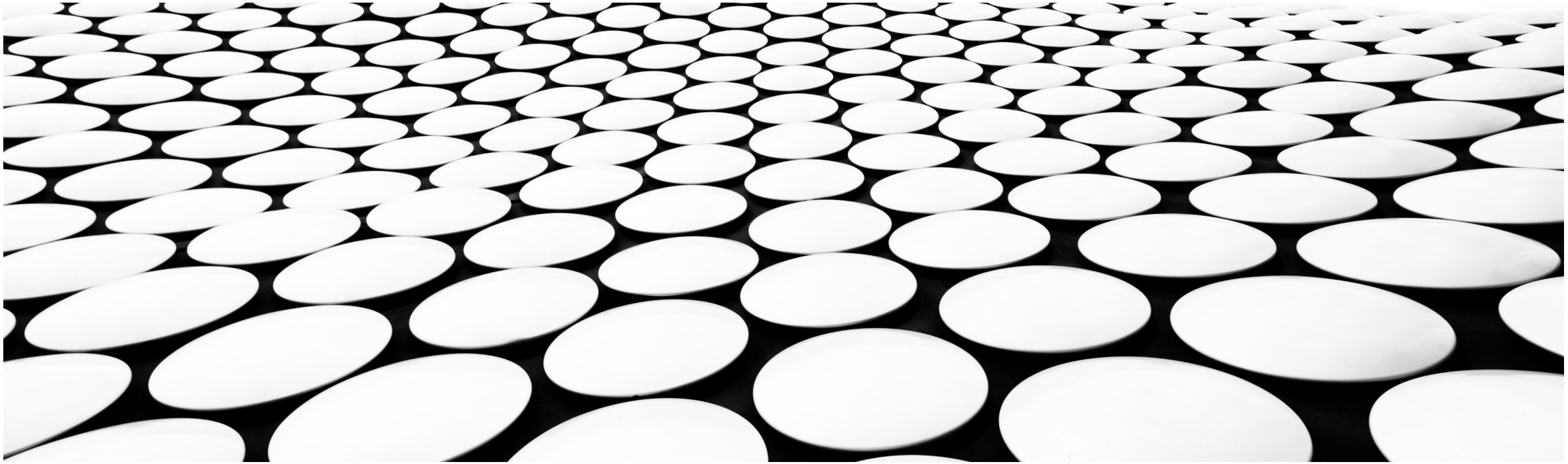
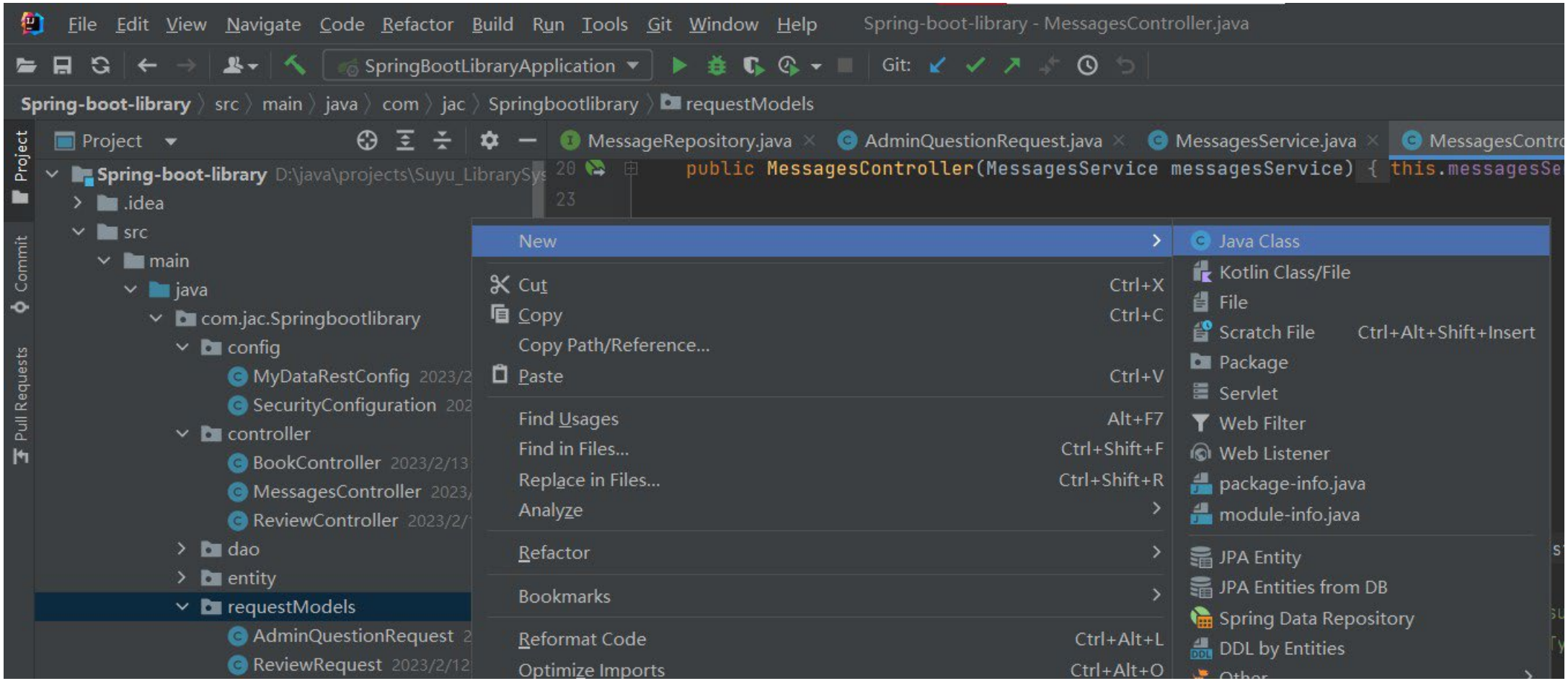
S20 ADD NEW BOOK ADMIN



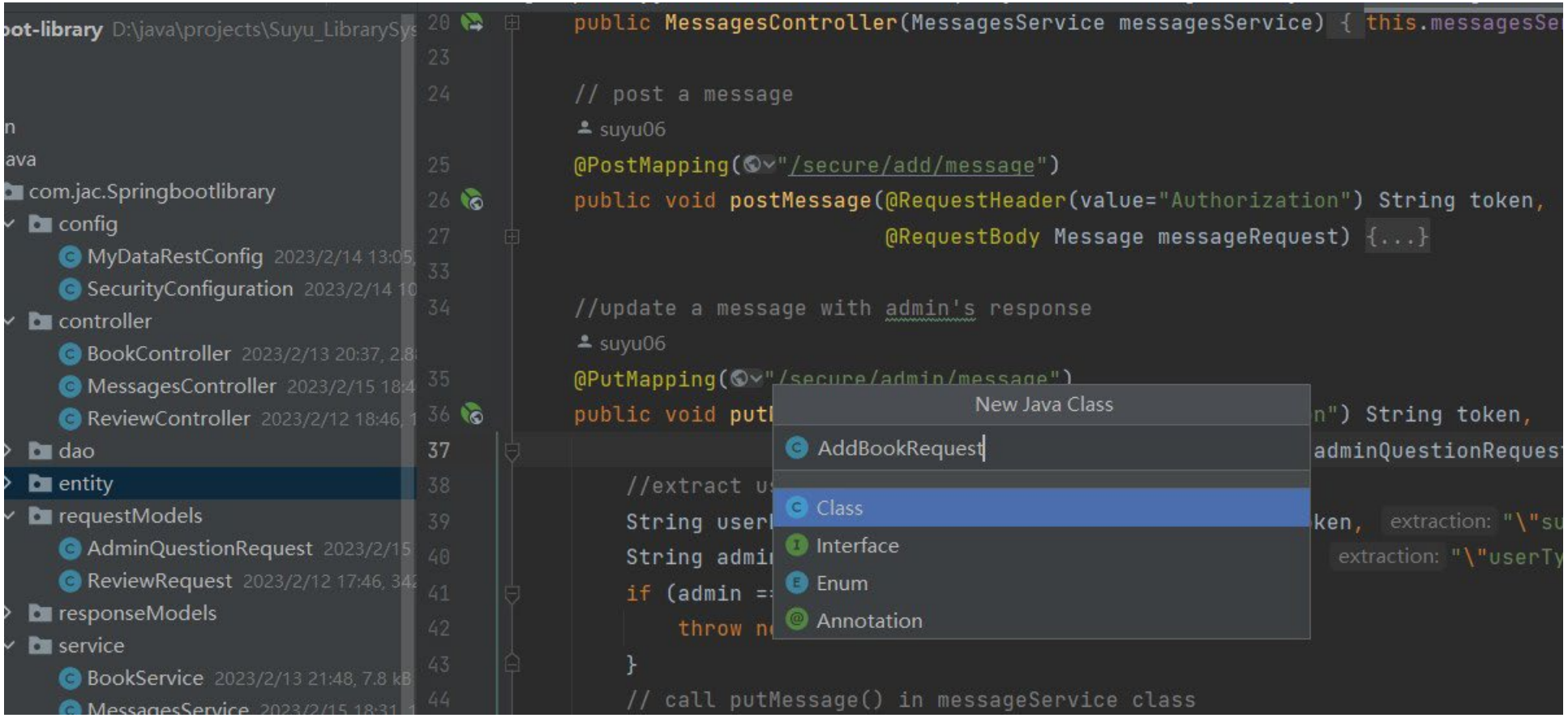
1. SPINGBOOT ADD BOOK REQUEST

To add a new book , the very first thing we need to do is to create a object, for the request that's coming from the client to the spring boot back end.

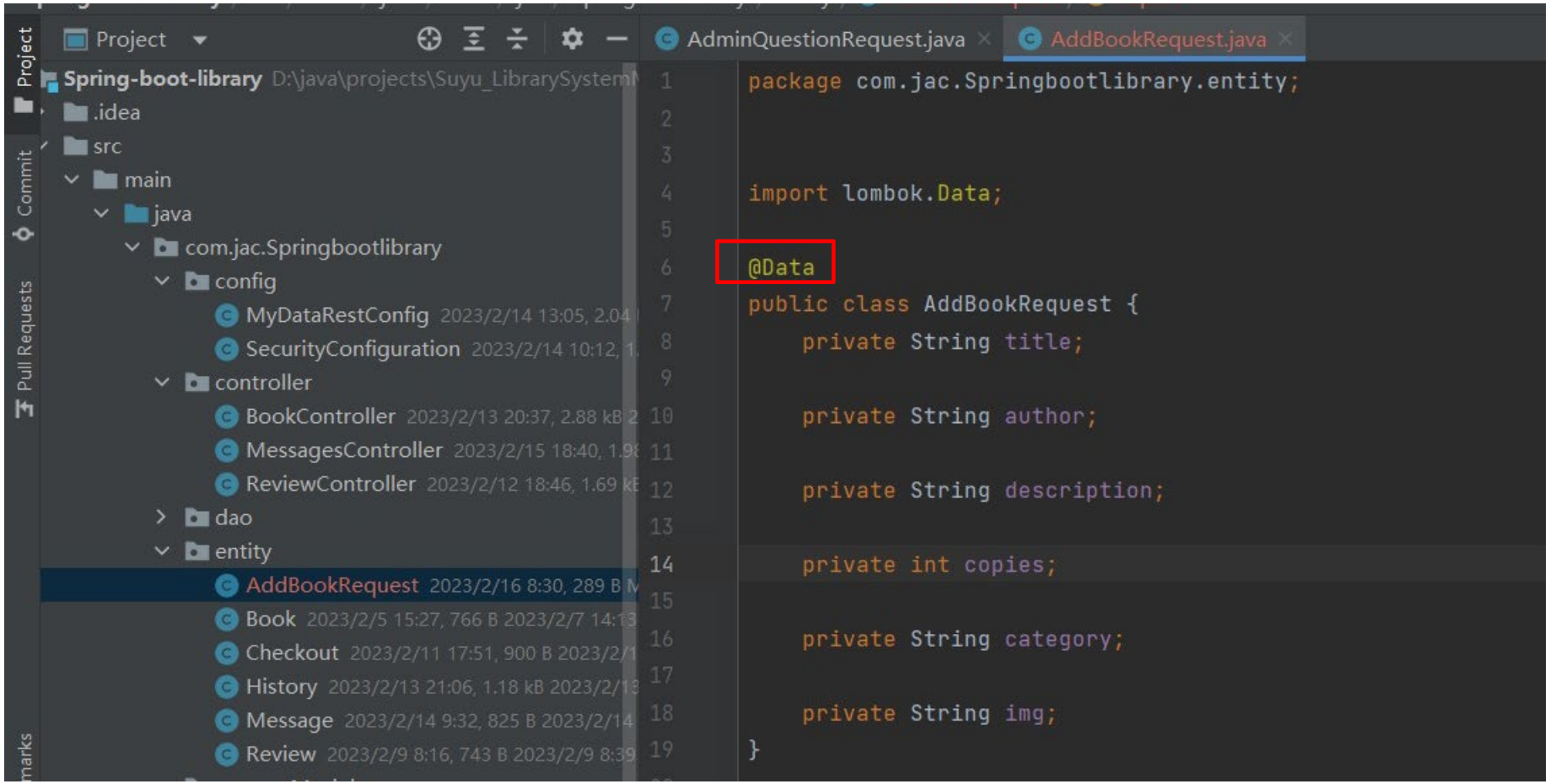
Step1 Right click “requestModels” package, choose “New”, “Java Class”.



Choose “Class”, input name “AddBookRequest”.

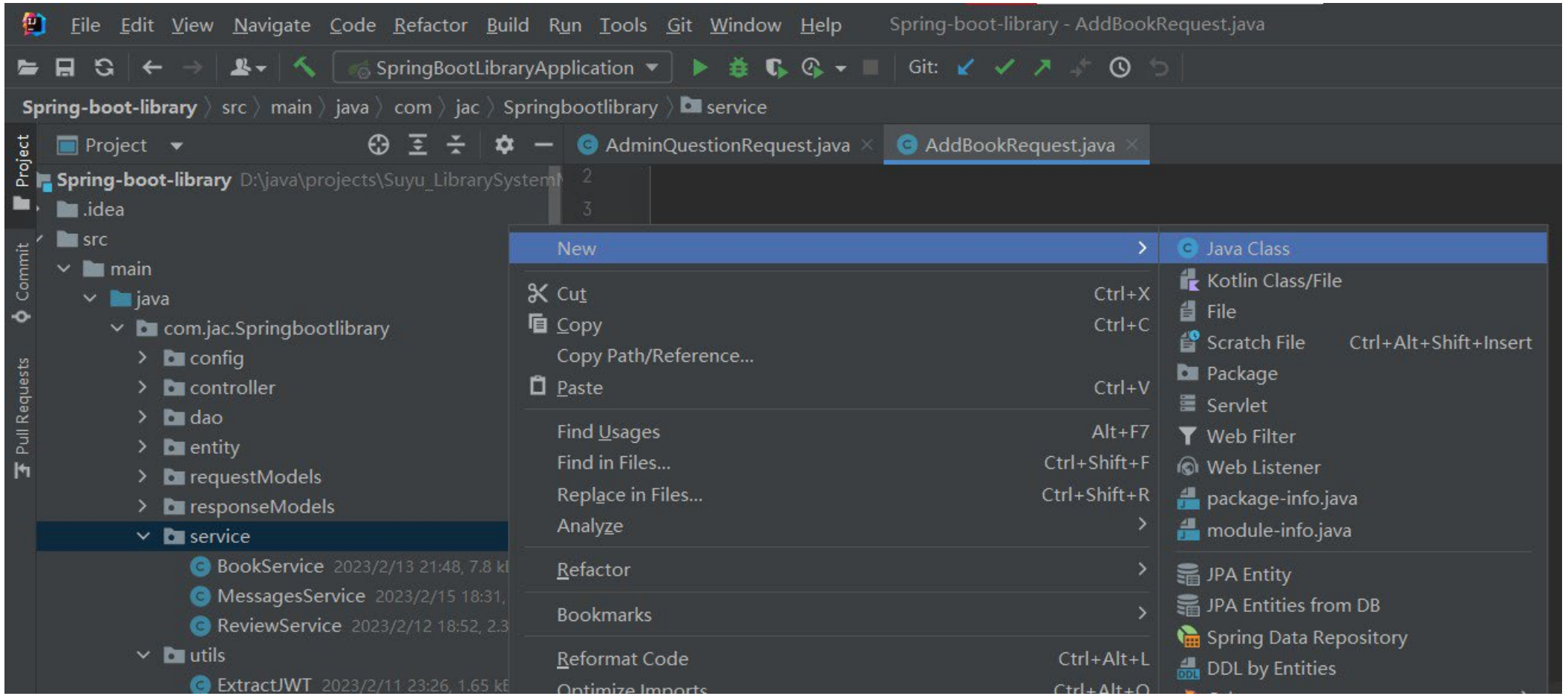


Step 2 add annotation @Data and class properties.

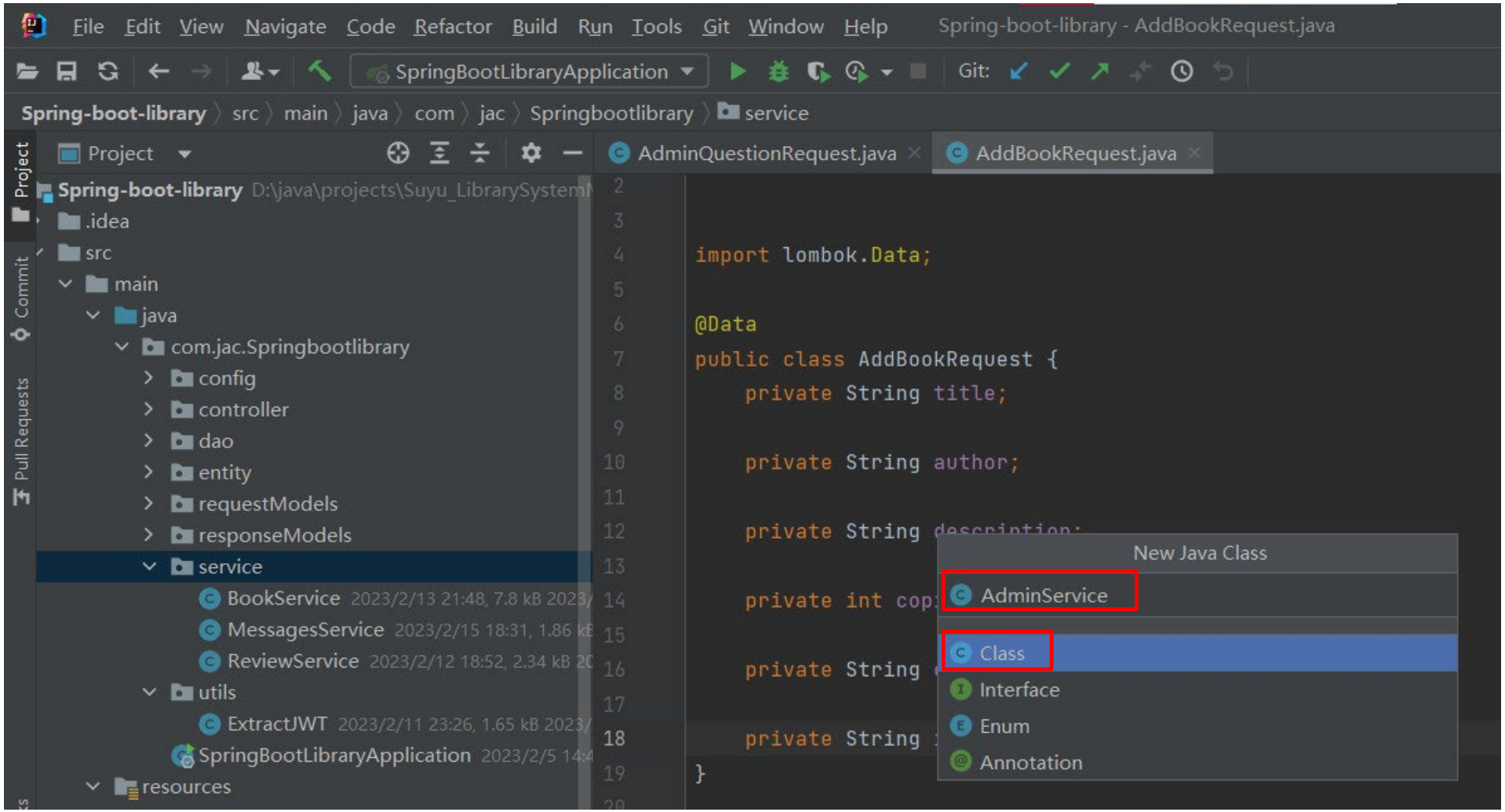


2. SPINGBOOT ADMIN SERVICE LAYER

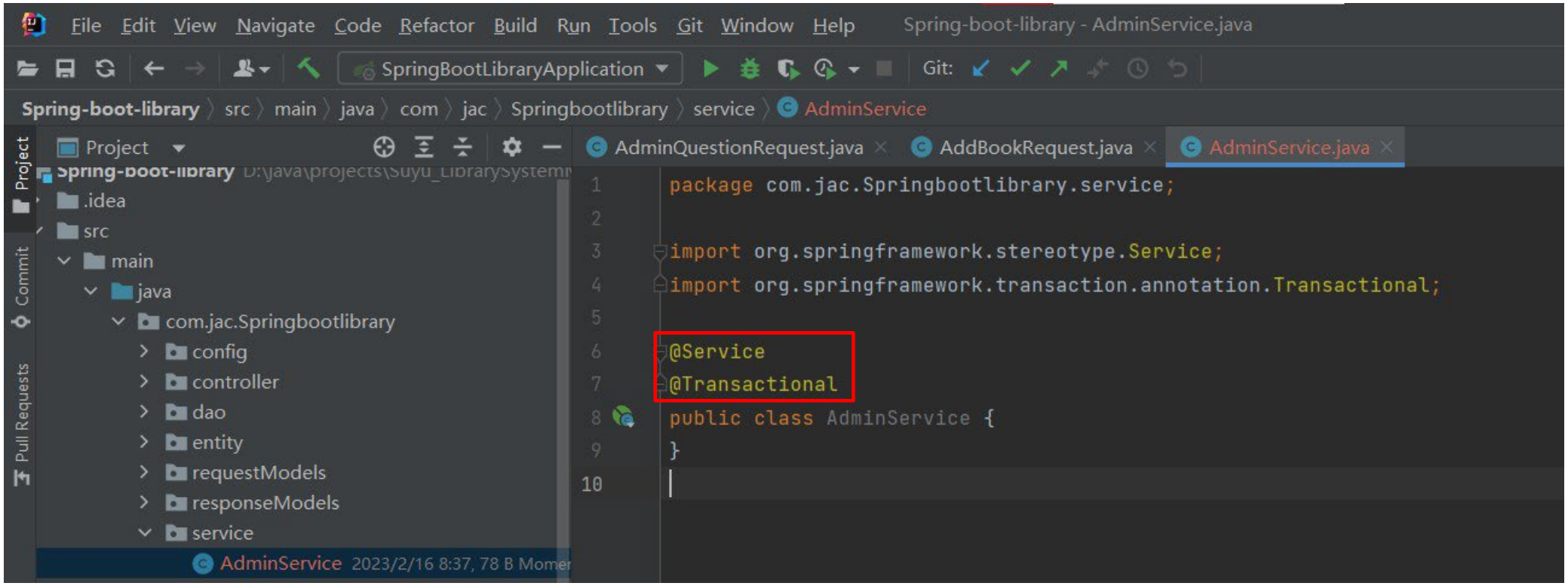
Step 1, Right click “service” package, choose “New”, “Java class”.



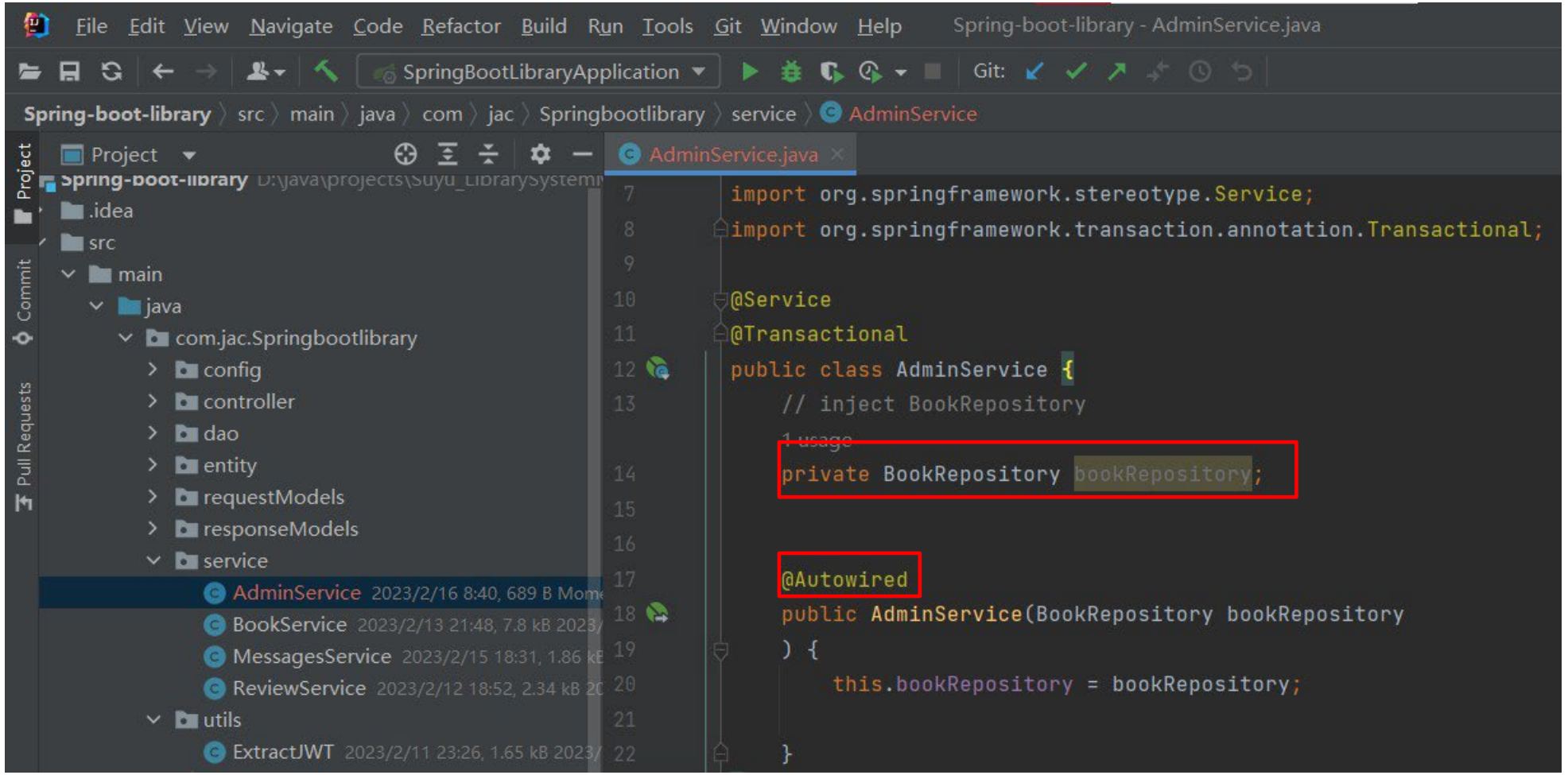
Choose “Class”, input the name “AdminService”.



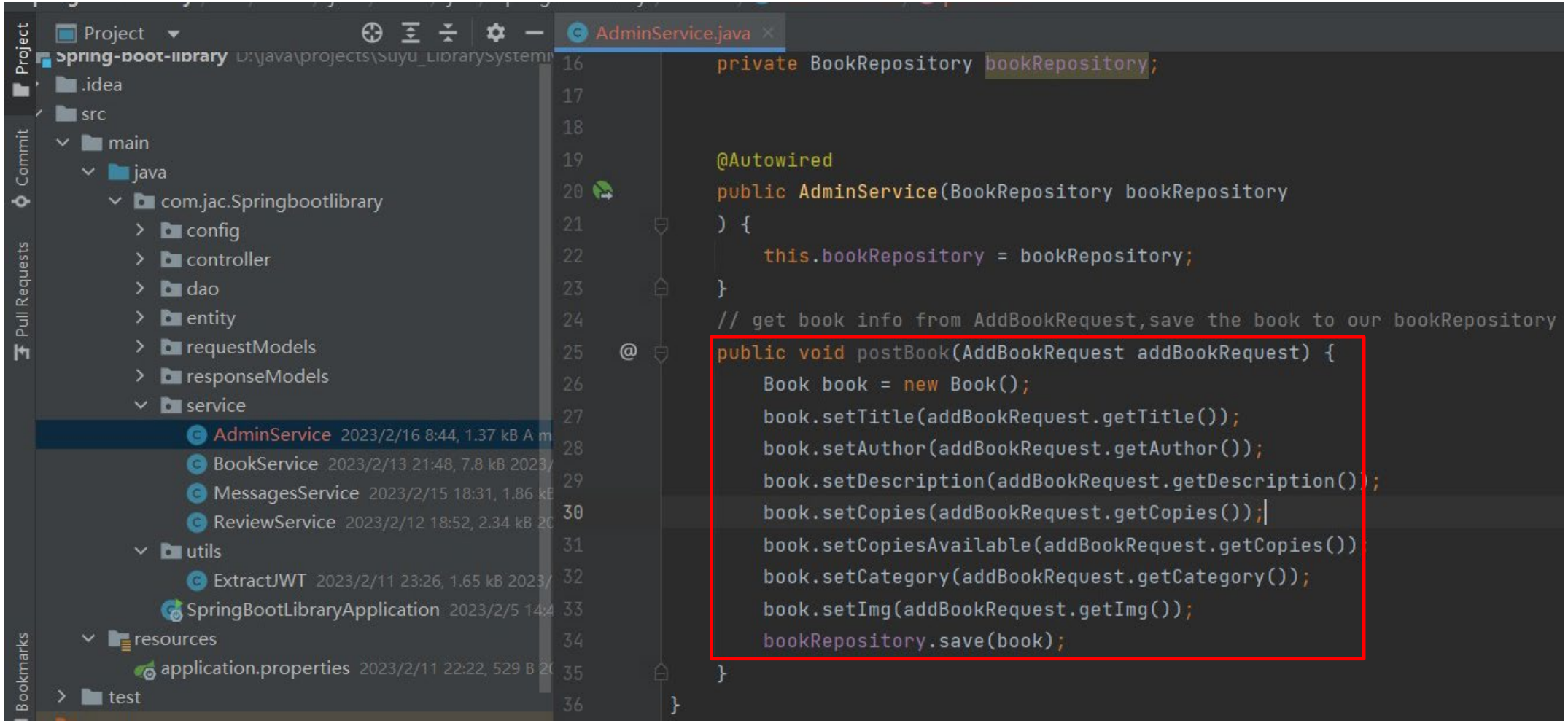
Step 2, add @Service ans @Transactional annotation.



Step 3, inject BookRepository and AutoWire our constructor of public AdminService constructor.



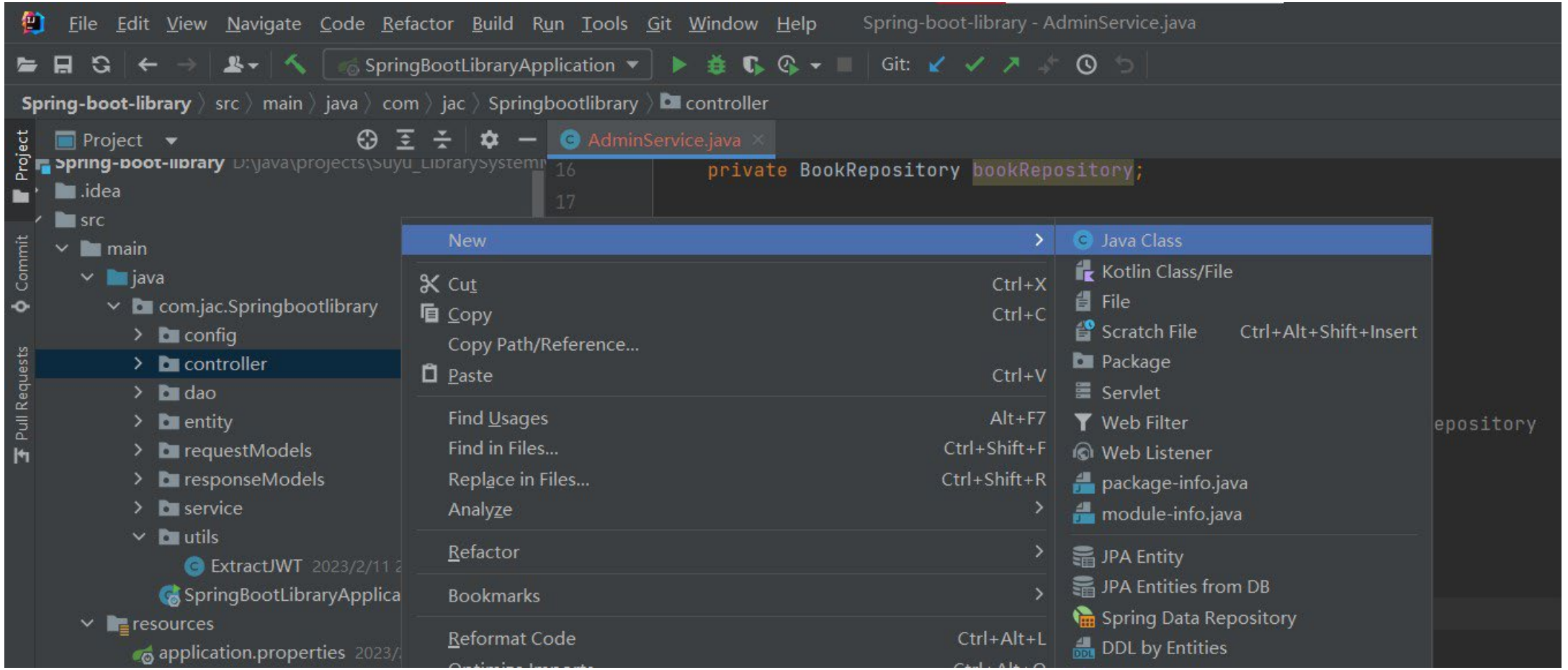
Step 4, create a postBook(), which allow us to get book info from AddBookRequest, save the book to our BookRepository.



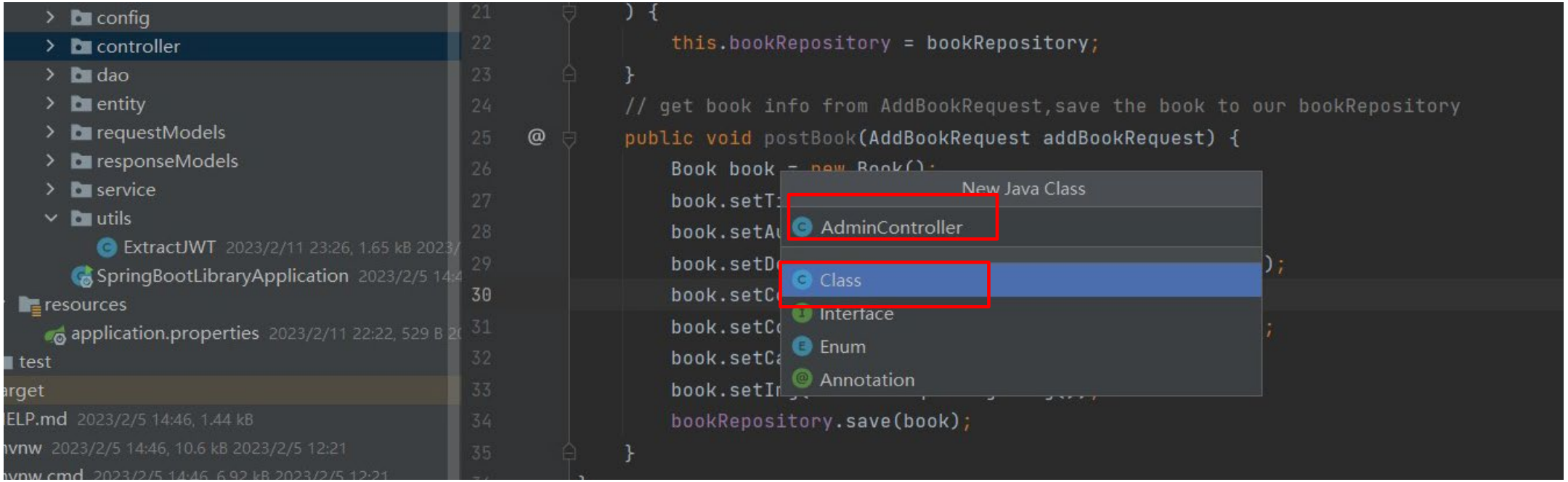
3. SPINGBOOT ADMIN CONTROLLER AND SECURITY

Now we will create a new admin controller with the endpoint to be able to create a new book.

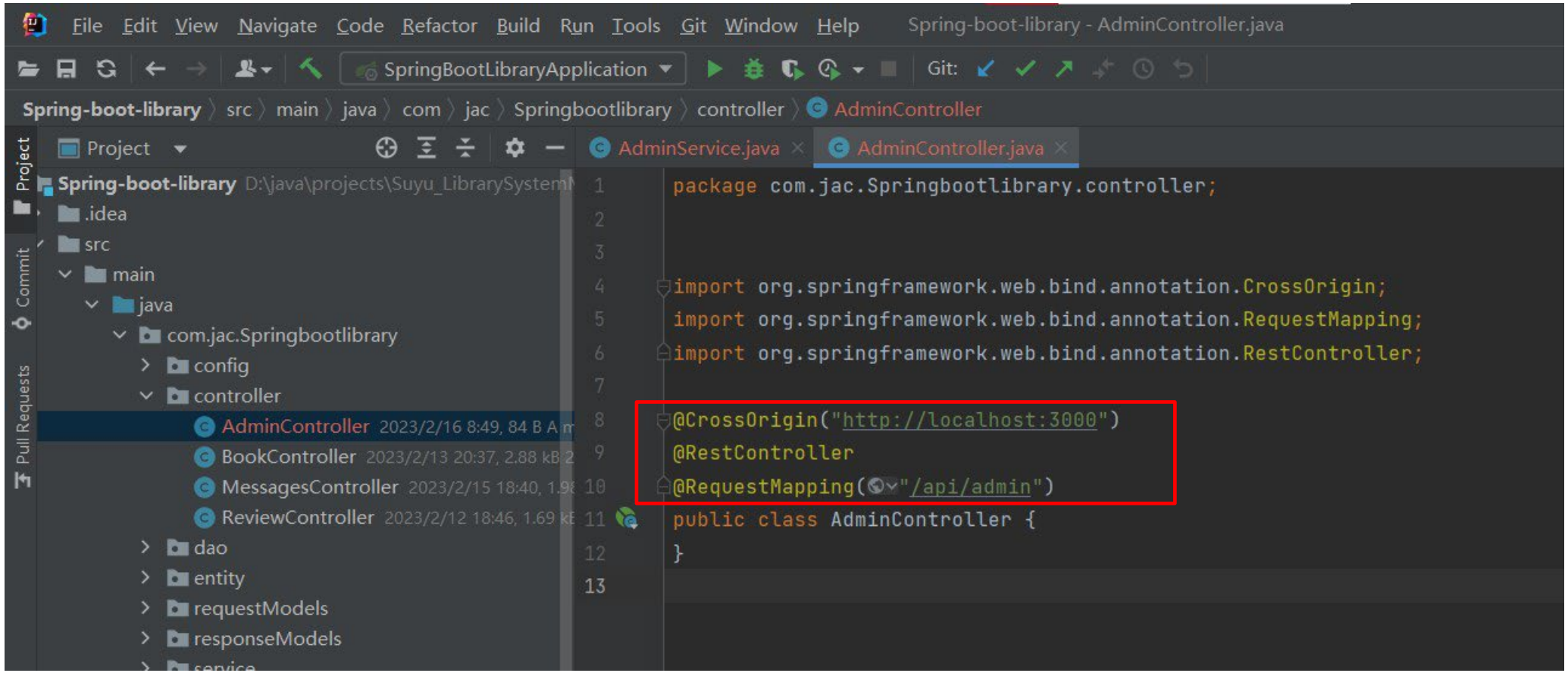
Step 1, Right click our “controller” package and choose “new”, “Java class”.



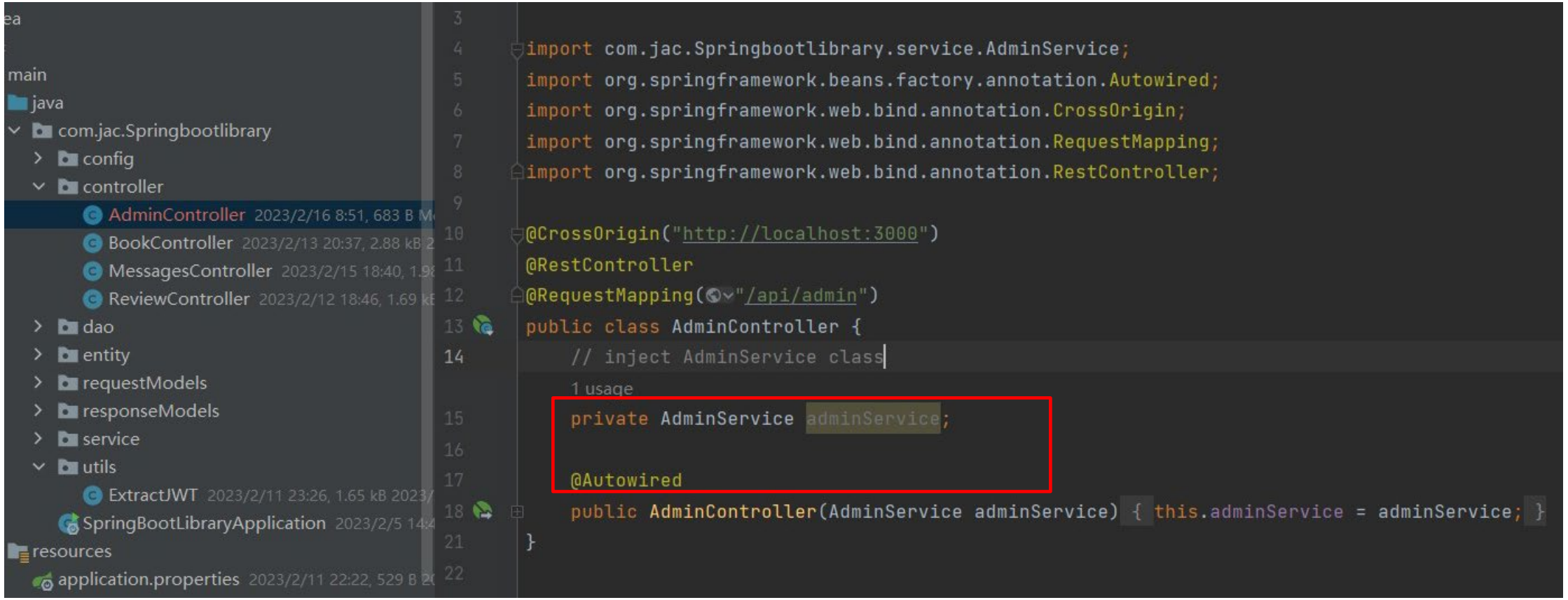
Choose “Class”, input the name “AdminController”.



Step 2, add annotations:
@CrossOrigin("http://localhost:3000")
@RestController
@RequestMapping("/api/admin")



Step 3, inject AdminService and then auto wire our AdminController constructor to our AdminService.

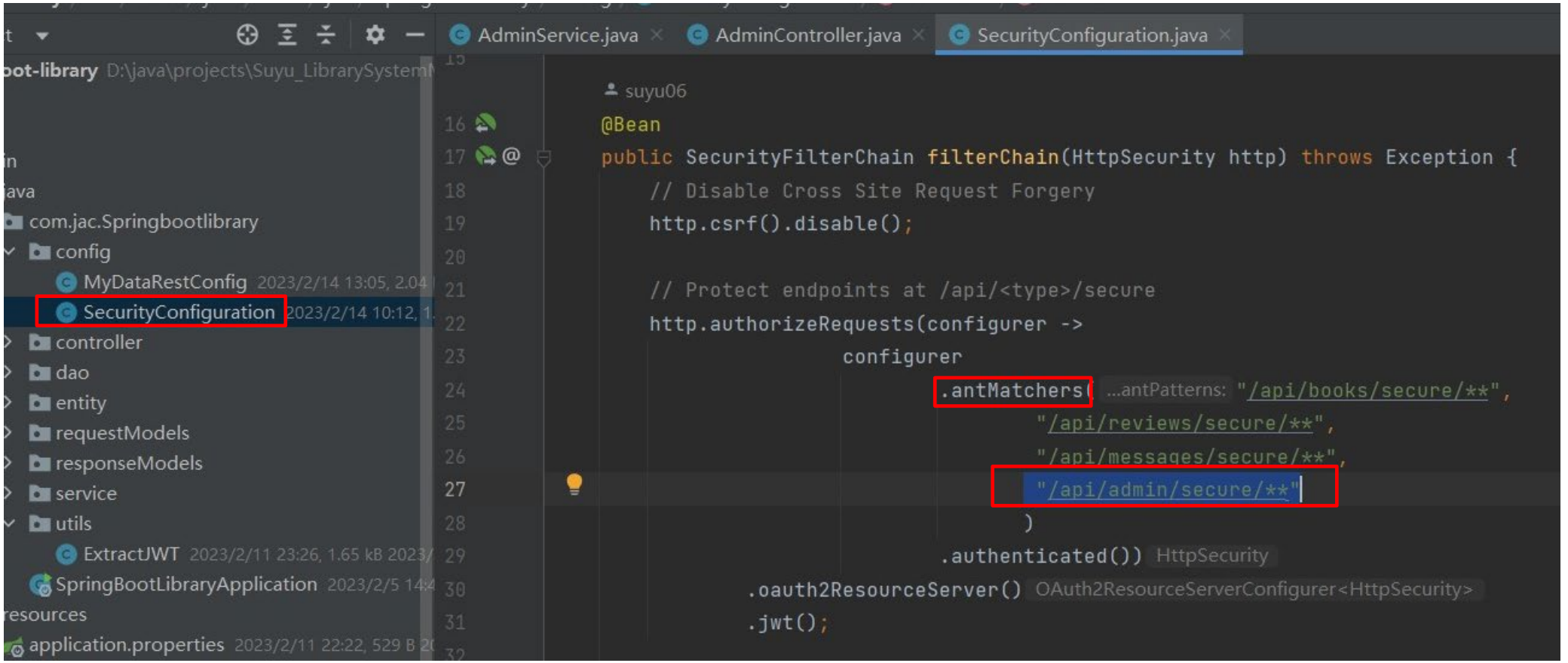


Step 4, use PostMapping method to create add book API.

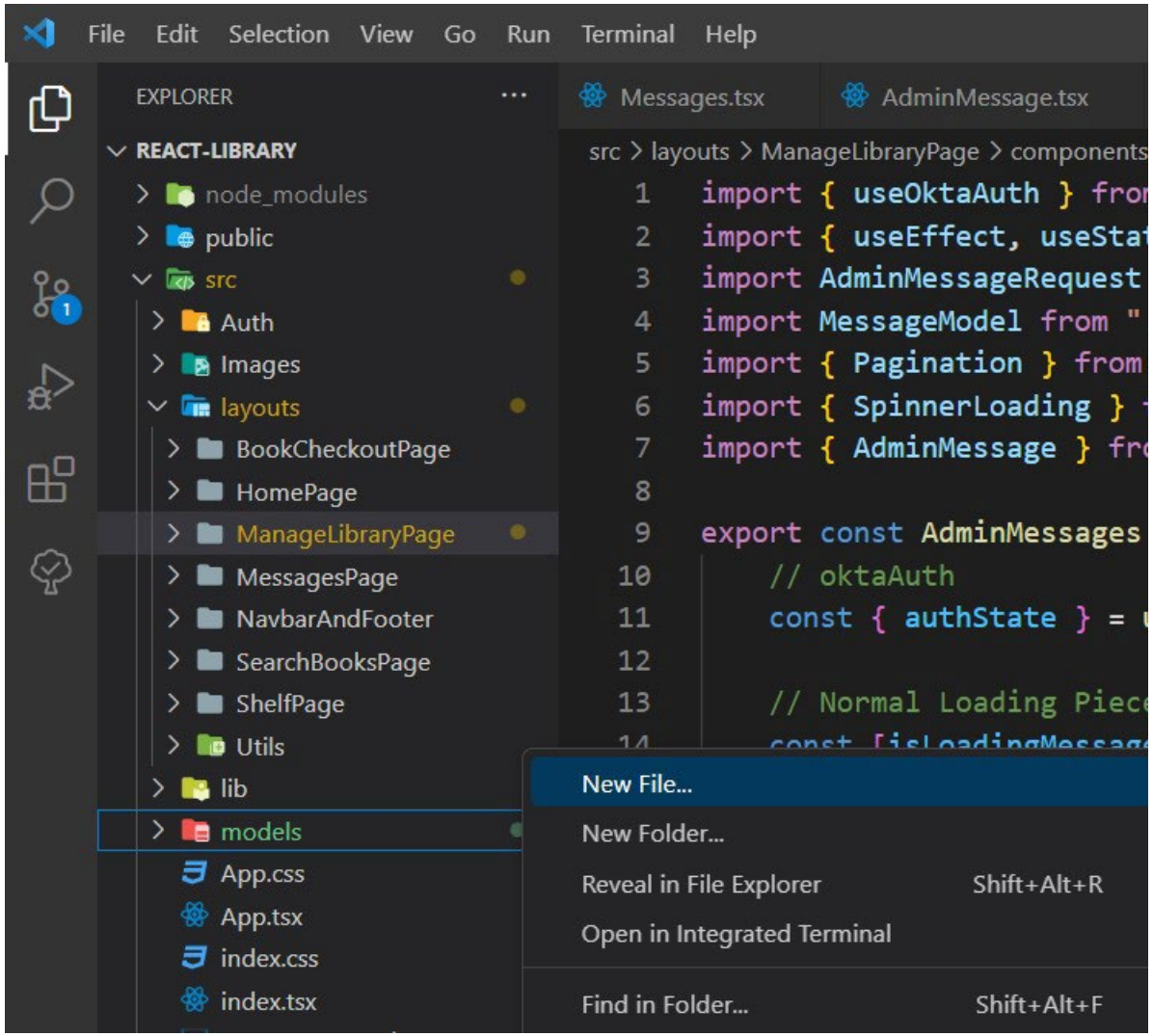
AdminService.java | AdminController.java

```
2 usages
15      private AdminService adminService;
16
17      @Autowired
18      public AdminController(AdminService adminService) {
19          this.adminService = adminService;
20      }
21      // add book api
22      @PostMapping("/secure/add/book")
23      public void postBook(@RequestHeader(value="Authorization") String token,
24                           @RequestBody AddBookRequest addBookRequest) throws Exception {
25          // extract user type from the token
26          String admin = ExtractJWT.payloadJWTExtraction(token, extraction: "\"userType\"");
27          //if not role of Admin, can not use this function
28          if (admin == null || !admin.equals("admin")) {
29              throw new Exception("Administration page only");
30          }
31          //call postBook method in adminService class to save the book from the request to our database
32          adminService.postBook(addBookRequest);
33      }
```

Step 5, Hop into our “config” package , “SecurityConfiguration” class,
the same thing that we ‘ve done before, add “admin/secure/**” to our .antMatchers().



4. REACT ADD BOOK REQUEST MODEL



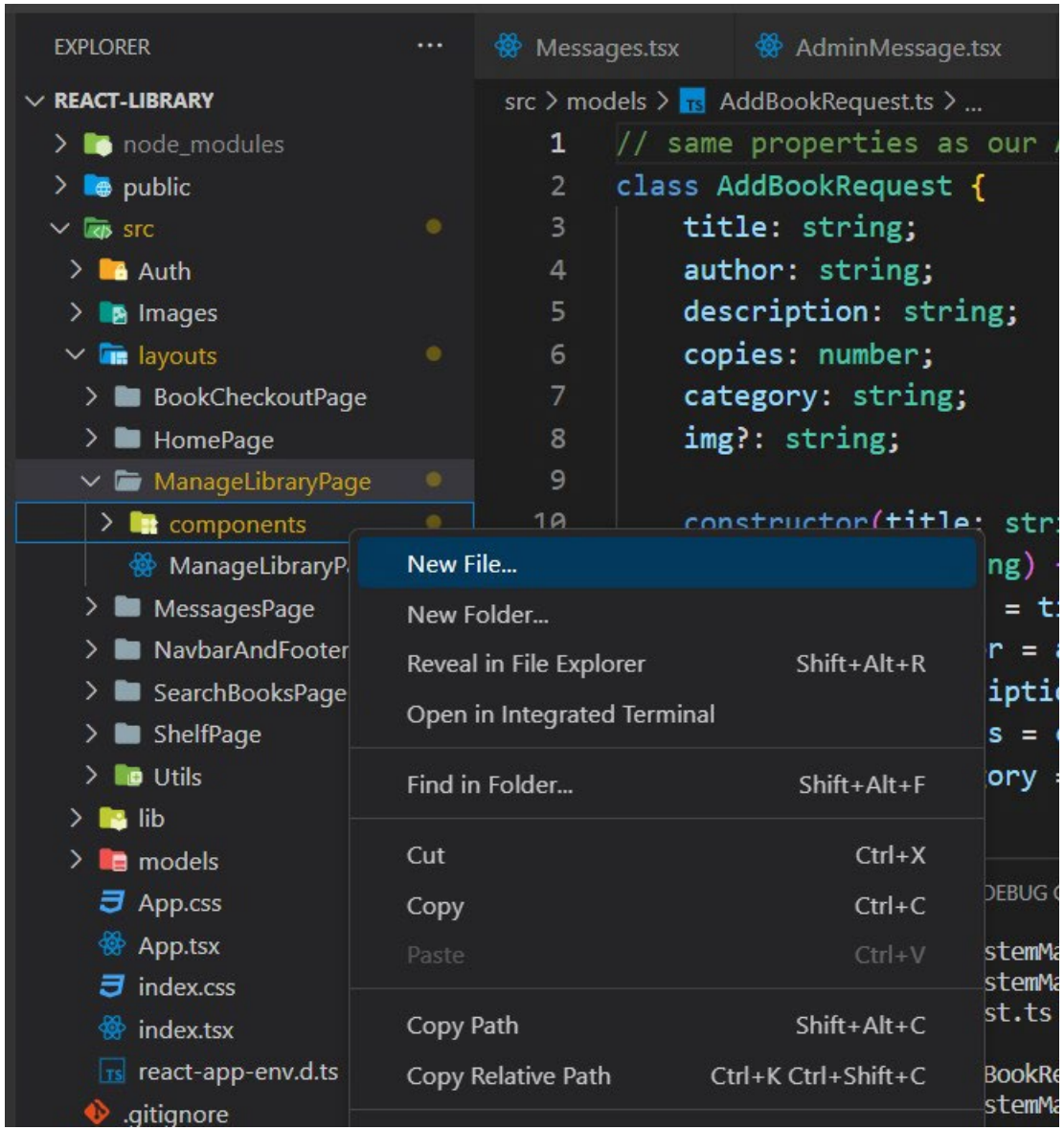
Step1 Let's go to our “models” folder, create a new typescript file that's called “AddBookRequest.ts”.

Step 2, add the same properties as our AddBookRequest class in springboot, and its constructor.

src > models > AddBookRequest.ts > default

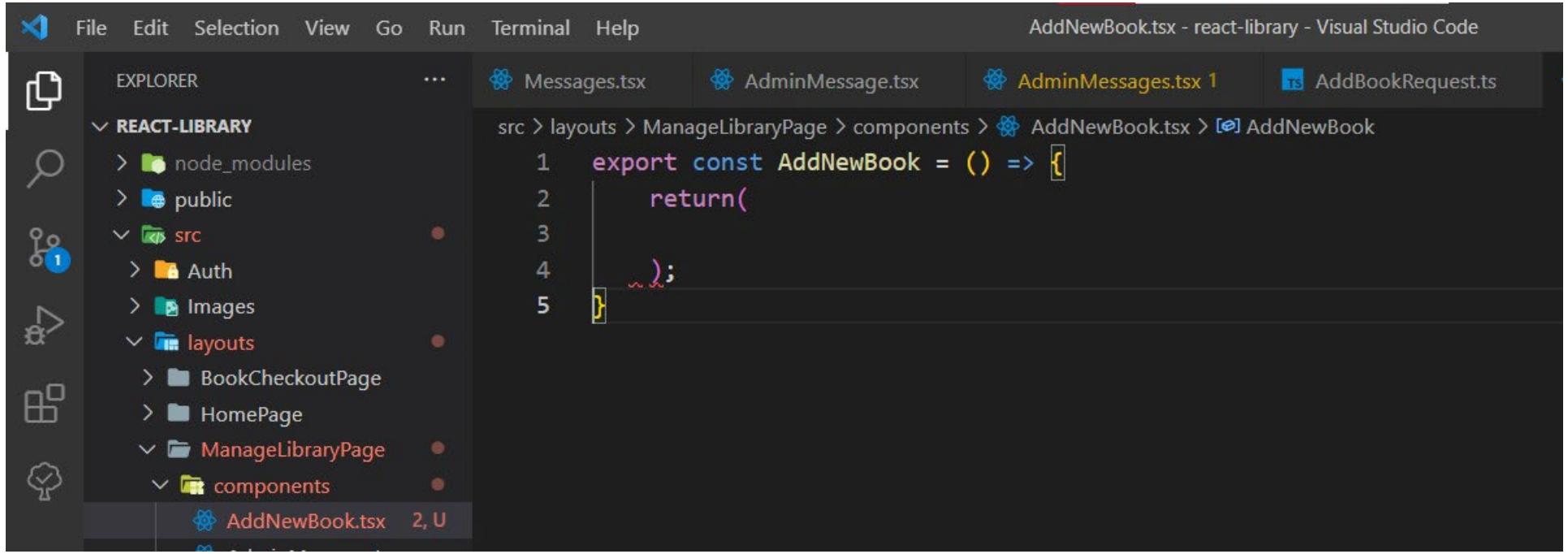
```ts
class AddBookRequest {
    title: string;
    author: string;
    description: string;
    copies: number;
    category: string;
    img?: string;

    constructor(title: string, author: string, description: string, copies: number,
        category: string) {
            this.title = title;
            this.author = author;
            this.description = description;
            this.copies = copies;
            this.category = category;
        }
}

export default AddBookRequest;
```

5. REACT ADD BOOK COMPONENT STATE AND HTML/CSS

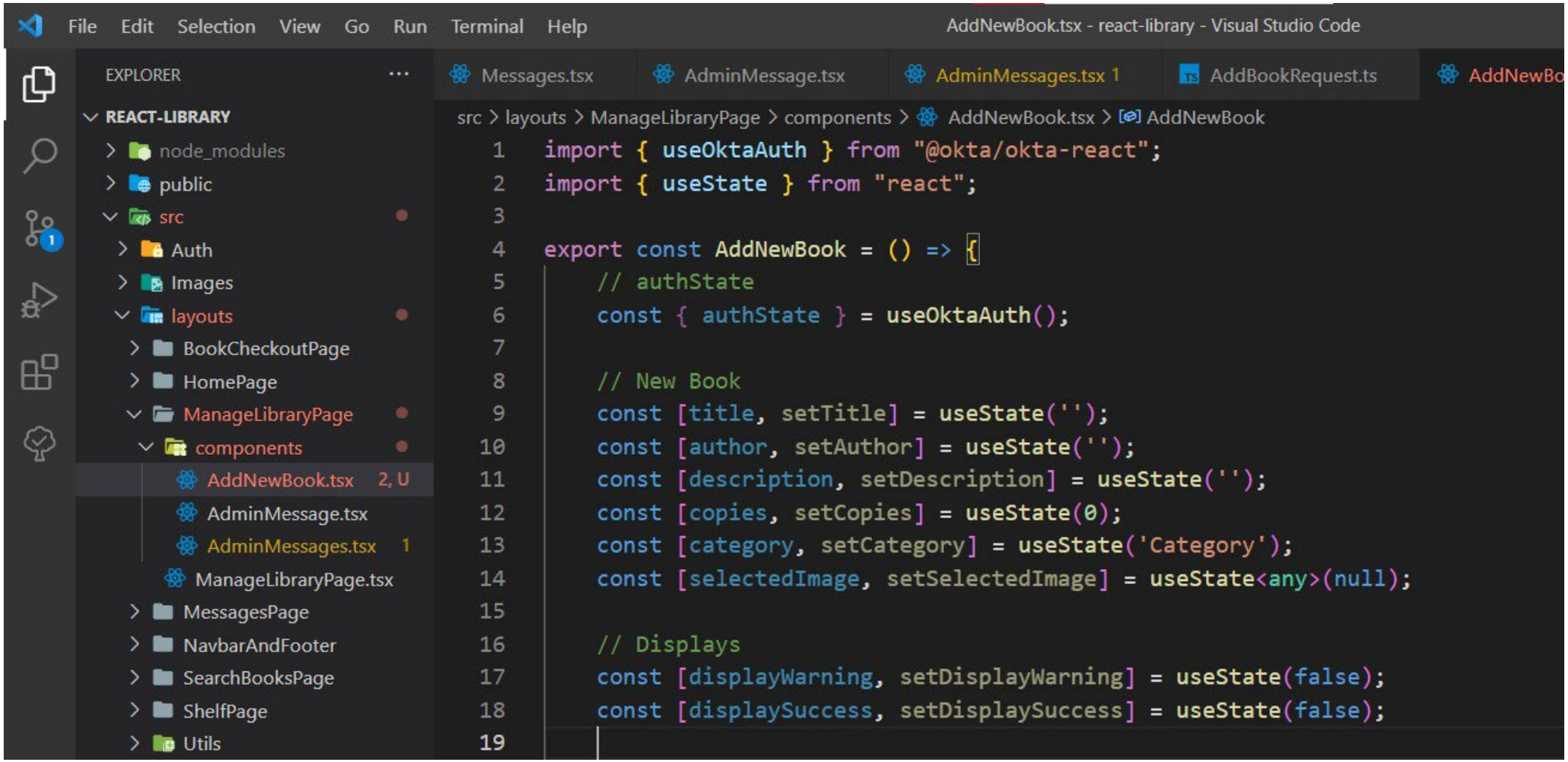


Step 1, jumping into our “layouts”, “ManageLibraryPage” , go into “components”, right click , choose “new file”, name it “AddNewBook.tsx”.

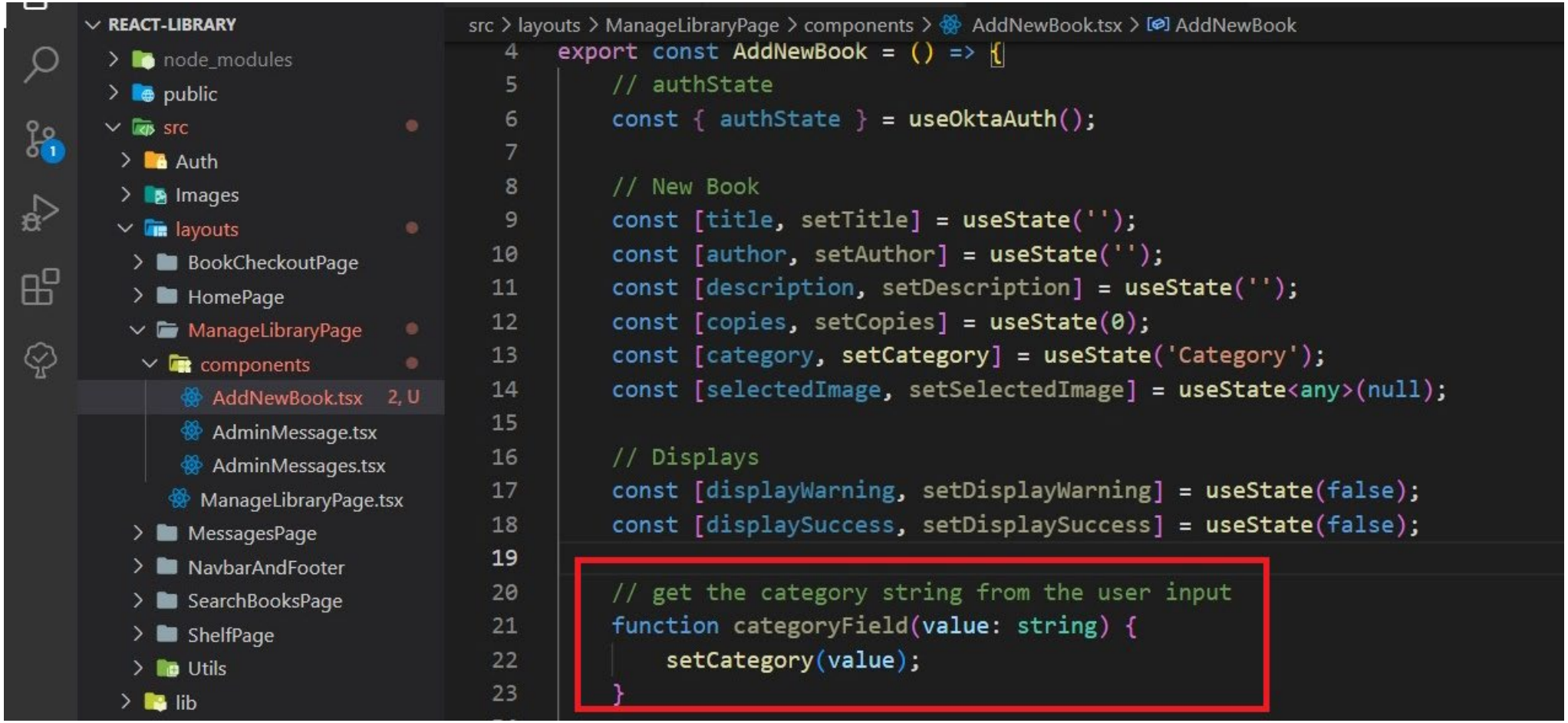
Step 2, In AddNewBook.tsx, as always, create export const AddNewBook() structure.



Step 3, import oktaAuth and create useState for new book and alert messages.



Step 4, create a categoryField() to fetch the category string from the user's choice in the category dropdown menu.



Step 5, fill in return part with html/CSS code.

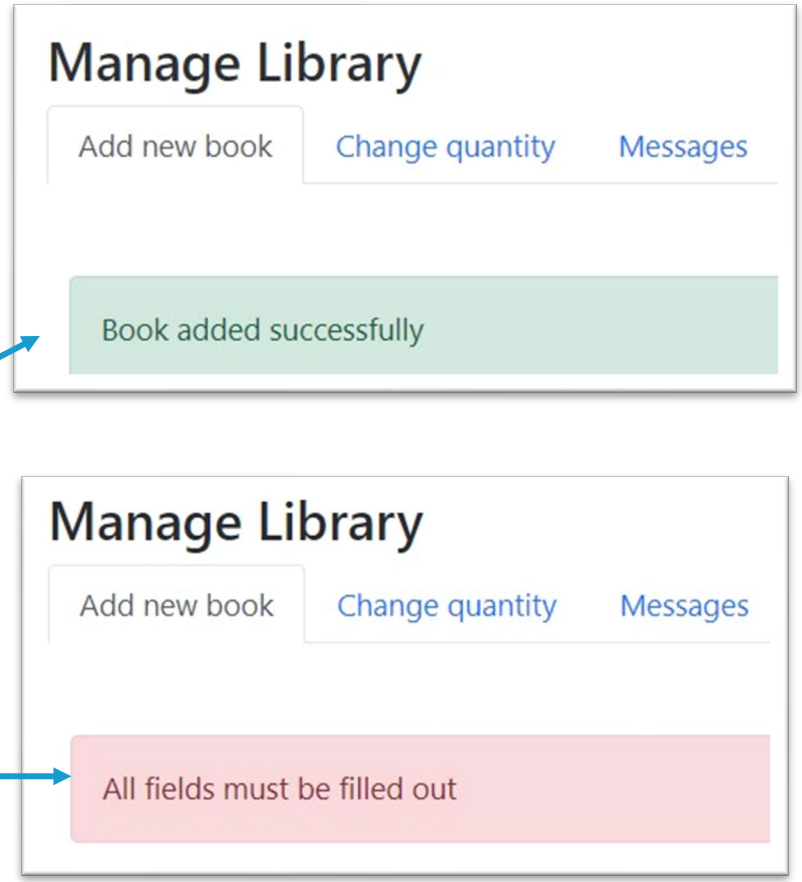
5.1 successful banner and alert banner

Terminal Help AddNewBook.tsx - react-library - Visual Studio Code

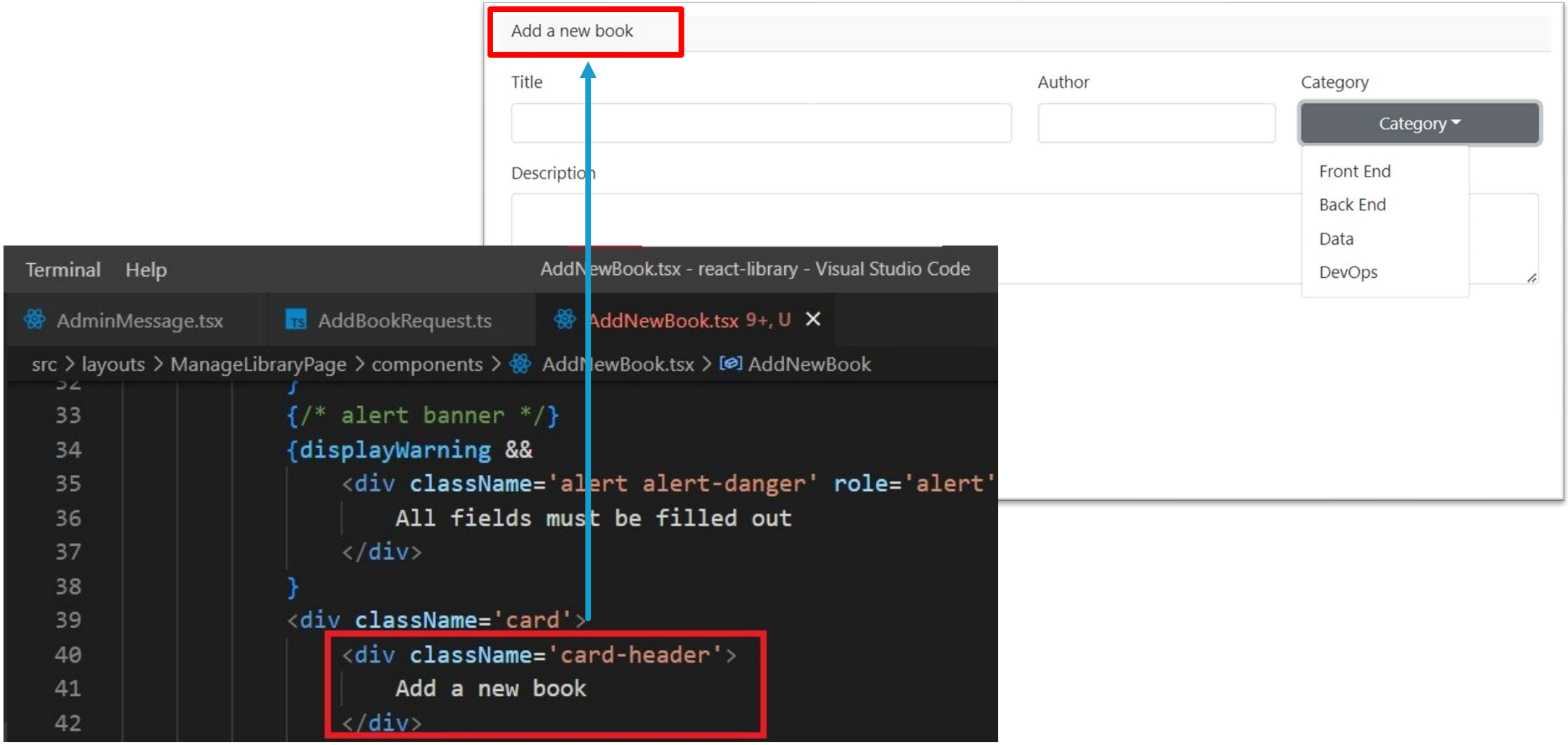
AdminMessage.tsx | AddBookRequest.ts | AddNewBook.tsx 9+, U

src > layouts > ManageLibraryPage > components > AddNewBook.tsx > AddNewBook

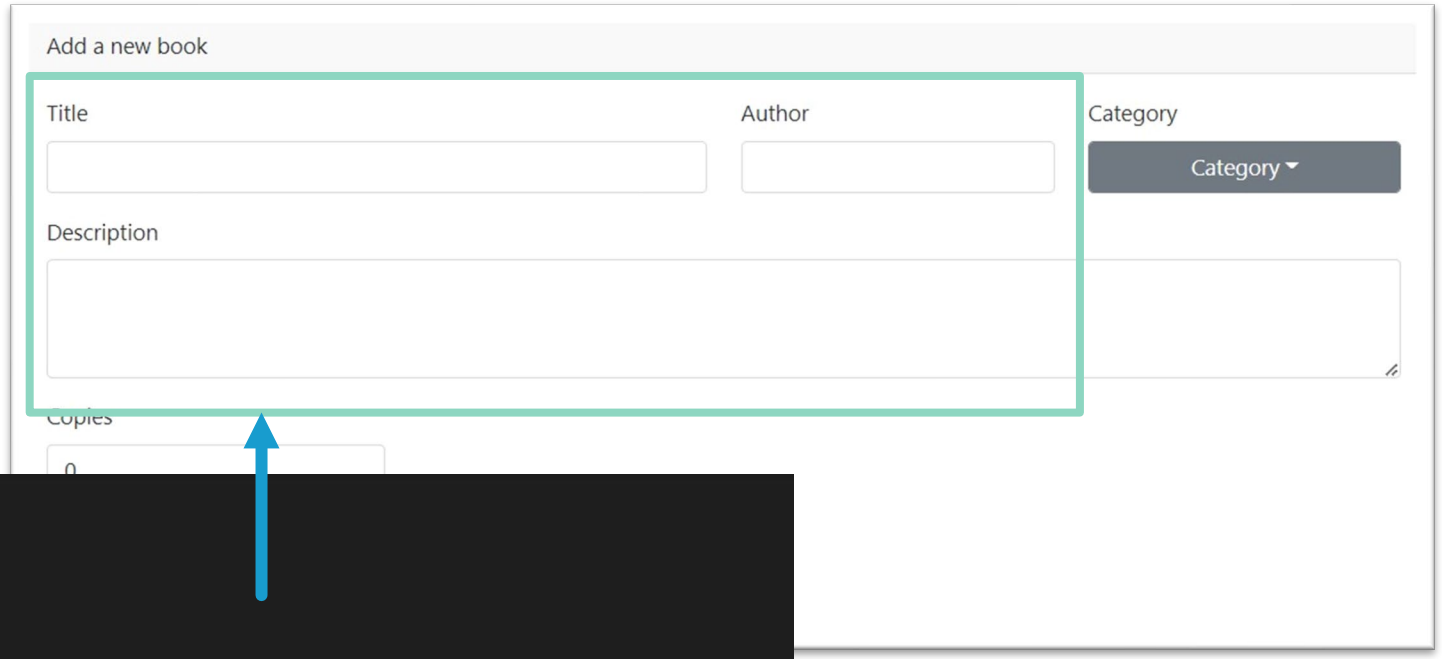
```
21      function categoryField(value: string) {
22          setCategory(value);
23      }
24
25      return(
26          <div className='container mt-5 mb-5'>
27              {/* successful banner */}
28              {displaySuccess &&
29                  <div className='alert alert-success' role='alert'>
30                      Book added successfully
31                  </div>
32              }
33              {/* alert banner */}
34              {displayWarning &&
35                  <div className='alert alert-danger' role='alert'>
36                      All fields must be filled out
37                  </div>
38              }
```



5.2 card header “Add a new book” part

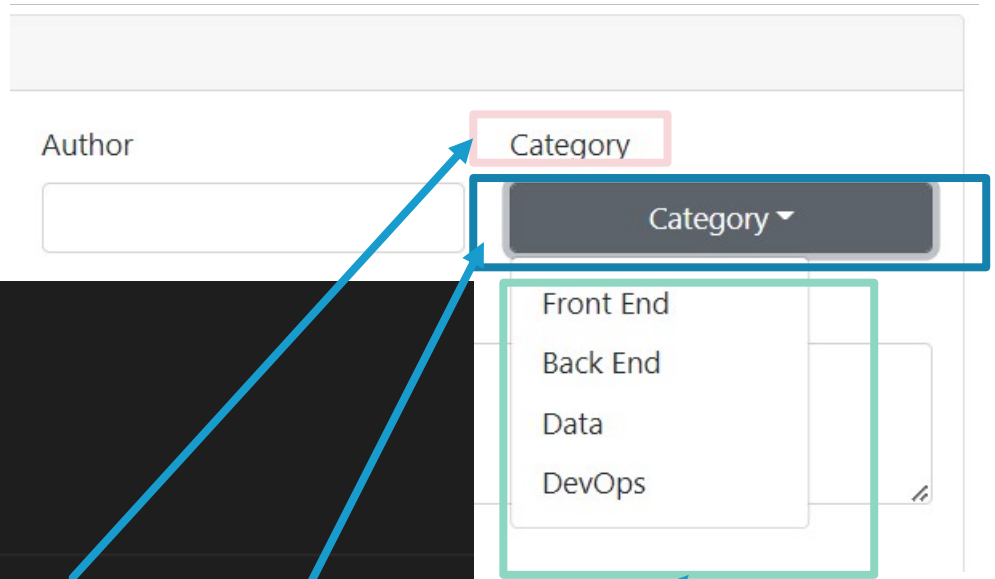


5.3 book info part



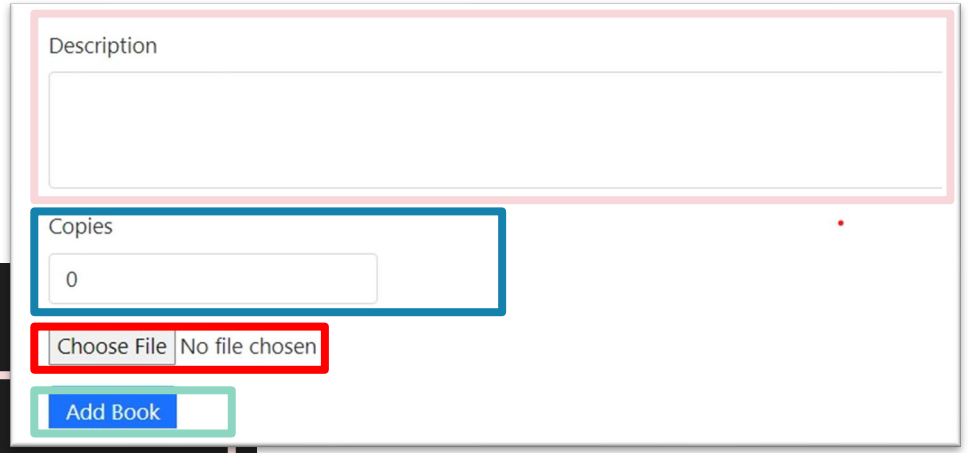

```
{/* form in the card body */}
<div className='card-body'>
    <form method='POST'>
        <div className='row'>
            {/* basic book info */}
            <div className='col-md-6 mb-3'>
                <label className='form-label'>Title</label>
                <input type="text" className='form-control' name='title' required
                    onChange={e => setTitle(e.target.value)} value={title} />
            </div>
            <div className='col-md-3 mb-3'>
                <label className='form-label'> Author </label>
                <input type="text" className='form-control' name='author' required
                    onChange={e => setAuthor(e.target.value)} value={author}/>
            </div>
            {/* drop down menu */}
```

5.4 “Category” dropdown menu part.




```
<div className='col-md-3 mb-3'>
    <label className='form-label'> Author </label>
    <input type="text" className='form-control' name='author' required
        onChange={e => setAuthor(e.target.value)} value={author}/>
</div>
{/* drop down menu */}
<div className='col-md-3 mb-3'>
    <label className='form-label'> Category</label>
    <button className='form-control btn btn-secondary dropdown-toggle' type='button'
        id='dropdownMenuButton1' data-bs-toggle='dropdown' aria-expanded='false'>
            {category}
    </button>
    {/* contene of drop down menu */}
    <ul id='addNewBookId' className='dropdown-menu' aria-labelledby='dropdownMenuButton1'>
        <li><a onClick={() => categoryField('FE')} className='dropdown-item'>Front End</a></li>
        <li><a onClick={() => categoryField('BE')} className='dropdown-item'>Back End</a></li>
        <li><a onClick={() => categoryField('Data')} className='dropdown-item'>Data</a></li>
        <li><a onClick={() => categoryField('DevOps')} className='dropdown-item'>DevOps</a></li>
    </ul>
</div>
```

5.5 input box part for book description, number of copies, choose book image file and “add book” button.




```
    </div>
    {/* input box part */}
    <div className='col-md-12 mb-3'>
        <label className='form-label'>Description</label>
        <textarea className='form-control' id='exampleFormControlTextarea1' rows={3}
            onChange={e => setDescription(e.target.value)} value={description}></textarea>
    </div>
    <div className='col-md-3 mb-3'>
        <label className='form-label'>Copies</label>
        <input type='number' className='form-control' name='Copies' required
            onChange={e => setCopies(Number(e.target.value))} value={copies}/>
    </div>
    <input type='file'/>
    {/* add book btn */}
    <div>
        <button type='button' className='btn btn-primary mt-3'>
            Add Book
        </button>
    </div>
</form>
```

Step 6, Jump back into our ManagLibraryPage,in the tab content part, replace the static word "Add new book" with our new AddNewBook component, and import it .

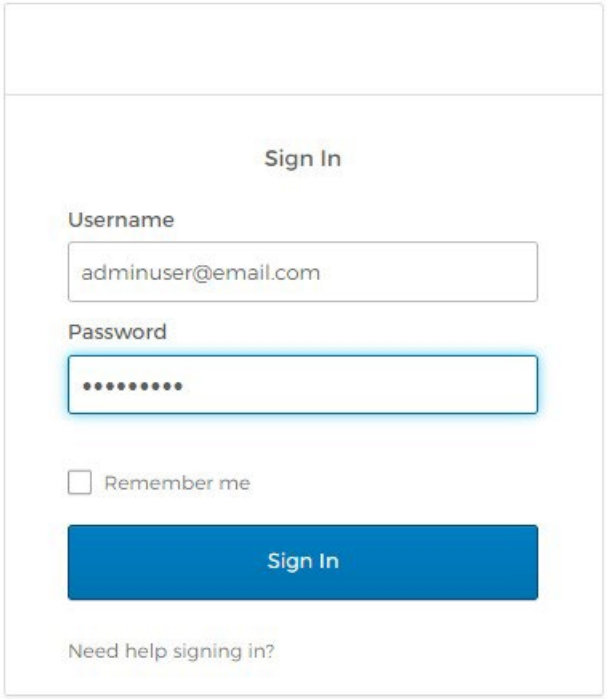
ManageLibraryPage.tsx

ryPage > ManageLibraryPage.tsx > ManageLibraryPage

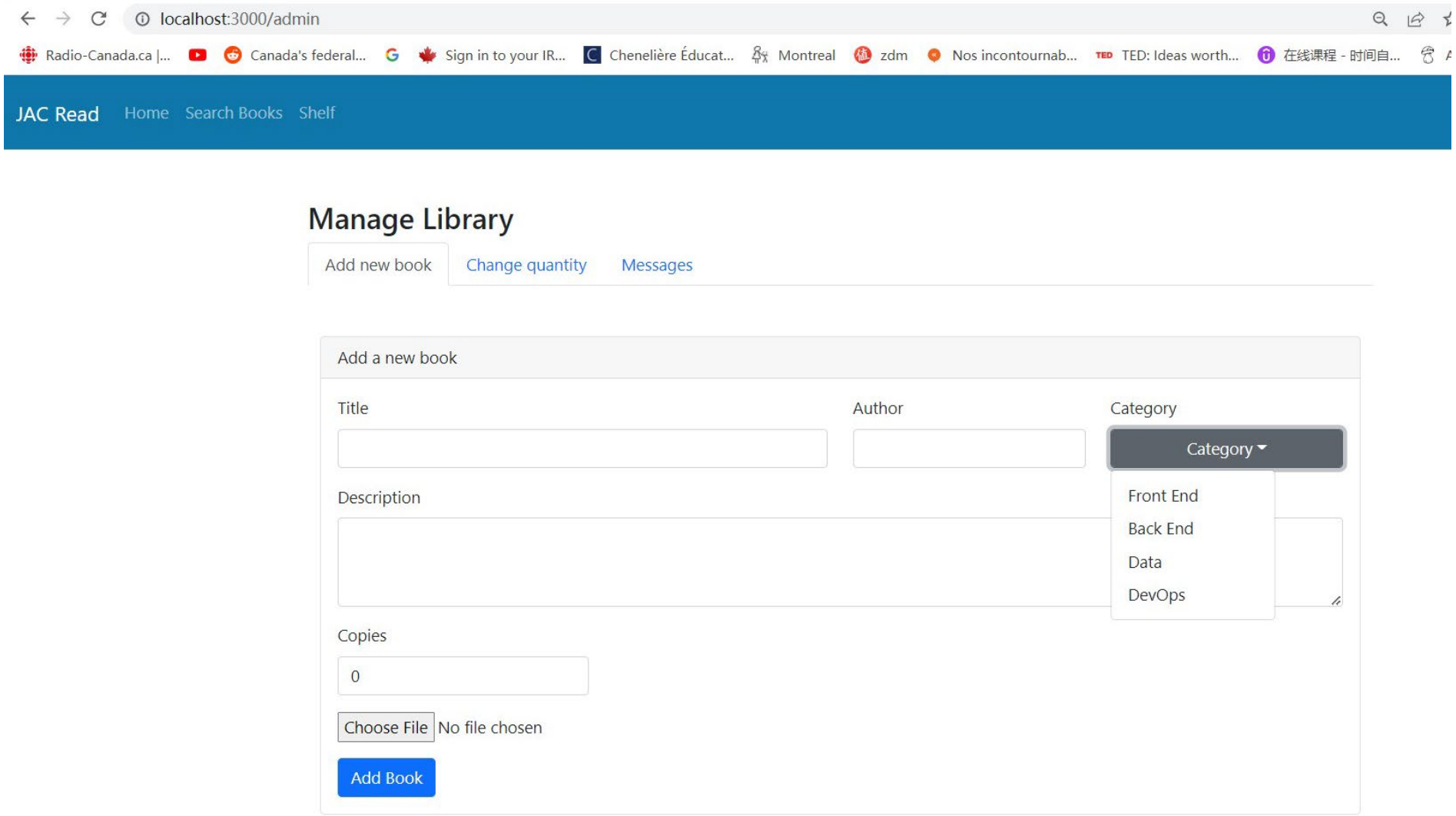
```
            </button>
        </div>
    </nav>
    {/*  content showed when the tab is clicked */}
    <div className='tab-content' id='nav-tabContent'>
        <div className='tab-pane fade show active' id='nav-add-book' role='tabpanel'
            aria-labelledby='nav-add-book-tab'>
            Add New Book
        </div>
```

```
                </nav>
                {/*  content showed when the tab is clicked */}
                <div className='tab-content' id='nav-tabContent'>
                    <div className='tab-pane fade show active' id='nav-add-book' role='tabpanel'
                        aria-labelledby='nav-add-book-tab'>
                        <AddNewBook/>
                    </div>
```

Sign in with our admin user’s email and password.



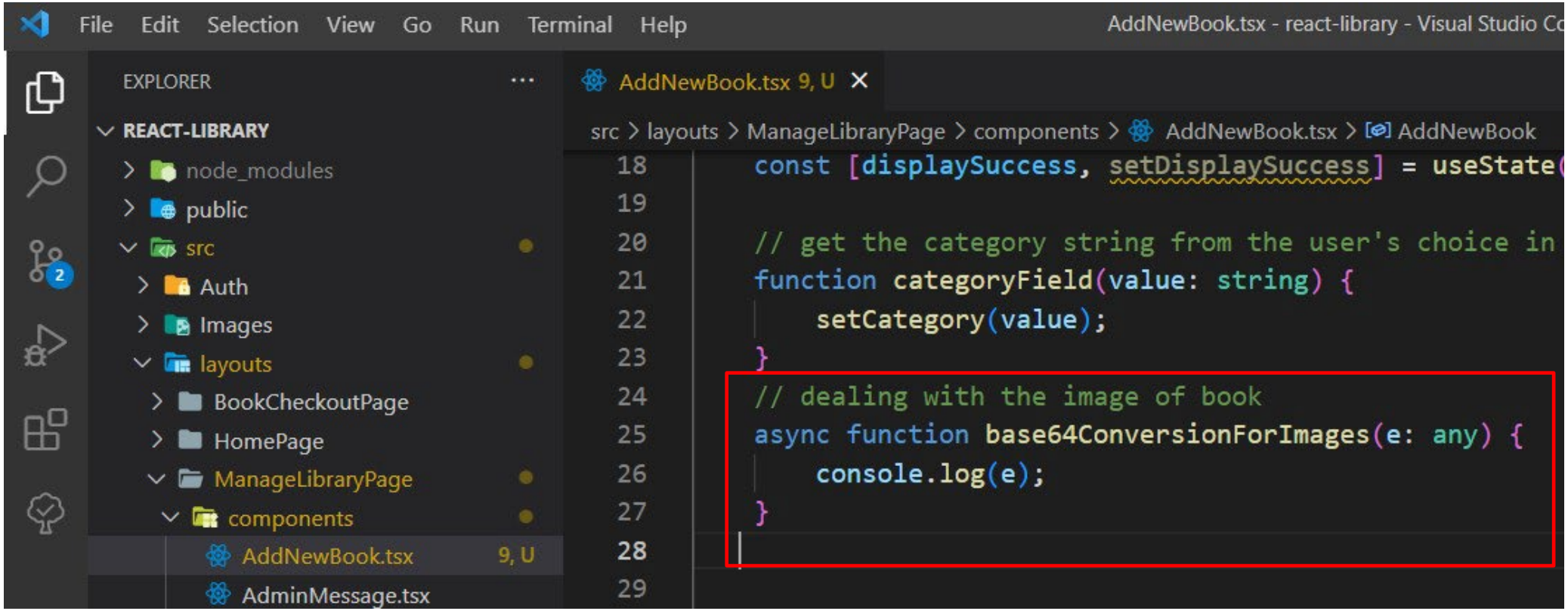
Visit http://localhost:3000/admin, we will get this add new book page.



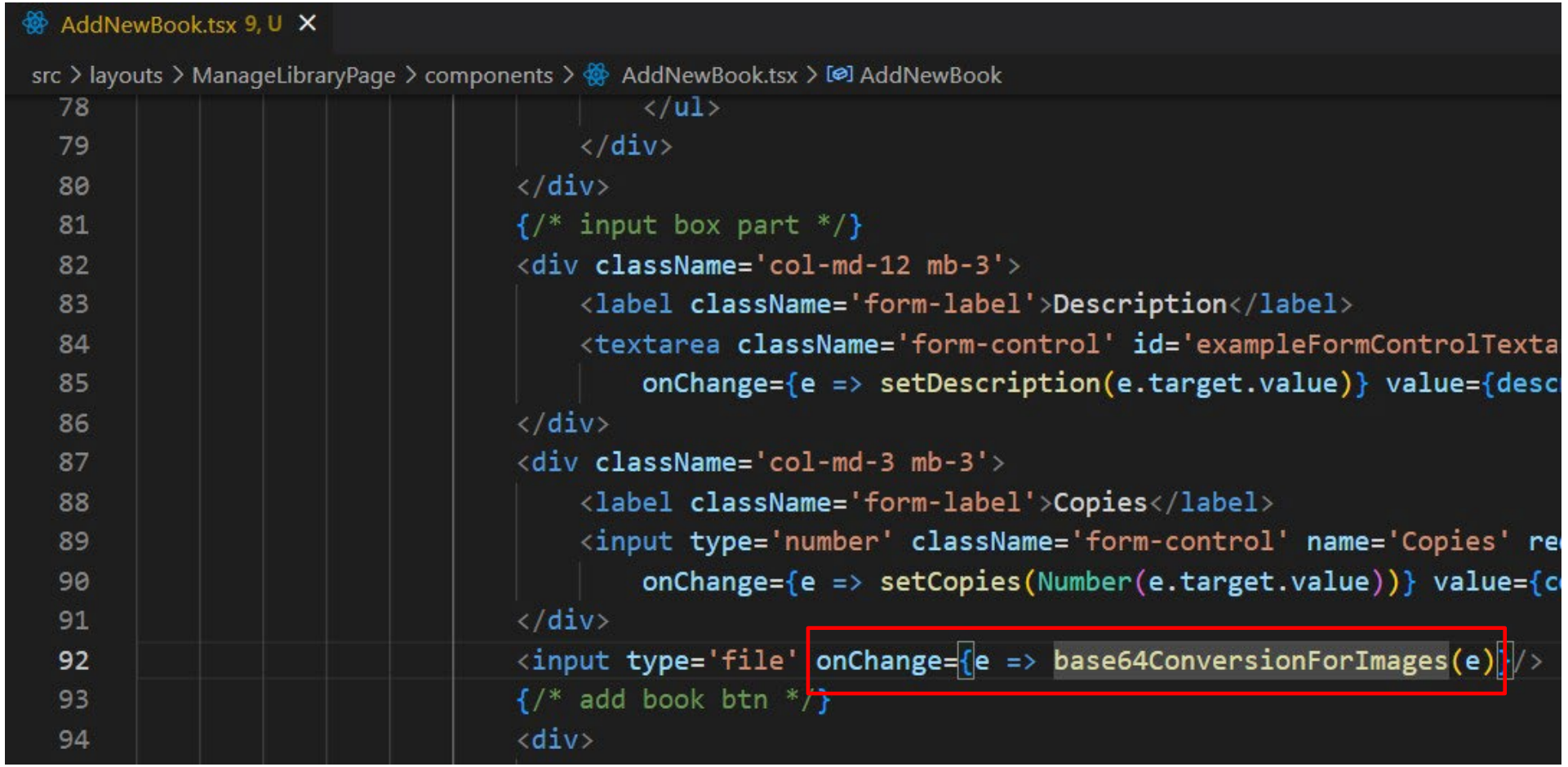
6. REACT SAVE IMAGE TO STATE

one thing that we need to do is when we select a file, allow our client side application to render this into a base 64. So we can send over this to the spring boot application.

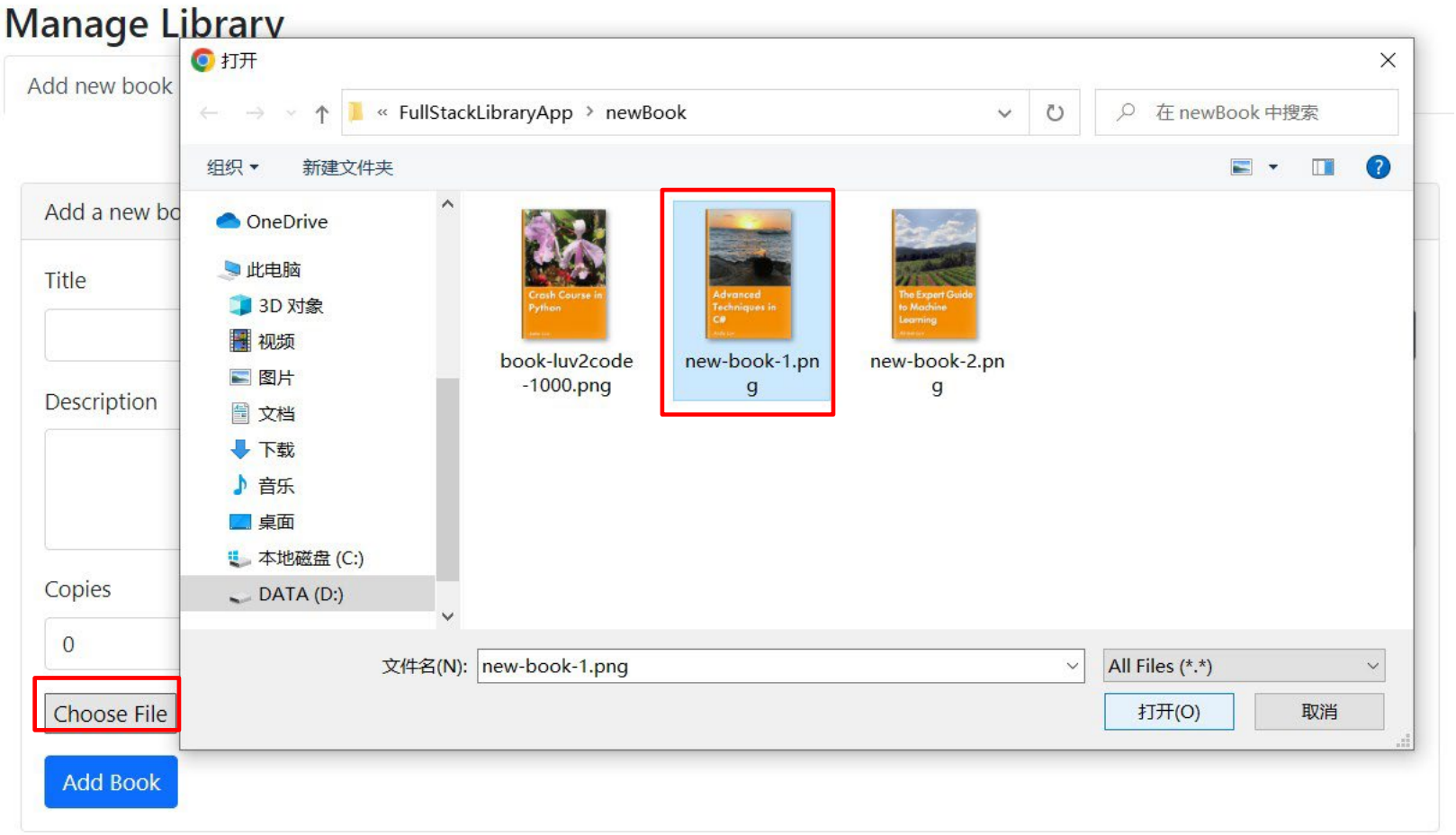
Step 1, In AddNewBook.tsx, under the function of categoryField(), create an async function base64ConversionForImages().



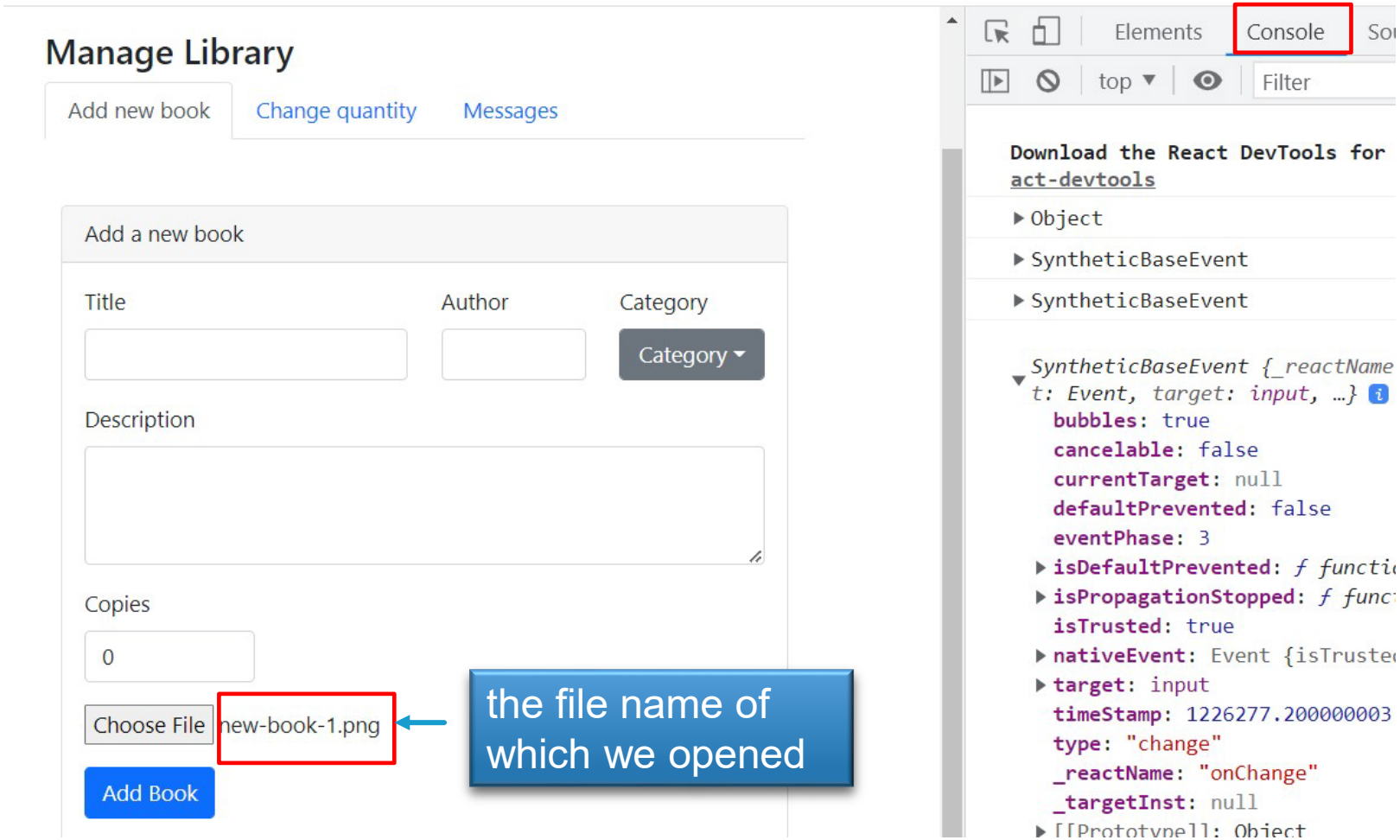
Step 2， add onChange listener to the file input.



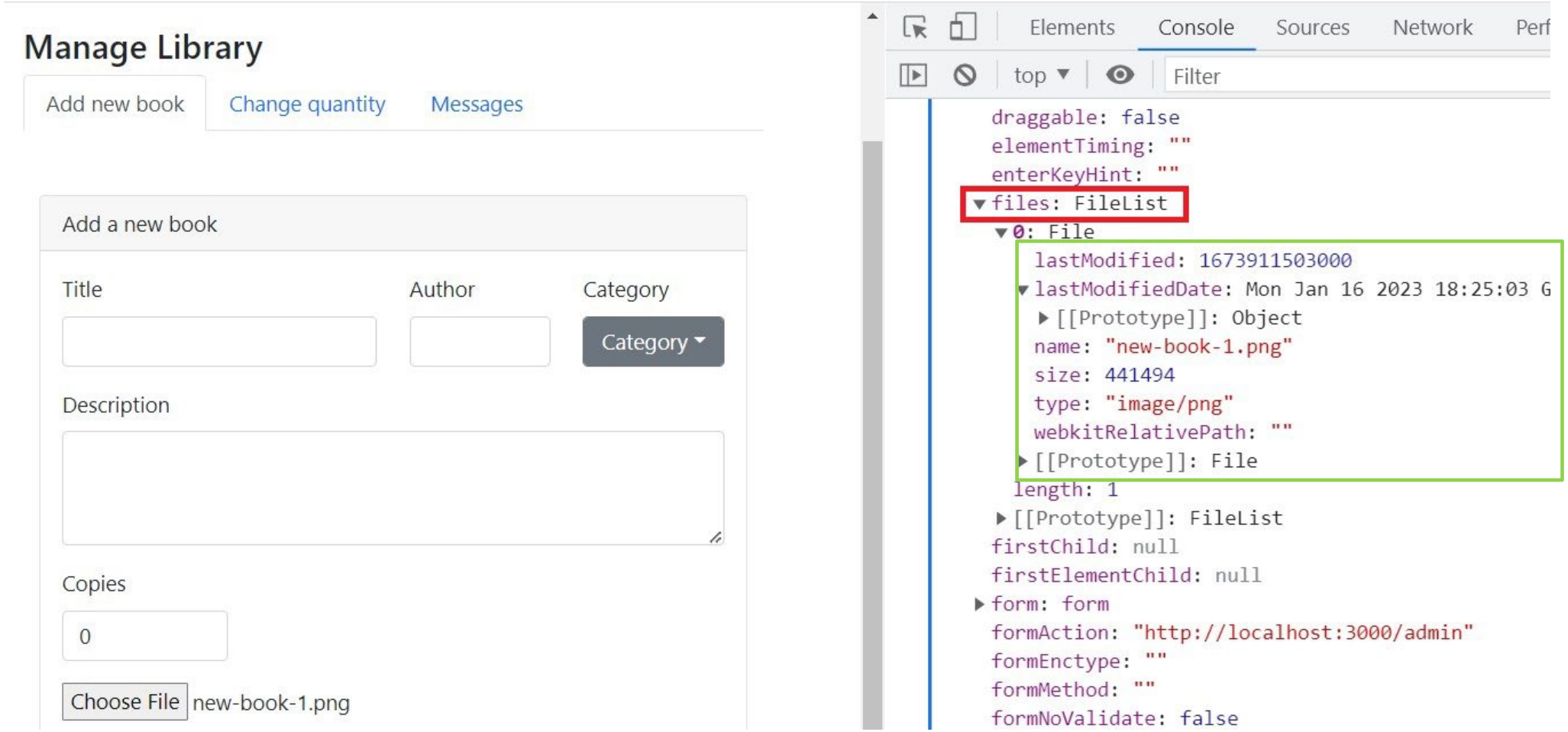
With admin user logged in status, go to our “add new book ” page, click on “Choose File”, open one image in our computer.
Here, I choose “new-book-1.png”.



Then, besides the “Choose File” , we can find the file name of which we opened.
Right click and choose “Inspect” or press “F12” to open the console,



Scroll down and find “files”, we can find all the information about the image file we chose, such as file name, file size ,file type etc.
We can see files are a list , and the file we chose is the element 0.



Step 3, create a function getBase64().
3.1 create a new fileReader.
3.2 then use readAsDataURL(). this method is used to read the contents of the specified Blob or File. When the read operation is finished, the readyState becomes DONE, and the loadend is triggered. At that time, the result attribute contains the data as a data: URL representing the file's data as a base64 encoded string.
3.3 Set the result of the reader to a piece of state that we can send over to our backend application.
3.4 if there is an error, print it in the console.

```
function getBase64(file: any) {
    // create a file readerread the contents of file of our ccomputer
    let reader = new FileReader();
    //read the contents of file,representing the file's data as a base64 encoded string.
    reader.readAsDataURL(file);
    // set the read result to our image
    reader.onload = function () {
        setSelectedImage(reader.result);
    };
    reader.onerror = function (error) {
        console.log('Error', error);
    }
}
```

Step 4, change the console.log into a if statement, which says if the index 0 element exists, call the function getBase64() and pass the index o file to the function.

```
// dealing with the image of book
async function base64ConversionForImages(e: any) {
    console.log(e);
}
```

```
// dealing with the image of book
async function base64ConversionForImages(e: any) {
    if (e.target.files[0]) {
        //passing this file into getBase64();
        getBase64(e.target.files[0]);
    }
}
```

7. SUBMIT NEW BOOK

As of right now, we have no way of being able to submit any data to our spring boot application. That‘s the thing we’re solving in this part.

Step 1, create a new async submitNewBook().
1.1 define the url string and specify the method and headers of add book request.

```
async function submitNewBook() {
    // url string
    const url = `http://localhost:8080/api/admin/secure/add/book`;
    // user is authenticated, all fields are not null:
    if (authState?.isAuthenticated && title !== '' && author !== '' && category !== 'Category'
        && description !== '' && copies >= 0) {
            // add the info input as a new book object's properties
            const book: AddBookRequest = new AddBookRequest(title, author, description, copies, category);
            book.img = selectedImage;
            const requestOptions = {
                method: 'POST',
                headers: {
                    Authorization: `Bearer ${authState?.accessToken?.accessToken}`,
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(book)
            };
```

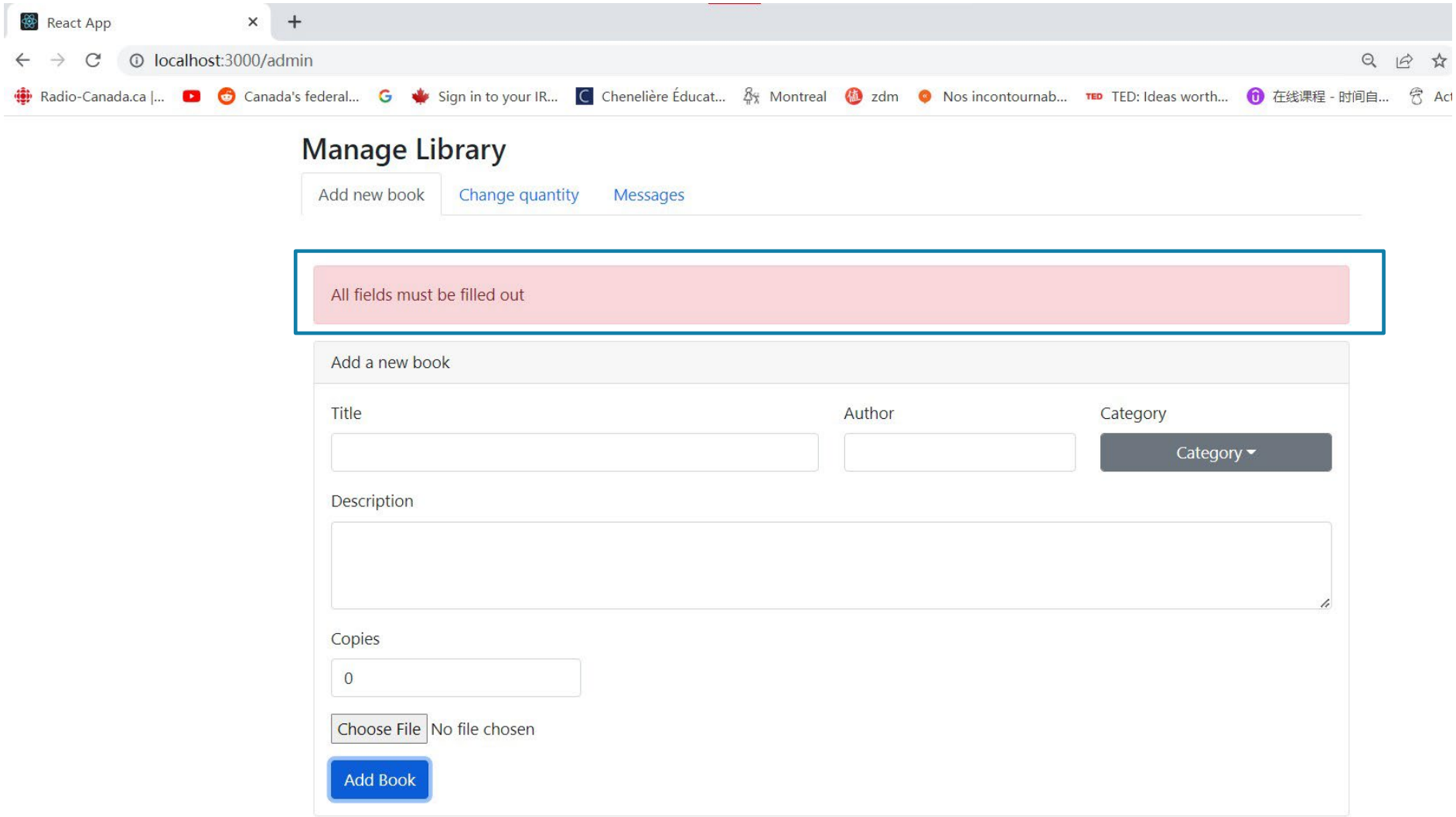
```
    };
    // fetch the data
    const submitNewBookResponse = await fetch(url, requestOptions);
    if (!submitNewBookResponse.ok) {
        throw new Error('Something went wrong!');
    }
    // reset all the input area to original empty state
    setTitle('');
    setAuthor('');
    setDescription('');
    setCopies(0);
    setCategory('Category');
    setSelectedImage(null);
    //display success banner
    setDisplayWarning(false);
    setDisplaySuccess(true);
} else {
    // display warning banner
    setDisplayWarning(true);
    setDisplaySuccess(false);
}
```

Step 2, pass the url and request info to the function await fetch() to call the API and fetch the data. Once we get the data, we set them as state.

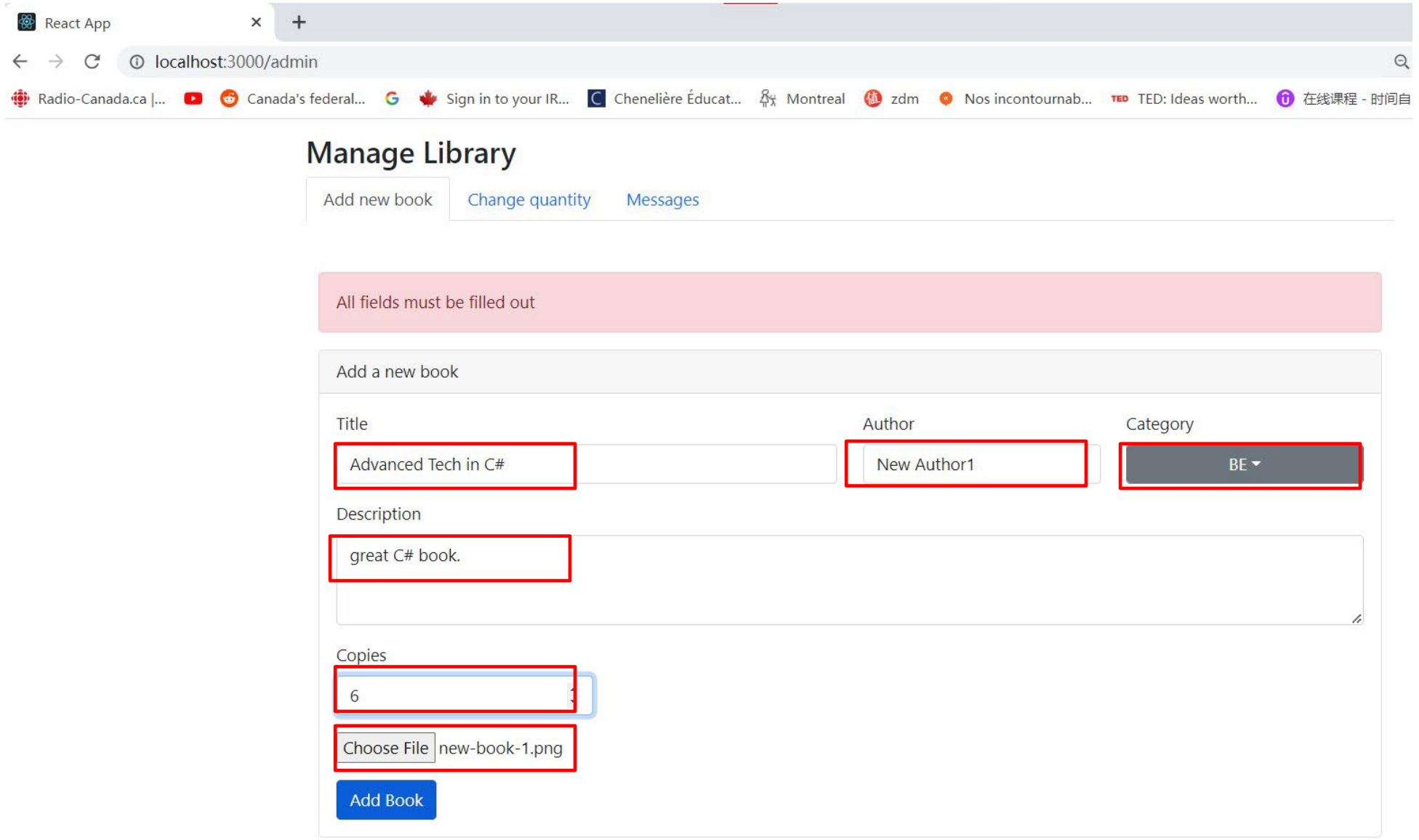
Step 2, add onClick listener to “add book” button.
Once the button is clicked, the submitNewBook function will be called.

```
{/* add book btn */}
<div>
    <button type='button' className='btn btn-primary mt-3'  onClick={submitNewBook}>
        Add Book
    </button>
</div>
```

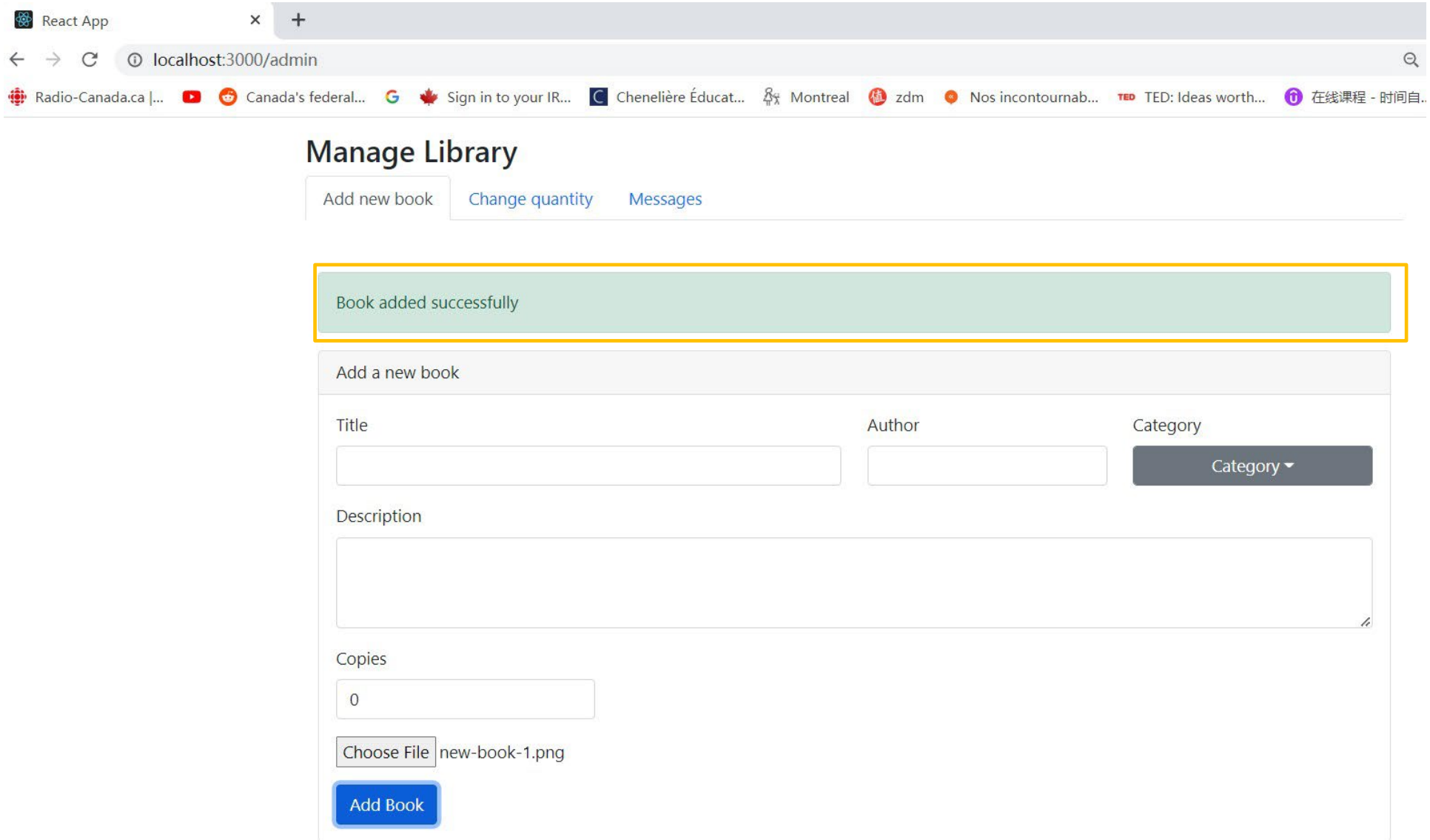
With admin user logged in status, visit http://localhost:3000/admin, let's try input nothing but click on the "Add Book" button, then an alert message "All fields must be filled out" will appear.



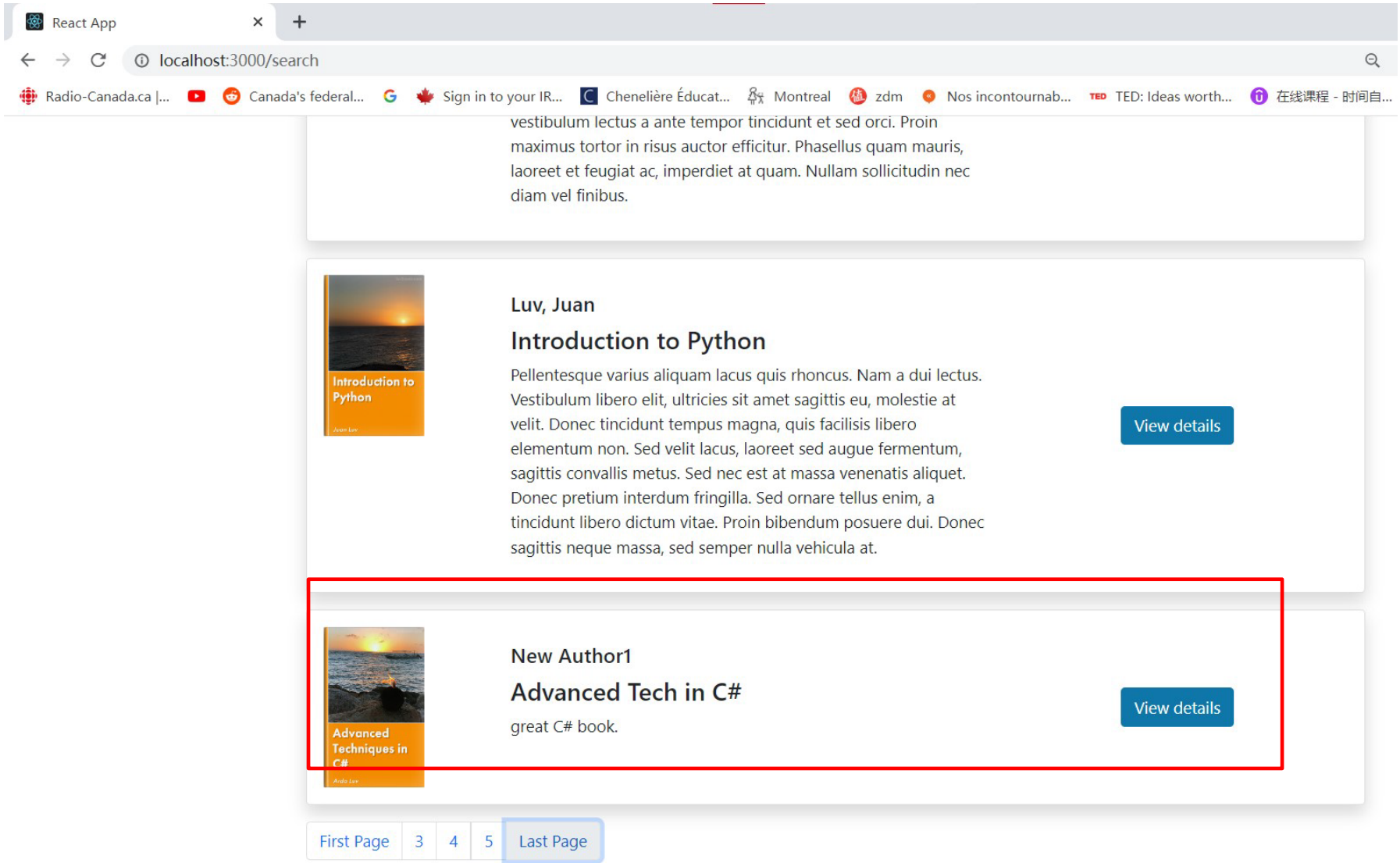
Then, input everything , click on the button.



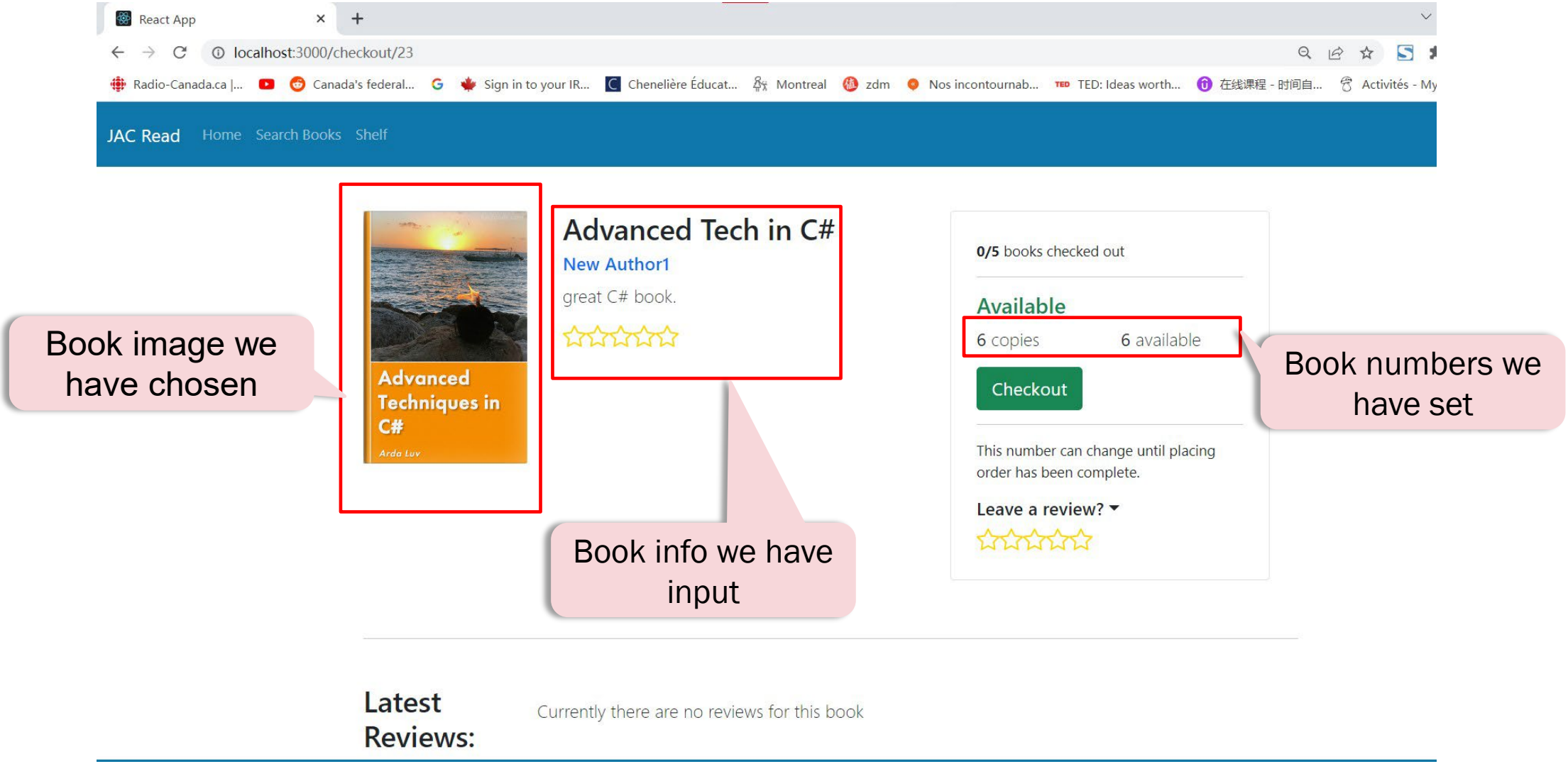
We will get this successful page.



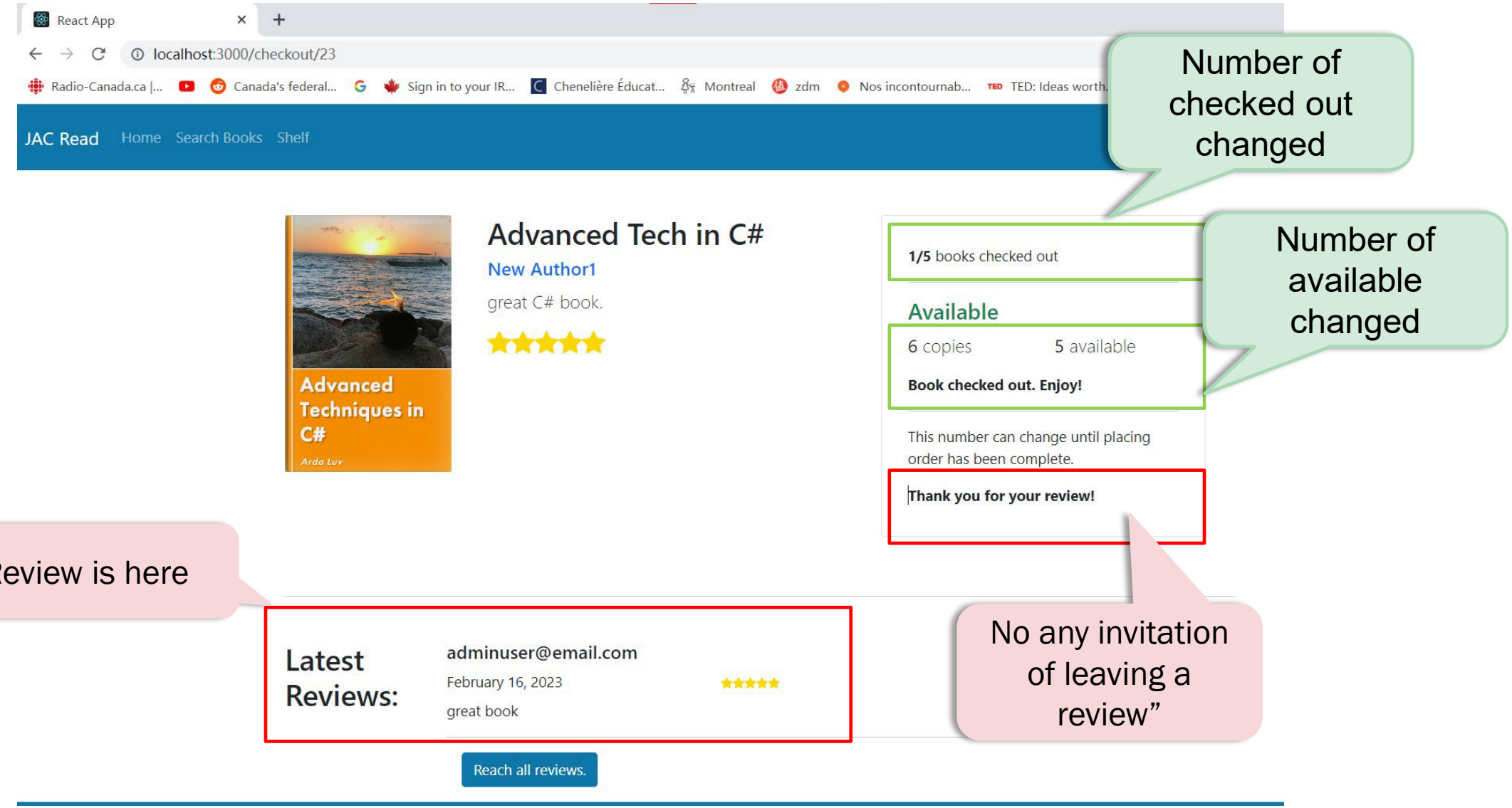
If we go to the last page of search pages, we can see the book we added is there. Click on the “View details” button, go to the details page of the book.



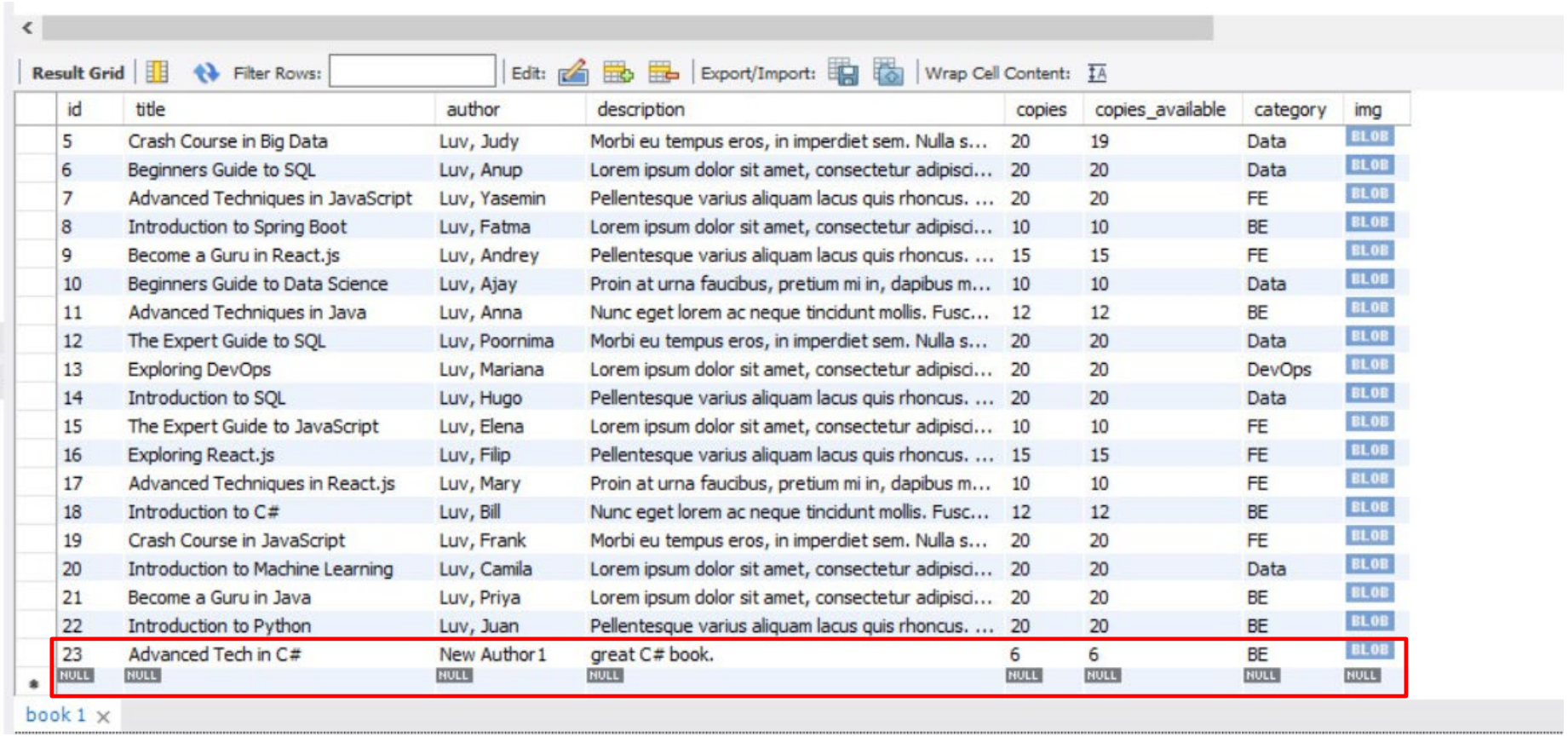
The available copy number and total copies number is as exactly as we set.
If we click on the “checkout” button, and try to leave a review.



We will get this page:



If we go to our database, open the “book” table, we can find the added book too!
Now, we have totally 23 books in our database.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	author	description	copies	copies_available	category	img
5	Crash Course in Big Data	Luv, Judy	Morbi eu tempus eros, in imperdiet sem. Nulla s...	20	19	Data	BLOB
6	Beginners Guide to SQL	Luv, Anup	Lorem ipsum dolor sit amet, consectetur adipisci...	20	20	Data	BLOB
7	Advanced Techniques in JavaScript	Luv, Yasemin	Pellentesque varius aliquam lacus quis rhoncus. ...	20	20	FE	BLOB
8	Introduction to Spring Boot	Luv, Fatma	Lorem ipsum dolor sit amet, consectetur adipisci...	10	10	BE	BLOB
9	Become a Guru in React.js	Luv, Andrey	Pellentesque varius aliquam lacus quis rhoncus. ...	15	15	FE	BLOB
10	Beginners Guide to Data Science	Luv, Ajay	Proin at urna faucibus, pretium mi in, dapibus m...	10	10	Data	BLOB
11	Advanced Techniques in Java	Luv, Anna	Nunc eget lorem ac neque tincidunt mollis. Fusc...	12	12	BE	BLOB
12	The Expert Guide to SQL	Luv, Poornima	Morbi eu tempus eros, in imperdiet sem. Nulla s...	20	20	Data	BLOB
13	Exploring DevOps	Luv, Mariana	Lorem ipsum dolor sit amet, consectetur adipisci...	20	20	DevOps	BLOB
14	Introduction to SQL	Luv, Hugo	Pellentesque varius aliquam lacus quis rhoncus. ...	20	20	Data	BLOB
15	The Expert Guide to JavaScript	Luv, Elena	Lorem ipsum dolor sit amet, consectetur adipisci...	10	10	FE	BLOB
16	Exploring React.js	Luv, Filip	Pellentesque varius aliquam lacus quis rhoncus. ...	15	15	FE	BLOB
17	Advanced Techniques in React.js	Luv, Mary	Proin at urna faucibus, pretium mi in, dapibus m...	10	10	FE	BLOB
18	Introduction to C#	Luv, Bill	Nunc eget lorem ac neque tincidunt mollis. Fusc...	12	12	BE	BLOB
19	Crash Course in JavaScript	Luv, Frank	Morbi eu tempus eros, in imperdiet sem. Nulla s...	20	20	FE	BLOB
20	Introduction to Machine Learning	Luv, Camila	Lorem ipsum dolor sit amet, consectetur adipisci...	20	20	Data	BLOB
21	Become a Guru in Java	Luv, Priya	Lorem ipsum dolor sit amet, consectetur adipisci...	20	20	BE	BLOB
22	Introduction to Python	Luv, Juan	Pellentesque varius aliquam lacus quis rhoncus. ...	20	20	BE	BLOB
23	Advanced Tech in C#	New Author1	great C# book.	6	6	BE	BLOB
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

book 1