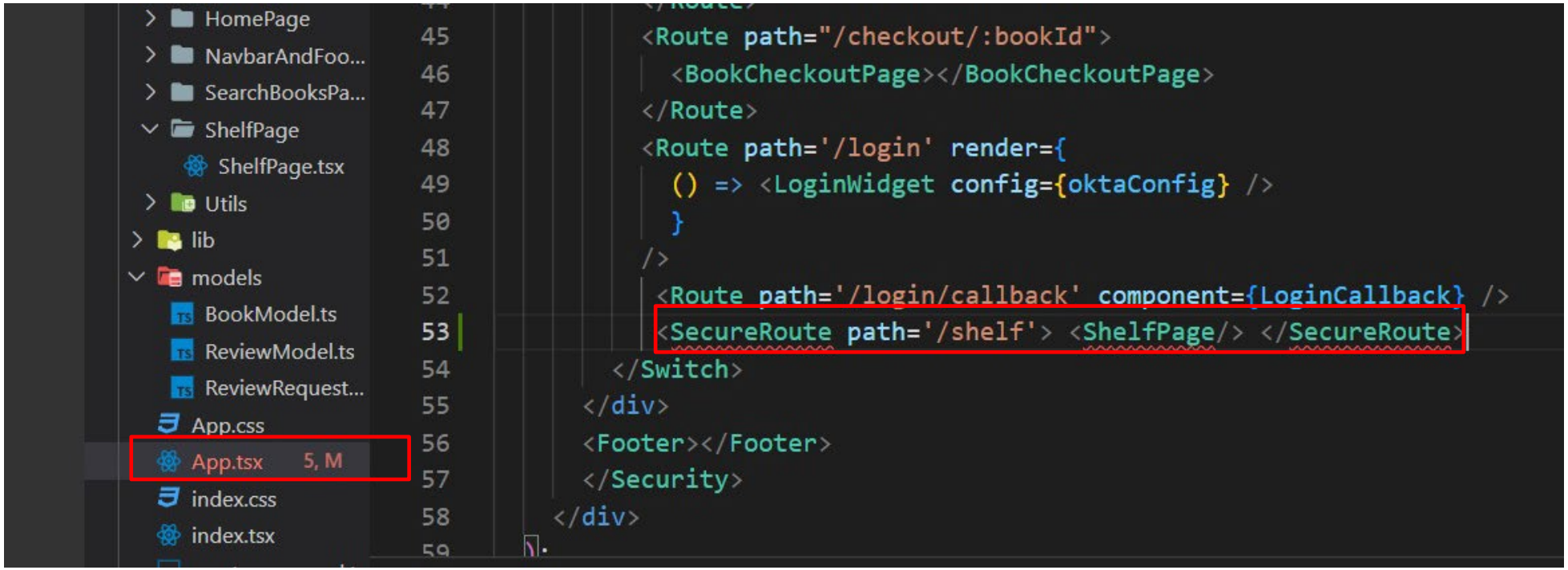


4.2 create two tab contents.

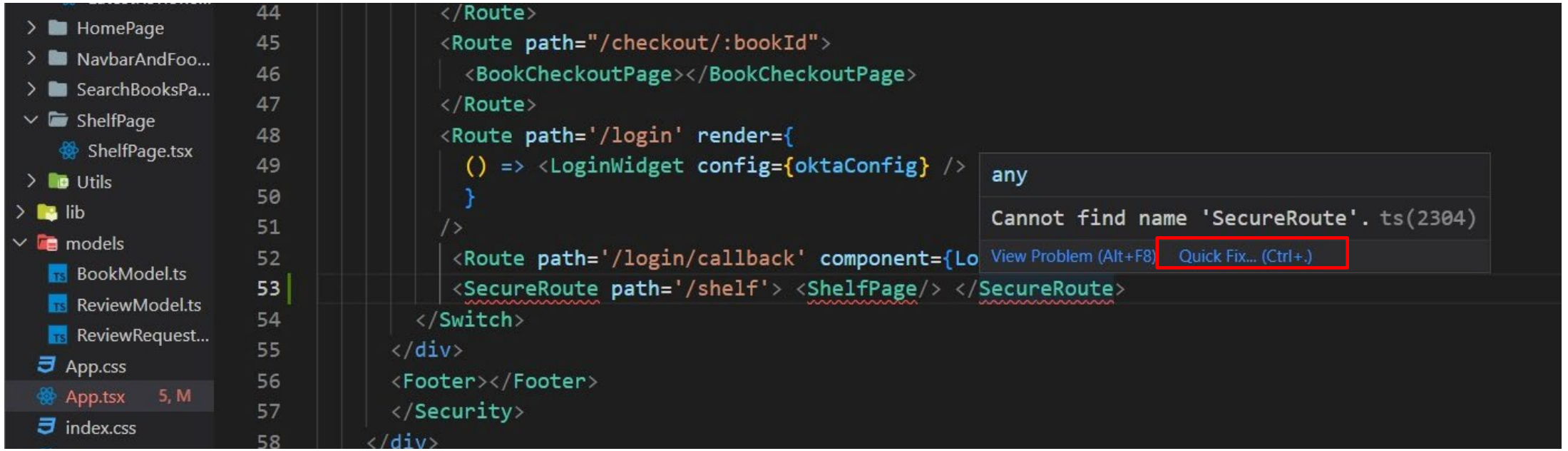

```
31          </div>
32        </nav>
33        <div className="tab-content" id="nav-tabContent">
34          <div className="tab-pane fade show active" id="nav-loans" role="tabpanel" aria-labelledby="nav-loans-tab">
35            <p> Loans</p>
36          </div>
37          <div className="tab-pane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

2. SECURE ROUTES

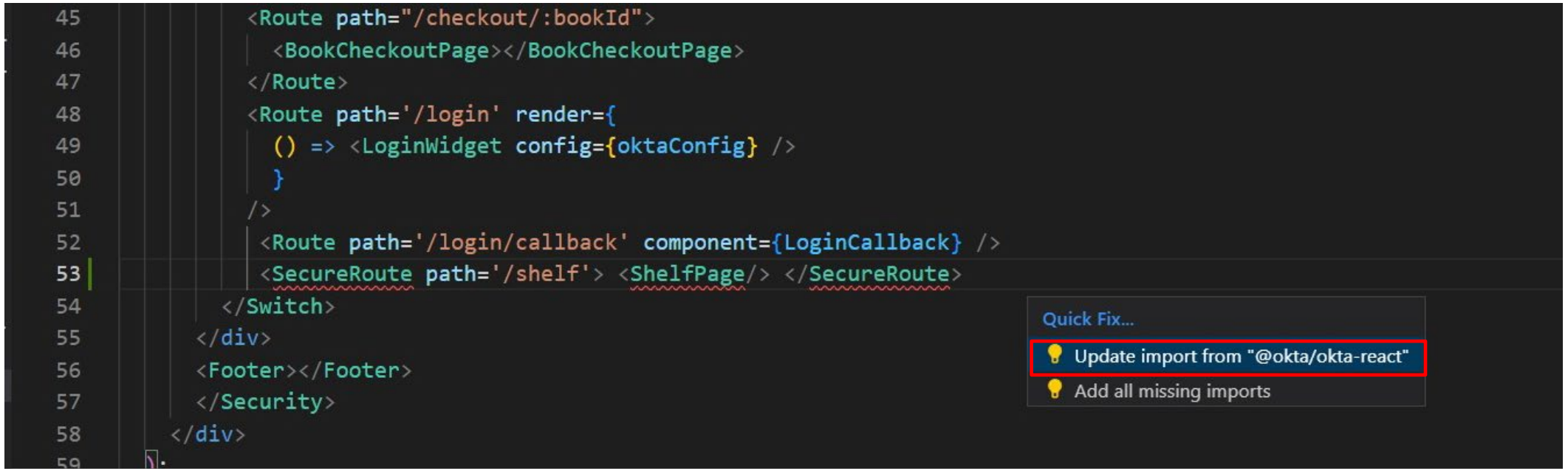
Step1 Go to App.tsx, add SecureRoute and specify the path.



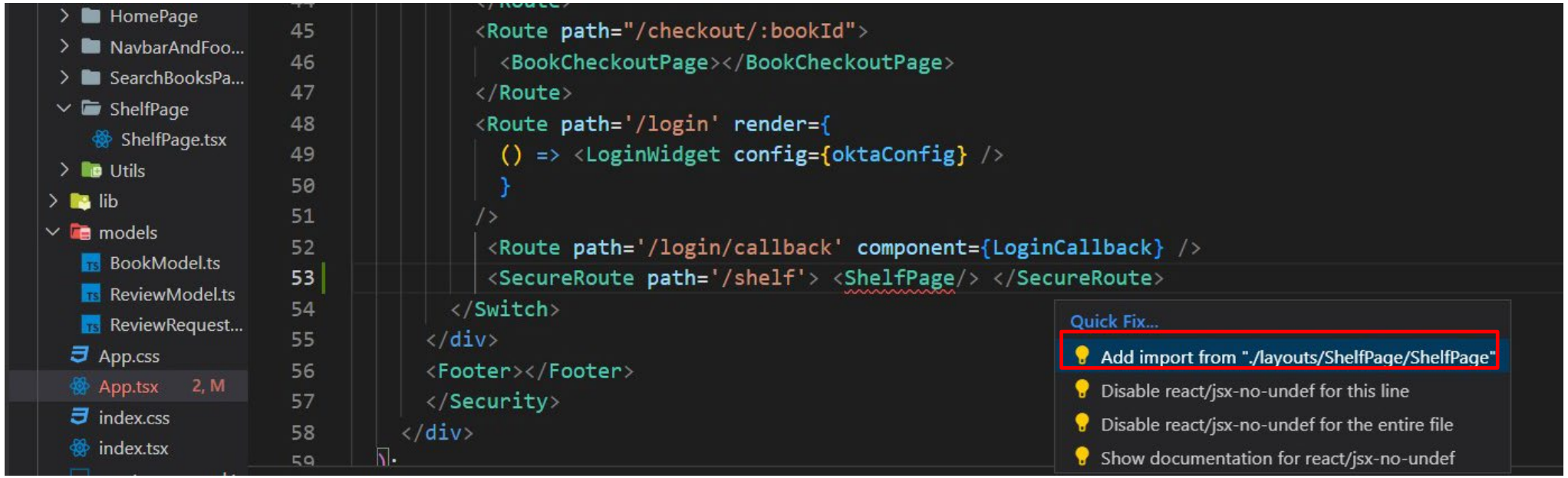
Step 2 Hover over the red tilde of SecureRoute, click on “Quick Fix”.



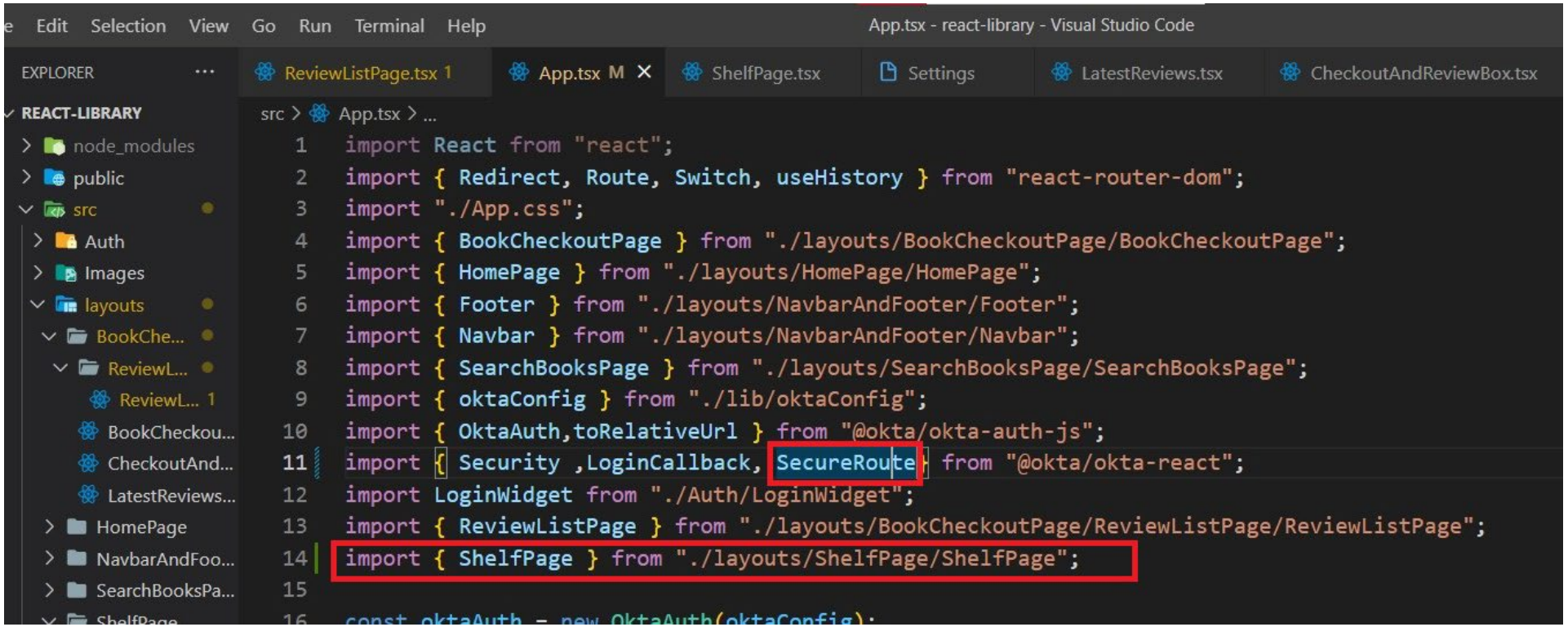
Choose “Update import from @okta/okta-react”.



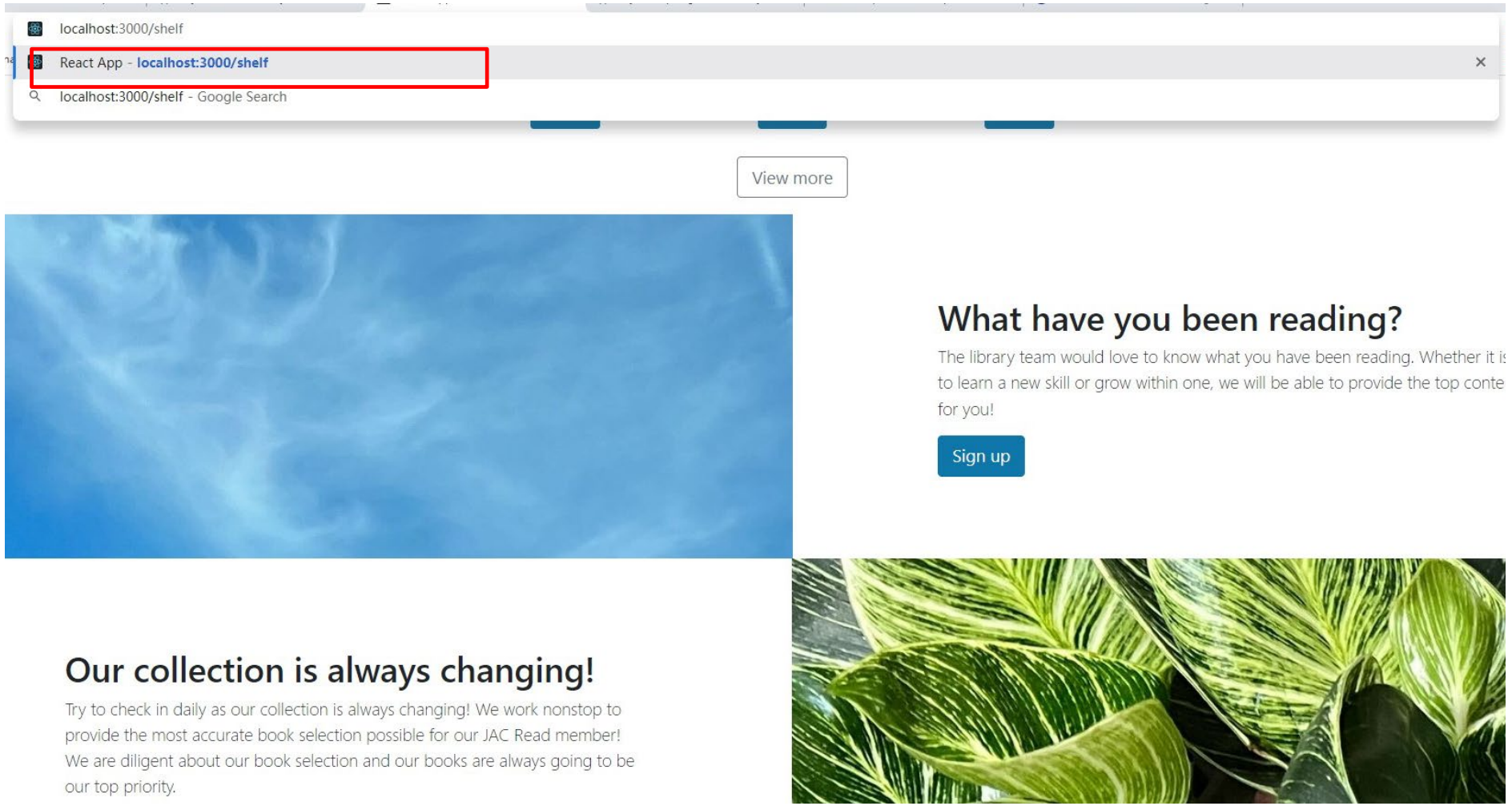
Step 3 Hover over the red tilde of ShelfPage, click on “Quick Fix”.
Choose “Add import from…”.



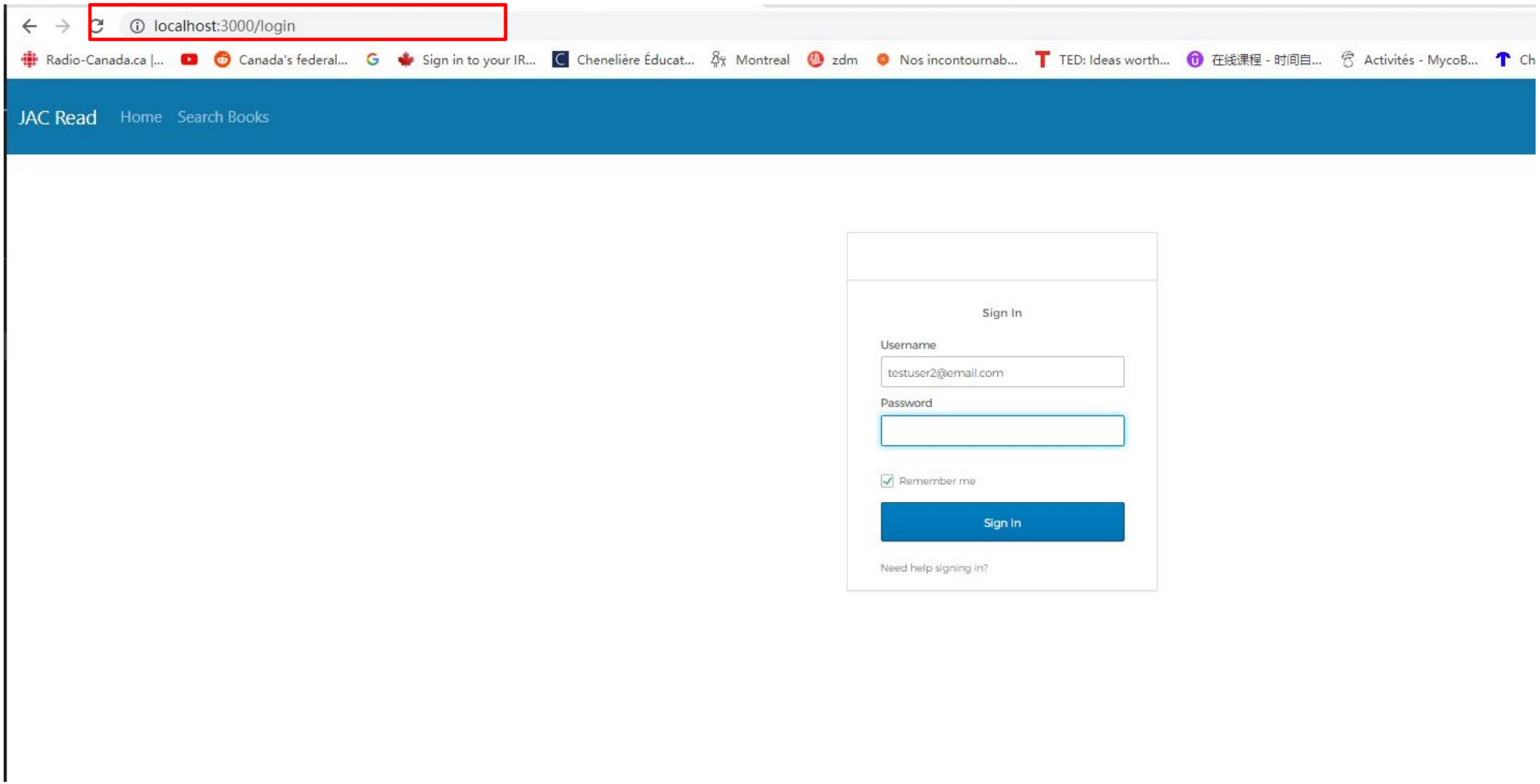
Then , on the top of the page, we can see both the two are imported.



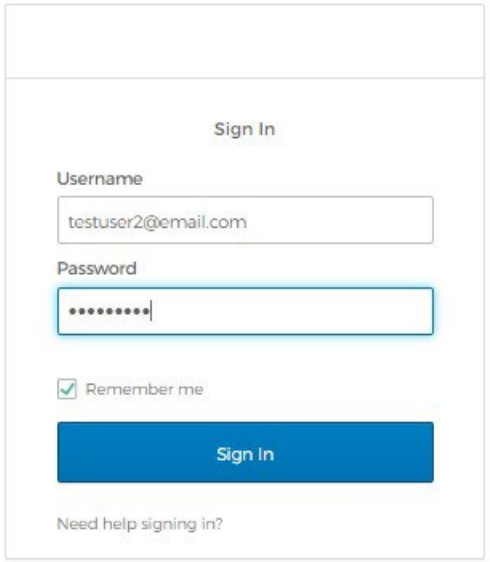
If we type http://localhost:3000/shelf ,



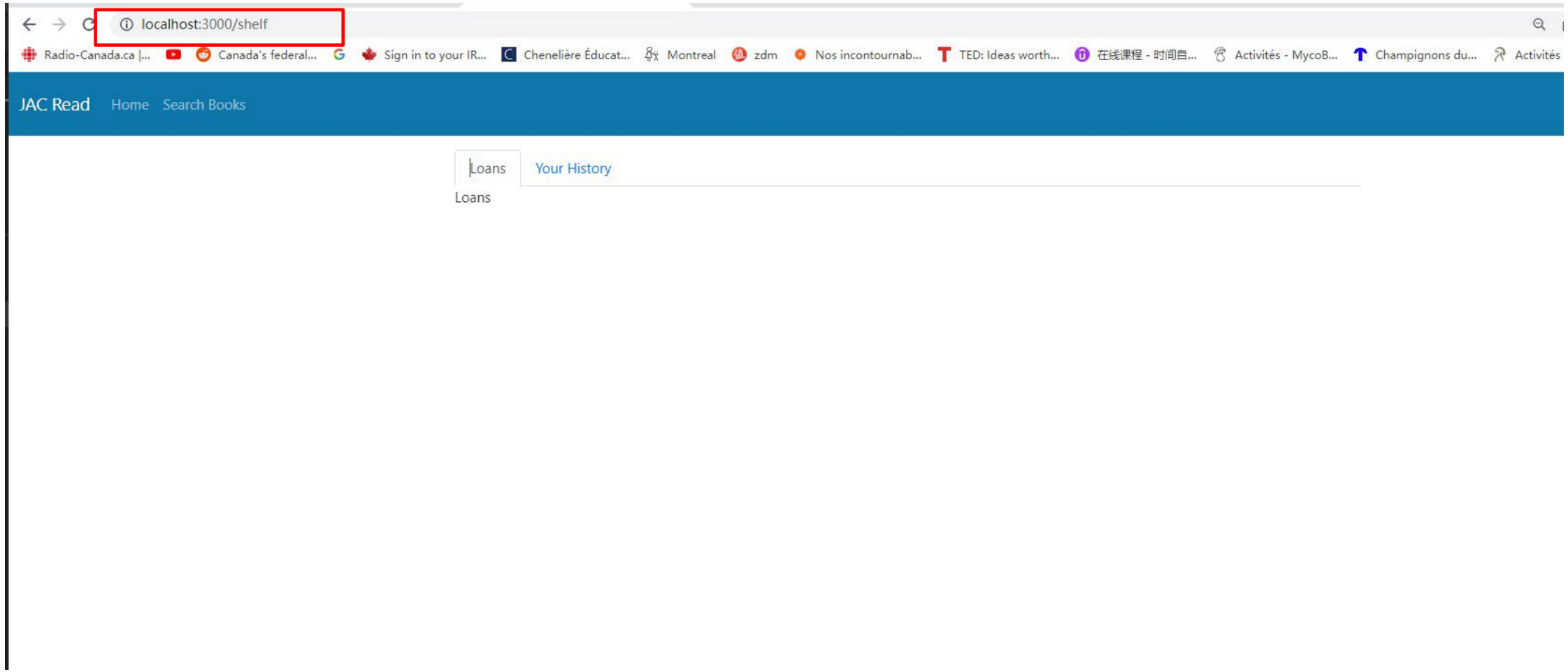
It will redirect us to the log in page.



Input our username and password :



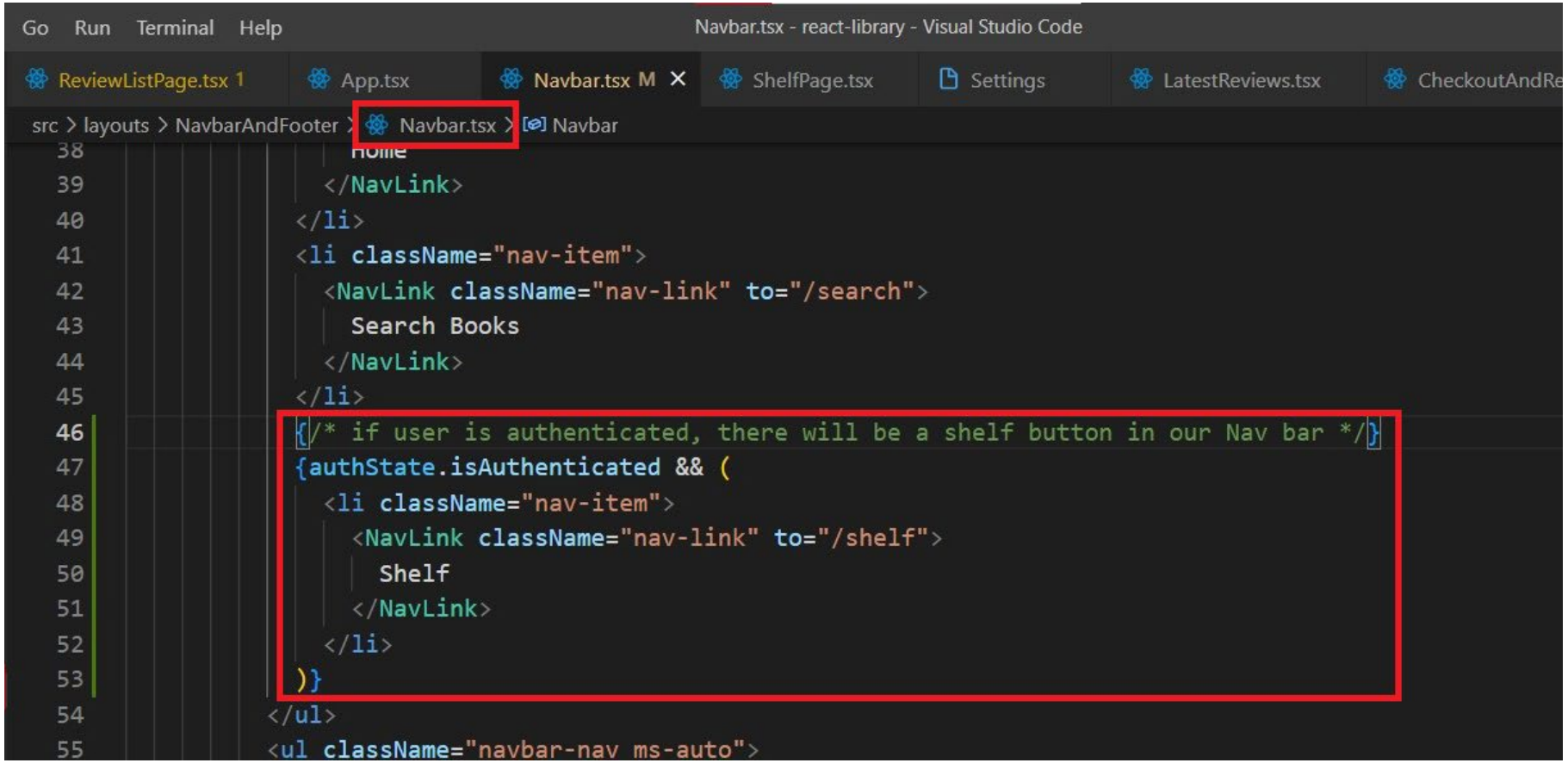
Then, we can visit shelf page.



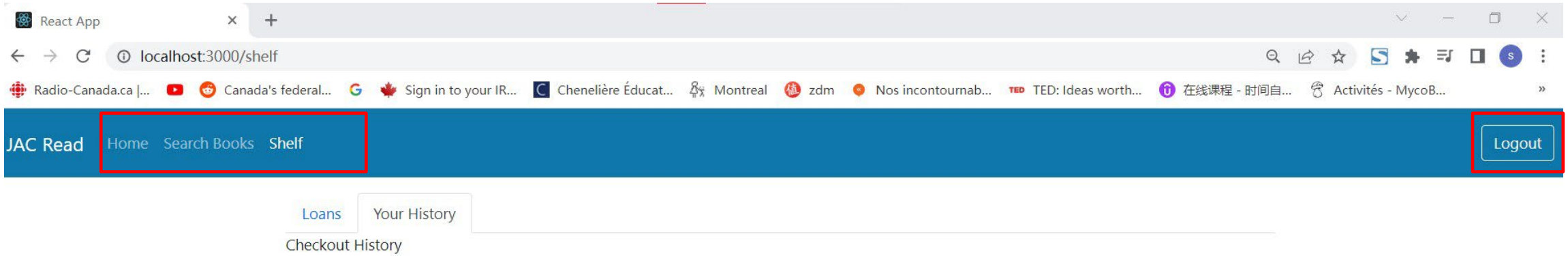
3.NAVIGAGTION BAR ENHANCEMENT

Let’s add shelf to our navigation bar.

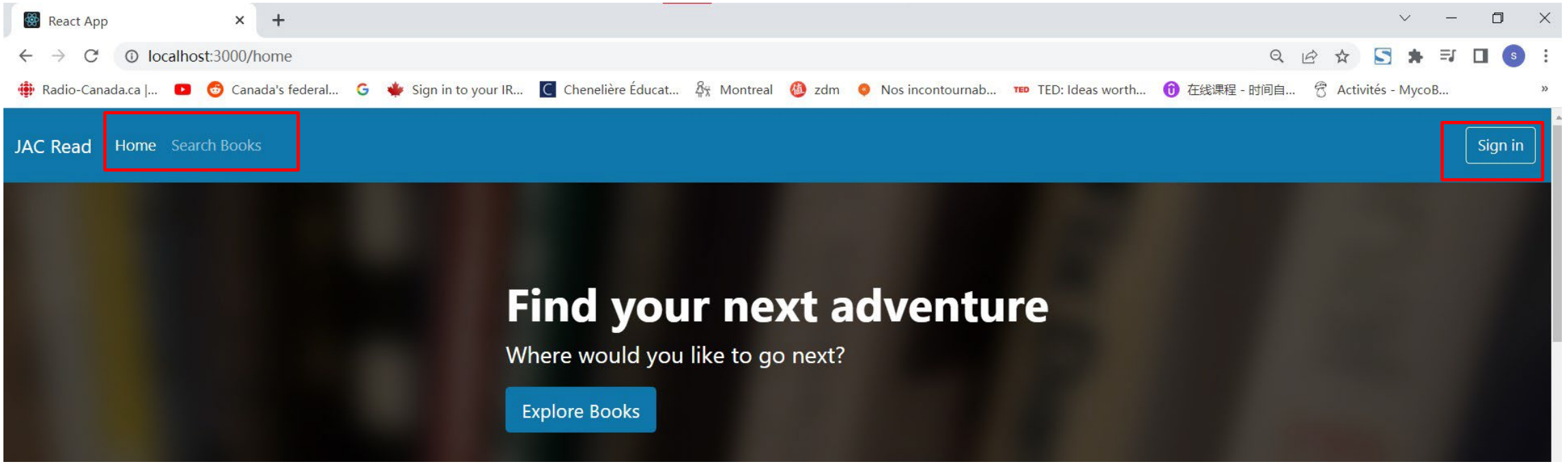
Step 1 Let‘s go to layouts -> NavbarAndFooter，open Navbar.tsx.
Add a NavLink of shelf, and add a authState judgement to it.



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

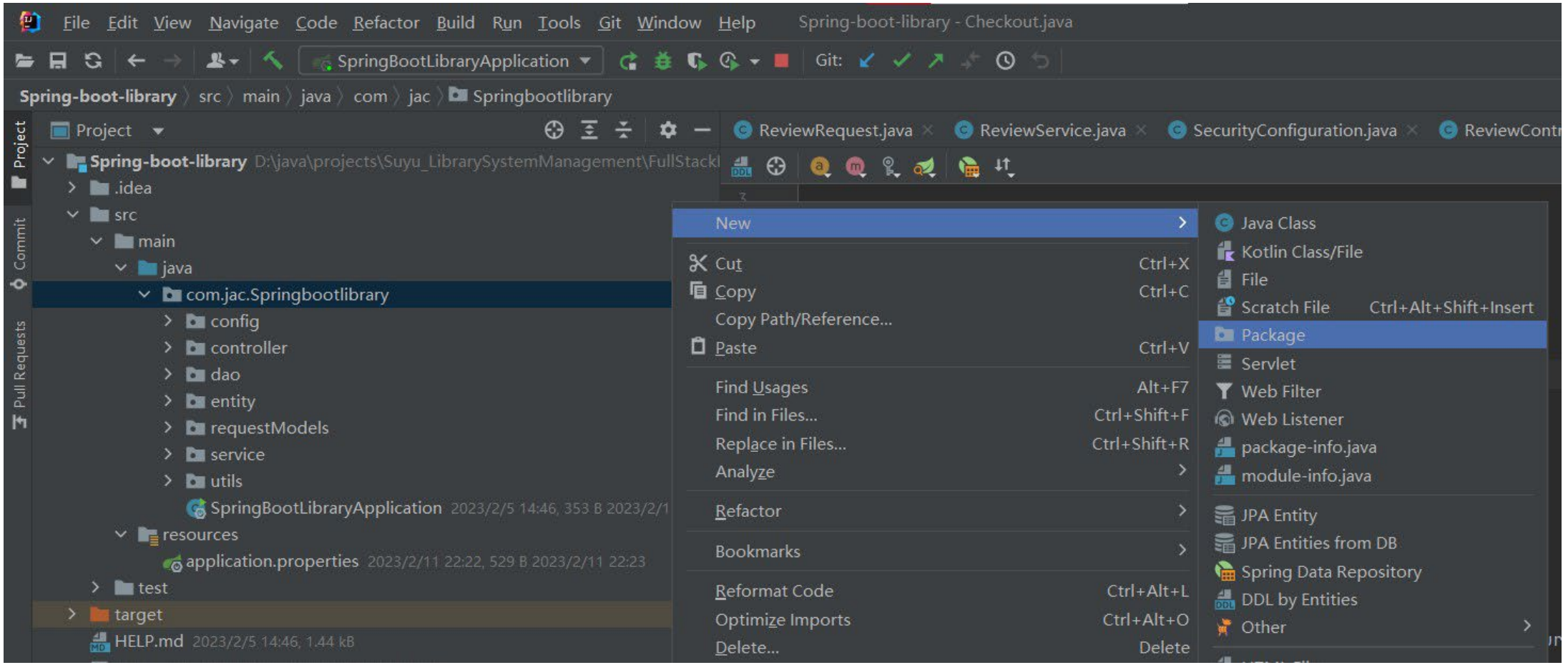


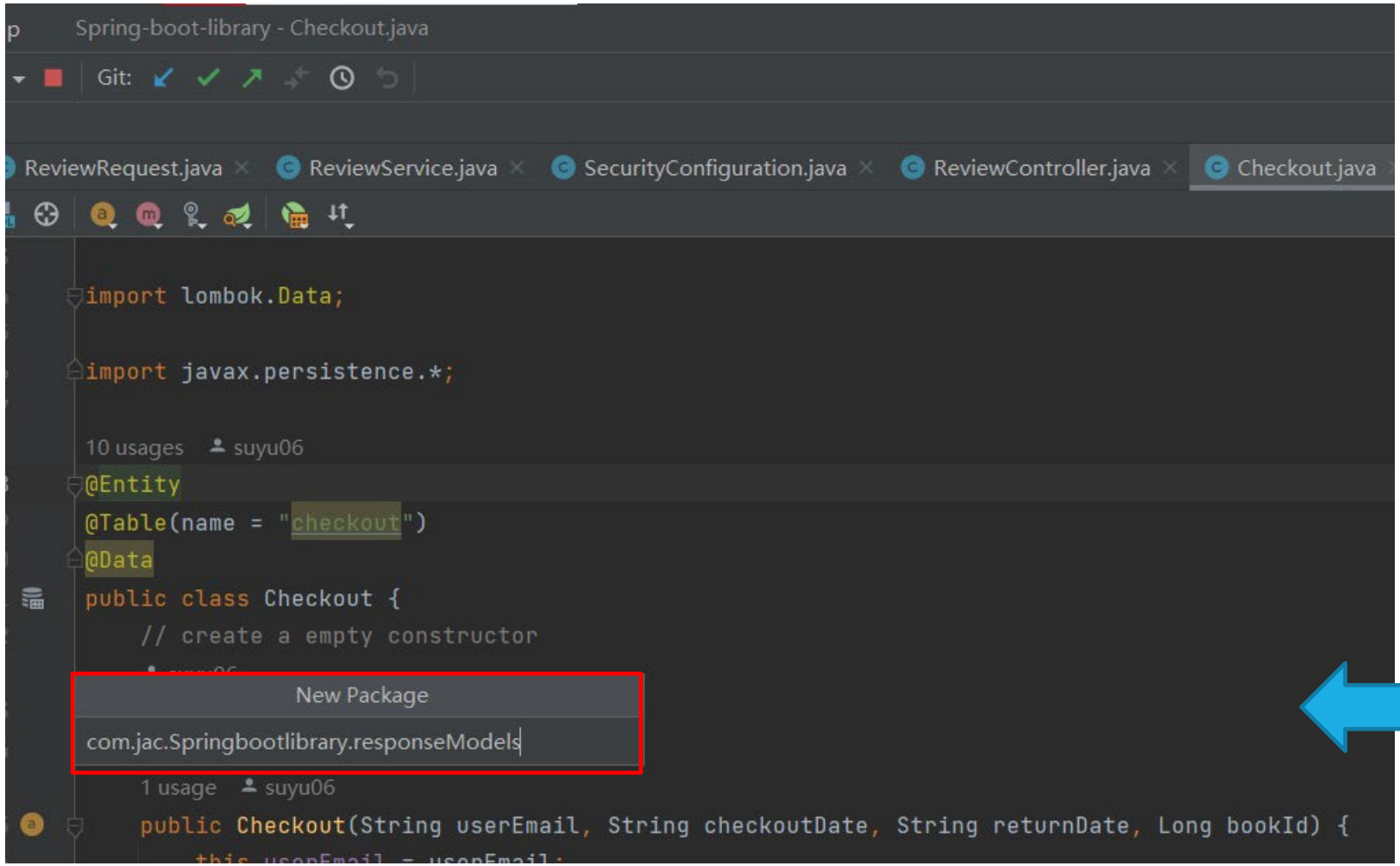
If we log out , in the navigation bar, there will be no “ shelf ” link.



4. SPRINGBOOT CREATE A RESPONSE

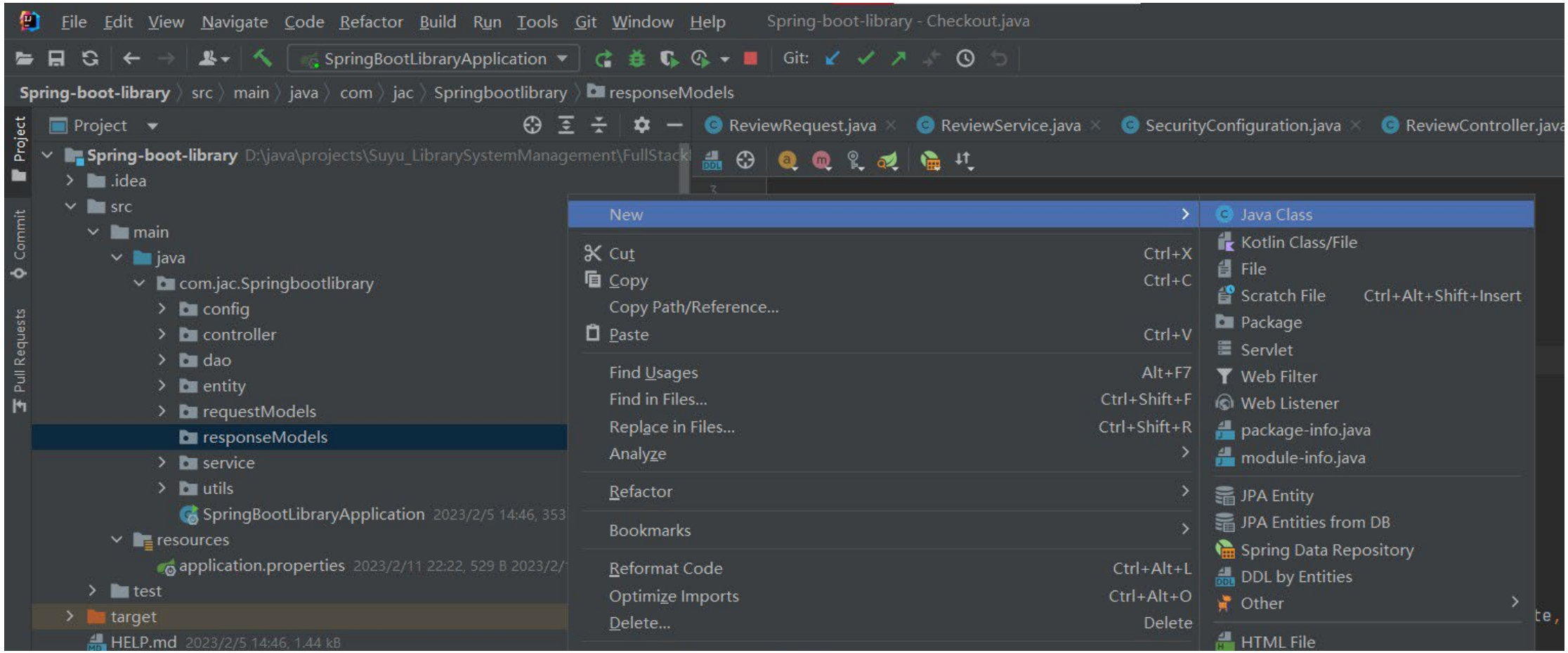
Step 1 So let's go ahead and right click on root directory and create a new package.

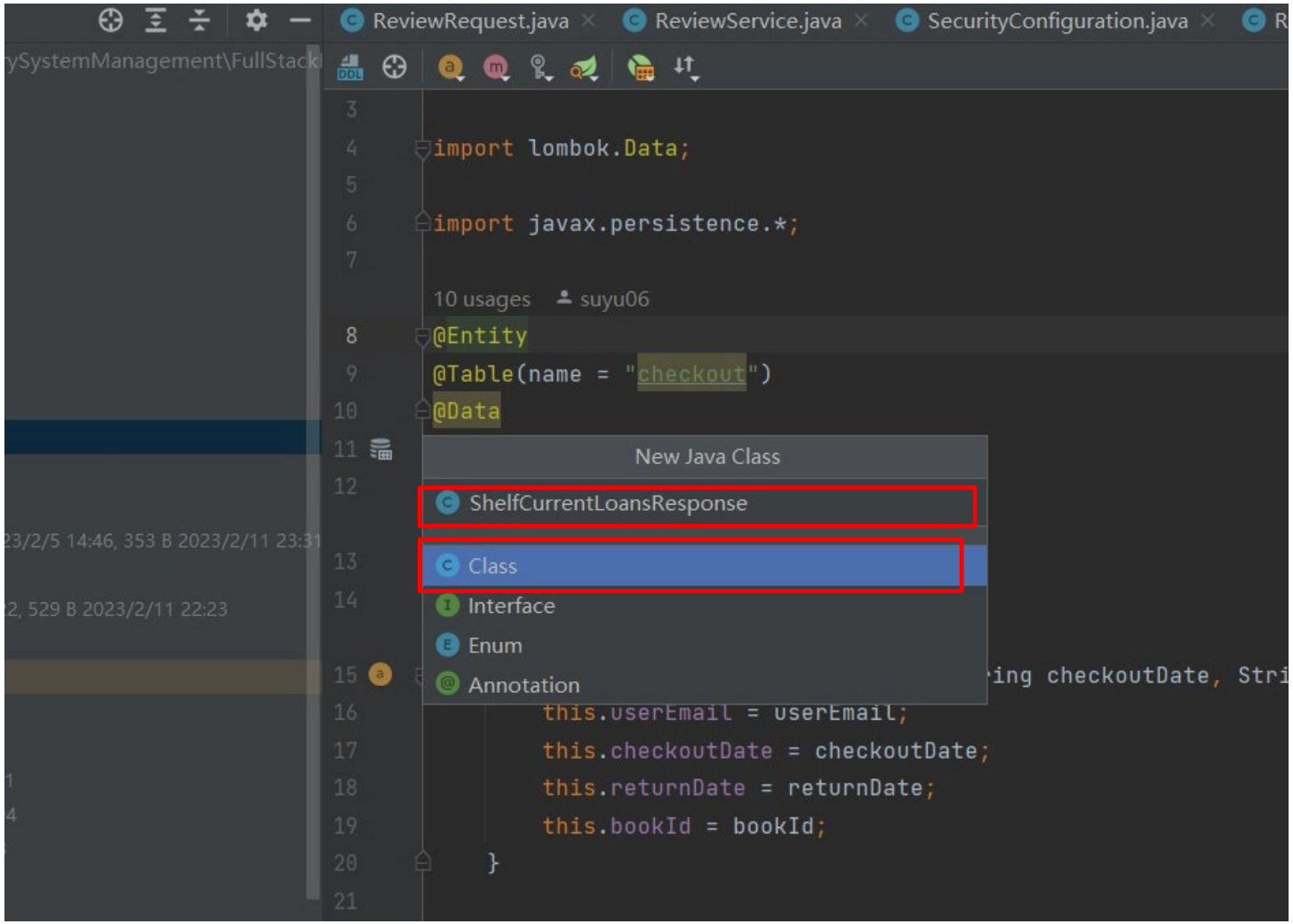




Name it responseModels

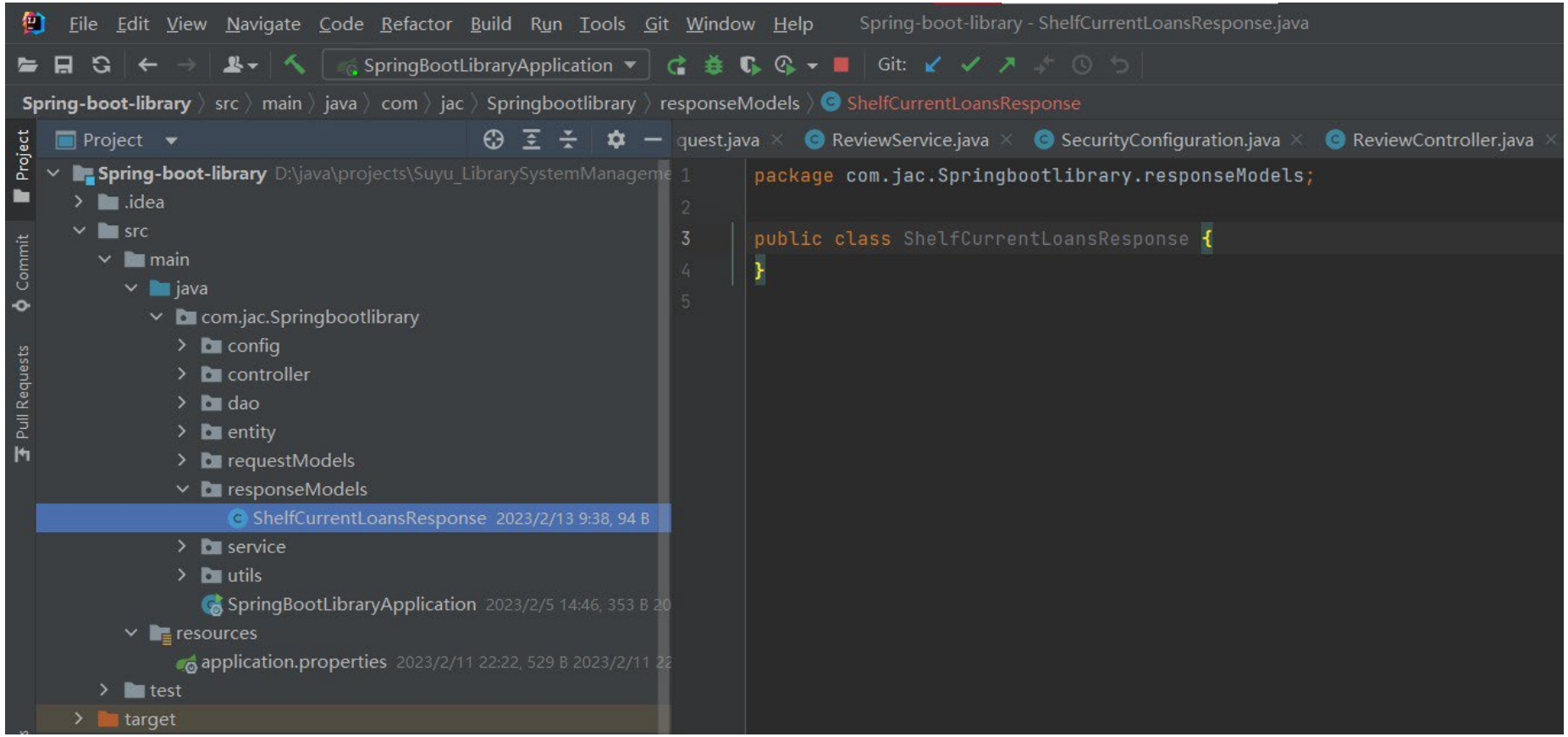
Step 2 Inside responseModels, right click and choose “new”, “Java class”



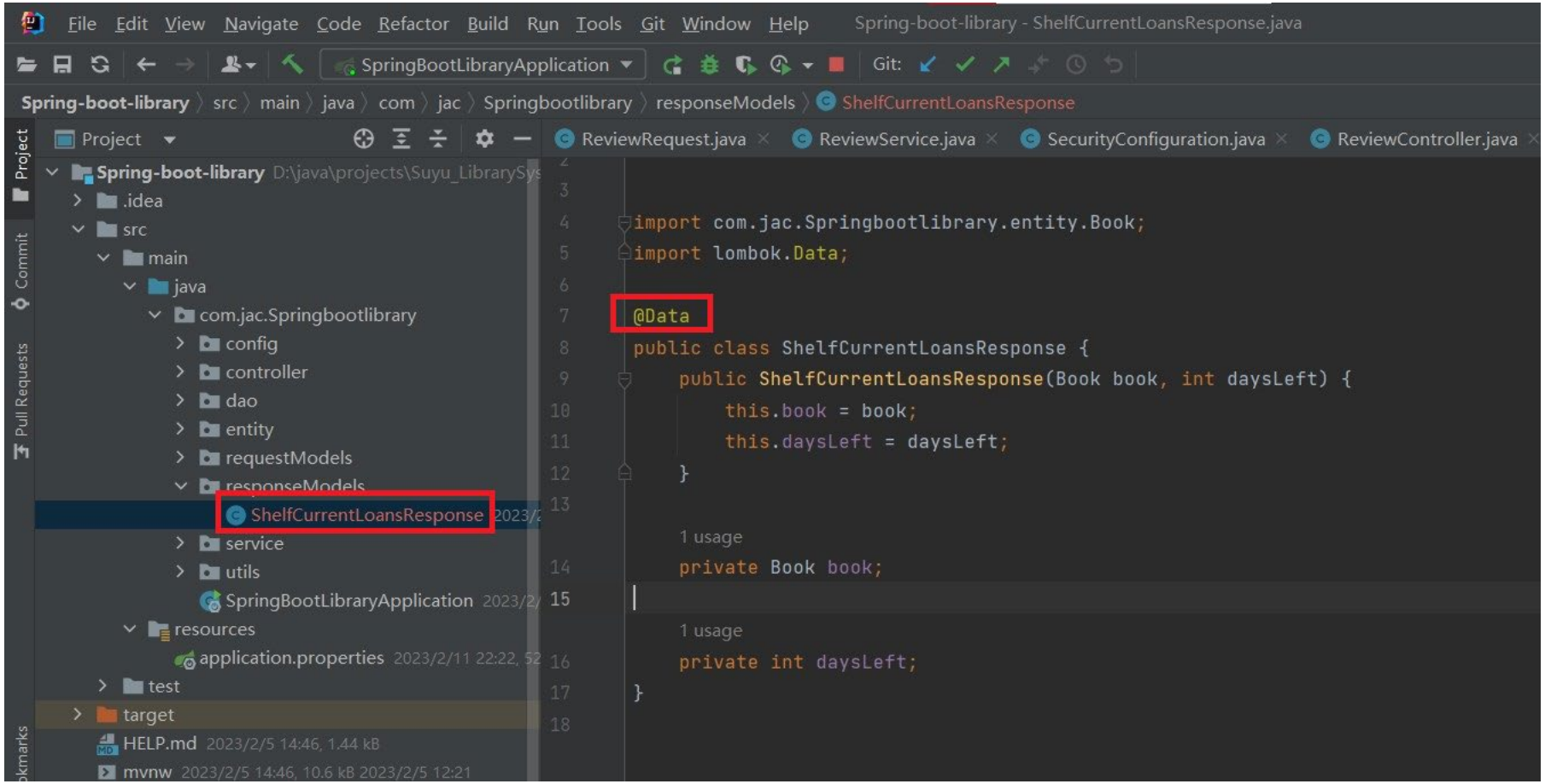


Choose “Class”, input the class name “ShelfCurrentLoansResponse”.

We will get this class:

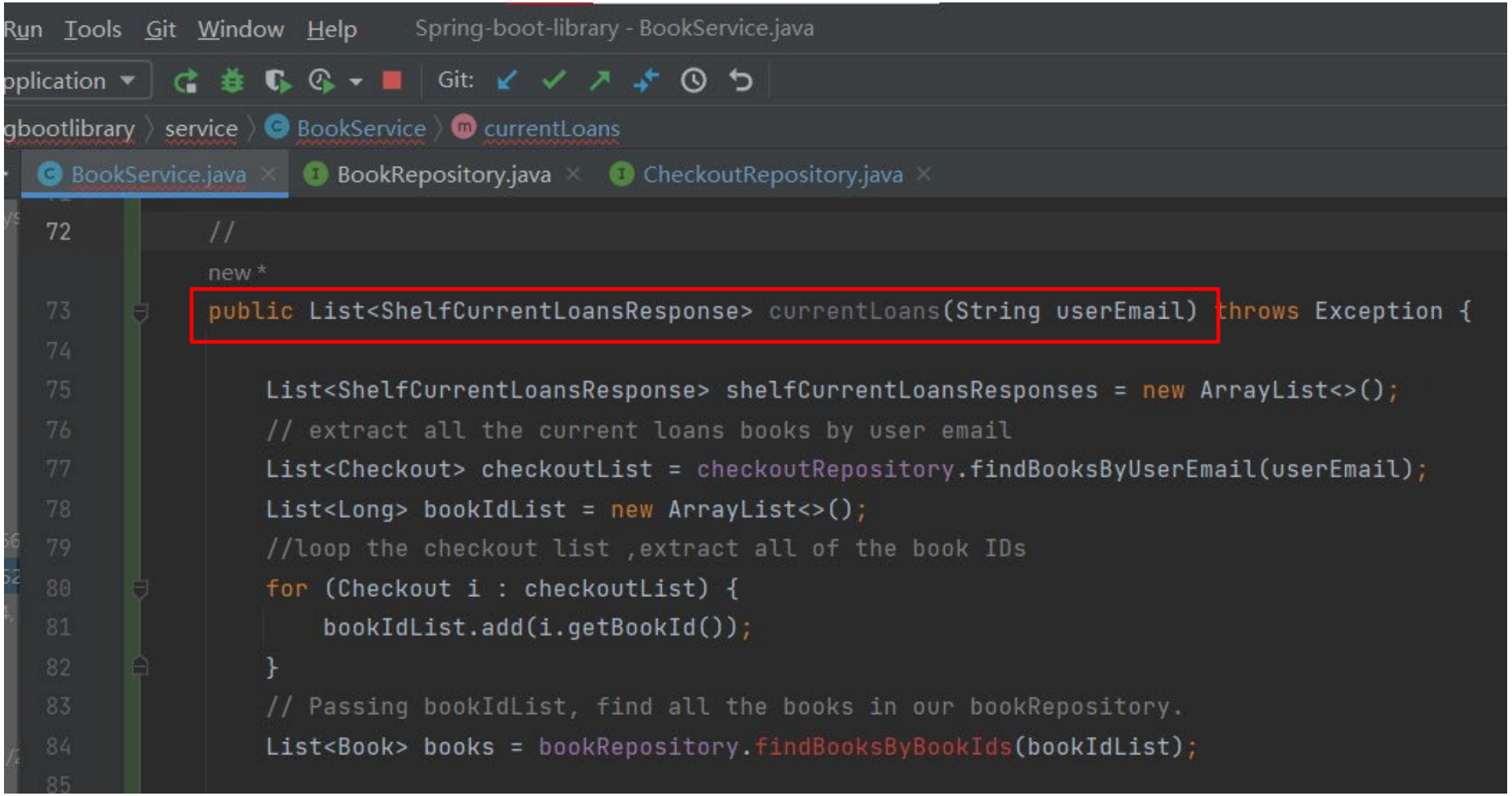


Step 3， add @Data annotation and add properties for this class. And create its two properties : “book” and “daysleft”.



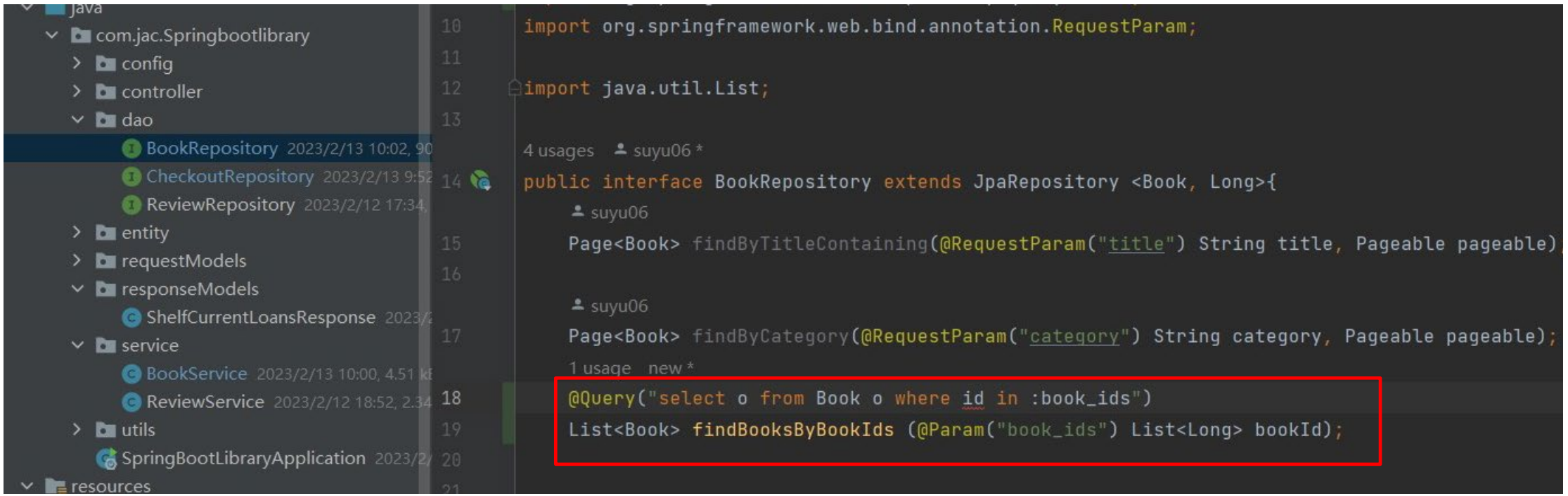
5. SPRINGBOOT CURRENT LOANS SERVICE

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

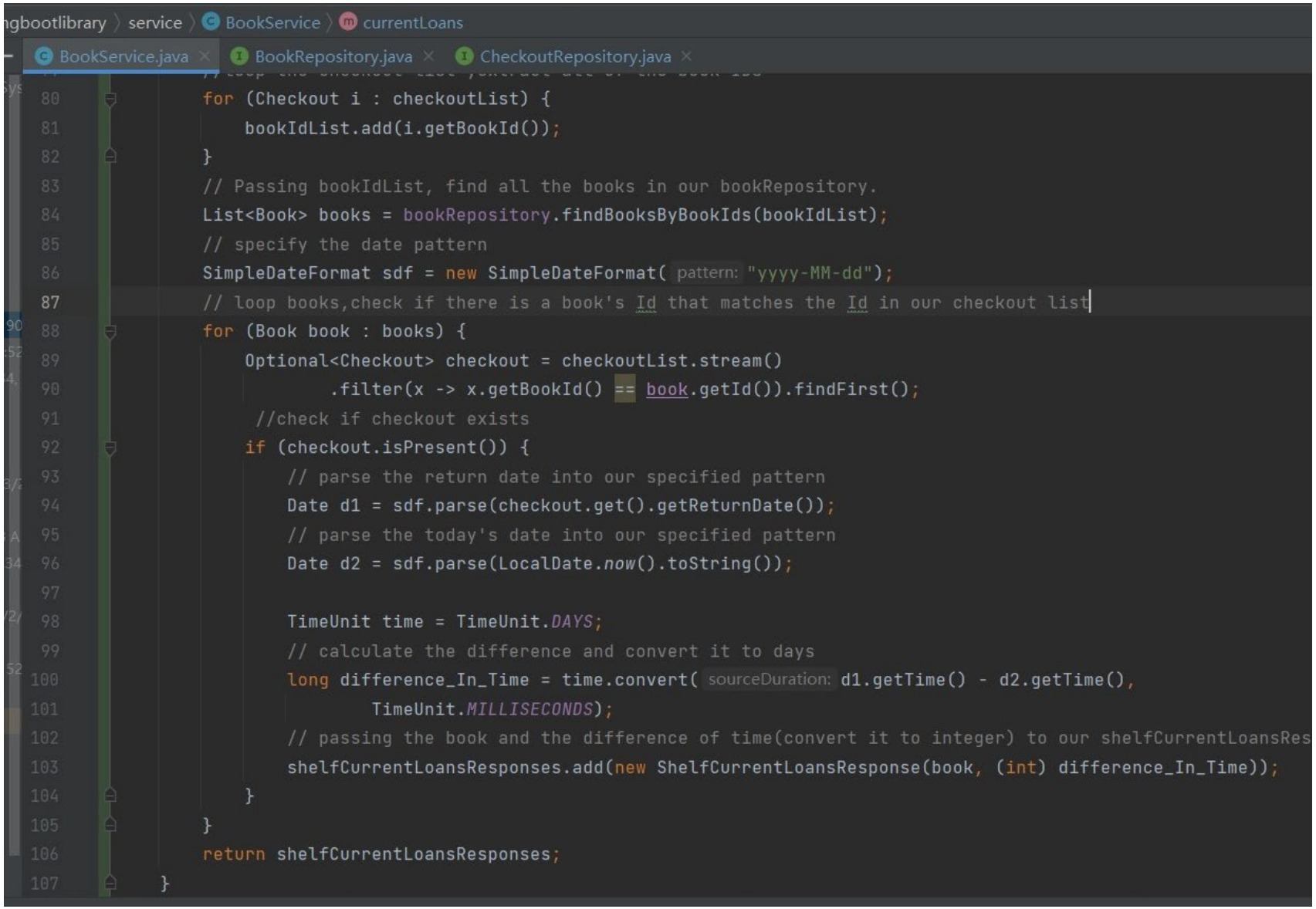


6. SPRINGBOOT QUERY ANNOTATION

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

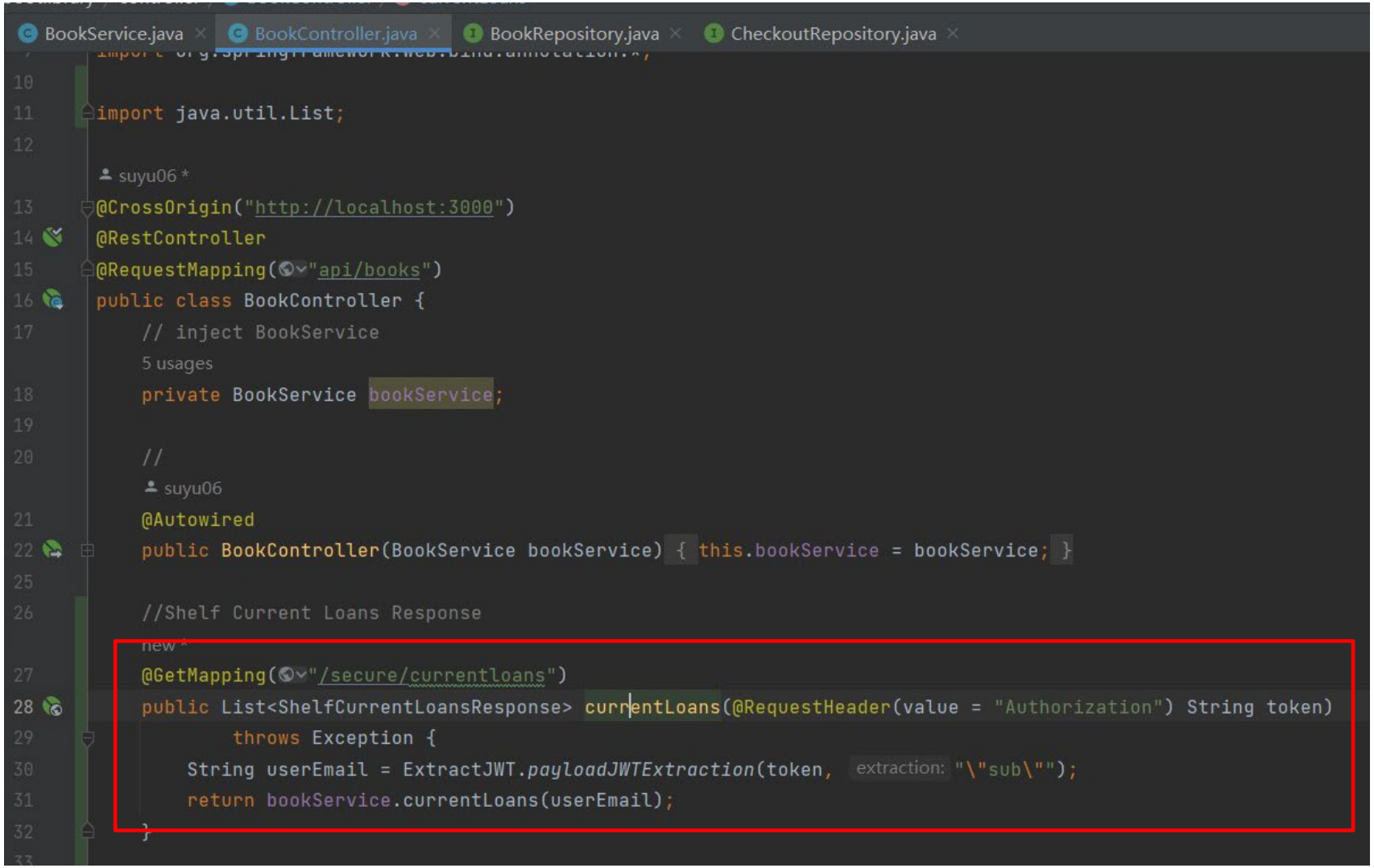


7. SPRINGBBOT CURRENT LOANS SERVICE IMPLEMENTATION



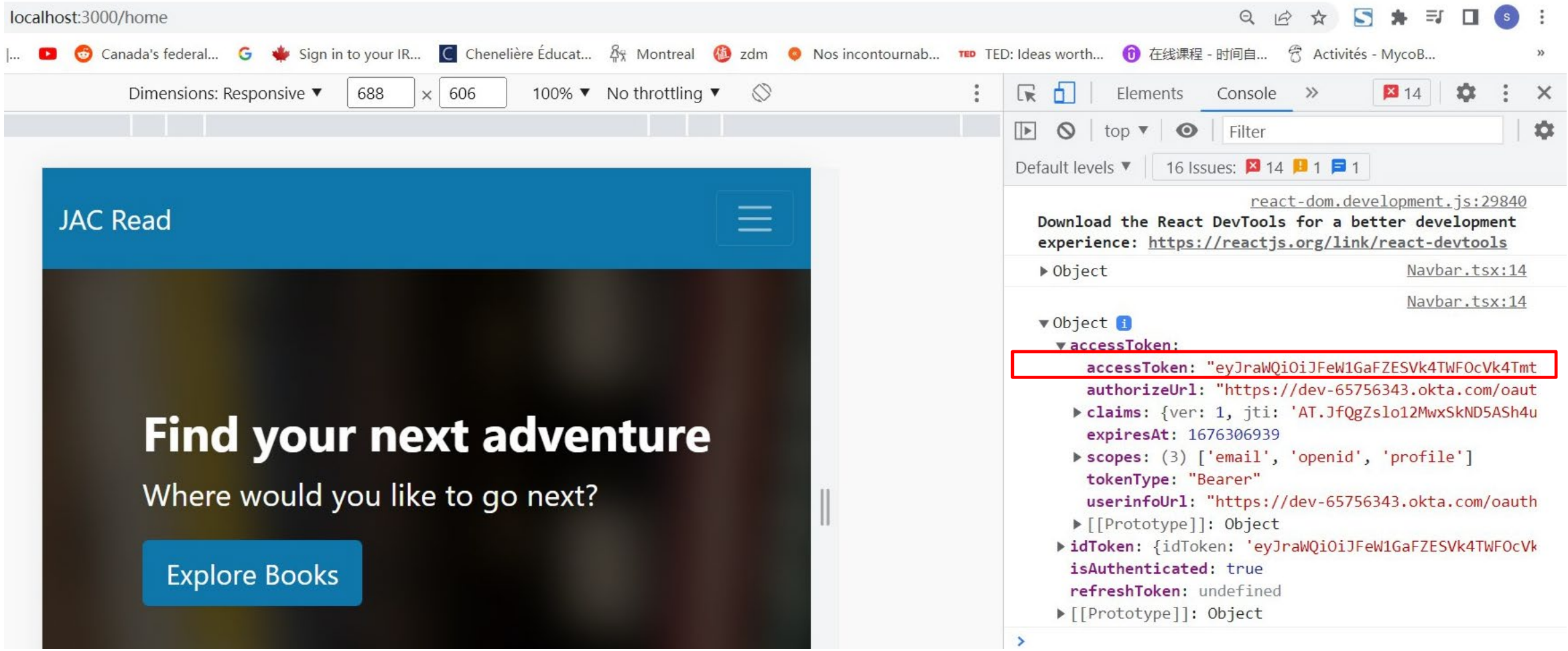
Back to BookService class, ShelfCurrentLoansResponse(), We specify the date type, and find the book, then calculate the days left

8. SPIRNGBOOT CONTROLLER ENDPOINT

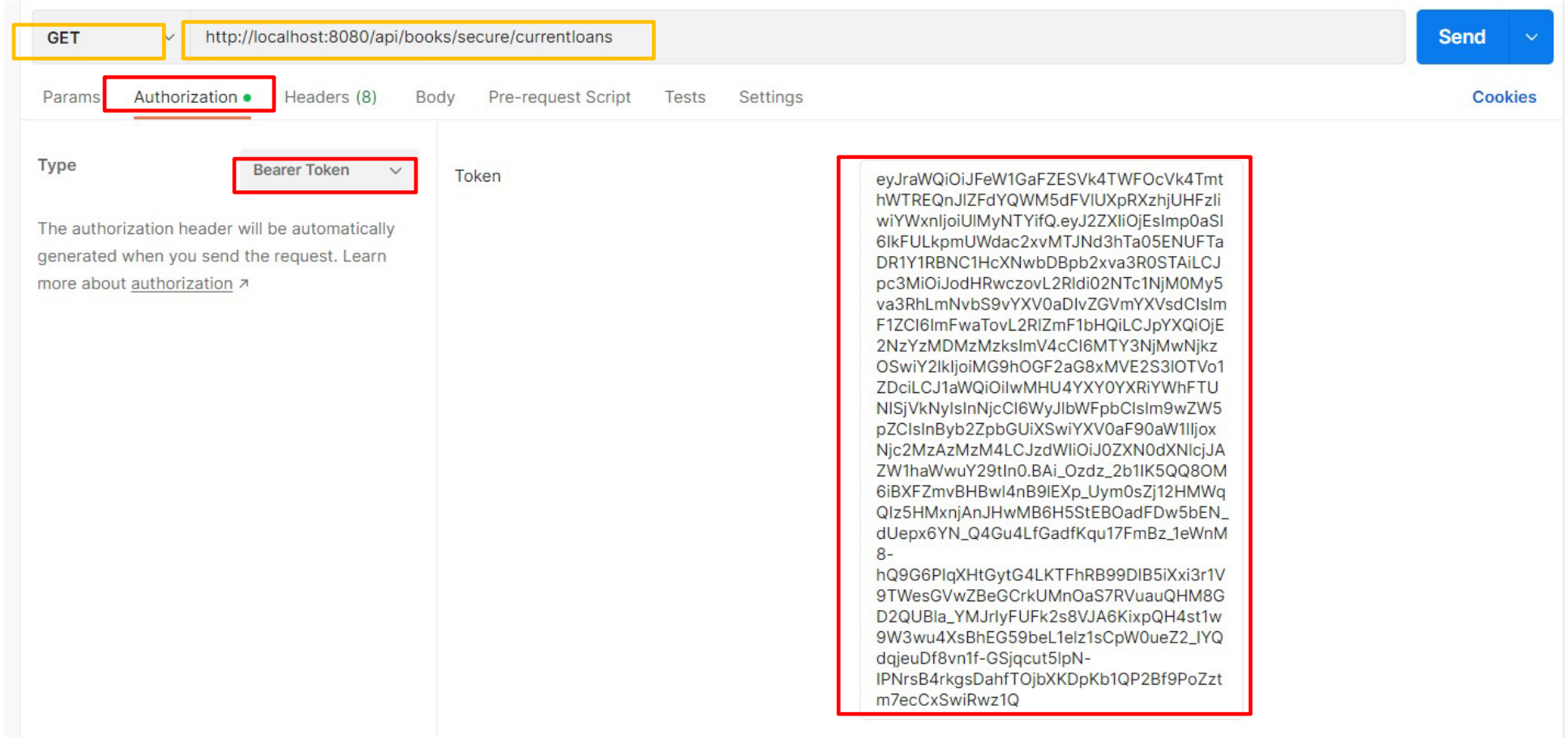


Go to BookController, Use Get method to create API for current loans.

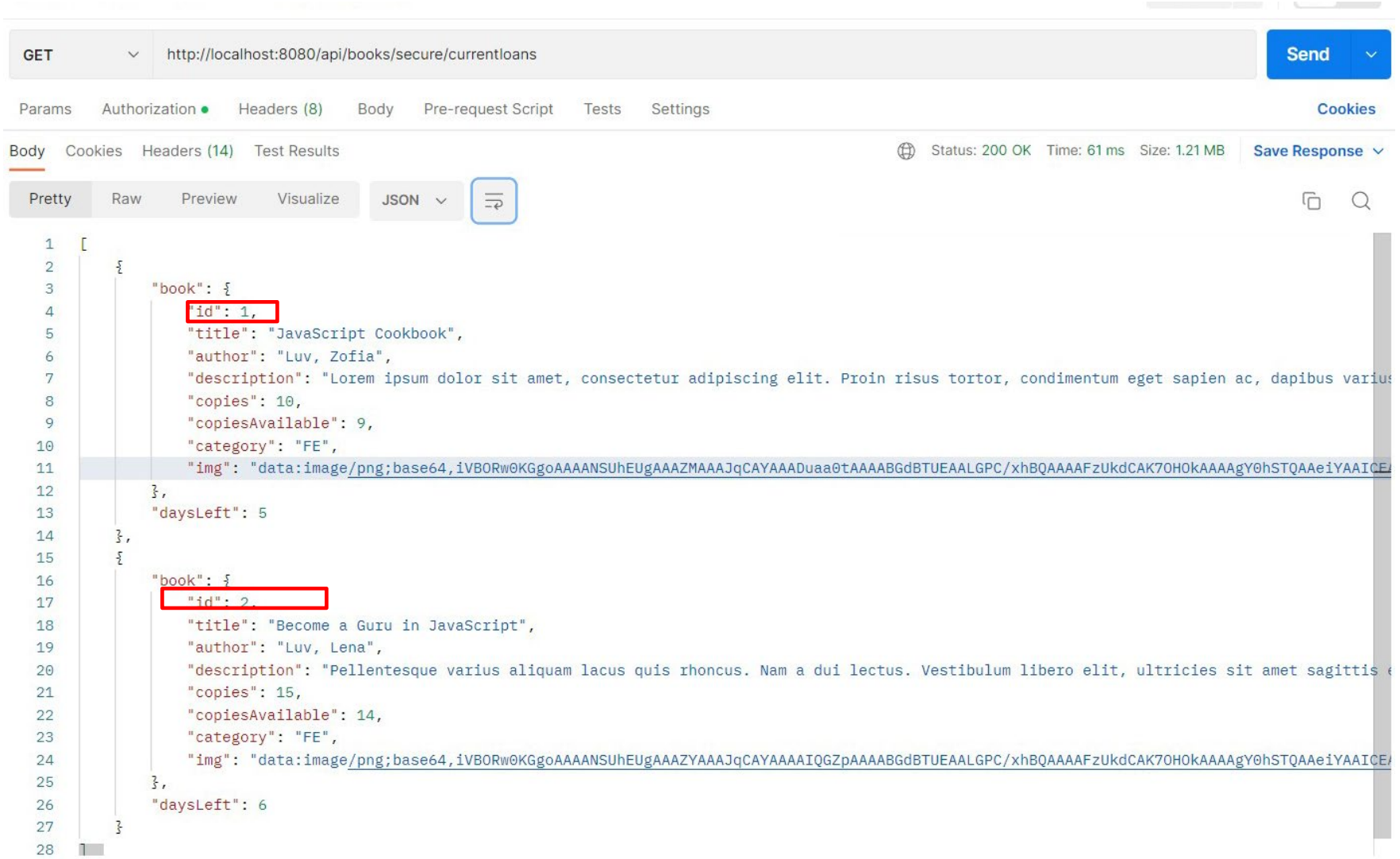
Let’s copy the accessToken in the console and test our API in Postman.



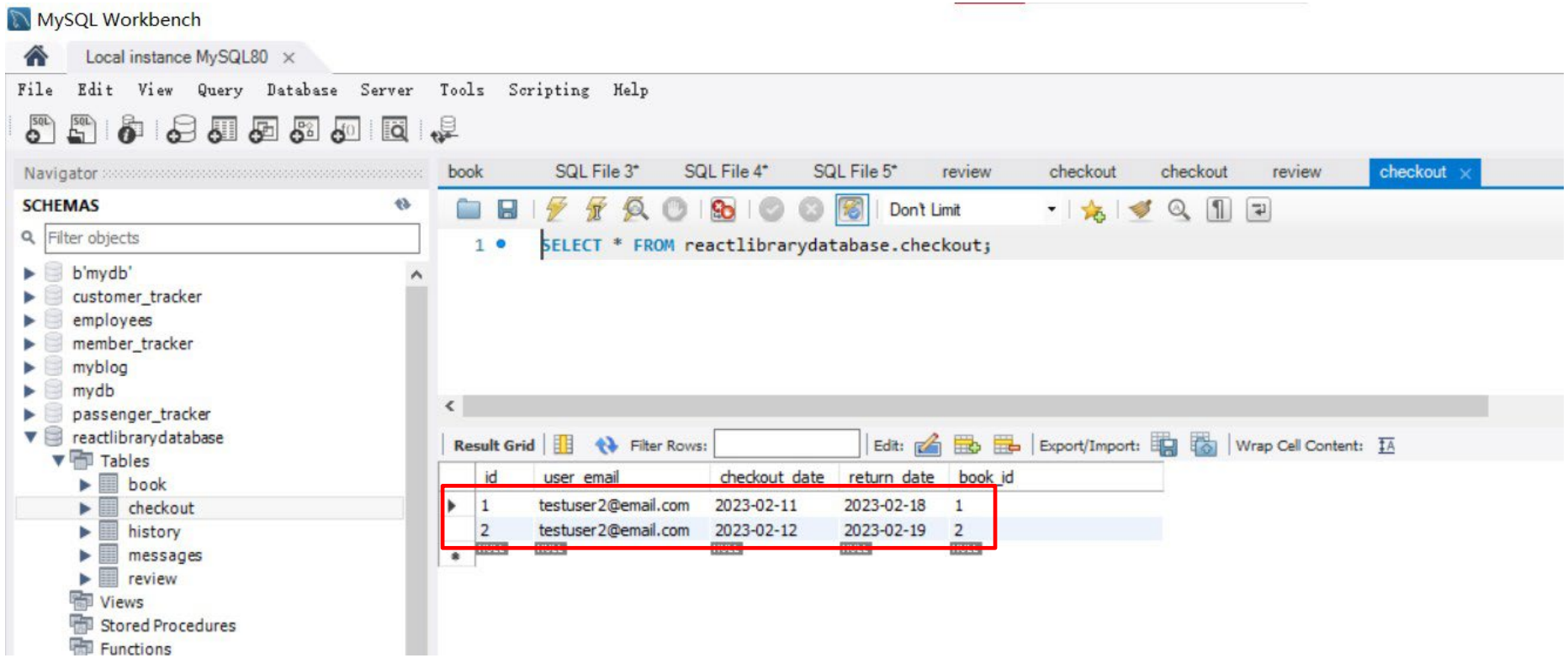
Choose “GET”, type http://localhost:8080/api/books/secure/currentloans.
Click on “Authorization”, choose “Bearer Token” in type , paste access token in the token box, then click on “Send”.



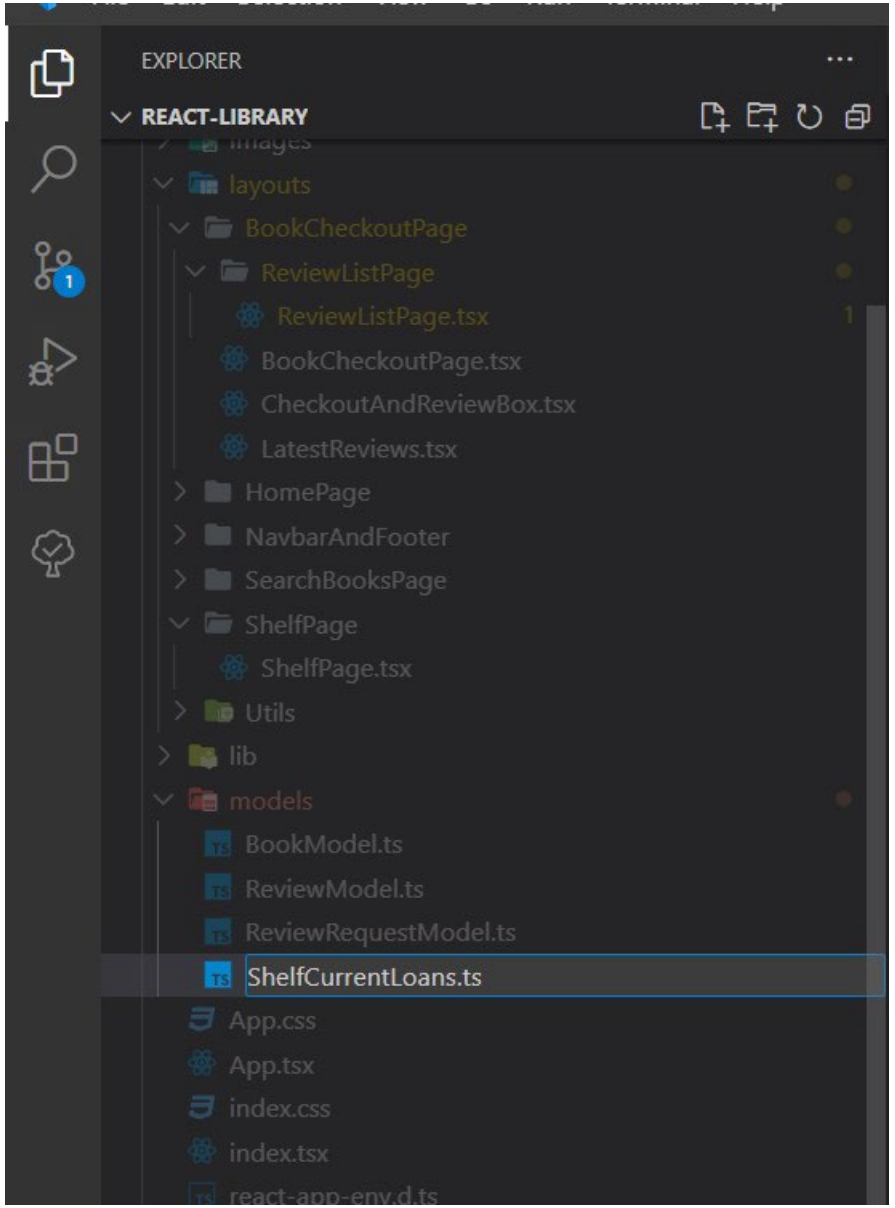
We will get two results returned.



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

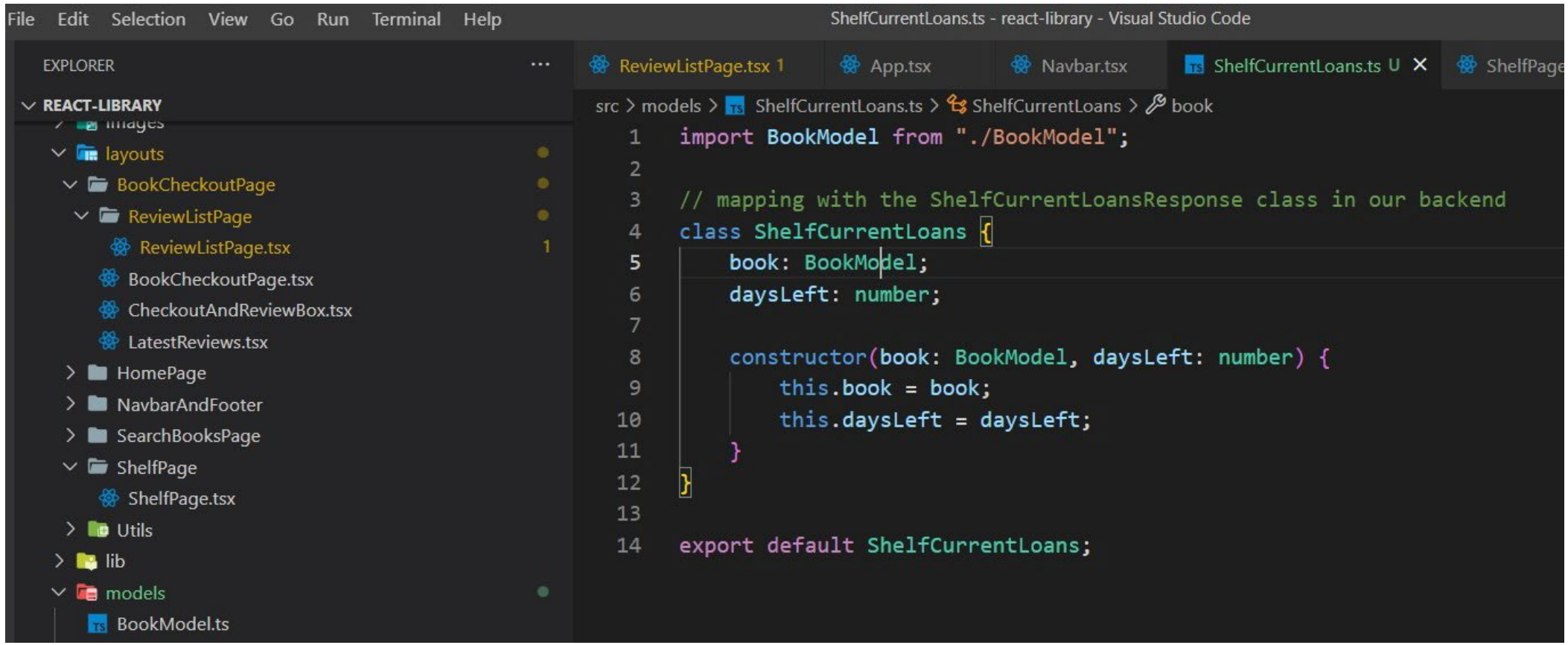


9. REACT SHELF CURRENT LOANS MODEL

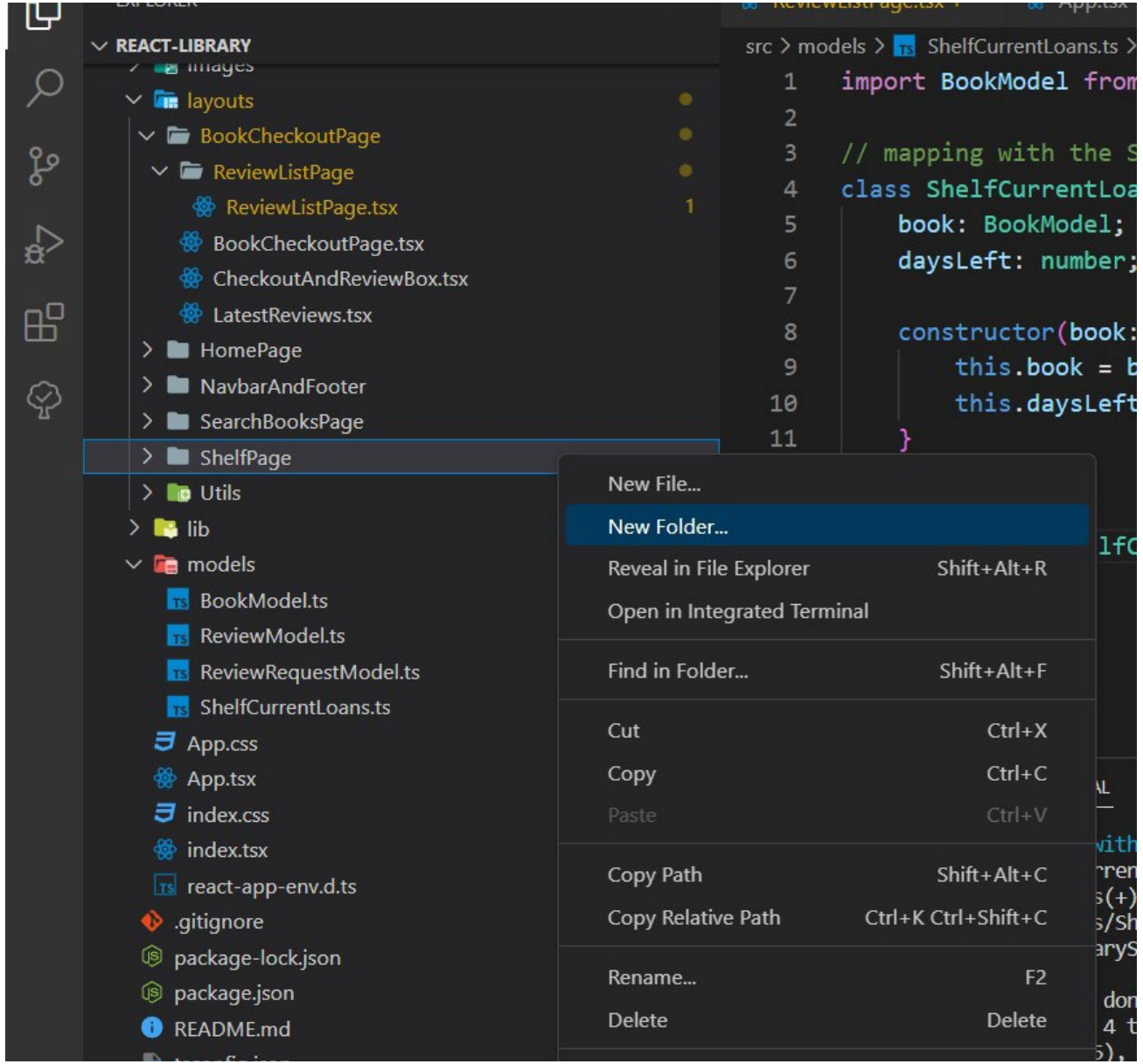


Step 1 Right click on models and choose “new file”.
Name it ShelfCurrentLoans.ts

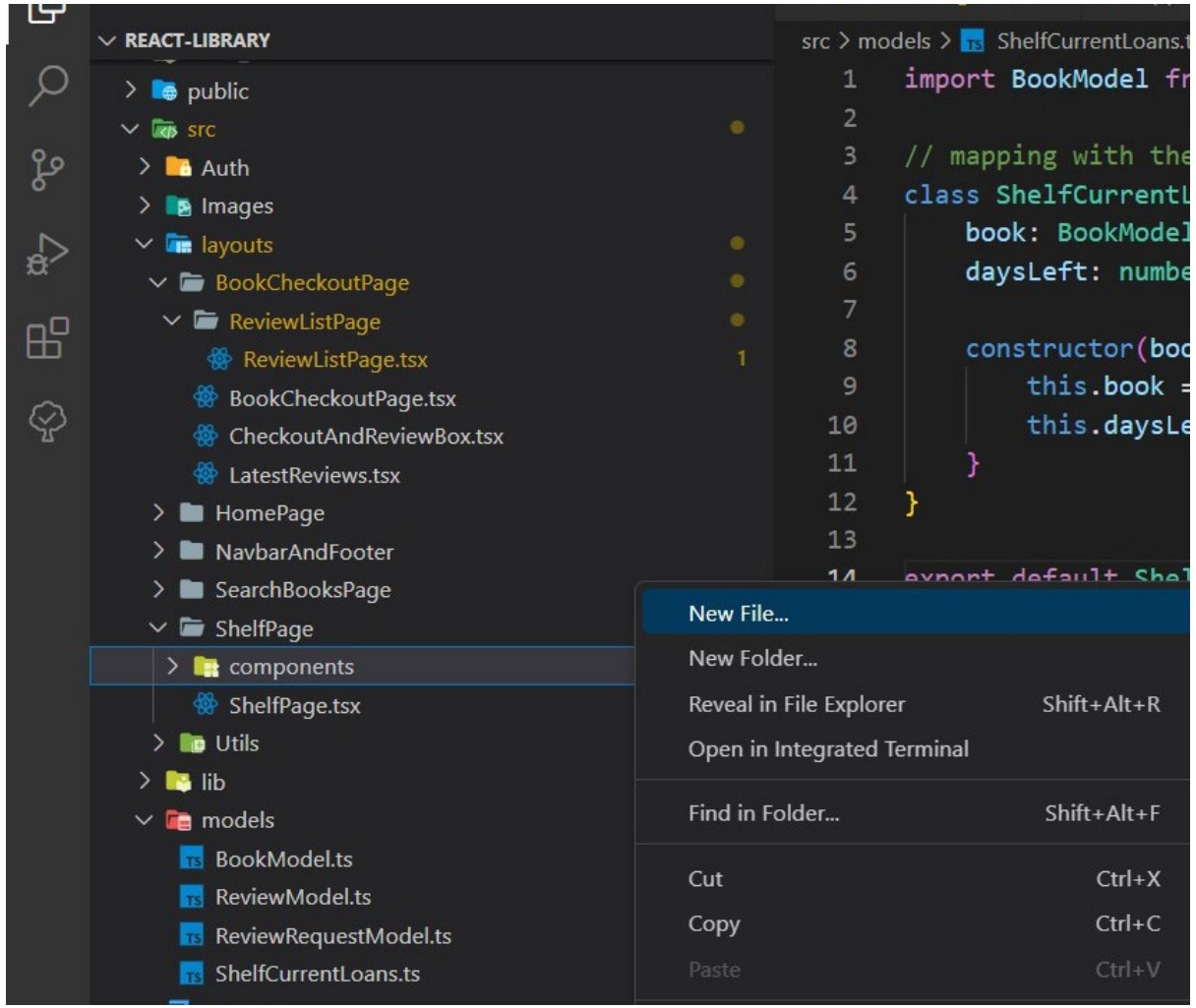
Step2 create the properties which are mapping to the ShelfCurrentLoansResponse class in our backend,



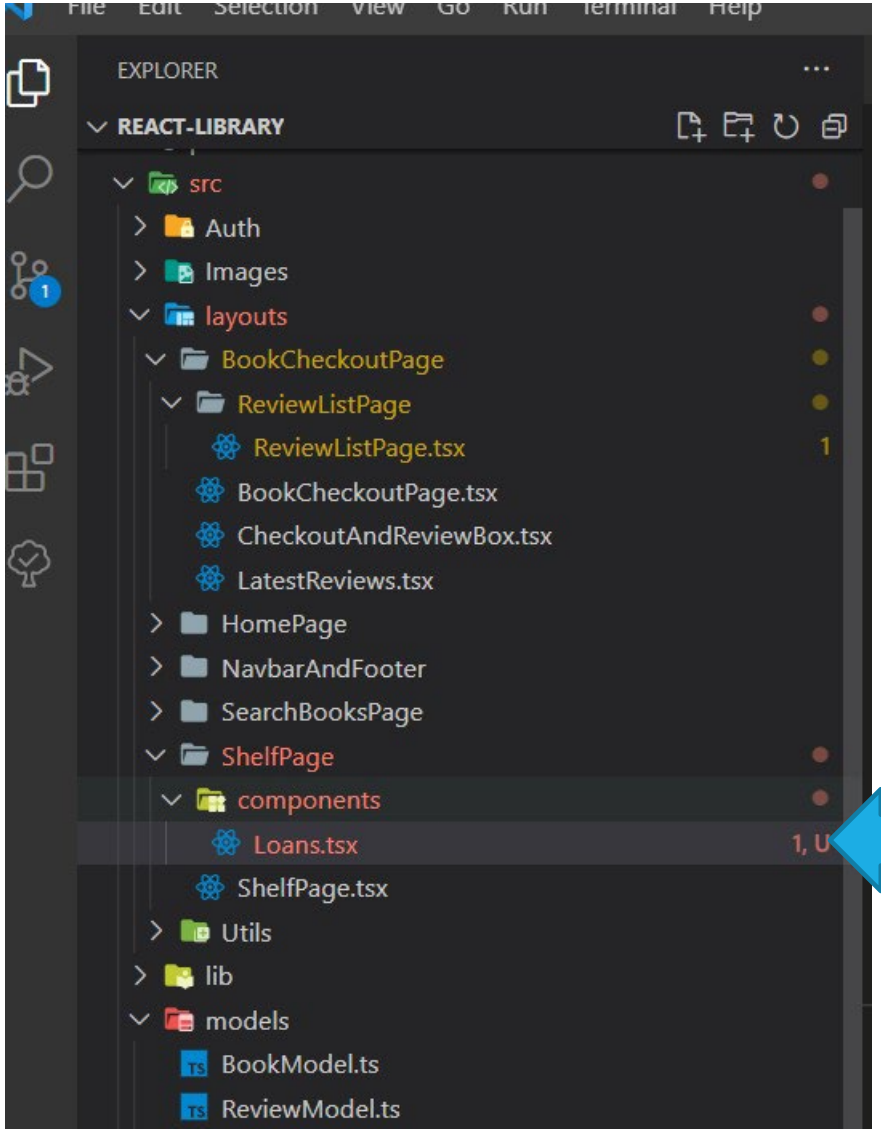
10. REACT NEW LOANS COMPONENT



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

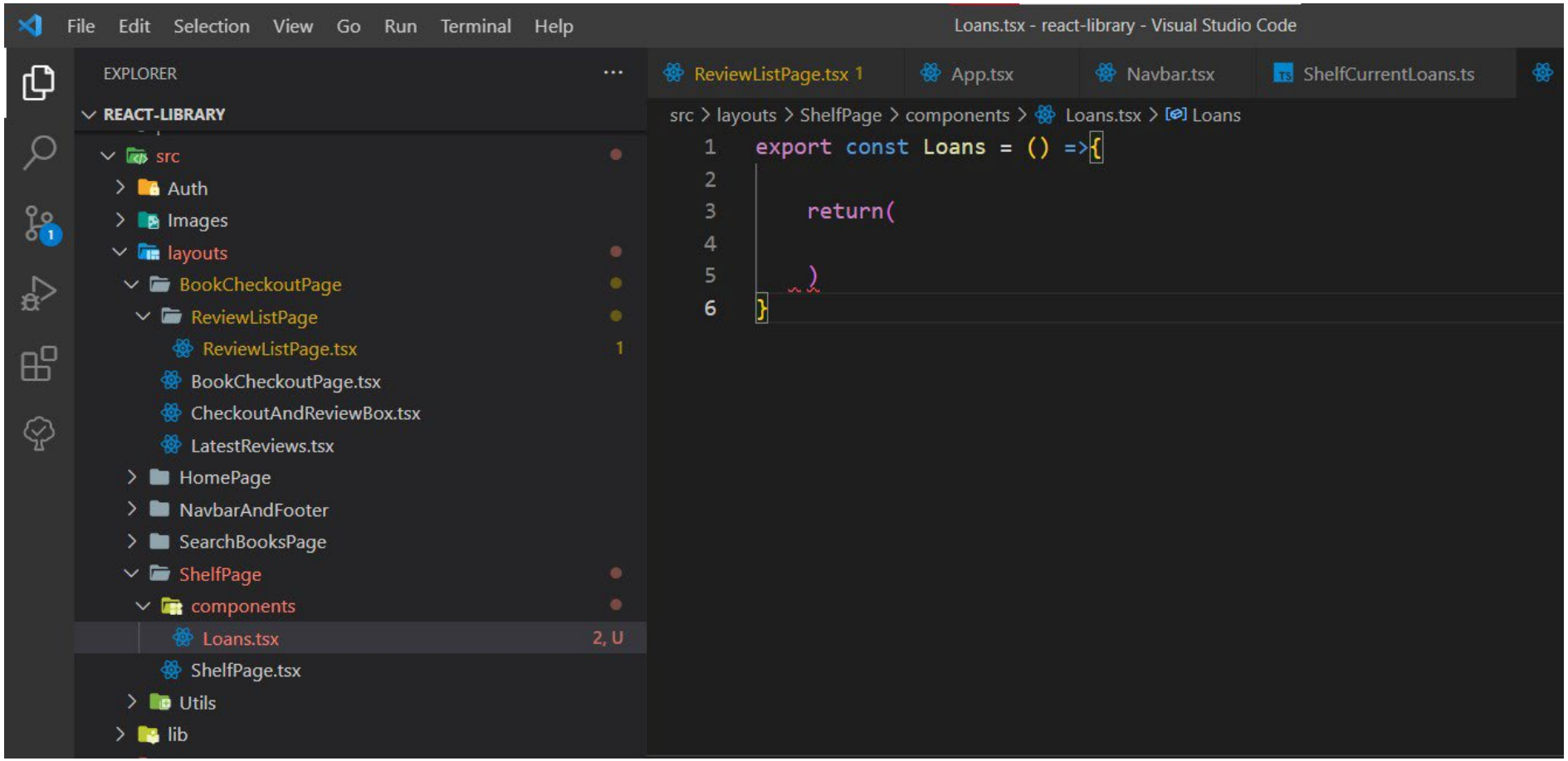


Stpe 2
Right click “components”, choose “New file.

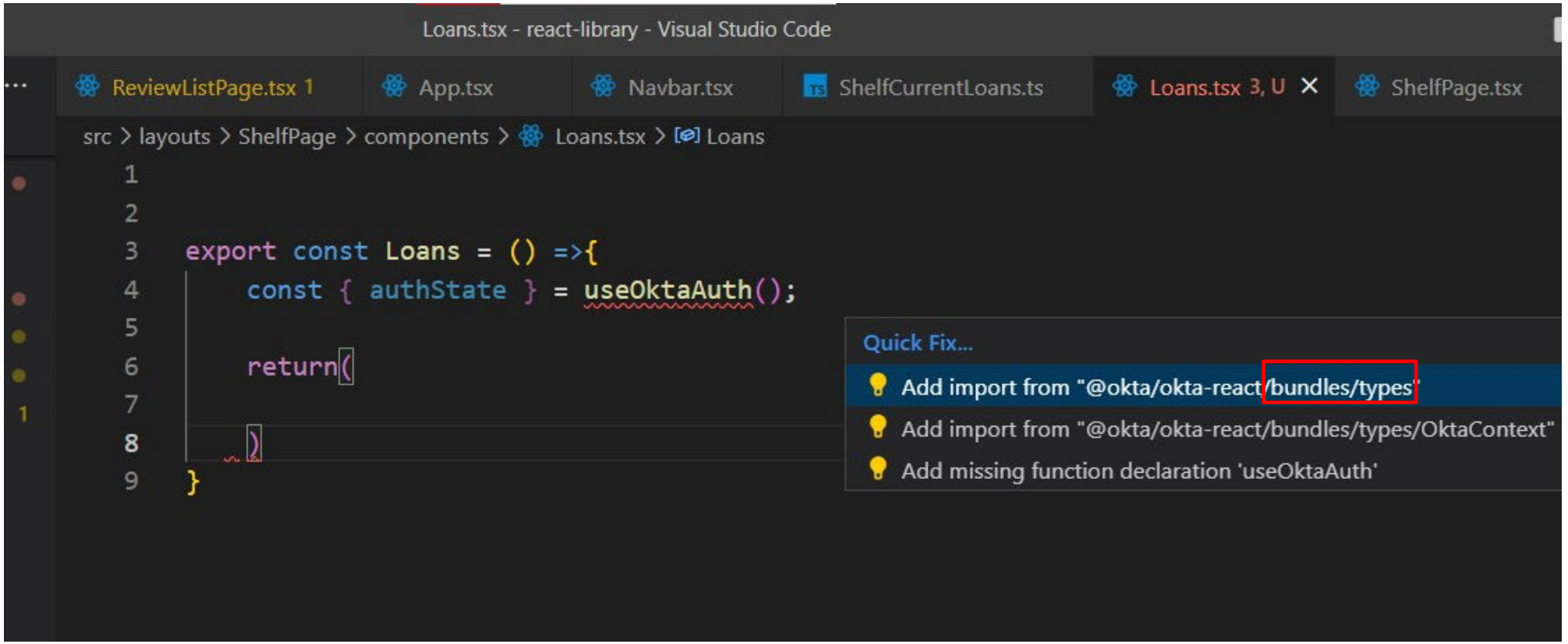


Name it Loans.tsx

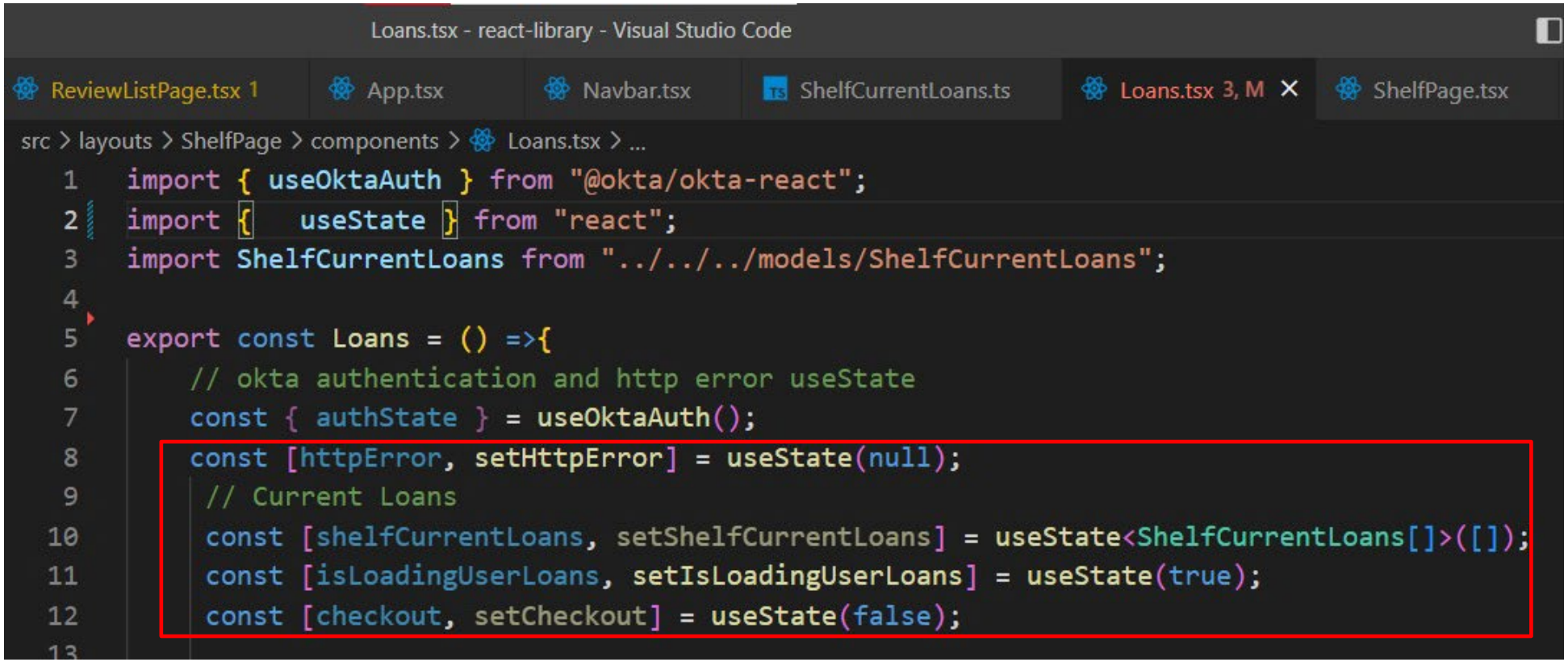
Stpe 3 create a export const structure as always.

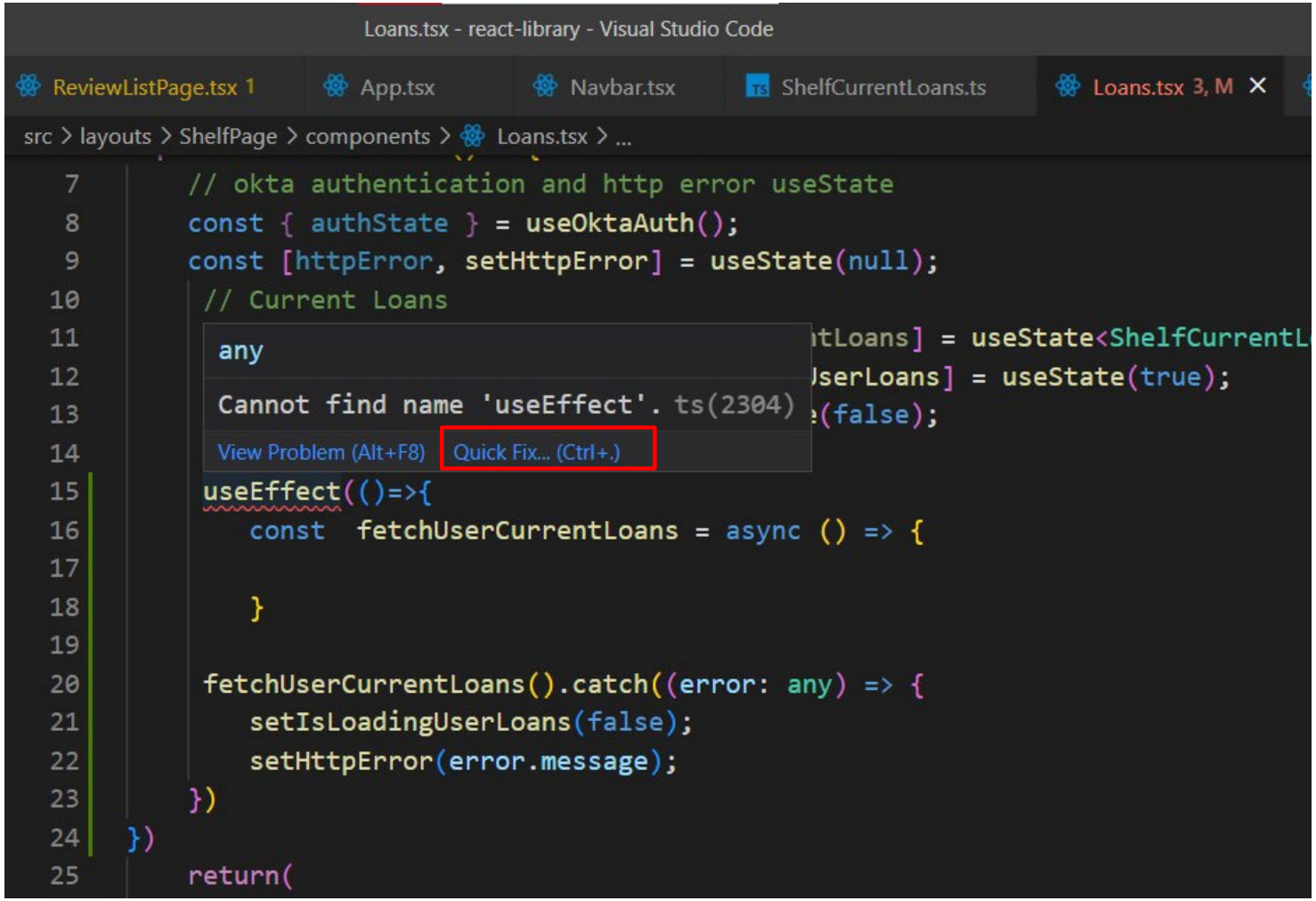


Step 4, create authState and import okta.
Remember to remove the “/bundles/types” from the path generated automatically.



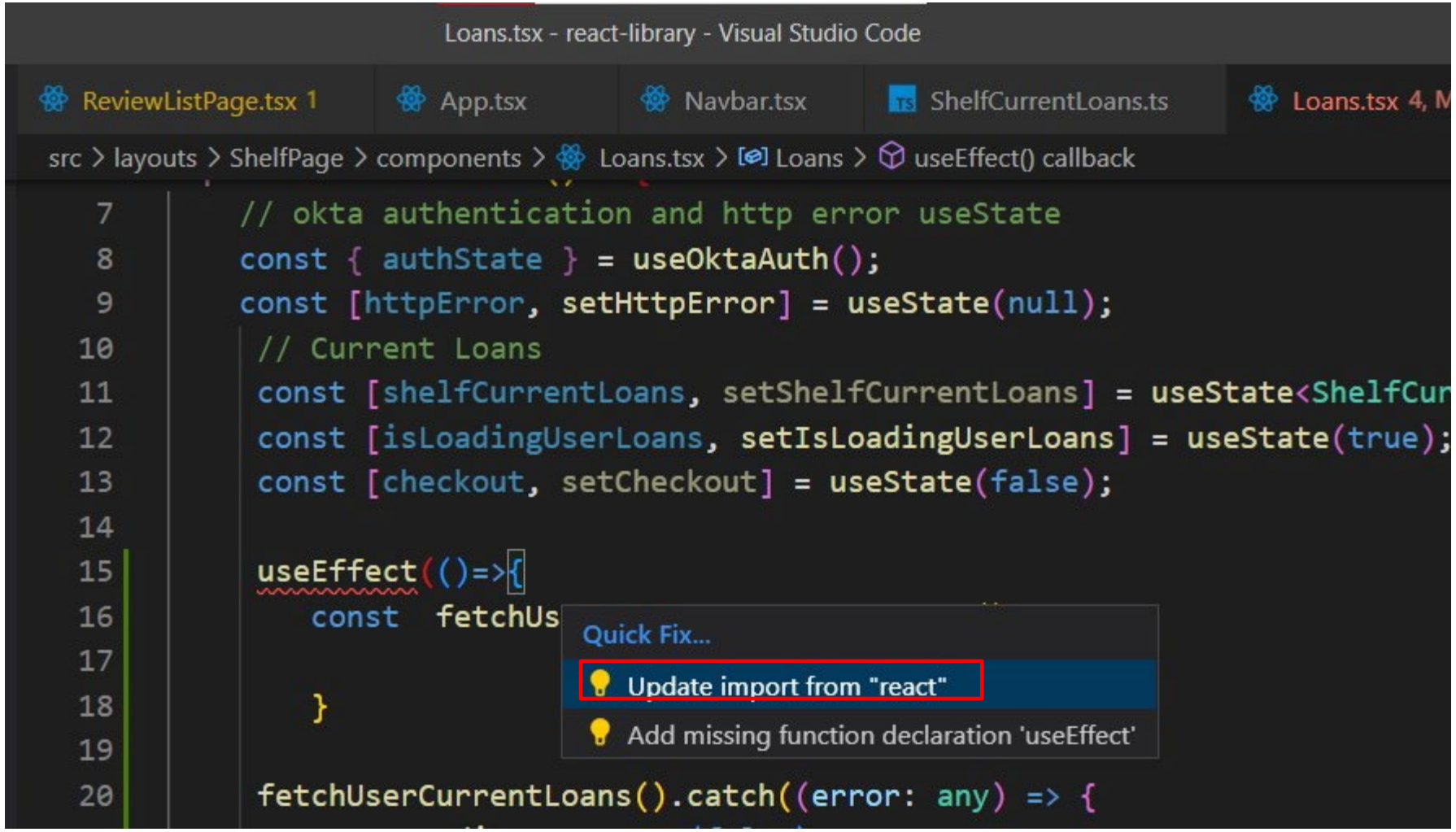
Step 5 create useState.



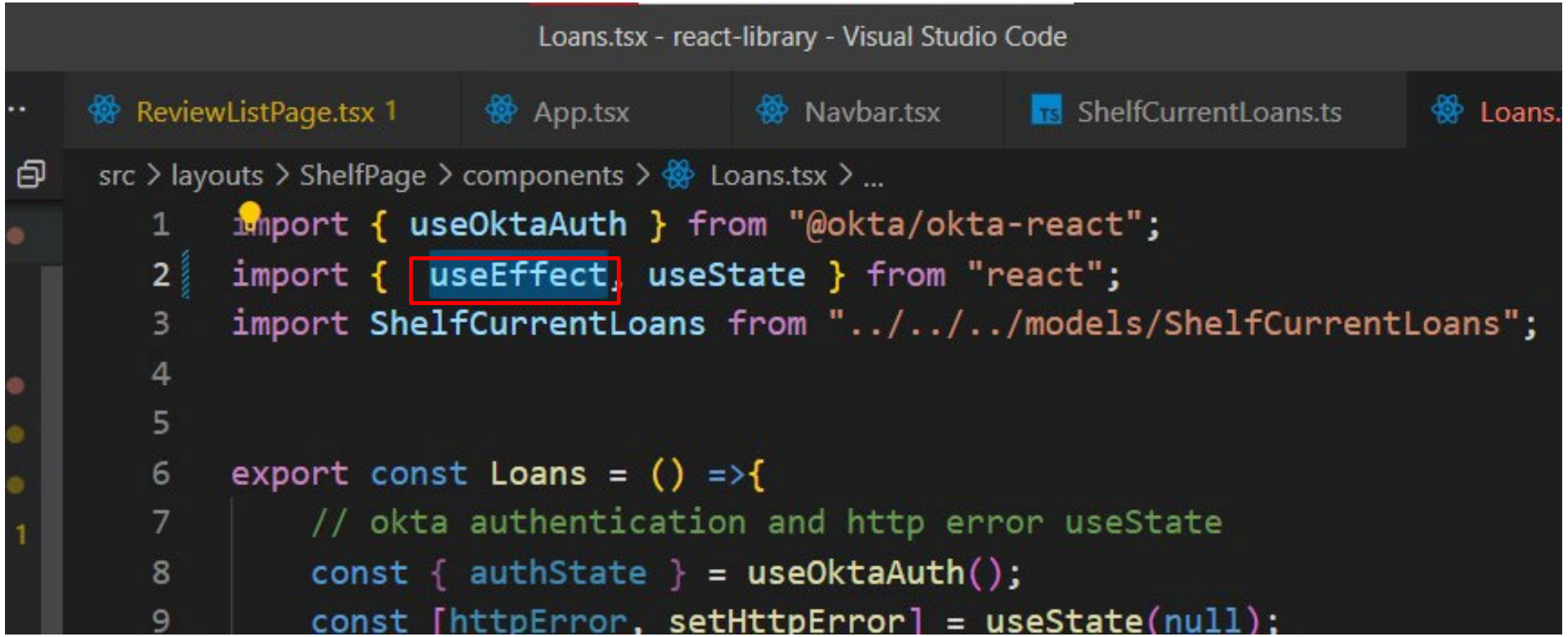


Step 6 create useEffect structure, which contains async function fetchUserCurrentLoans and its error catch function.
Import useEffct :
Hover over the useEffect, choose “Quick Fix”.

Click on “Update import from react”.



useEffect is imported now.



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

```
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] = useSt
11       const [isLoadingUserLoans, setIsLoadingUserLoans] = use
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 USEEFFECT

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          }
```

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 th
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

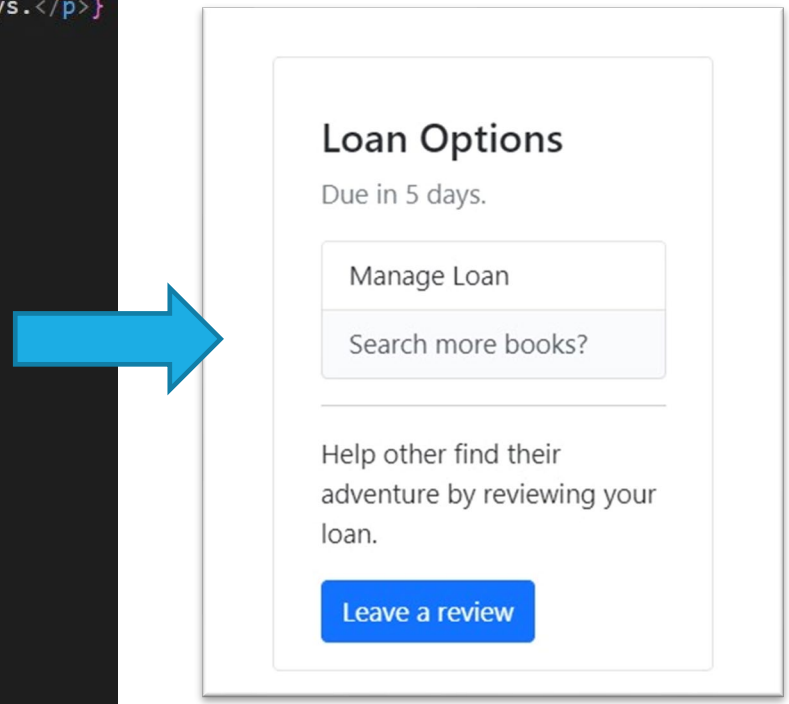
```
//html and css
return (
  <div>
    {/* Desktop */}
    <div className="d-none d-lg-block mt-2">
      {shelfCurrentLoans.length > 0 ? (
        <>
        {/*user has borrowed books*/}
          <h5>Current Loans: </h5>

          {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 className="card col-3 col-md-3 container d-flex">
                  <div className="card-body">
```

Step 1 image part.

```jsx
<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${shelfCurrentLoan.book.id}`}
        >
          Manage Loan
        </button>
        <Link to={"search"} className="list-group-item list-group-item-action">
          Search more books?
        </Link>
      </div>
    </div>
    <hr />
    <p className="mt-3">Help other find their adventure by reviewing your loan.</p>
    <Link className="btn btn-primary" to={`/checkout/${shelfCurrentLoan.book.id}`}>
      Leave a review
    </Link>
  </div>
</div>
```

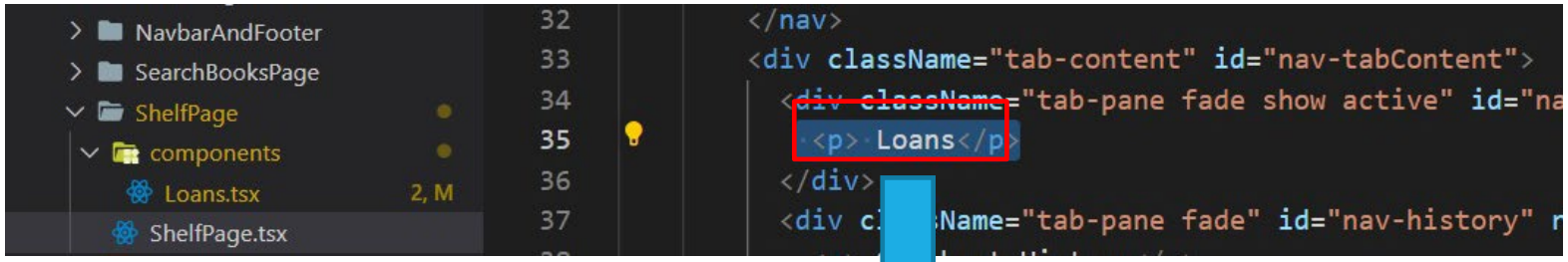
Step2 card part: loans option,manage loan, leave a review

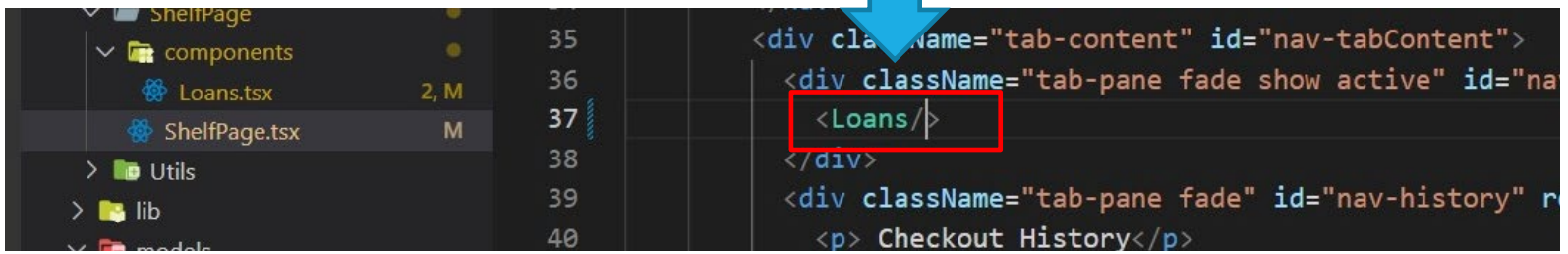


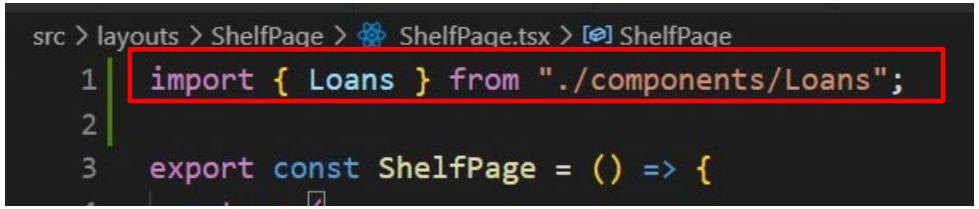
```
        ) : (
          //  user has not borrowed books yet
          <>
            <h3 className="mt-3">Currently no loans</h3>
            <Link className="btn btn-primary" to={`search`}>
              Search for a new book
            </Link>
          </>
        )}
      </div>
    </div>
  );
```

Step 3 if current user had no loan yet.

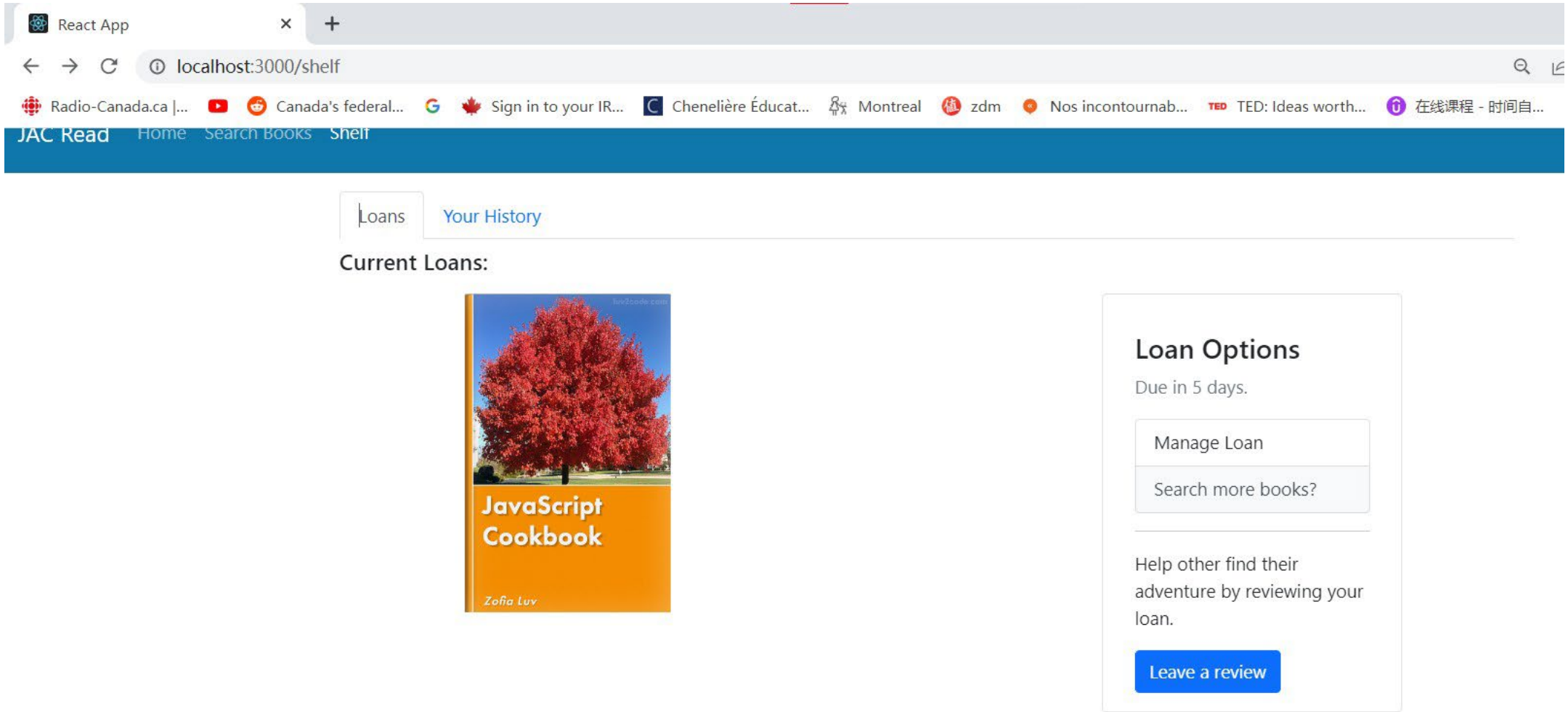
Step 4 , go to ShelfPage.tsx, replace <p>loans with loans component.
And import loans component.



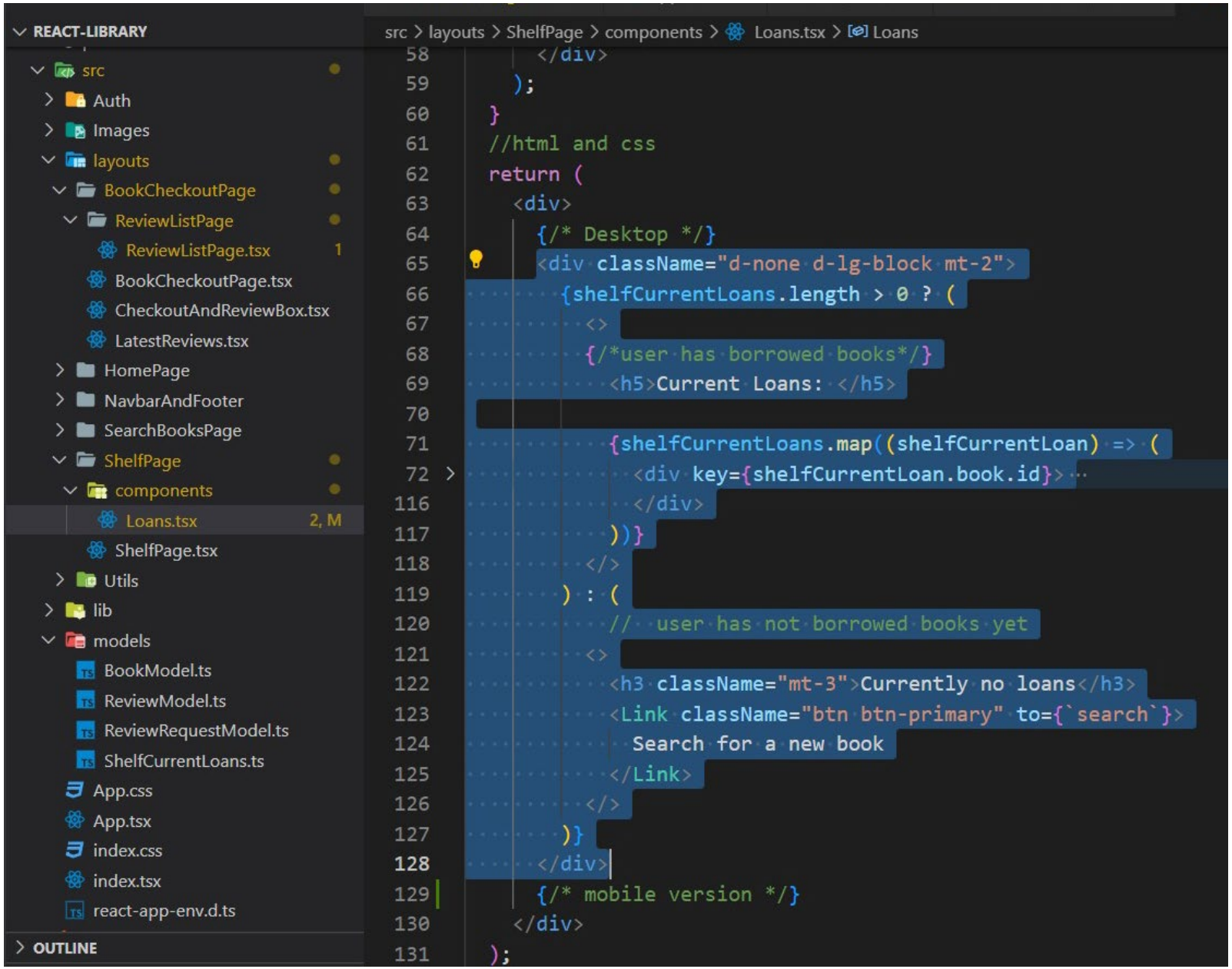




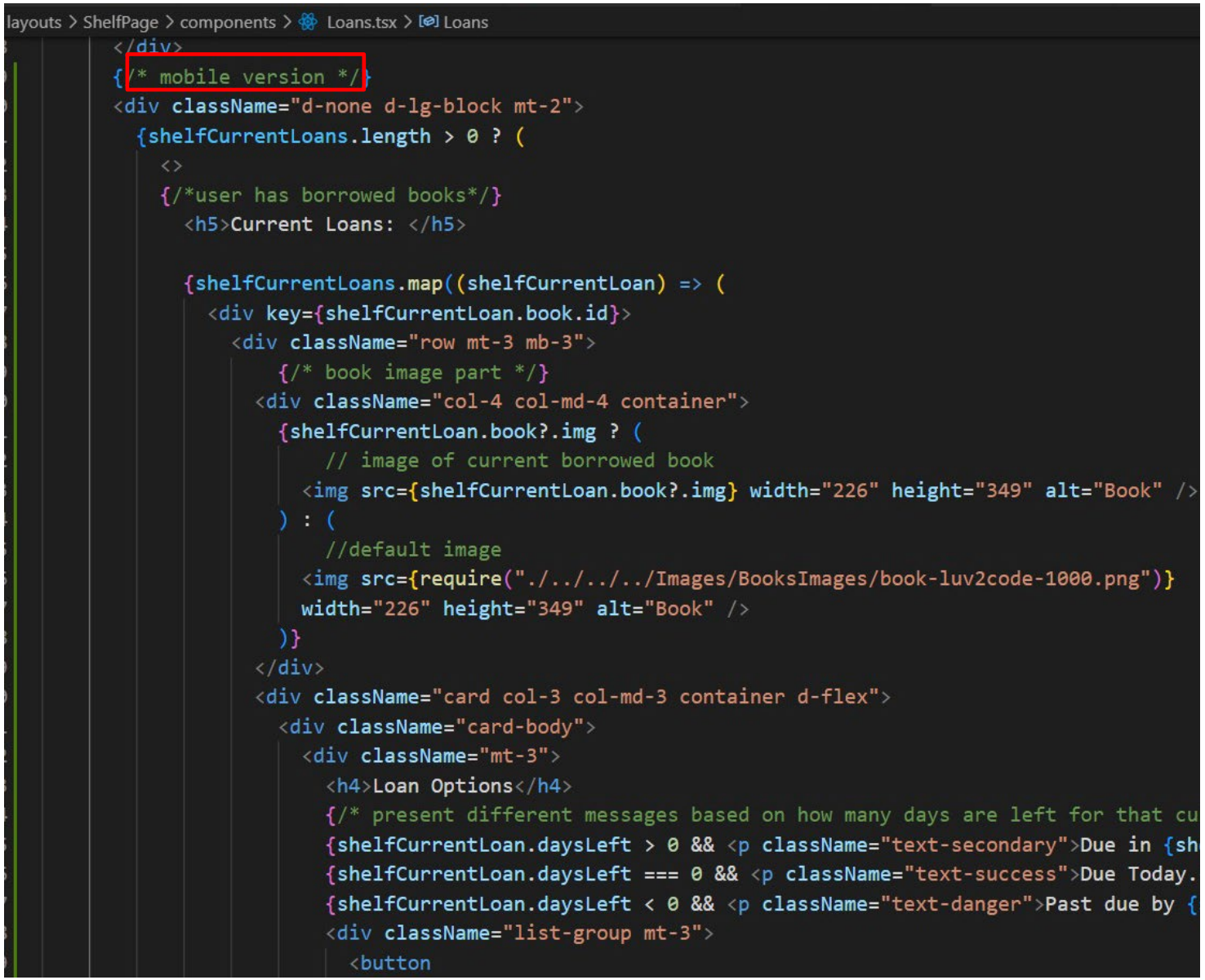
Final shelf page：



13. REACT LOANS COMPONENT MOBILE HTML/CSS

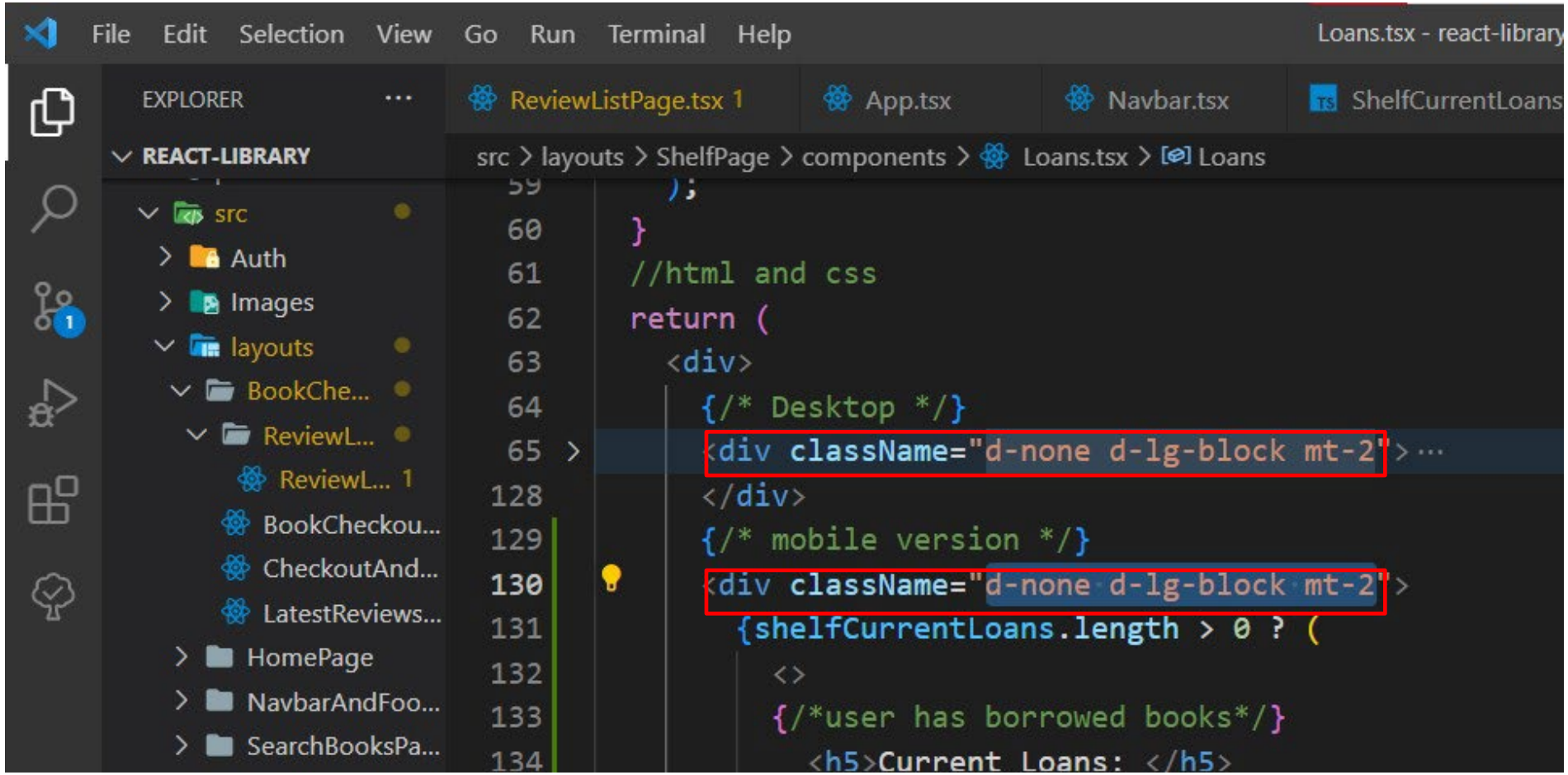


Step 1 copy desktop version code

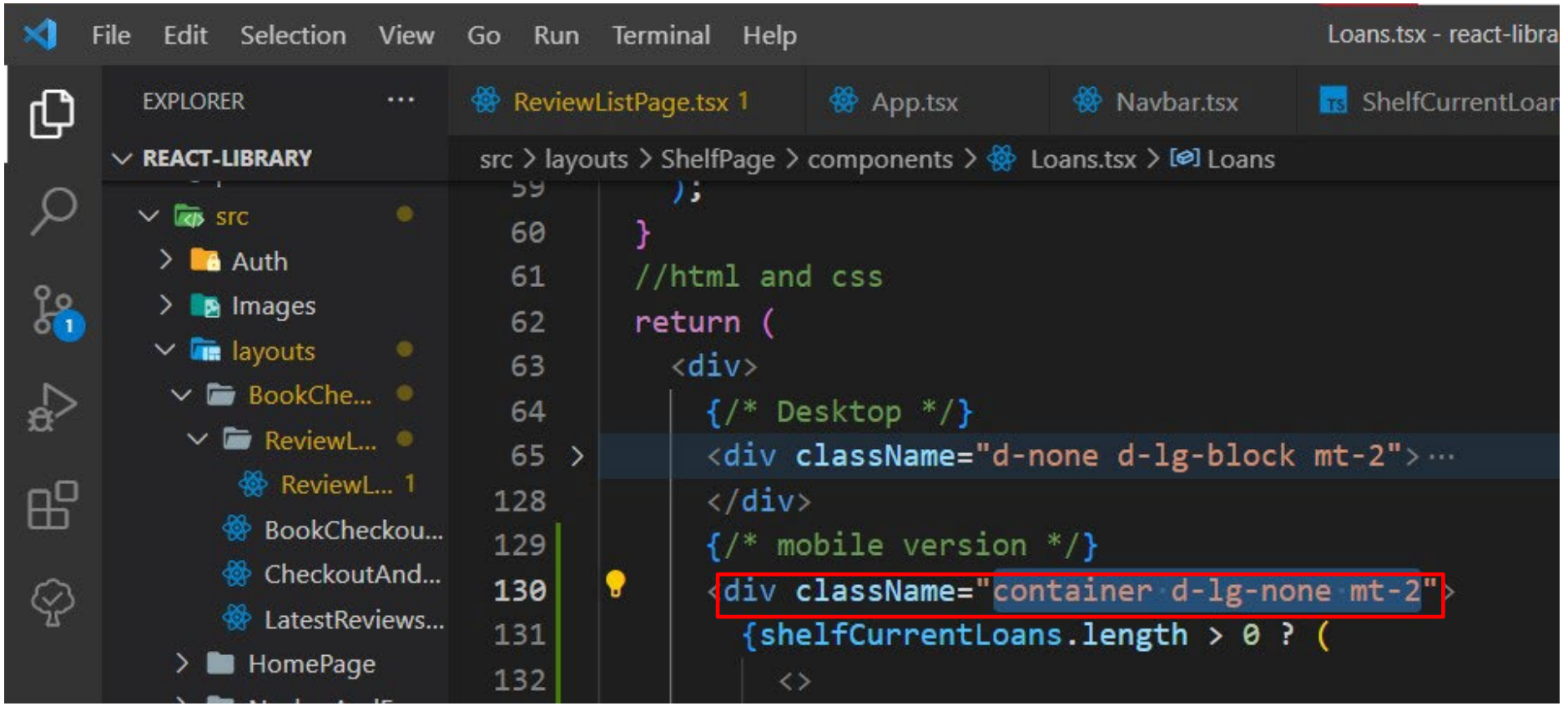


Step 2 paste it into our mobile section.

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



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

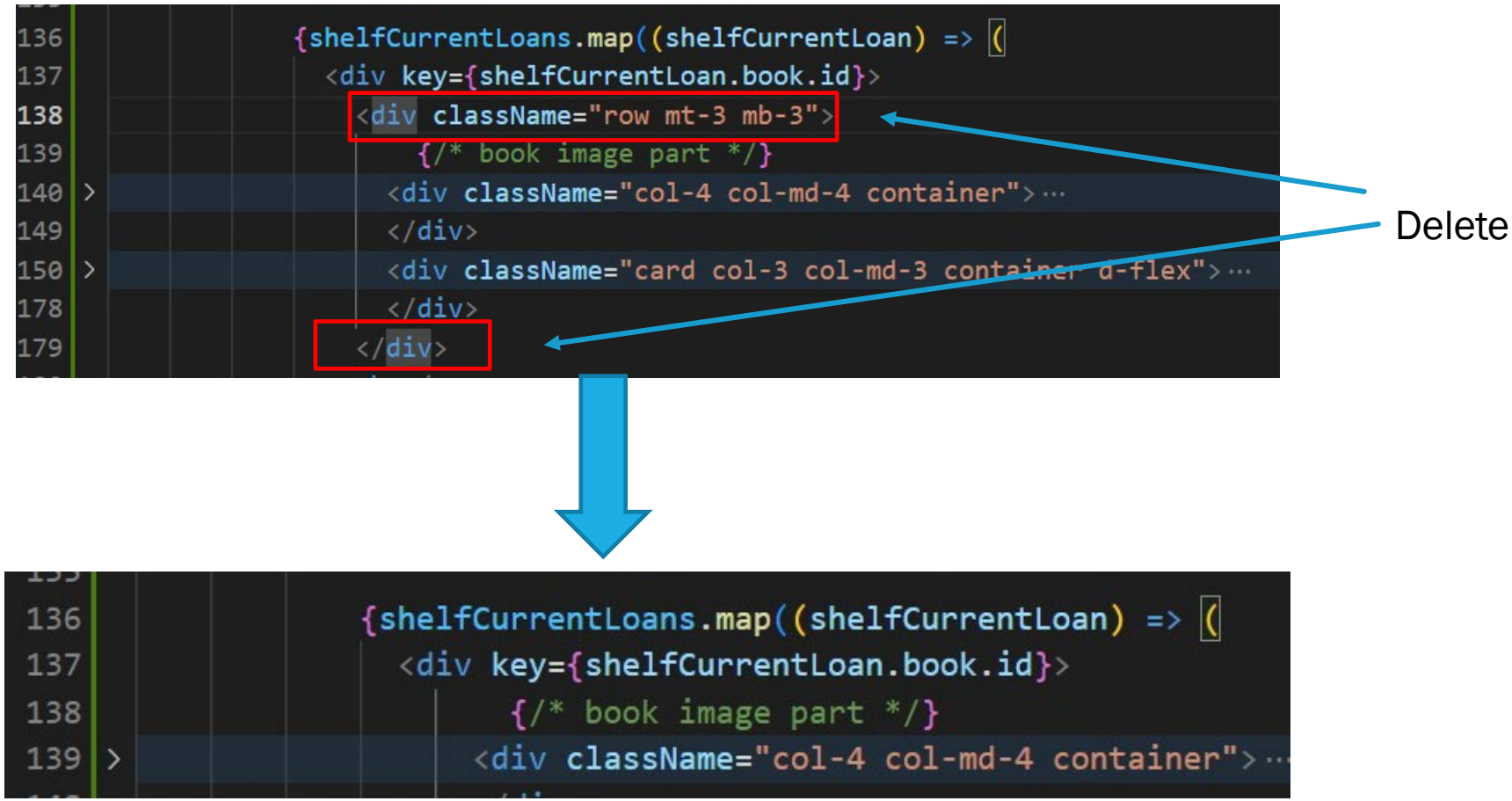


3.2 add className “mb-3” for the <h5> Current Loans<h5/>

```
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((shelfCurrentLo
```

```
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

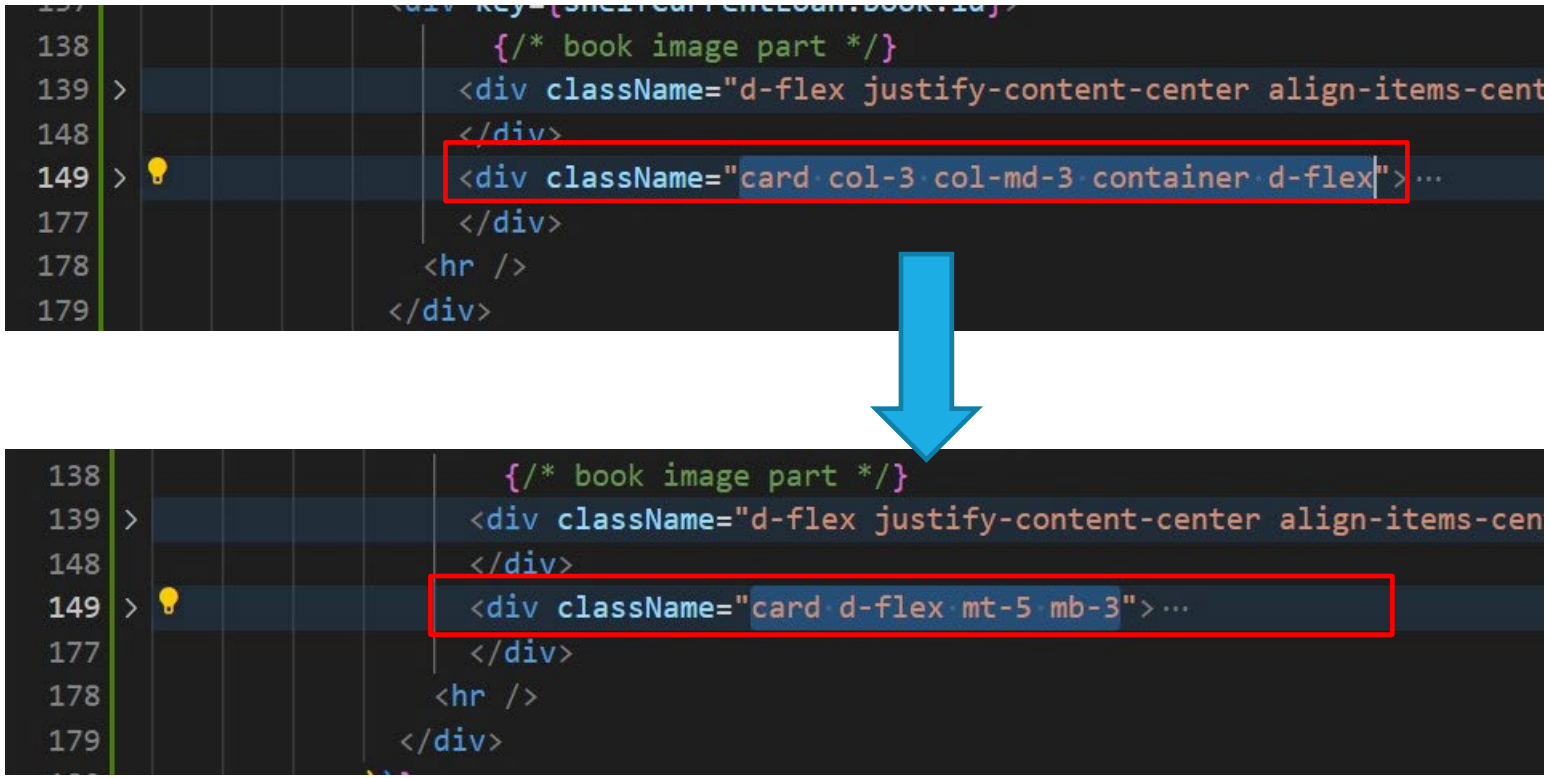

136 {shelfCurrentLoans.map((shelfCurrentLoan) => (
137
138
139 {/* book image part */}
140
149
150
178
179
Delete
136 {shelfCurrentLoans.map((shelfCurrentLoan) => (
137
138 {/* book image part */}
139

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"
```

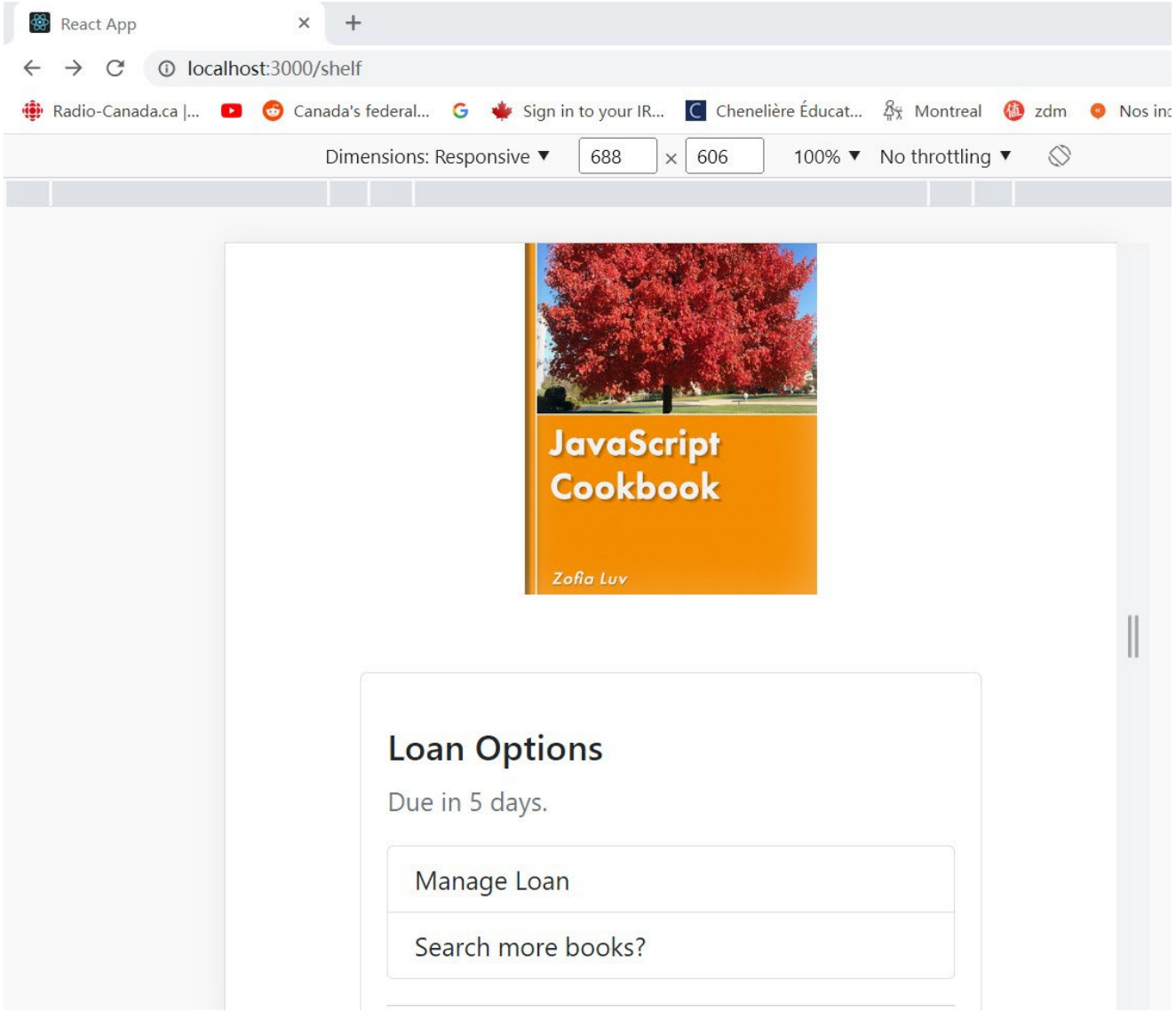
3.5.change card div’s className to "card d-flex mt-5 mb-3"



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

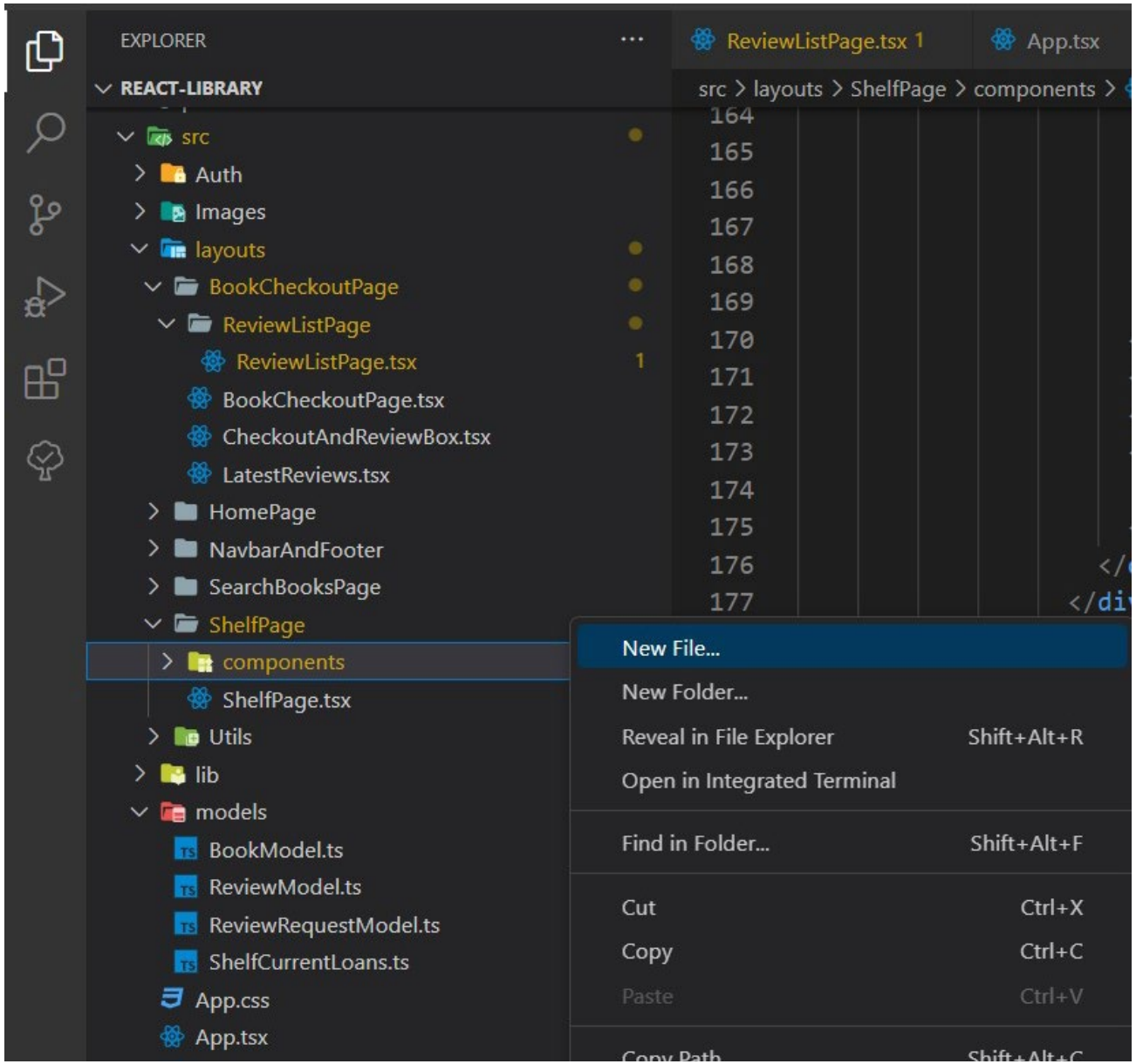
```
158   <button
159     className="list-group-item list-group-item-action"
160     aria-current="true"
161     data-bs-toggle="modal"
162     data-bs-target={`#modal${shelfCurrentLoan.book.id}`}
163   >
164     Manage Loan
165   </button>
```

```
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>
```

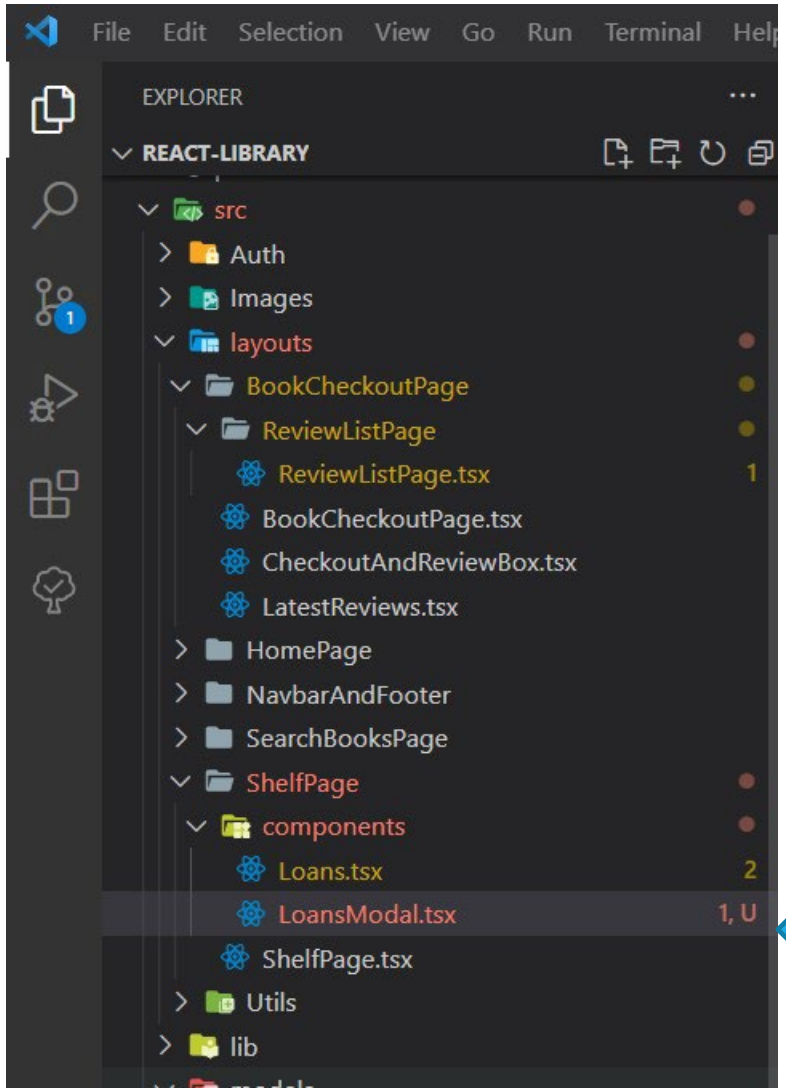


Refresh the page, and zoom out it, we can get a mobile version page.

14. REACT LOANS MODAL HTML/CSS

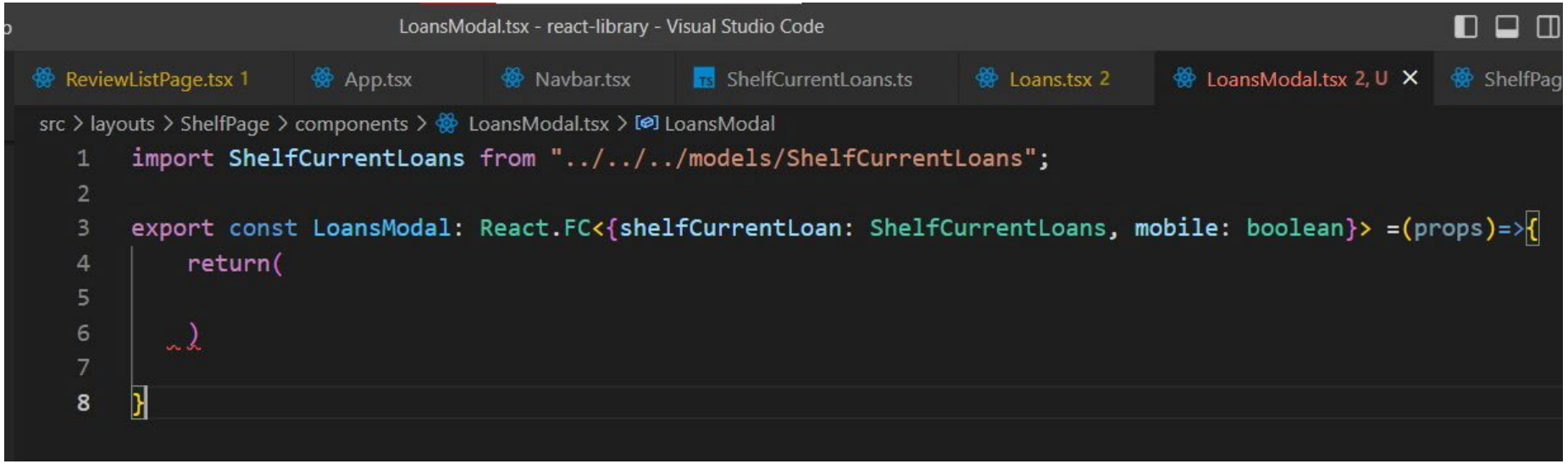


Step 1 Go to ShelfPage folder, then components folder, right click and choose “new file”.



name it LoansModal.tsx.

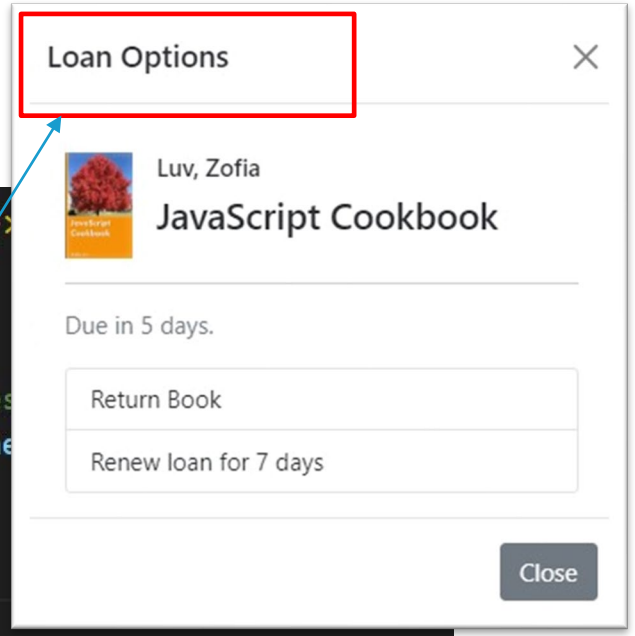
Step2， create export const structure.



Step 3, create html and css in return part.

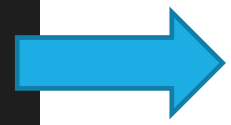
3.1 title part.

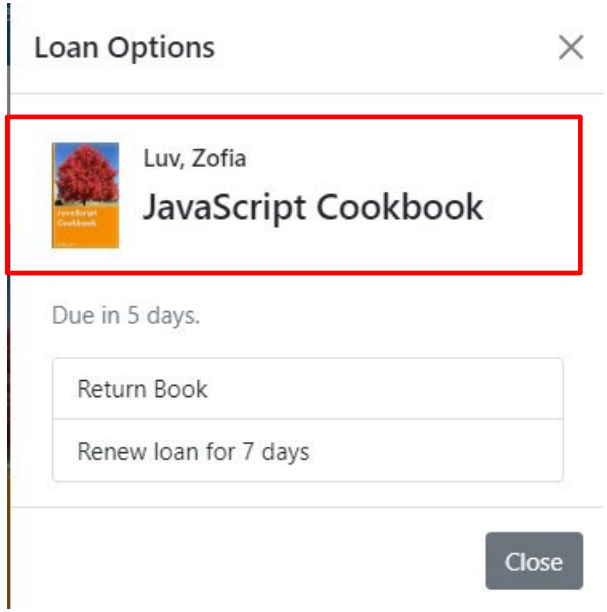
```
export const LoansModal: React.FC<{ shelfCurrentLoan: ShelfCurrentLoans; mobile: boolean }>
  return (
    <div
      className="modal fade"
      // give this div a id, if it equals to the mobilemodal, show mobile version, else des
      id={props.mobile ? `mobilemodal${props.shelfCurrentLoan.book.id}` : `modal${props.she
      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>
```



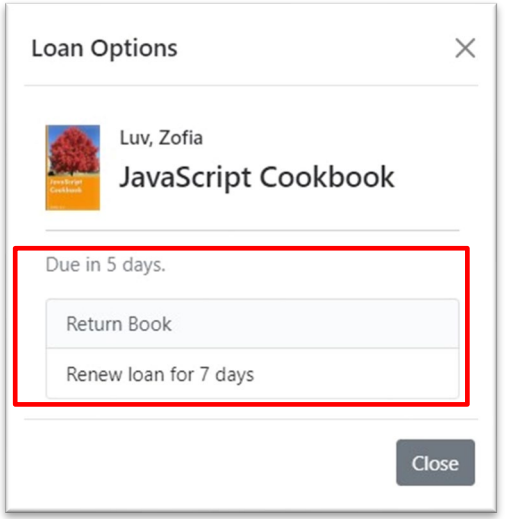
3.2 book info part.

```
</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" heigh
          ) : (
            // default image
            <img src={require("./../../../Images/BooksImages/book-luv2co
          )}
        </div>
        <div className="col-10">
          {/* author and title part */}
          <h6>{props.shelfCurrentLoan.book.author}</h6>
          <h4>{props.shelfCurrentLoan.book.title}</h4>
        </div>
      </div>
      <hr />
```





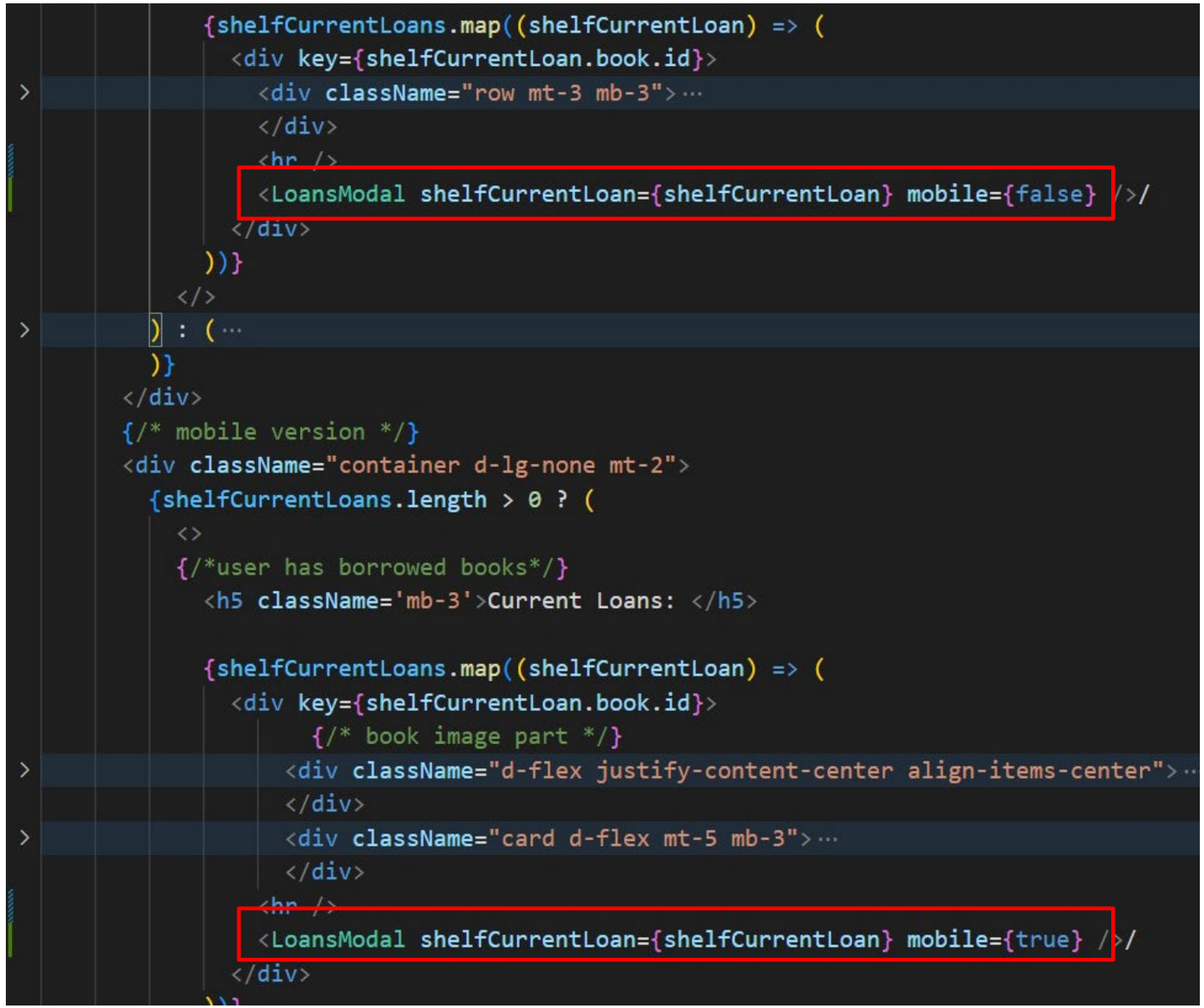
3.3 days reminder, return book, renew book part.




```
<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 clik 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>
```

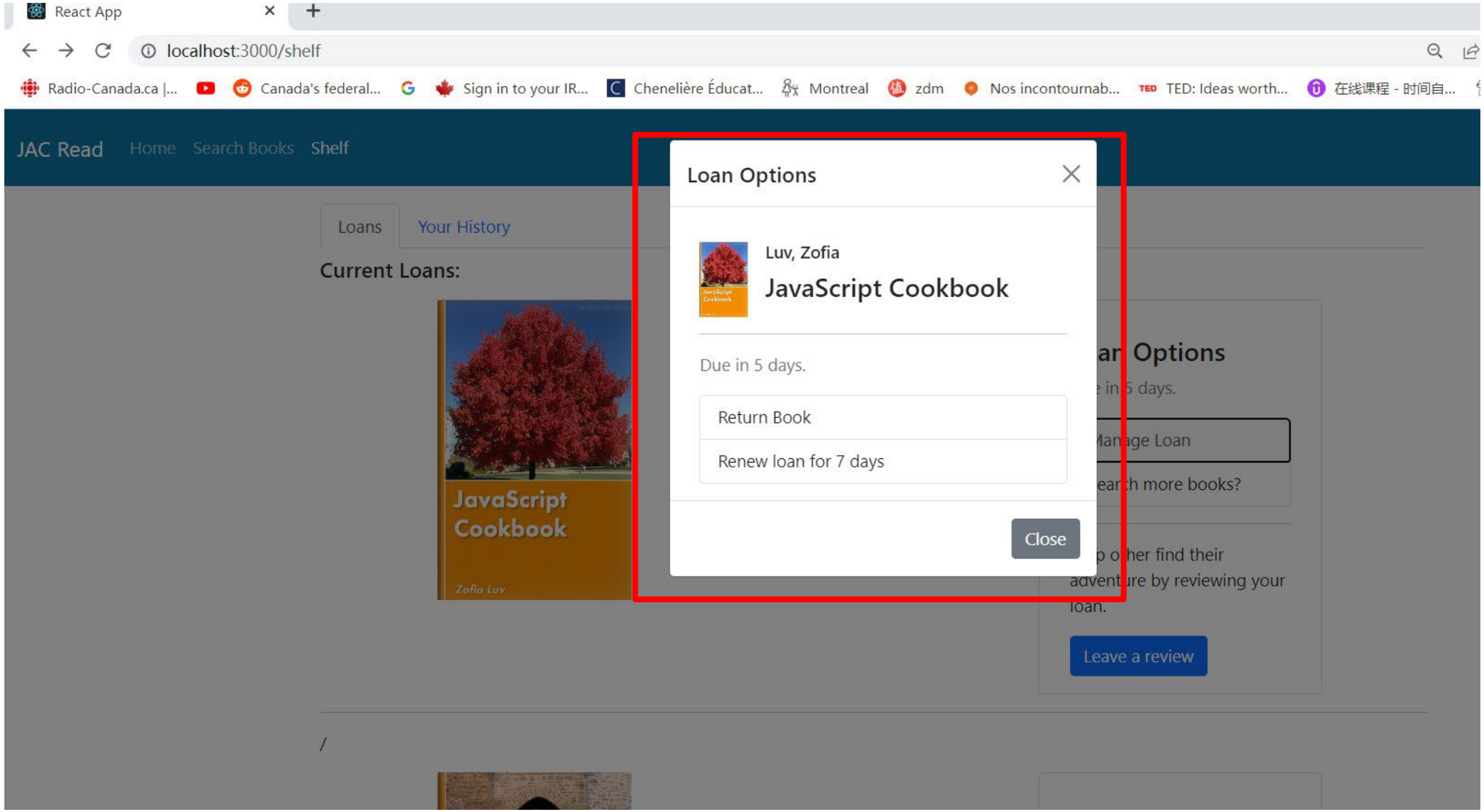
3.4 “close” button part.


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



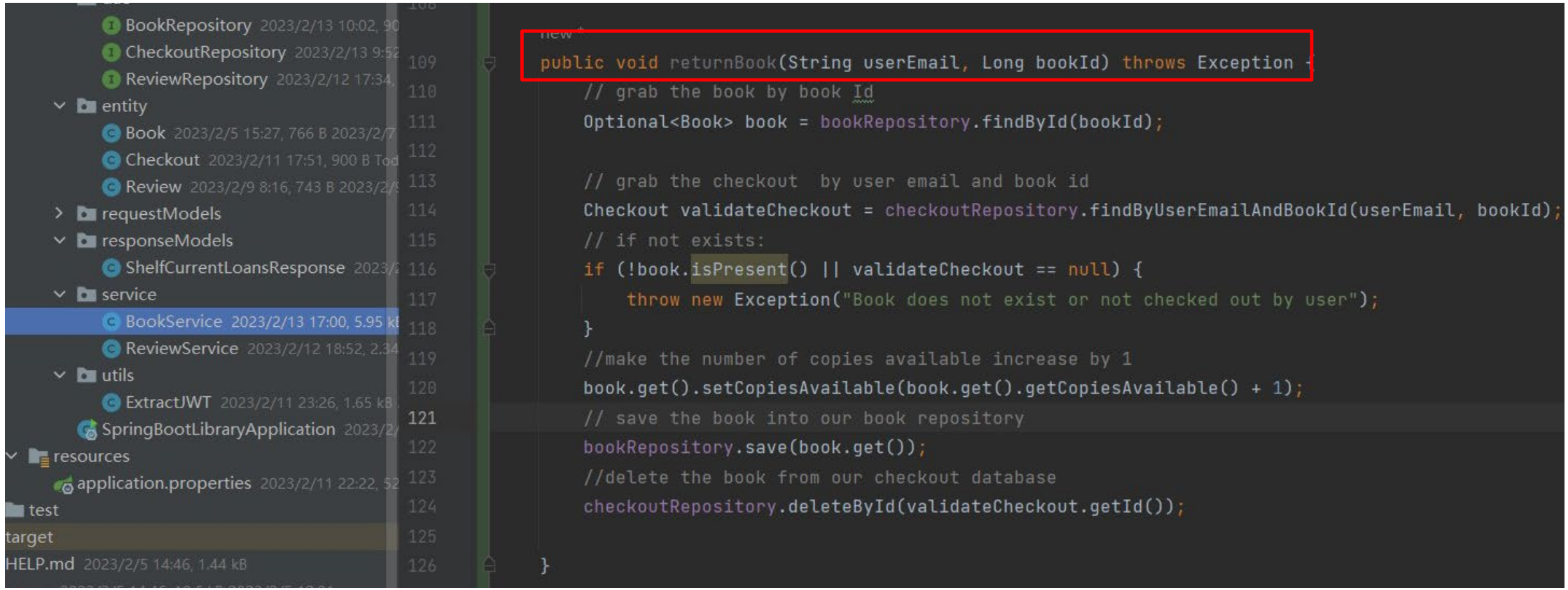
Step 4 Go to Loans.tsx, add “<LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} />” to both version, but for the mobile version, change the value of mobile to “true”.

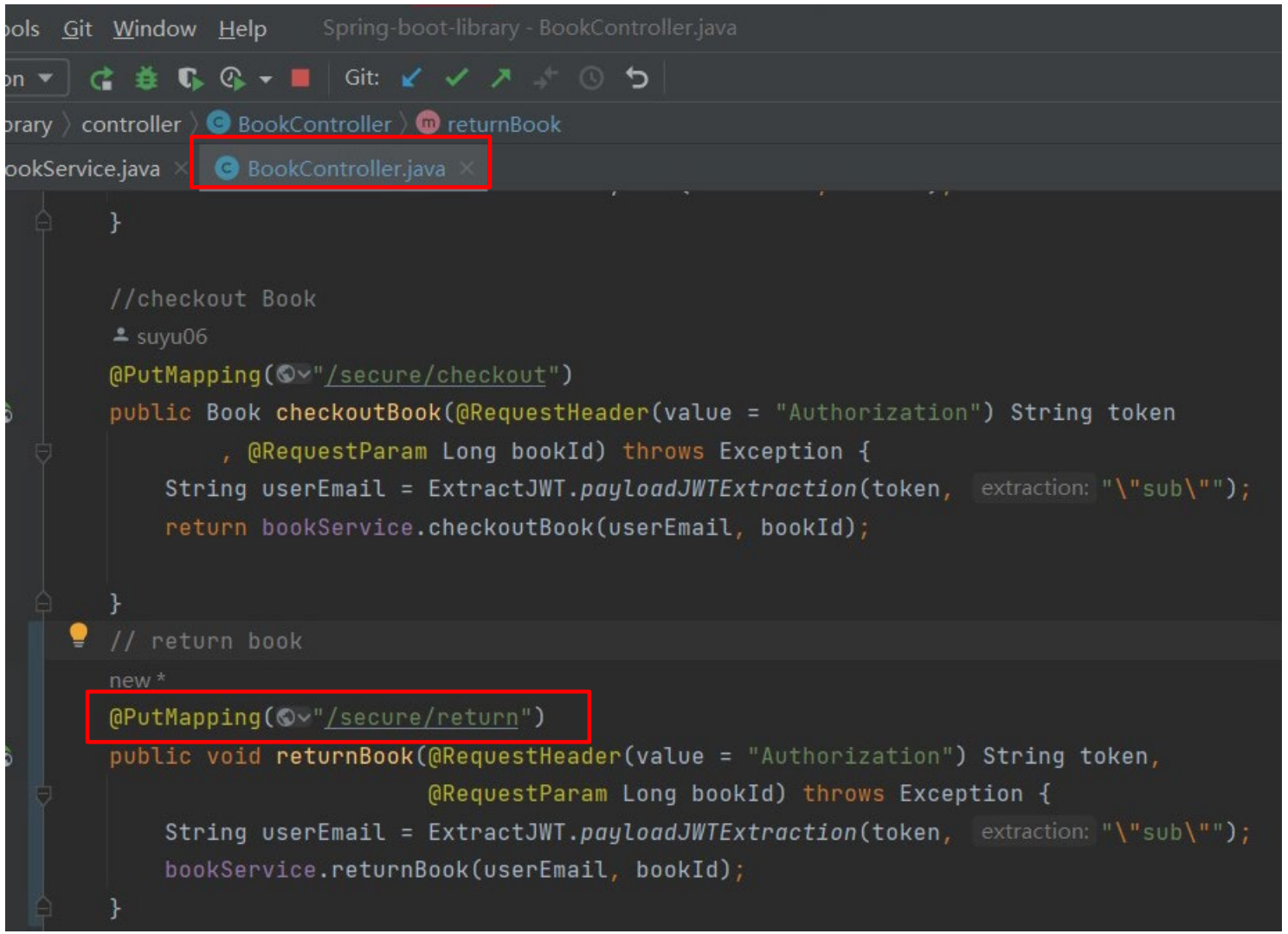
The final pop out dialogue box is as below:



15. SPRINGBOOT RETURN BOOK SERVICE AND ENDPOINT LAYER

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

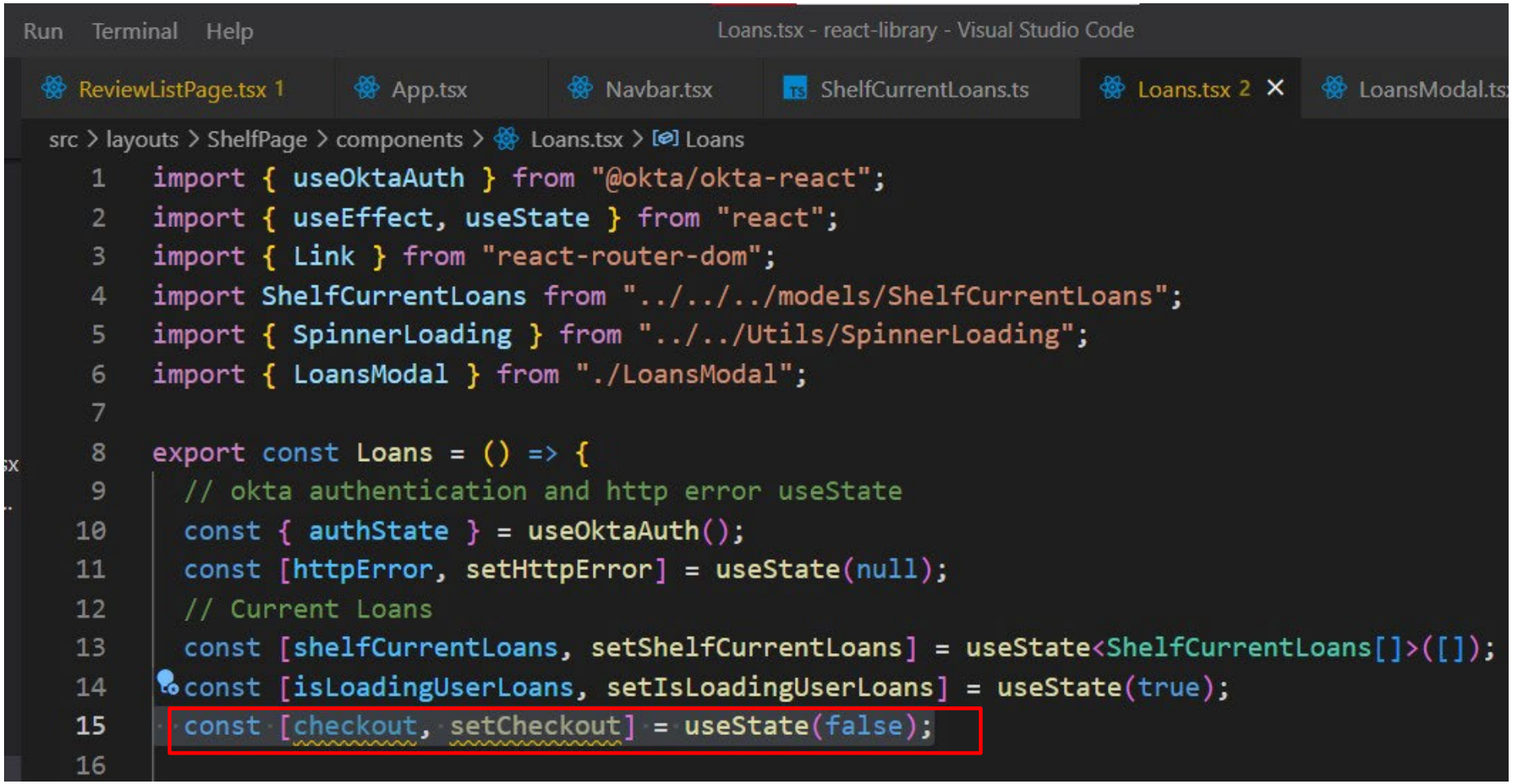




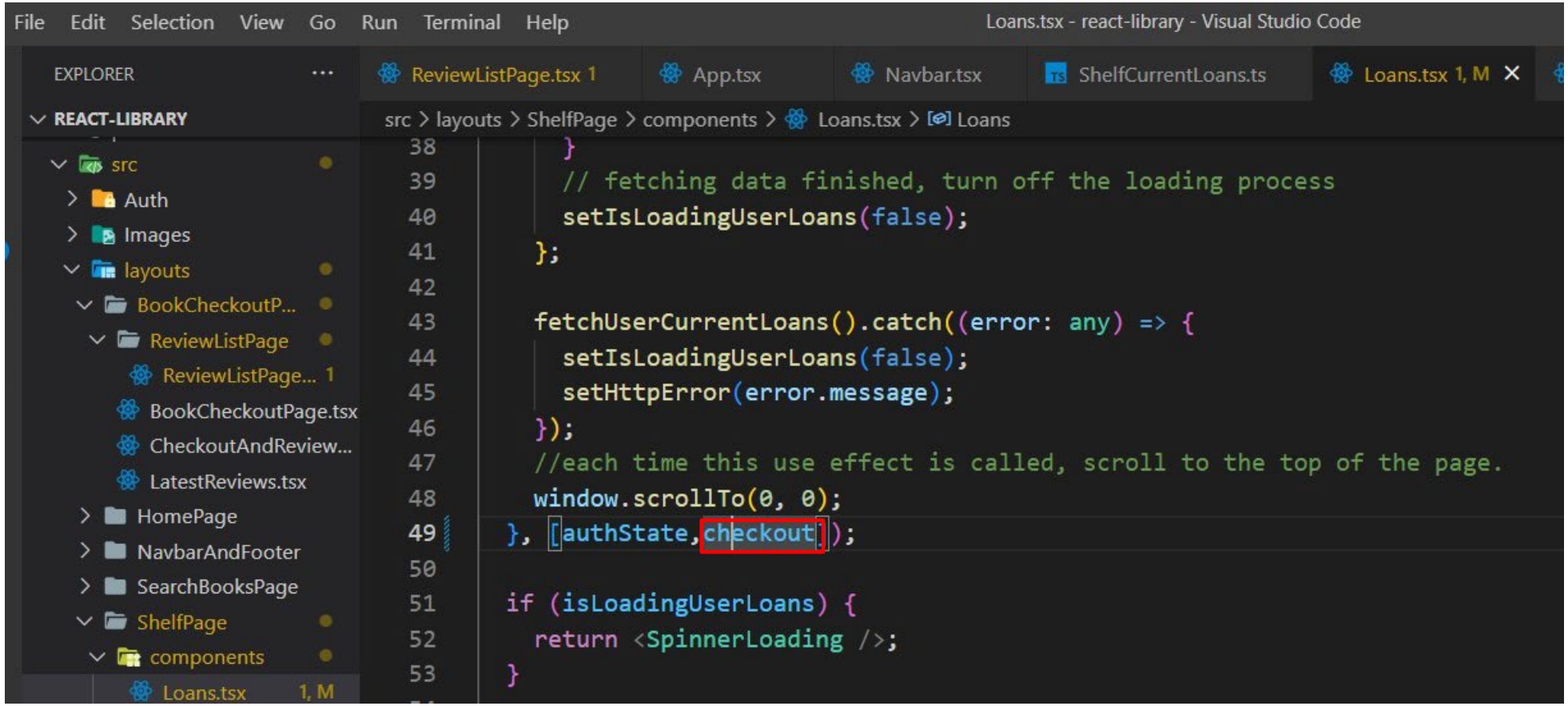
Step 2 Go into our controller package, then BookController, use “Put” method to create the endpoint for book return.

16. REACT RETURN BOOK

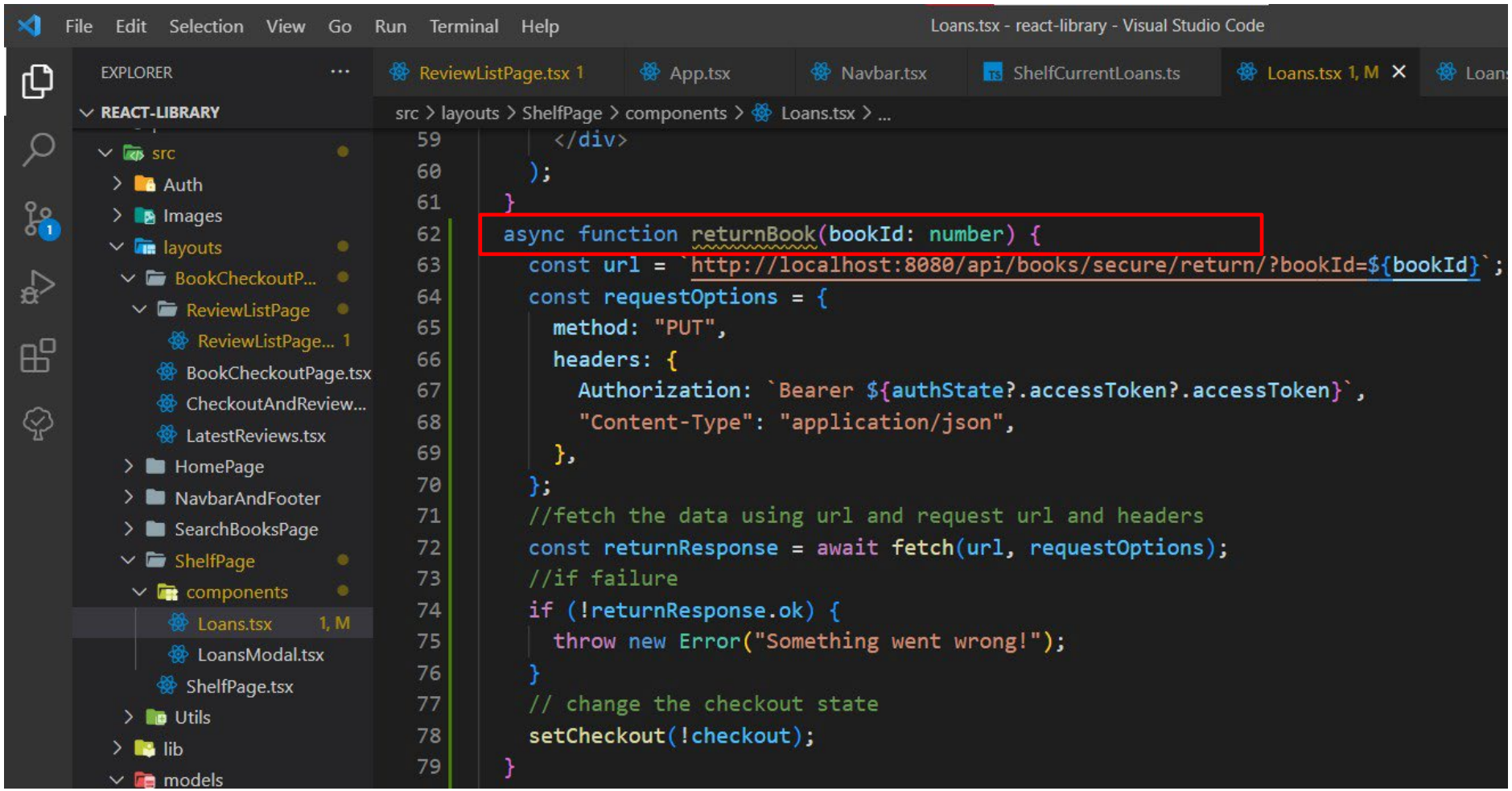
Step 1 Go to Loans.tsx,add useState.



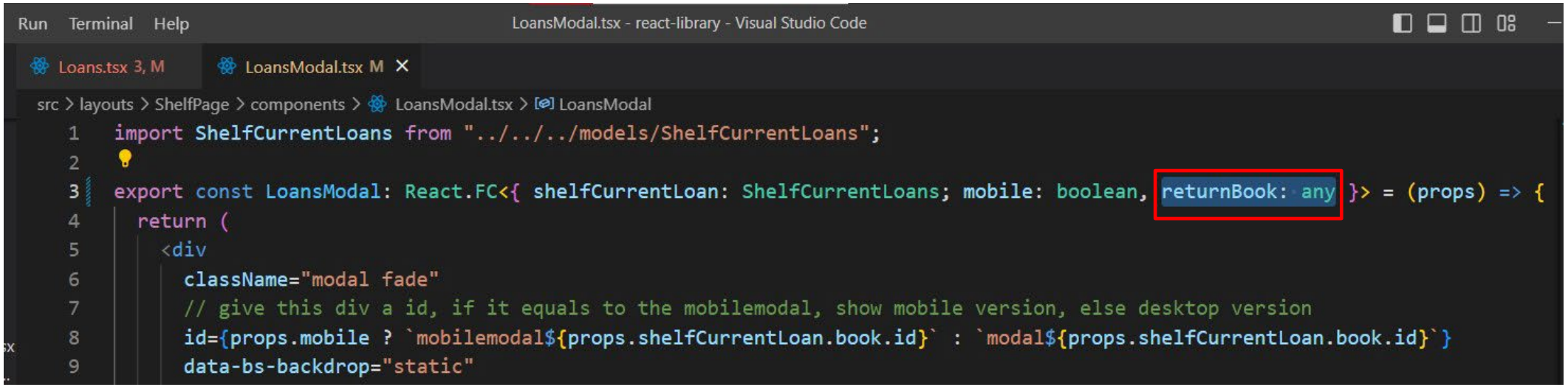
Step2 , add checkout useState to the end of the array of useEffect.
So authState and checkout will be part of the state changer for this use effect.



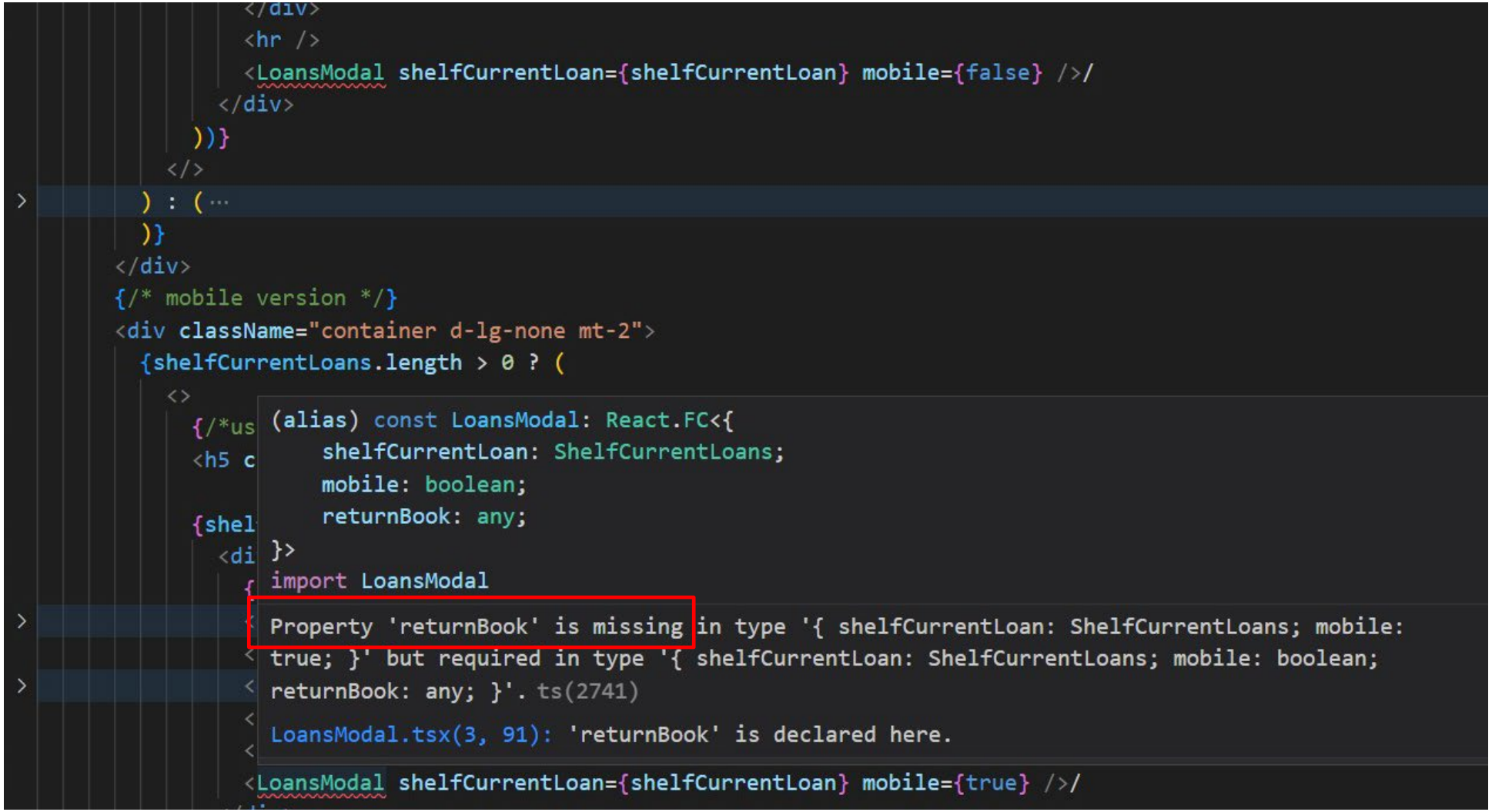
Step 3 create async function returnBook to fetch data from backend.



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



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

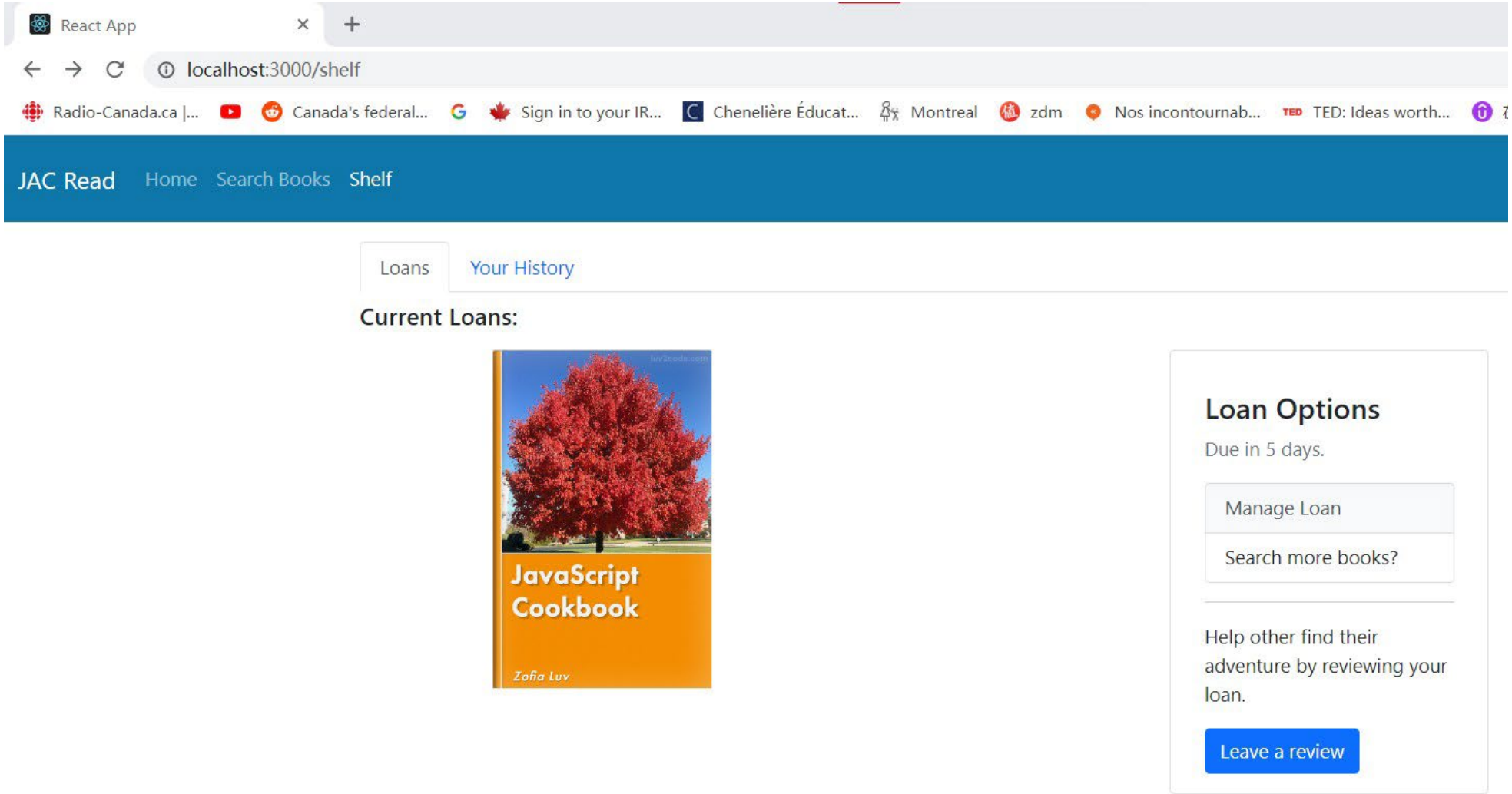


Add returnBook={returnBook} to LoansModal properties.

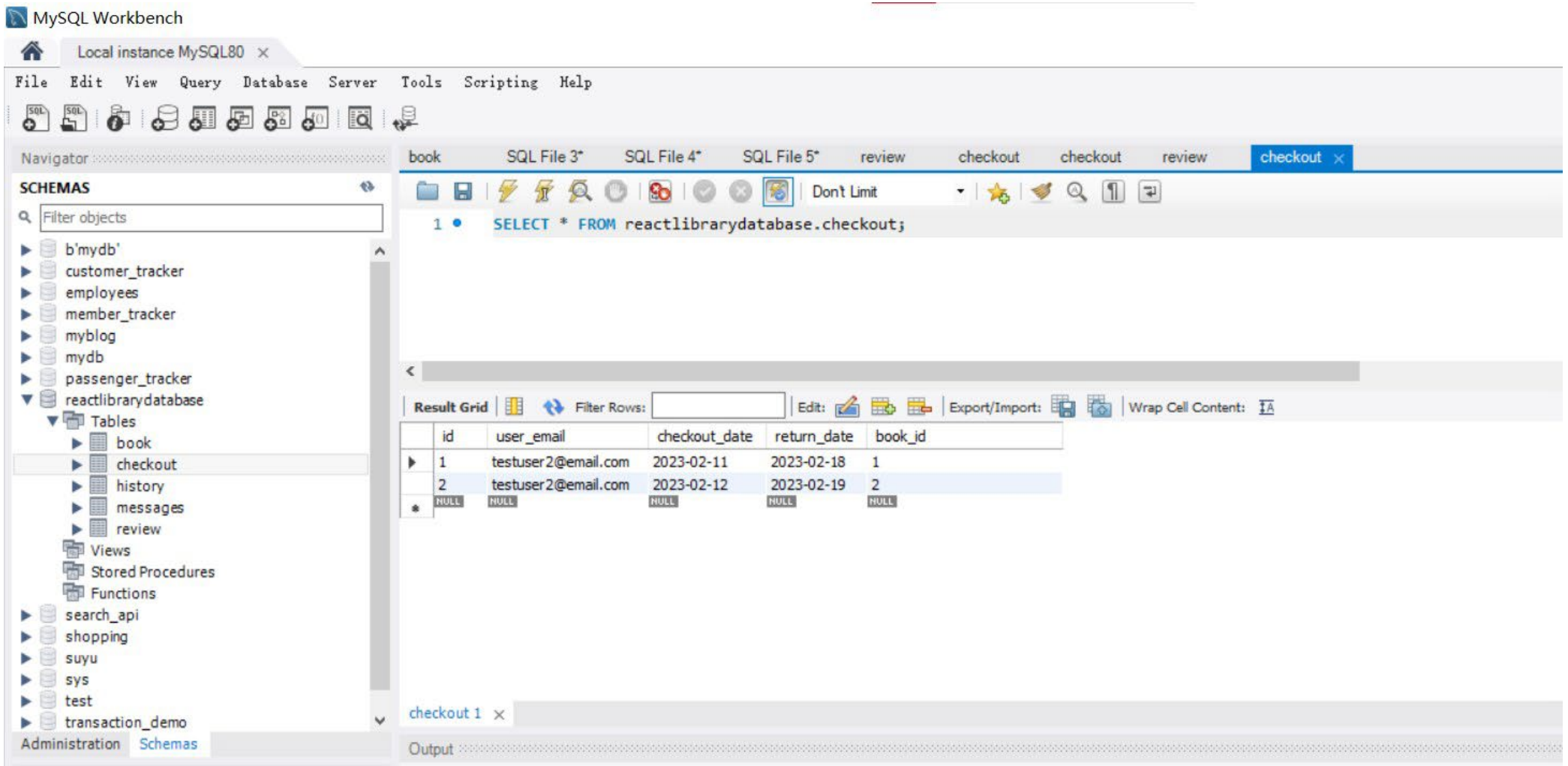
Step 6 Go to LoansModal.tsx, add onClick listener to “return book” btn.

```
{/* 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.daysLef
{props.shelfCurrentLoan.daysLeft === 0 && <p className="text-success">Due Today.</p>}
{props.shelfCurrentLoan.daysLeft < 0 && <p className="text-danger">Past due by {props.shelfCurrentLoan.daysL
<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>
```

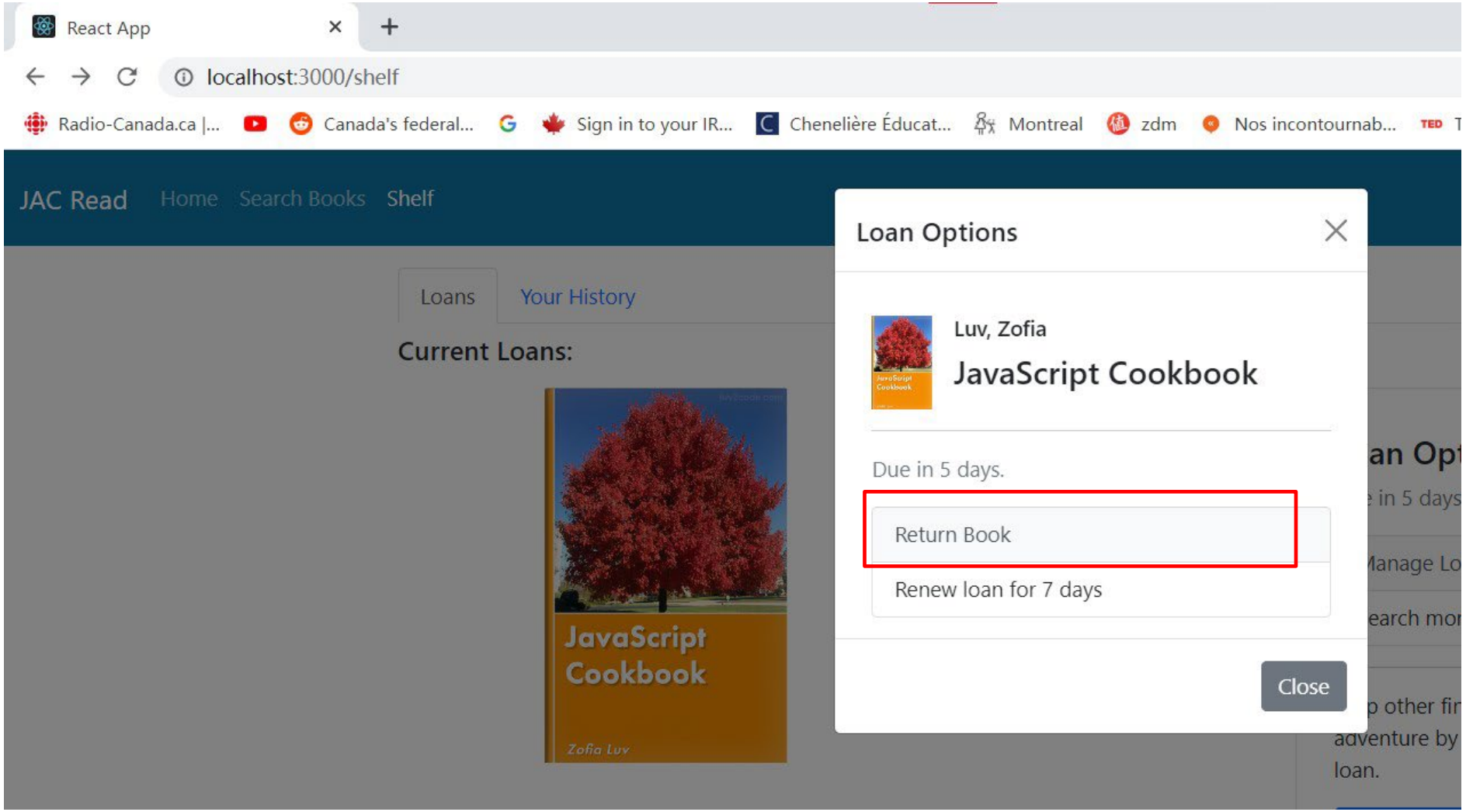
With the status of logged in, open our shelf page, try to manage our shelf.



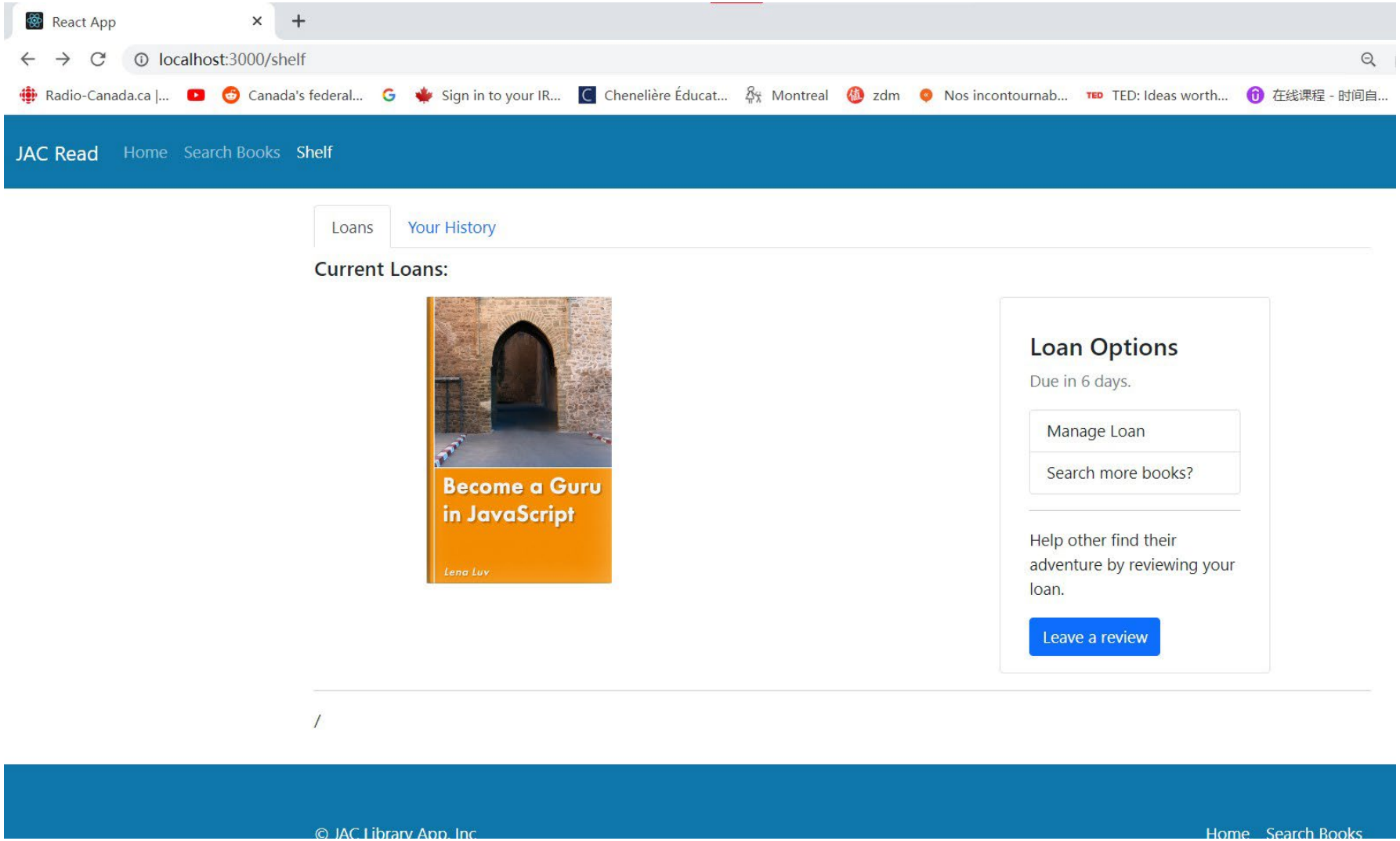
The user has now borrowed 2 books: bookId 1 and bookId 2.



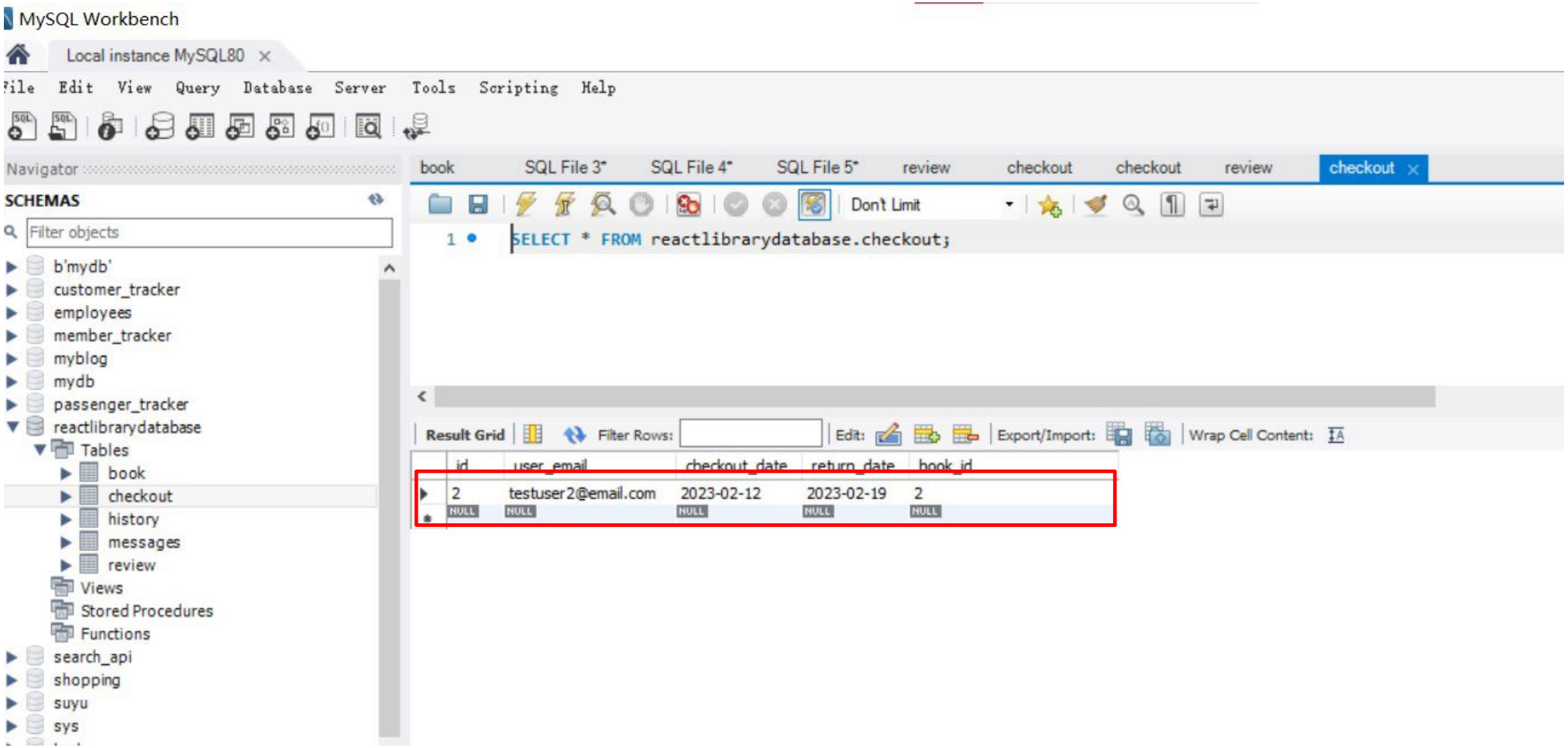
Now, if we click on “Return book” in the pop out window,



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



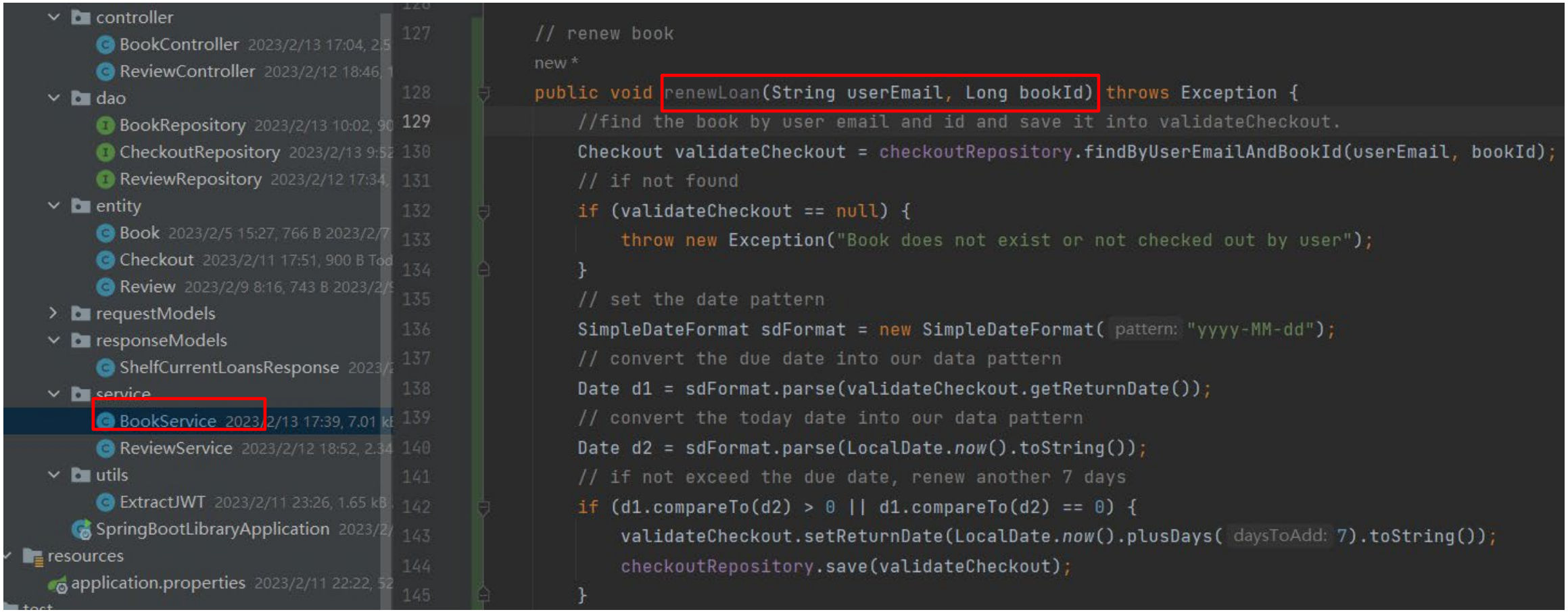
Go to our database, checkout table, only the second book is there.



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

17. SPRINGBOOT RENEW LOANS SERVICE AND ENDPOINT LAYER

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



BookService.java | BookController.java

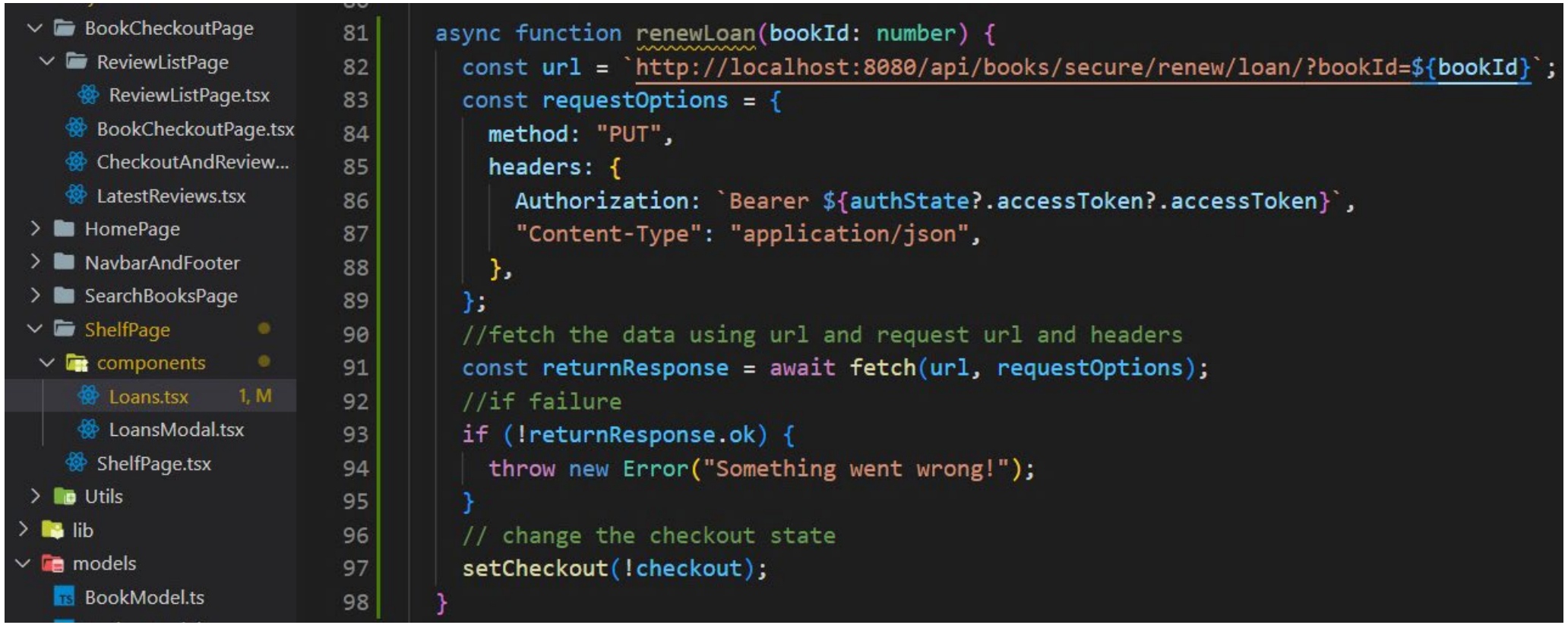
```
@PutMapping("/secure/checkout")
public Book checkoutBook(@RequestHeader(value = "Authorization") String token
        , @RequestParam Long bookId) throws Exception {
    String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
    return bookService.checkoutBook(userEmail,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

}
// return book
suyu06
@PutMapping("/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);
}
// renew another 7 days for the book
suyu06
@PutMapping("/secure/renew/loan")
public void renewLoan(@RequestHeader(value = "Authorization") String token,
                      @RequestParam Long bookId) throws Exception {
    String userEmail = ExtractJWT.payloadJWTExtraction(token, extraction: "\"sub\"");
    bookService.renewLoan(userEmail, bookId);
}
```

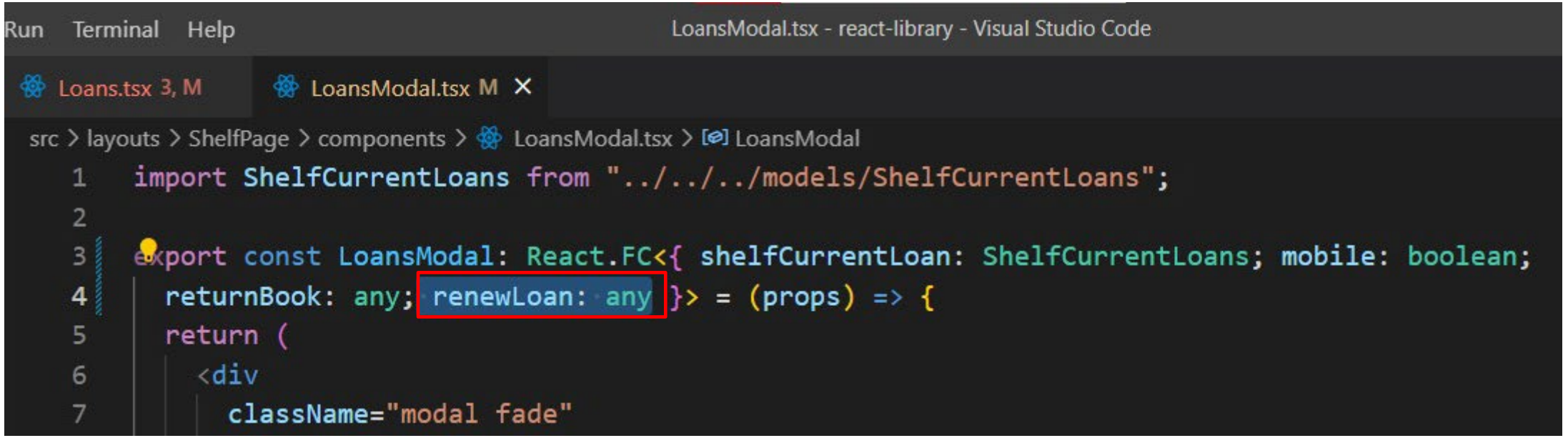
Step 2 Go into BookController class, use “Put” method to create an endpoint for renew the book.

18. REACT RENEW LOAN

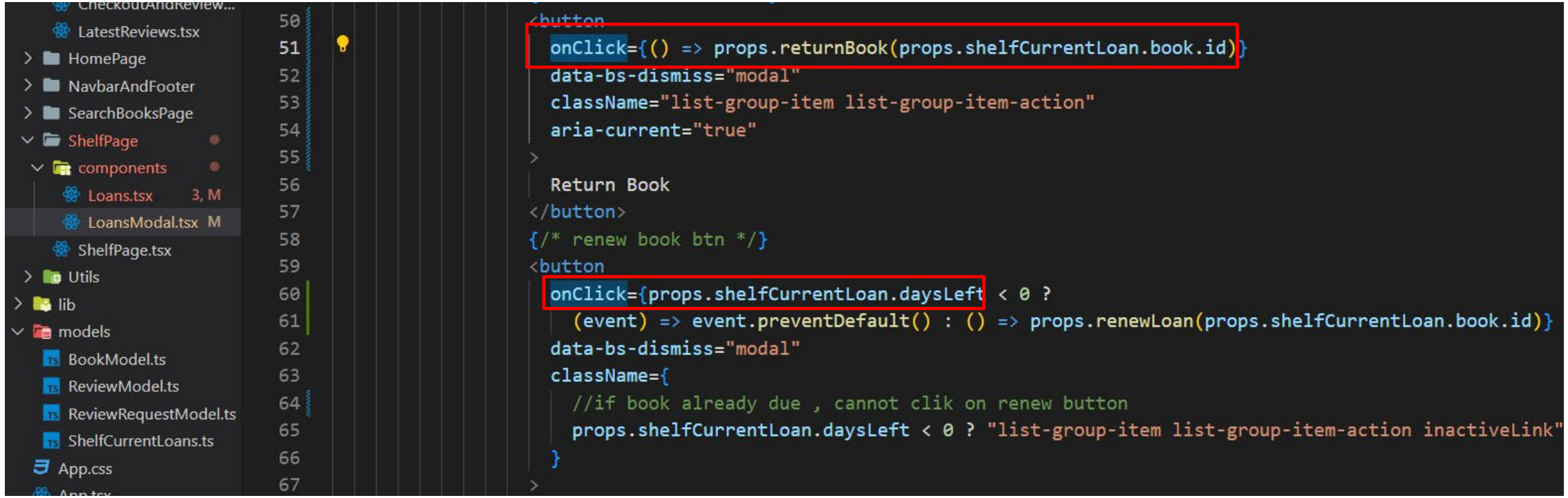
Step 1, In Loans.tsx, scroll until we can find the async function returnBook.
Right under this function, create a new async function renewLoan().



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



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



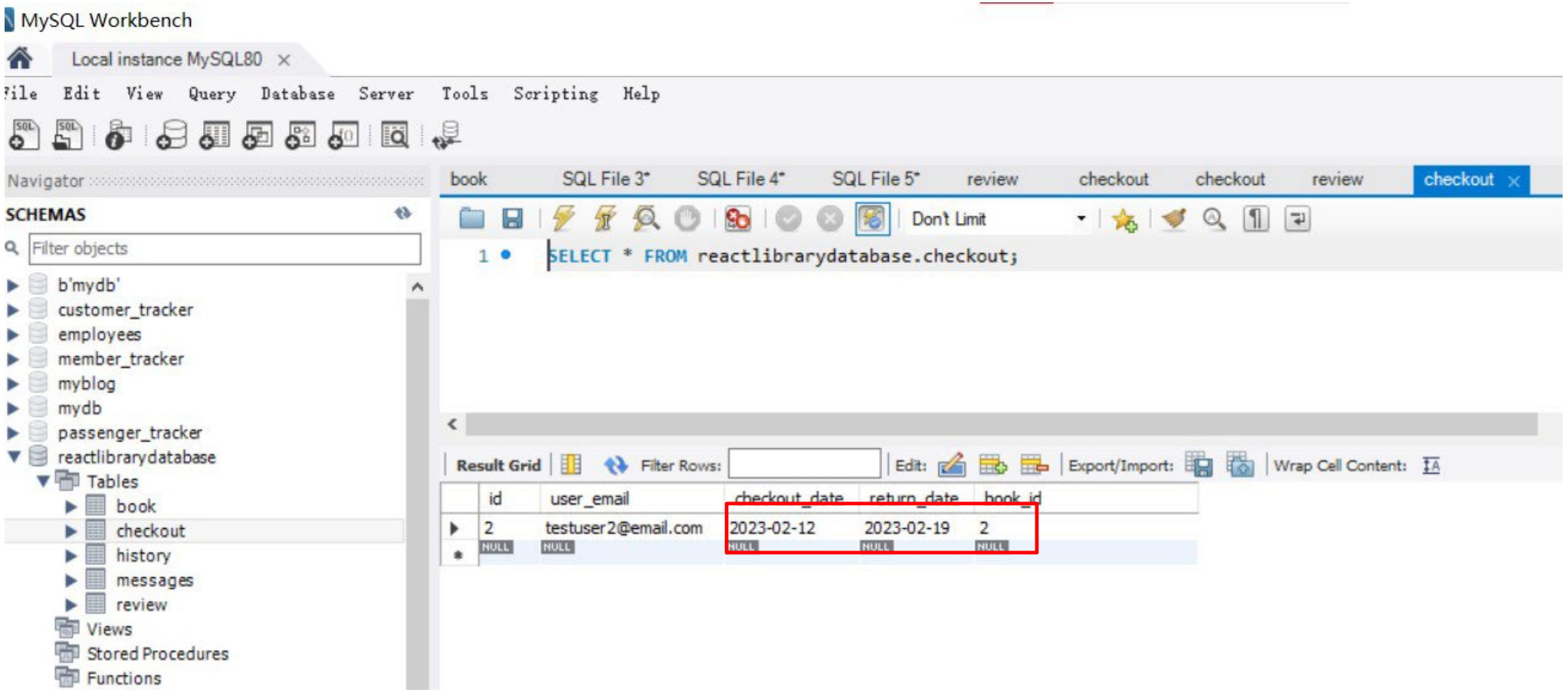
Step 4 Go to Loans.tsx, fix the error of LoansModal component.

Add renewLoan to LoansModal component's property.

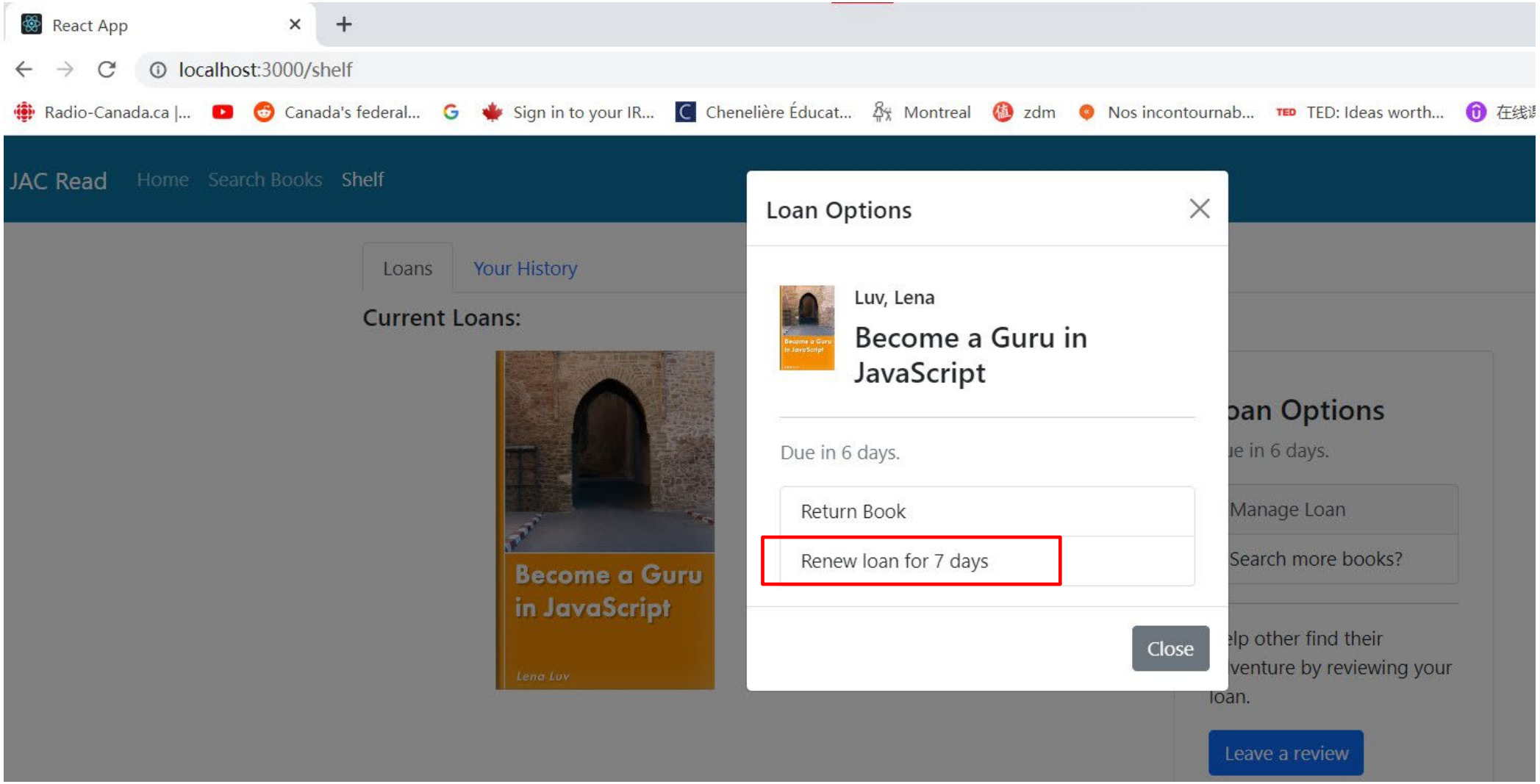
```
            <hr />
            <LoansModal shelfCurrentLoan={shelfCurrentLoan} mobile={false} returnBook={returnBook} renewLoan={renewLoan}/>
          </div>
        ))}
      </>
    ) : (…
    )}
  </div>
  {/* mobile version */}
  <div className="container d-lg-none mt-2">
    {shelfCurrentLoans.length > 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>
```

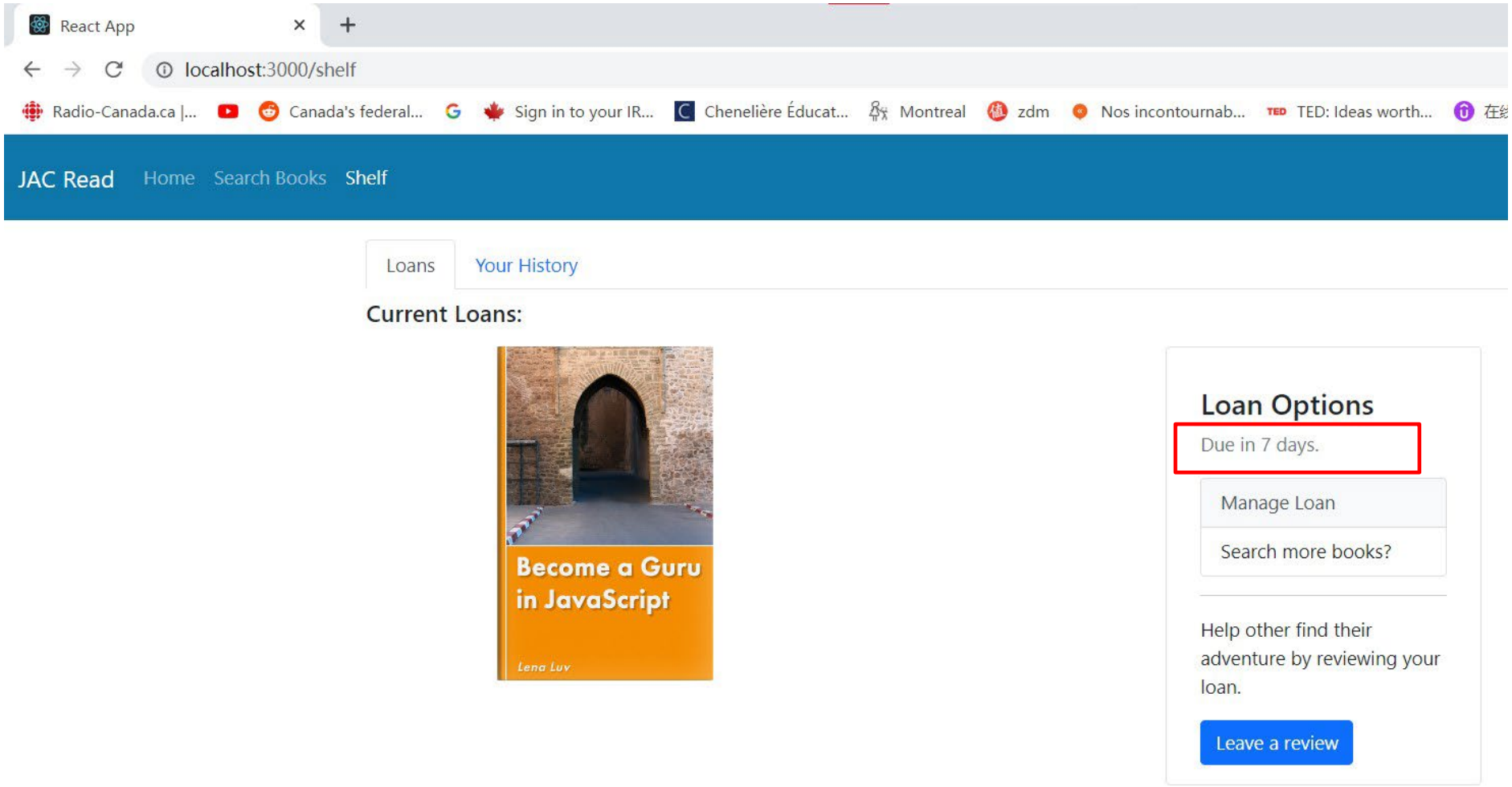
Now, open our database, table checkout, we can see the second book was checked out on 12 Feb 2023, and will due on 19th Feb 2023.



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



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



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.