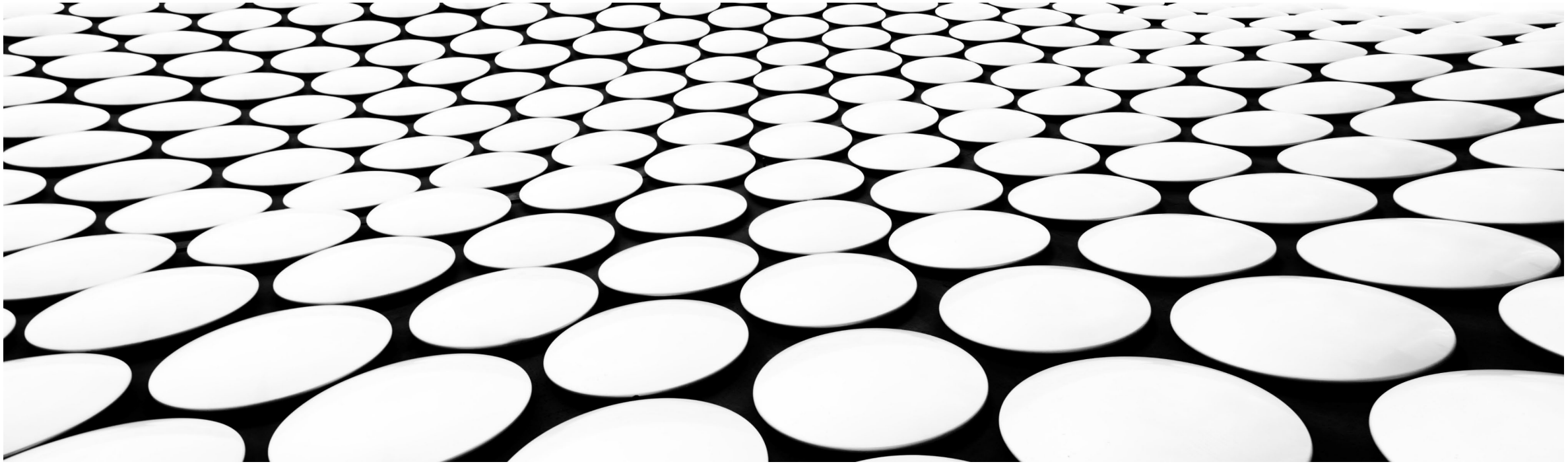
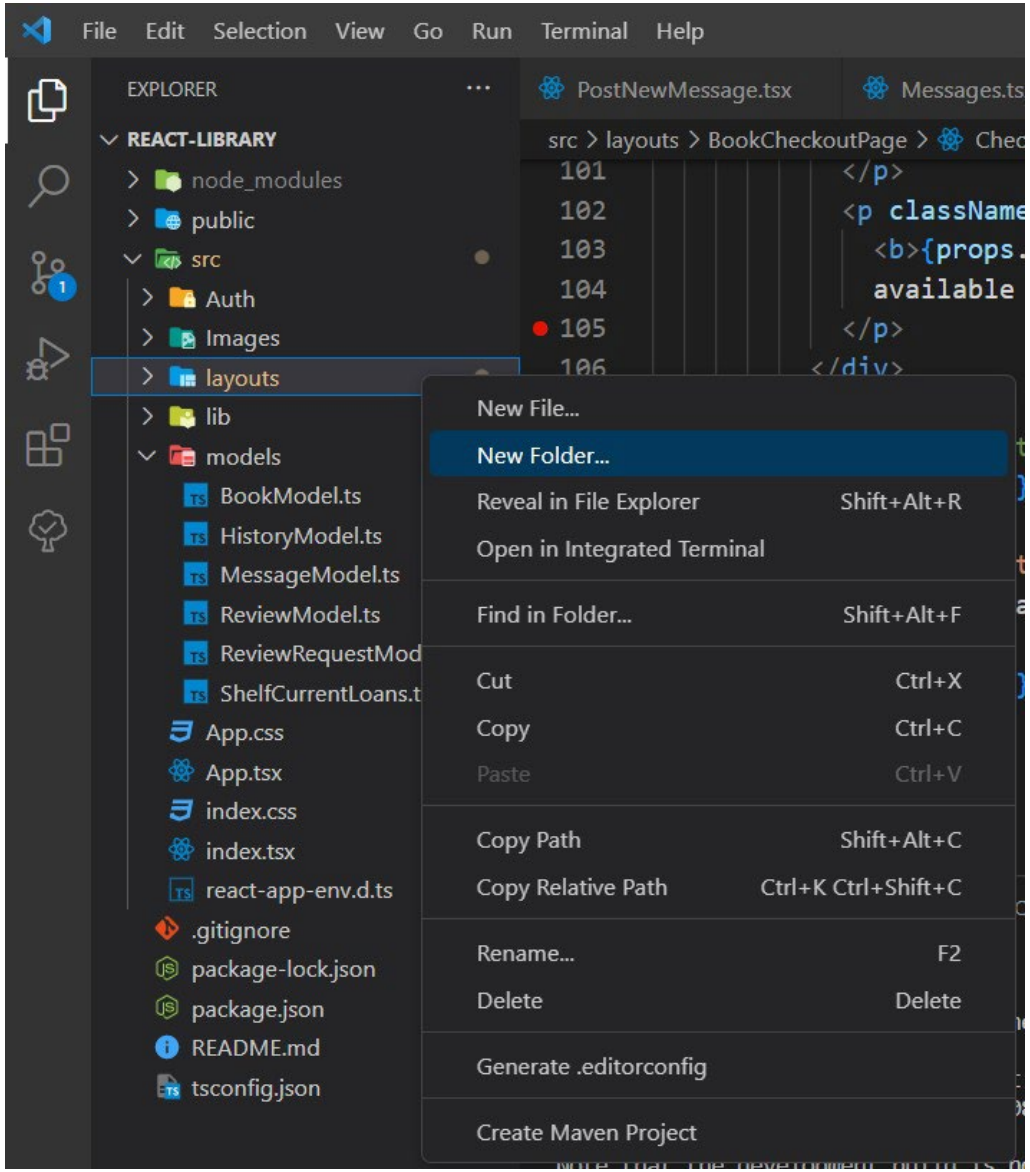
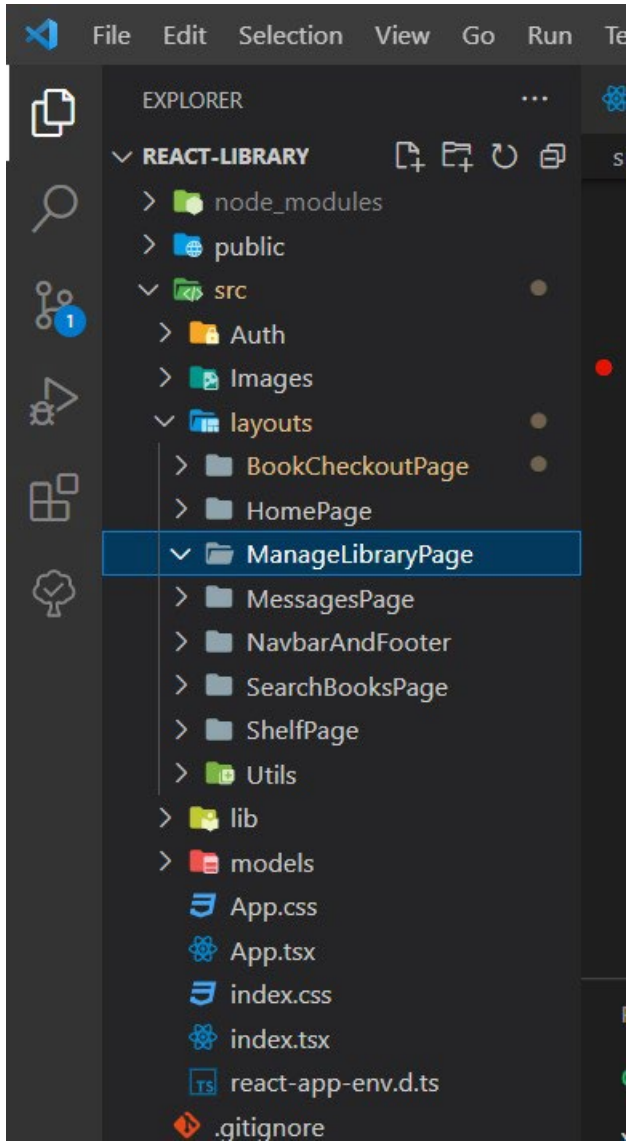
S19 ADMIN SERVICES



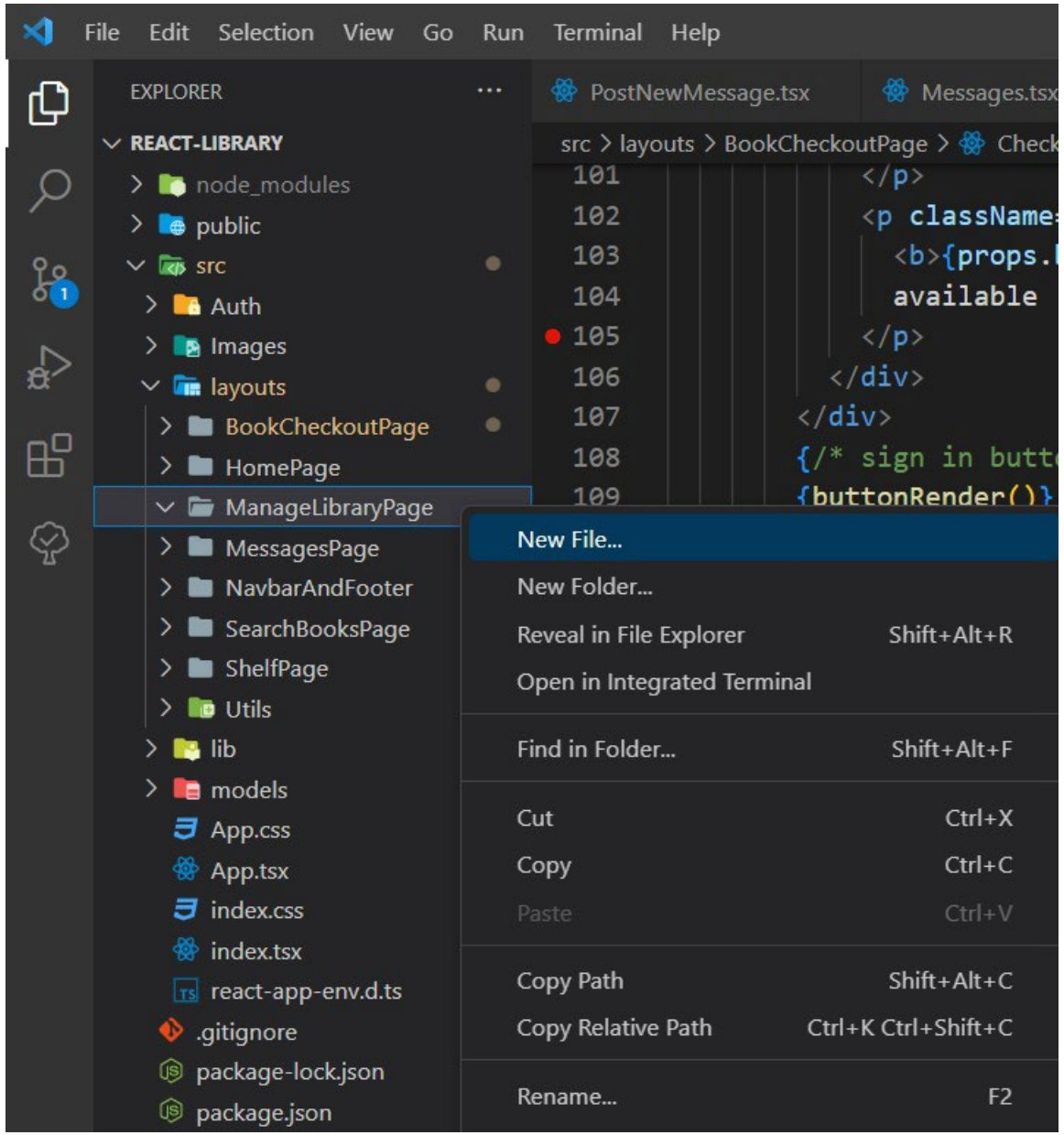
1.REACT MANAGELIBRARYPAGE



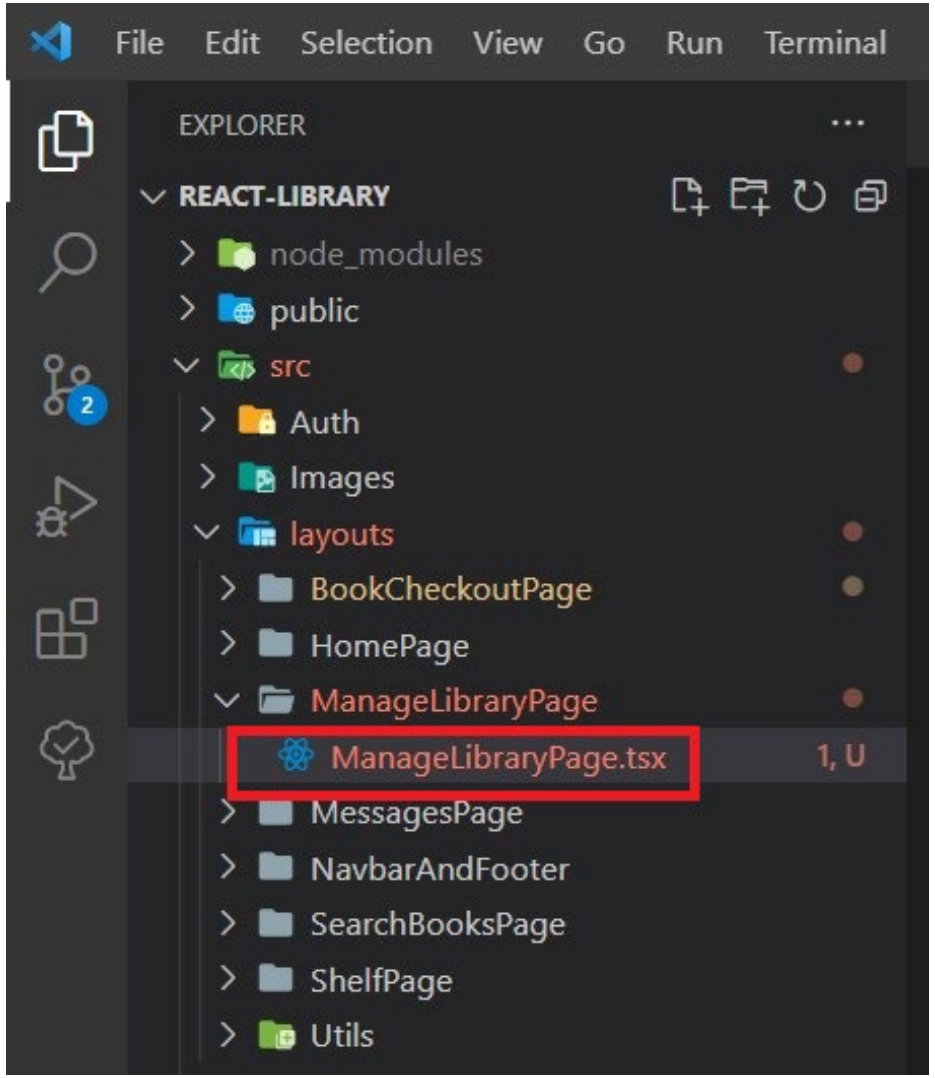
Step1, Jump into our layouts, create a new directory. Right click on the “layouts”, choose “new folder”, and we're going to call this ManageLibraryPage.



ManageLibraryPage folder created.

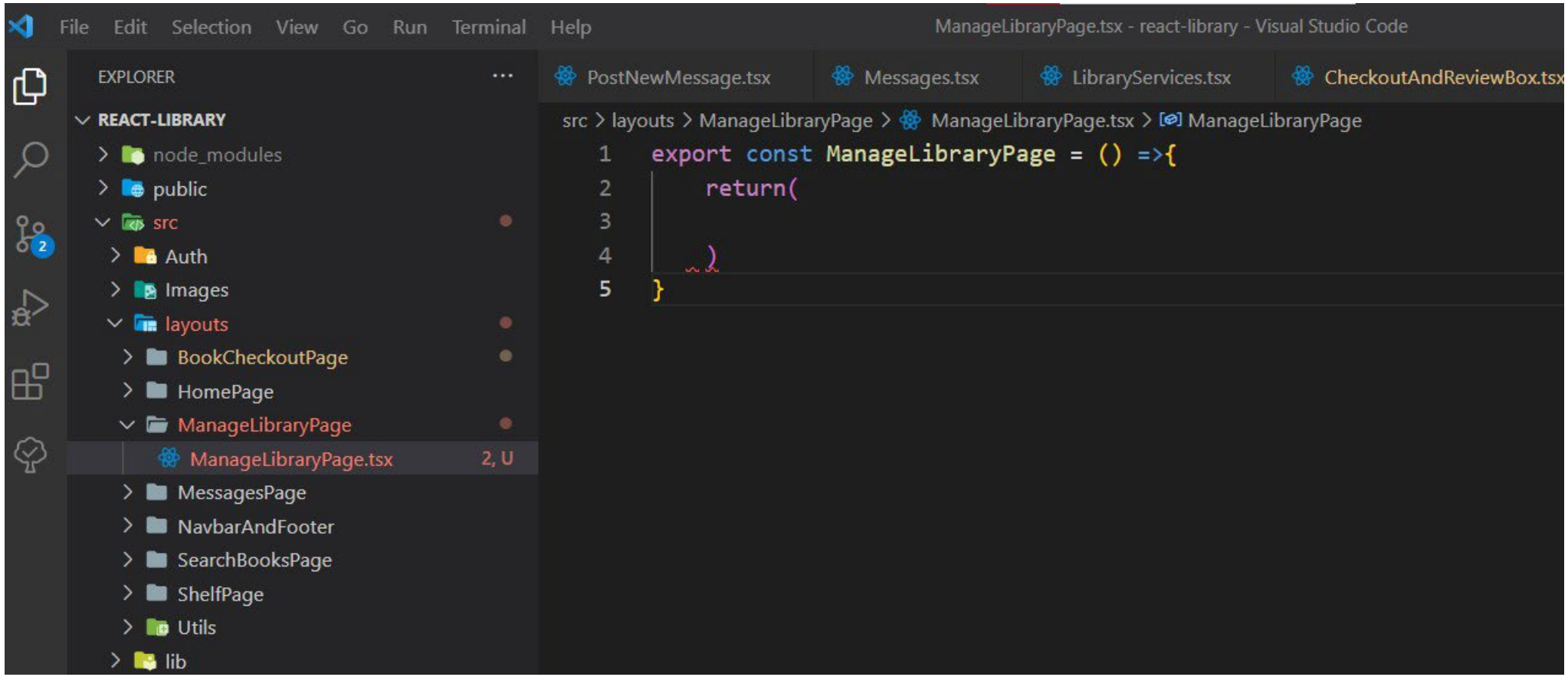


Step2, Right click this ManageLibraryPage, Choose “new file” , create a file named ManageLibraryPage.tsx.

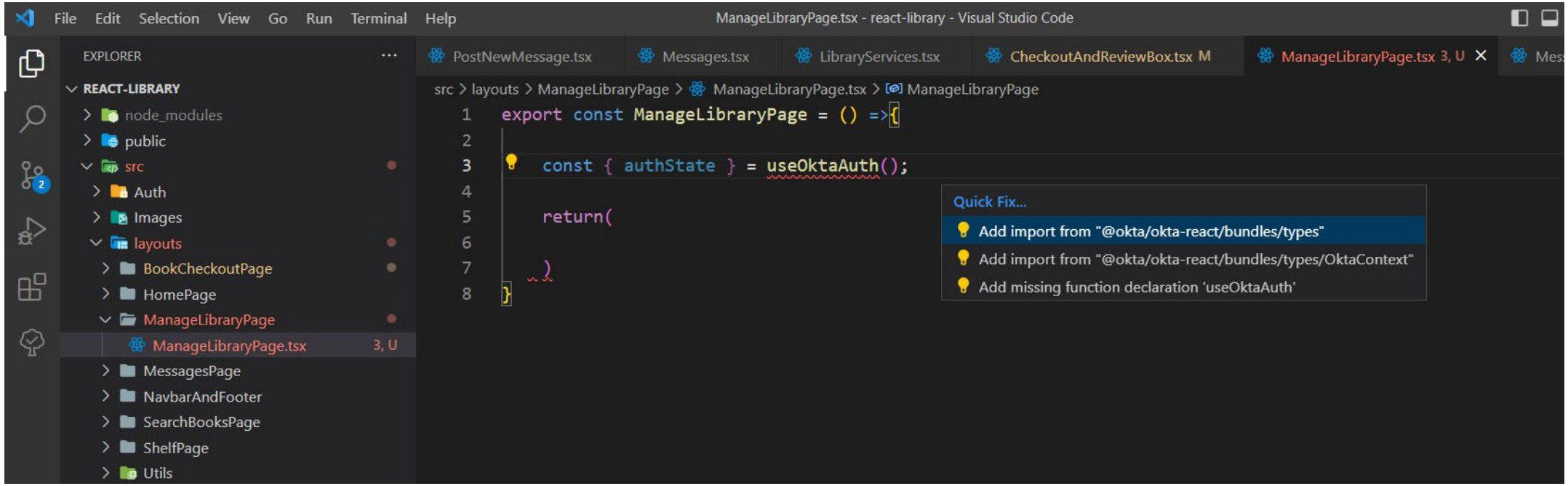


ManageLibraryPage.tsx. created.

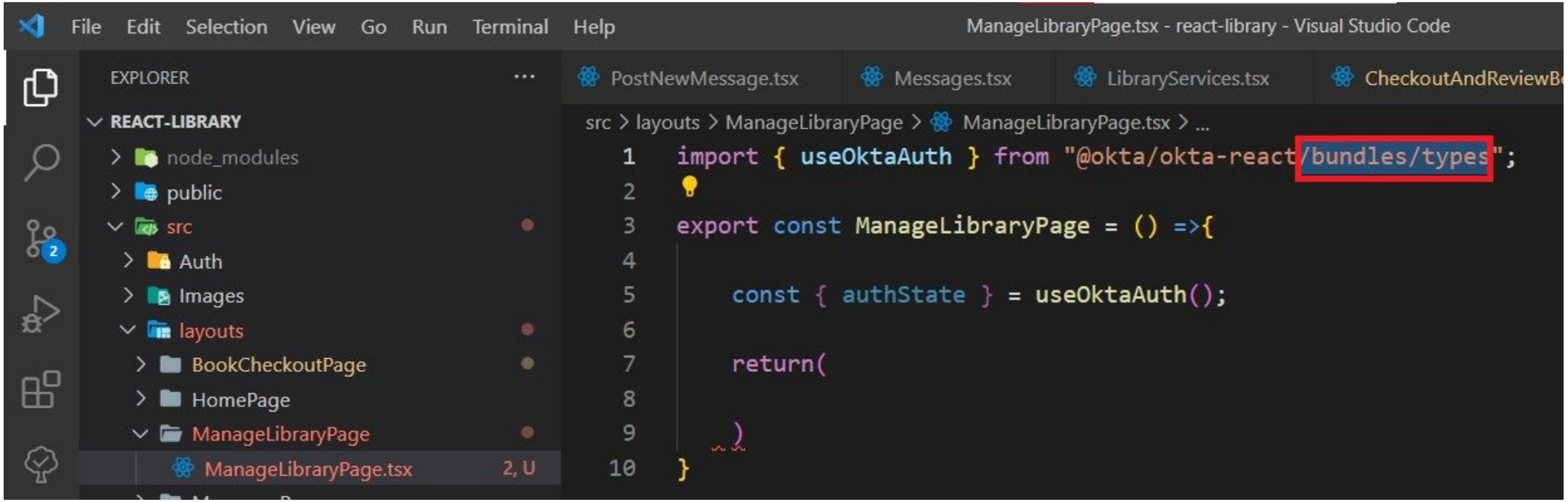
Step 3, now the very first thing we want to do, as we ‘ve done with every single new component, is to create export constant function structure.



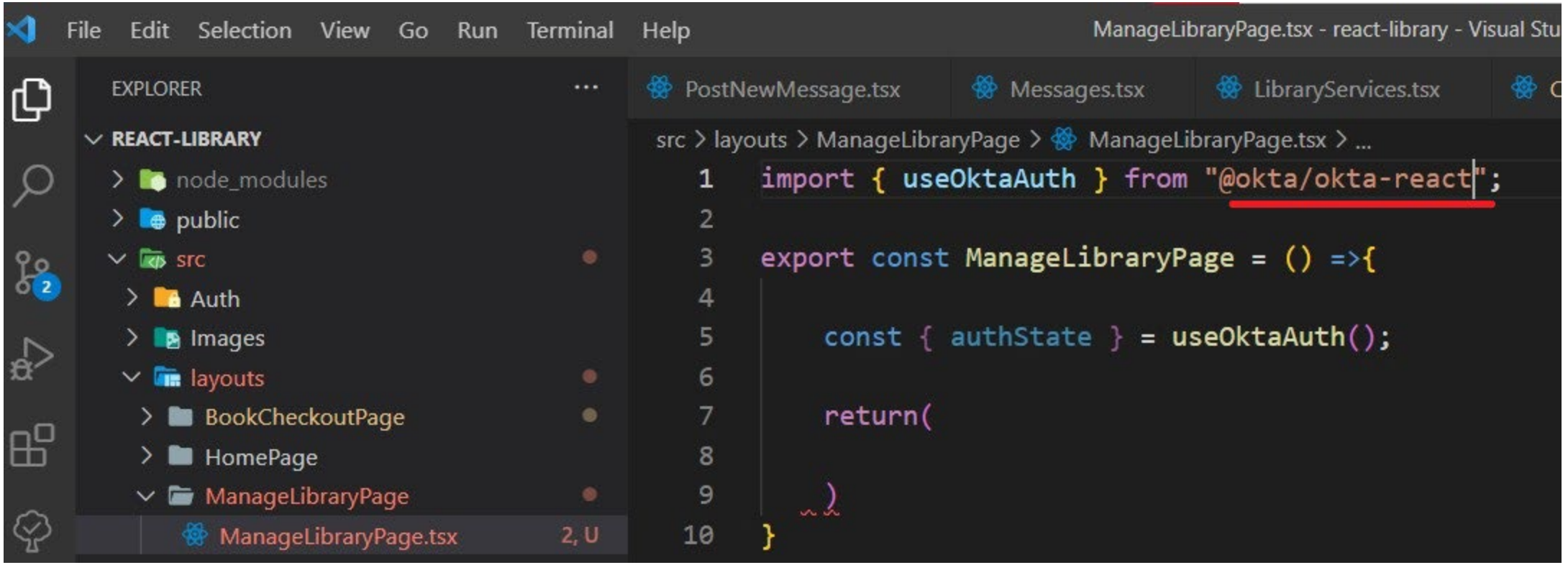
Step 4 , add authState and import {useOkataAuth}.



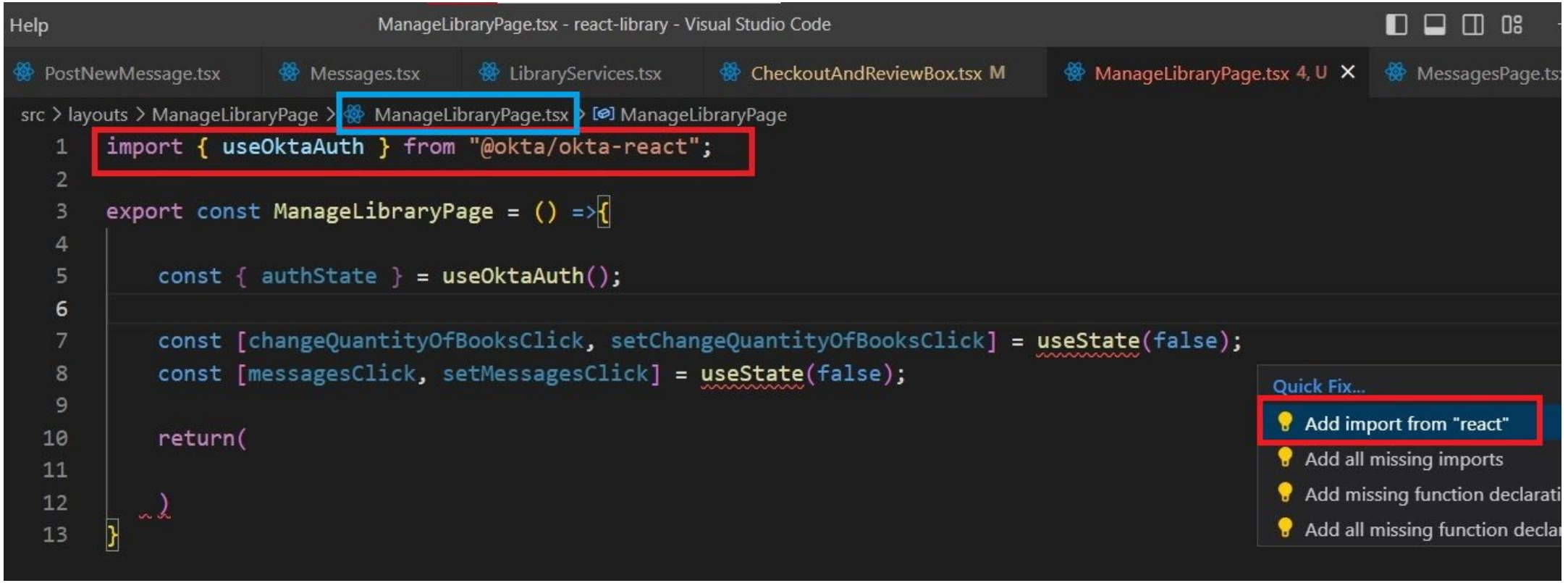
Remove the “/bundle/types” in the import directory for the useOktaAuth. Or else, we will get an compile error when our app starts.



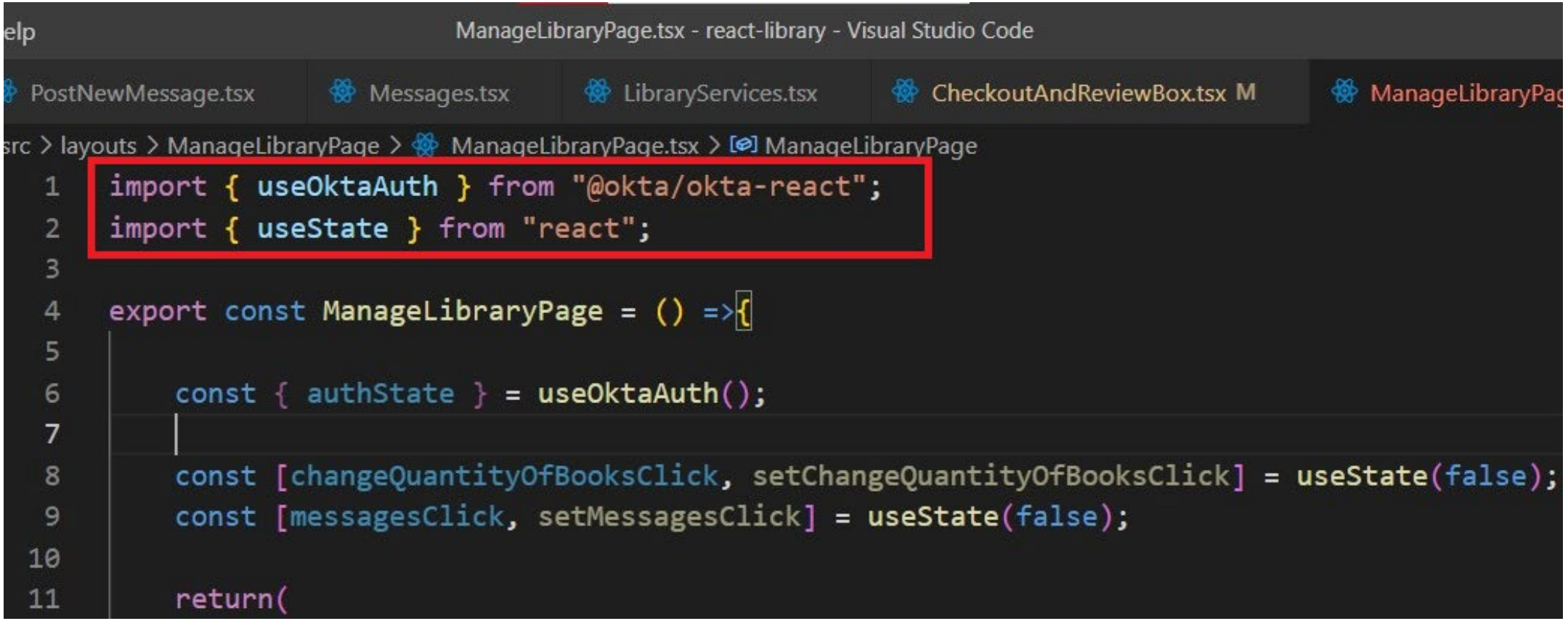
useOktaAuth import directory should be like this:



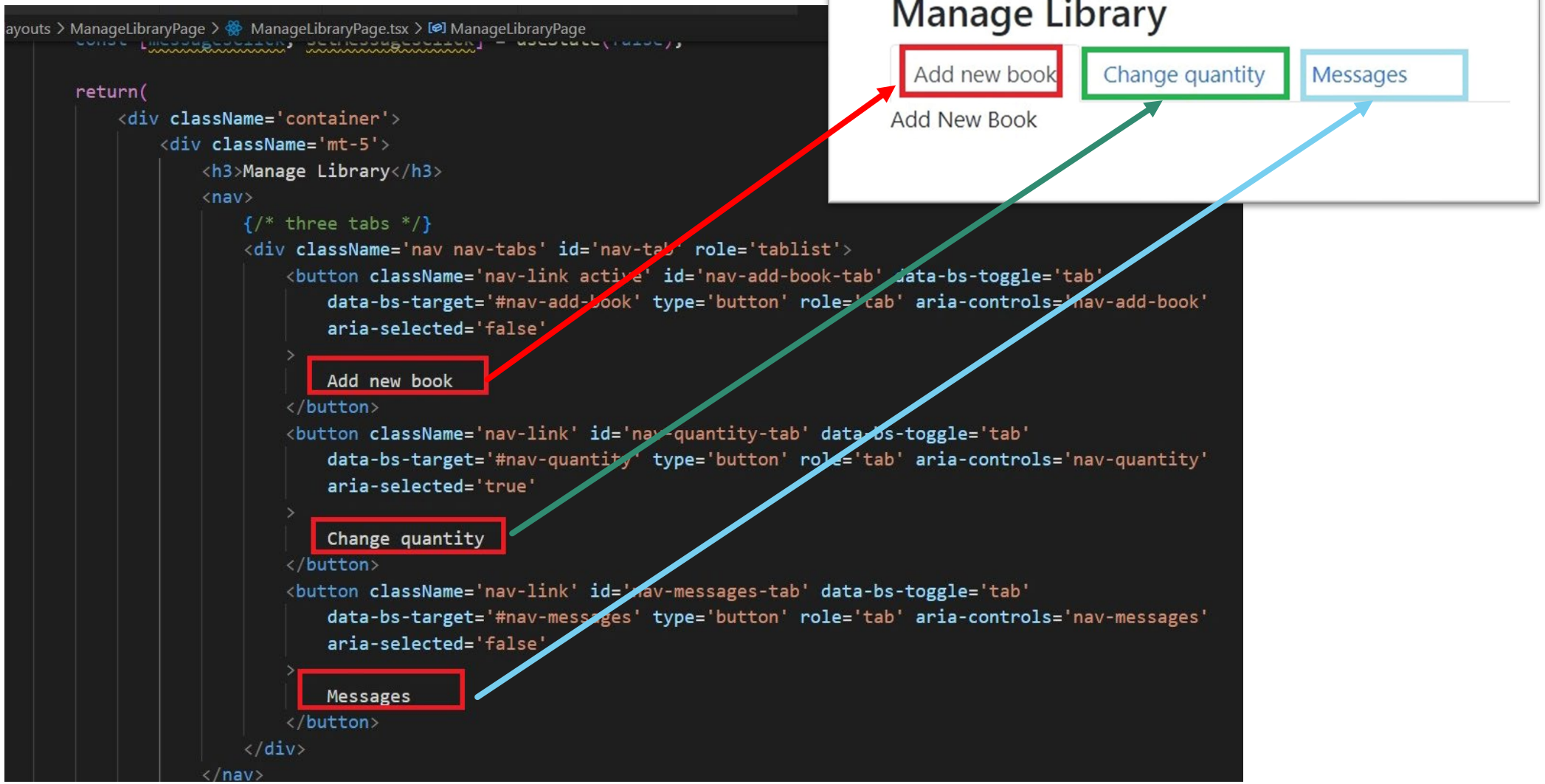
Step 5 , add another two useState and import the {useState}.

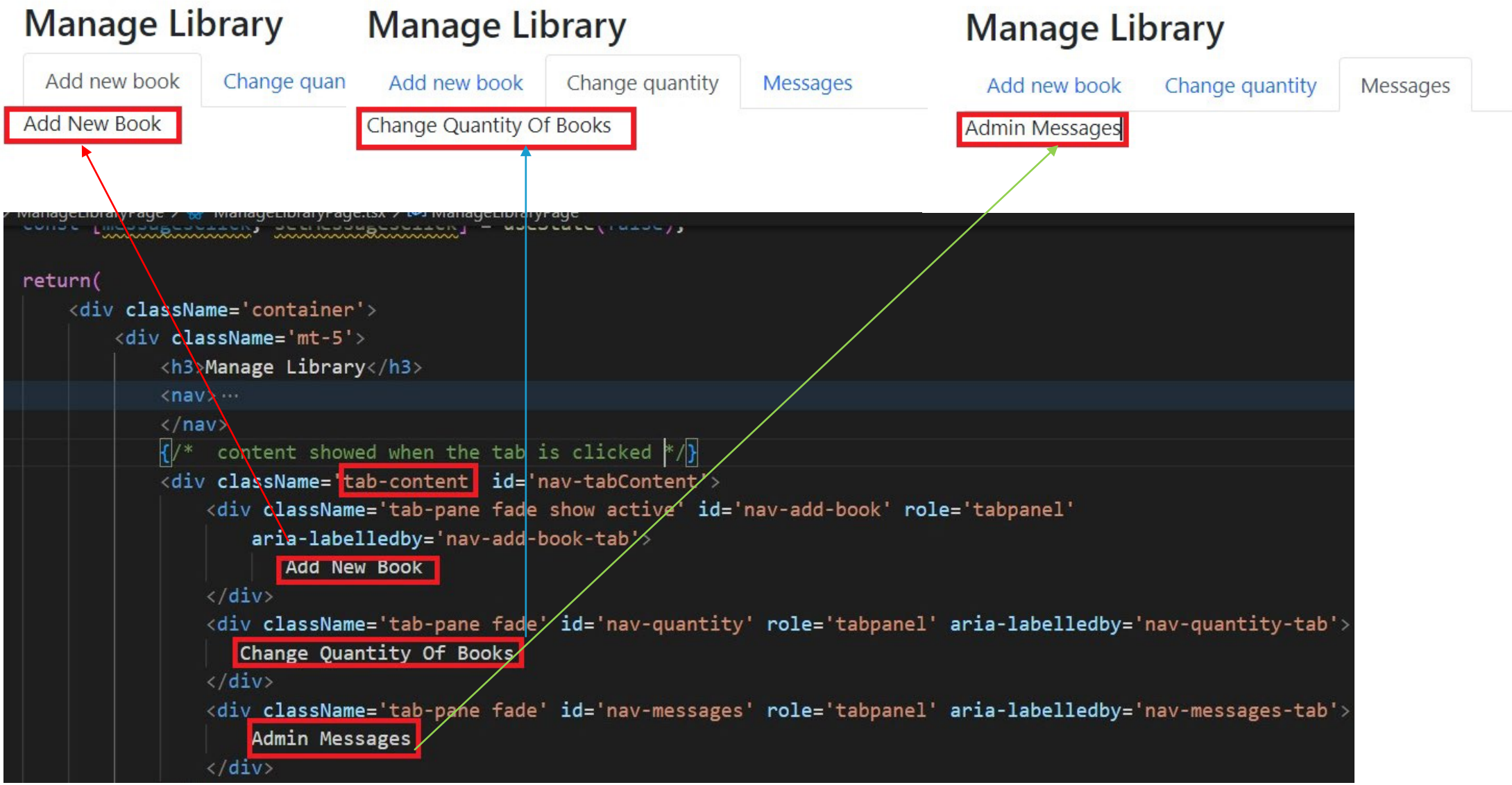


Until now, the import directory in the top should be as below:



Step 6, fill in Return part with html and CSS code.




Manage Library
Add new book
Change quan
Add New Book
Manage Library
Add new book
Change quantity
Messages
Change Quantity Of Books
Manage Library
Add new book
Change quantity
Messages
Admin Messages
return(

<h3>Manage Library</h3>
<nav>
</nav>
{/* content showed when the tab is clicked */}

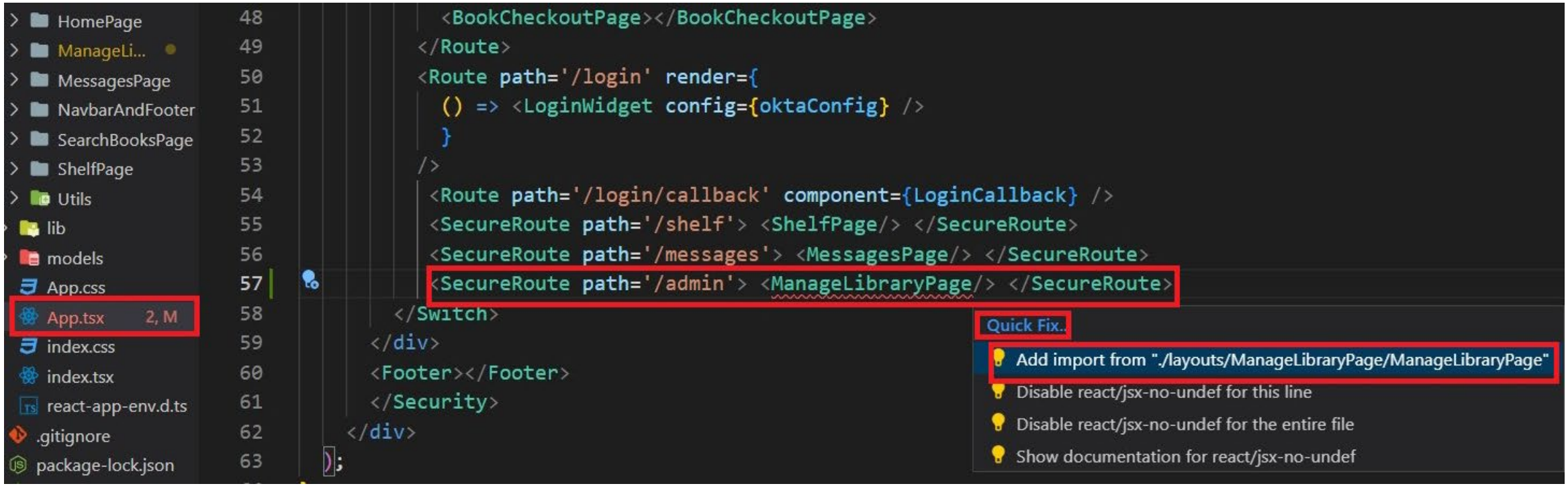
<div className='tab-pane fade show active' id='nav-add-book' role='tabpanel'
aria-labelledby='nav-add-book-tab'>
Add New Book

Change Quantity Of Books

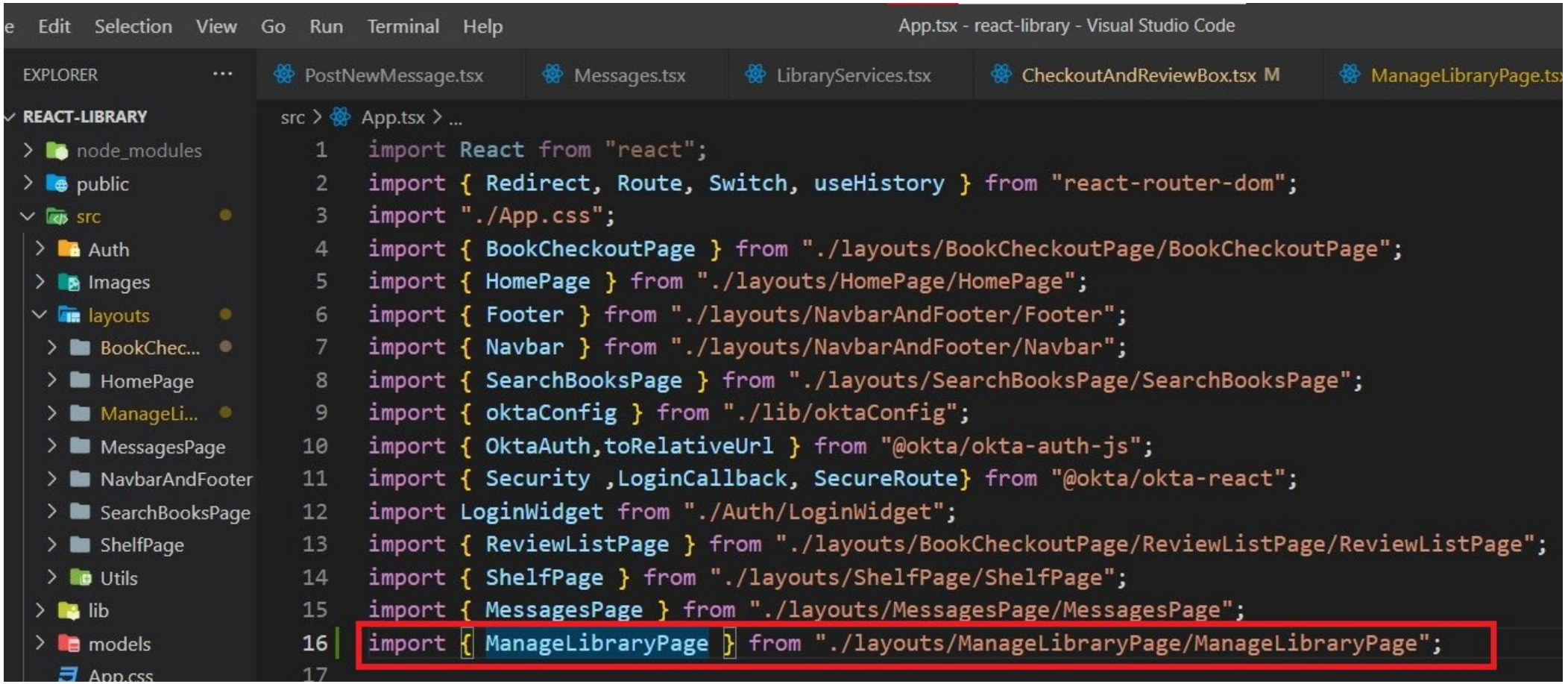
Admin Messages

2. REACT SECURE ROUTES FOR MANAGELIBRARY

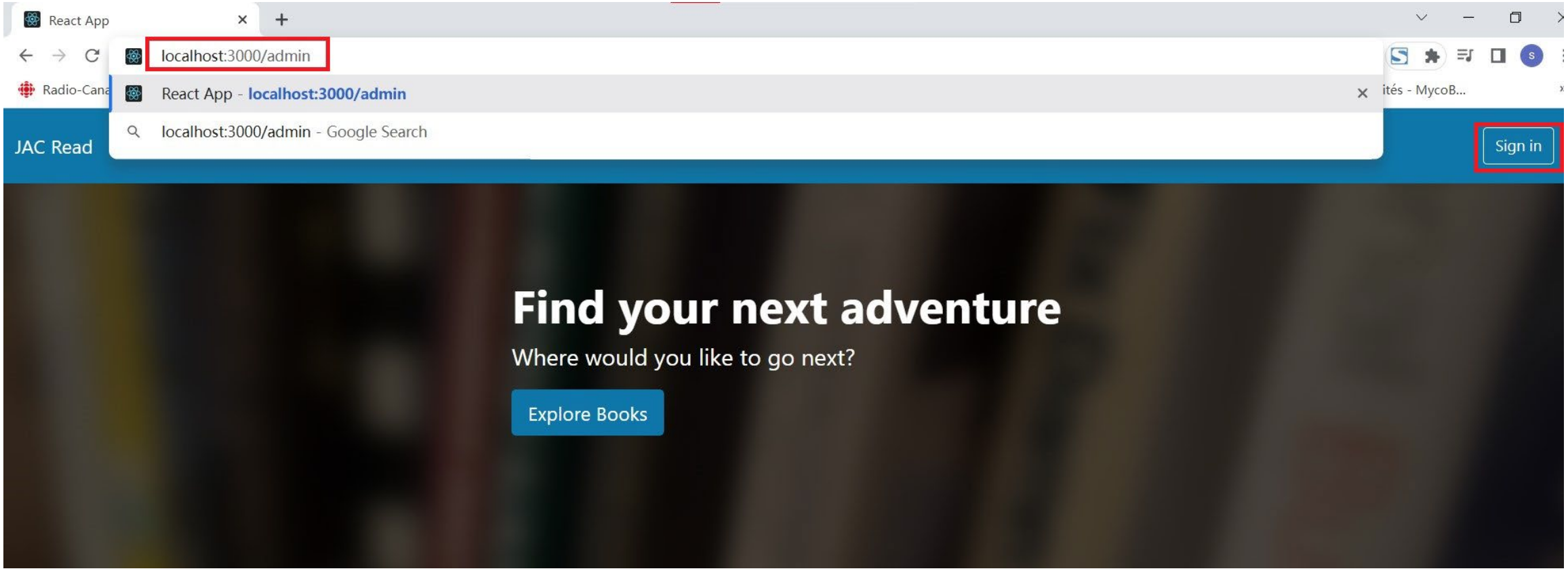
let's hop into our App.tsx.
And let's create a new secure route for our new ManageLlibraryPage.
Right click red tilde, click on “Quick Fix”.
Choose “add import”.



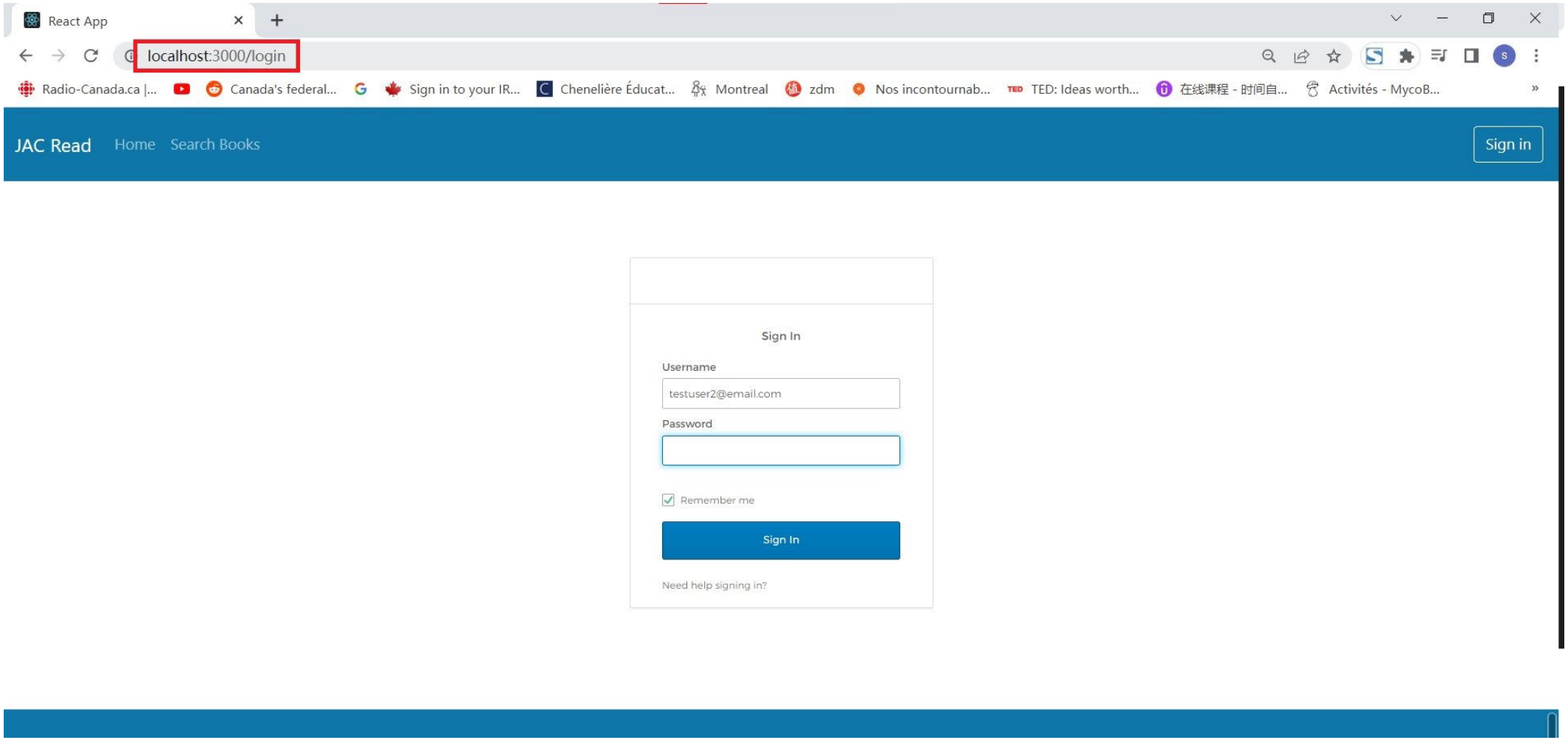
ManageLlibraryPage is imported.



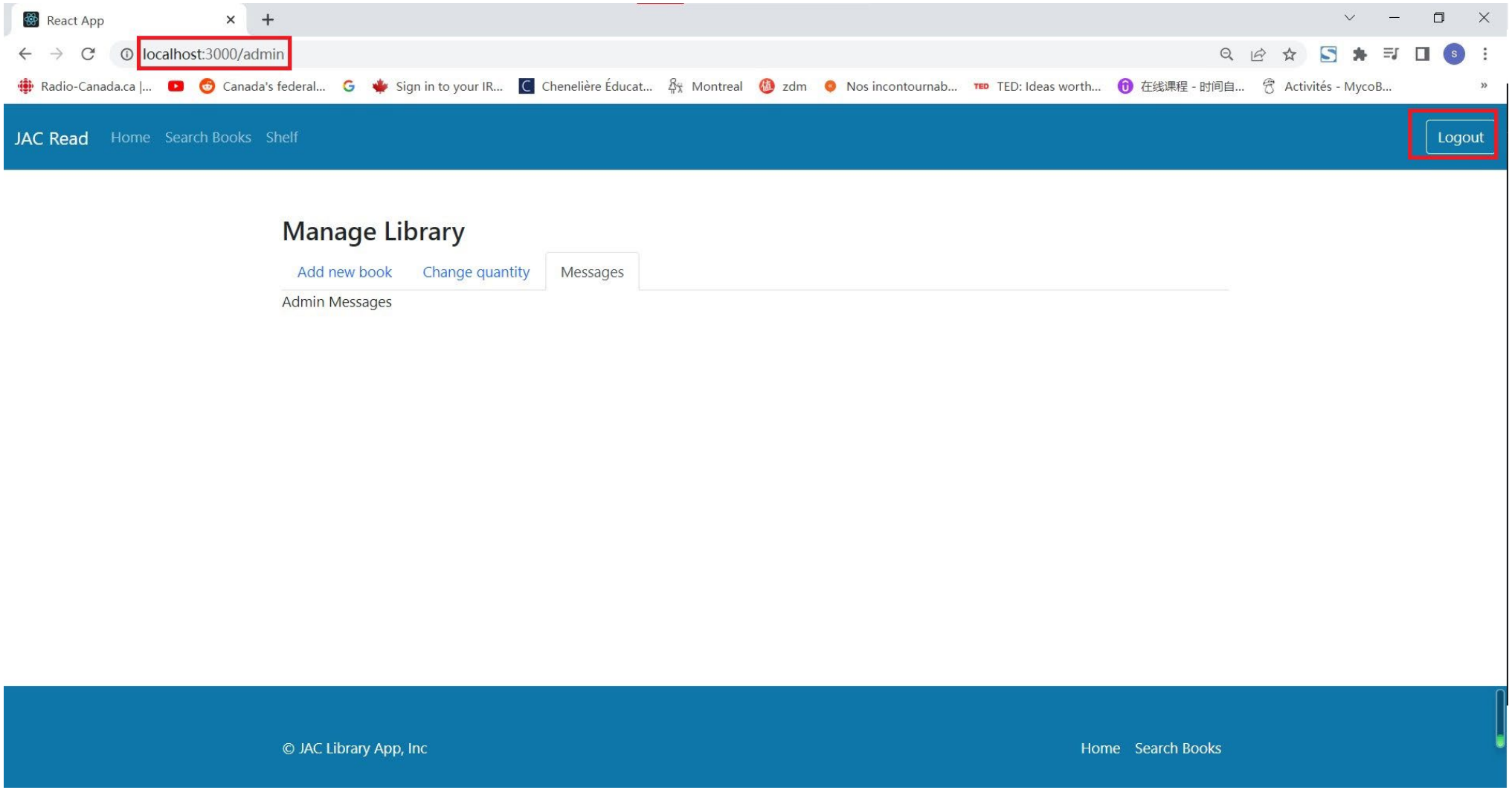
With the log out status, Visit http://localhost:3000/admin :



It will redirect us to the sign in page, beacuse we add a secure route to this page.

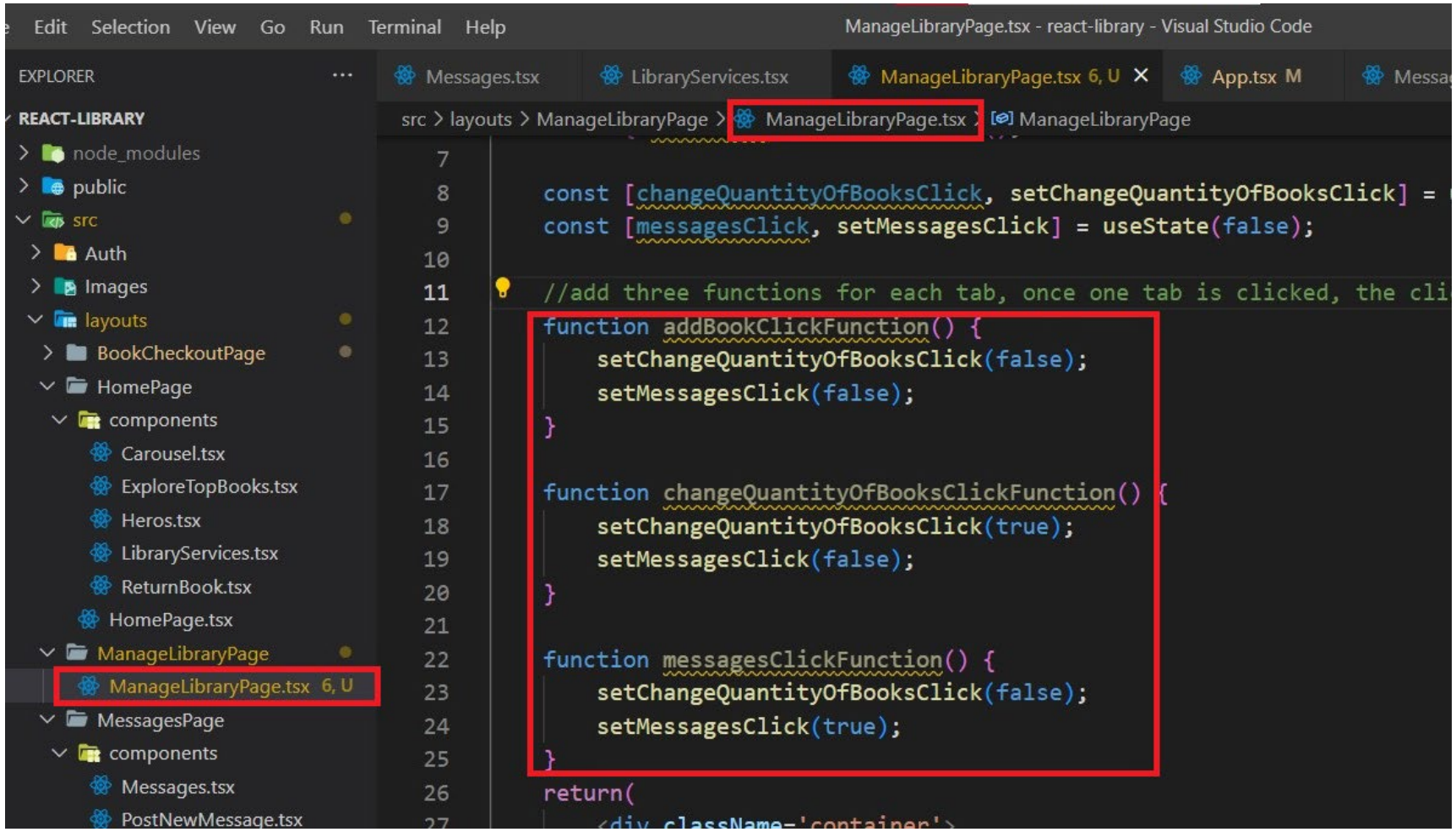


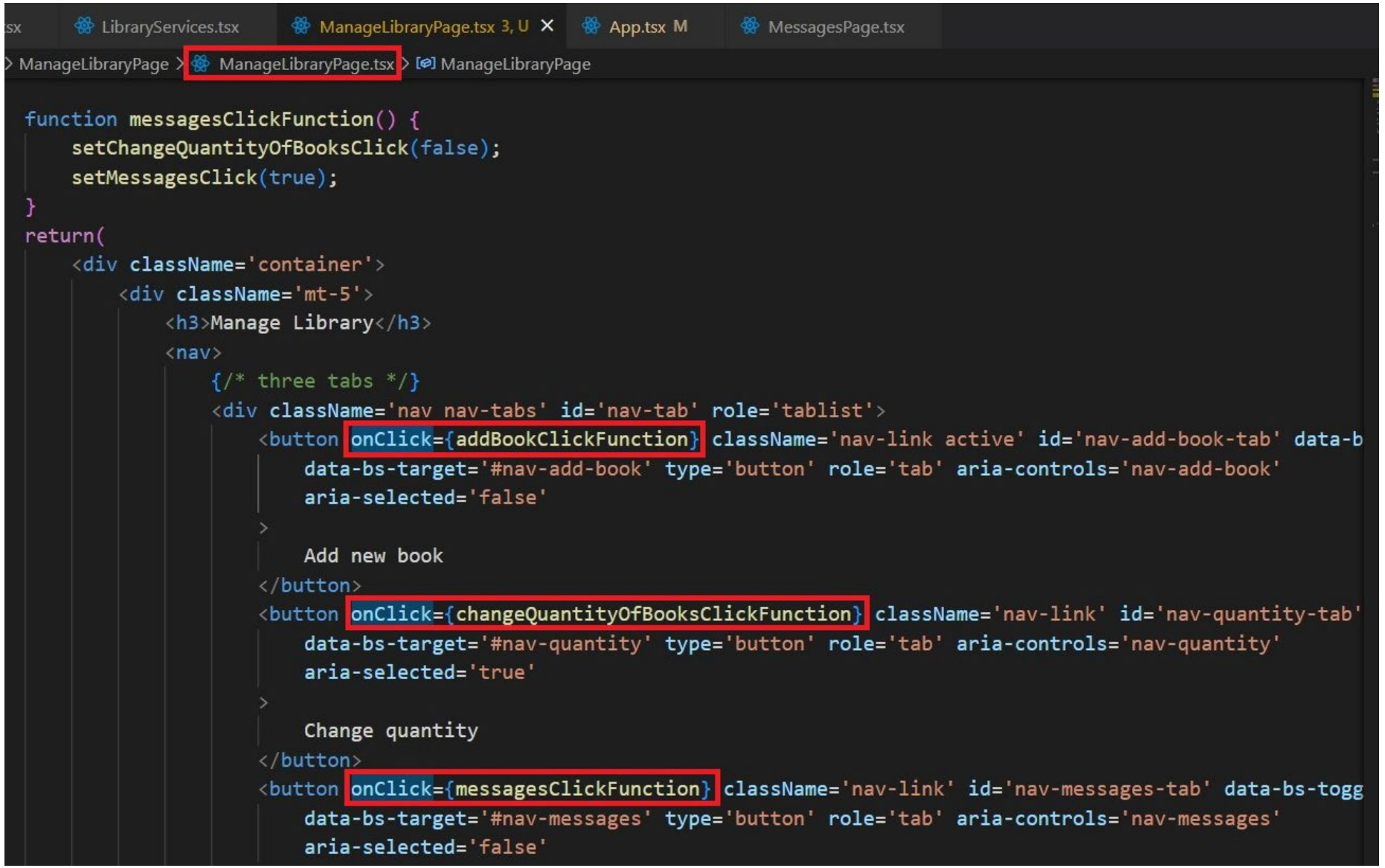
once logged in, it will show the ManageLlibraryPage.



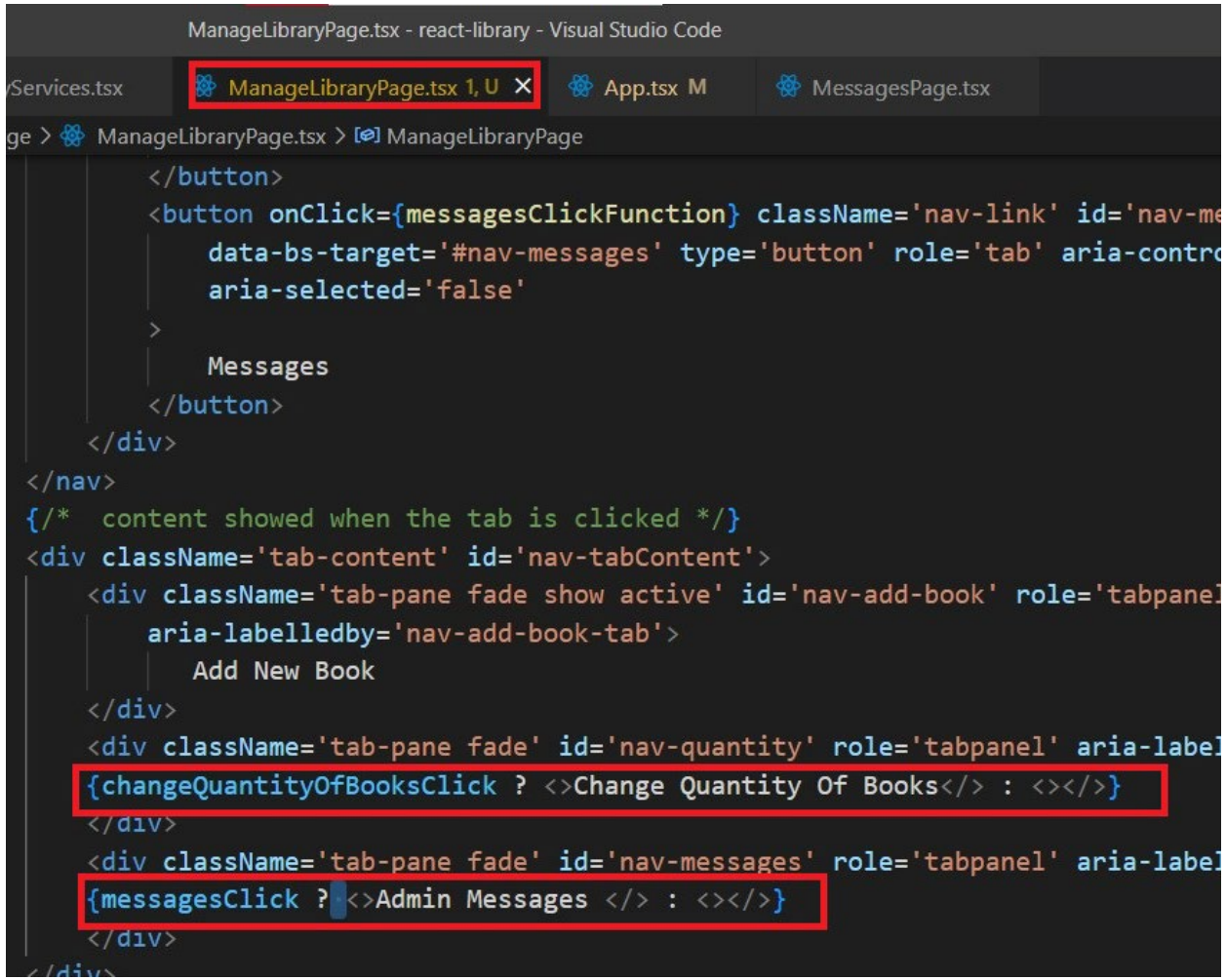
3. REACT STATE FOR MANAGELIBRARY

Step1, add three functions for each tab, once one tab is clicked, the click state of the other two tabs should be false.



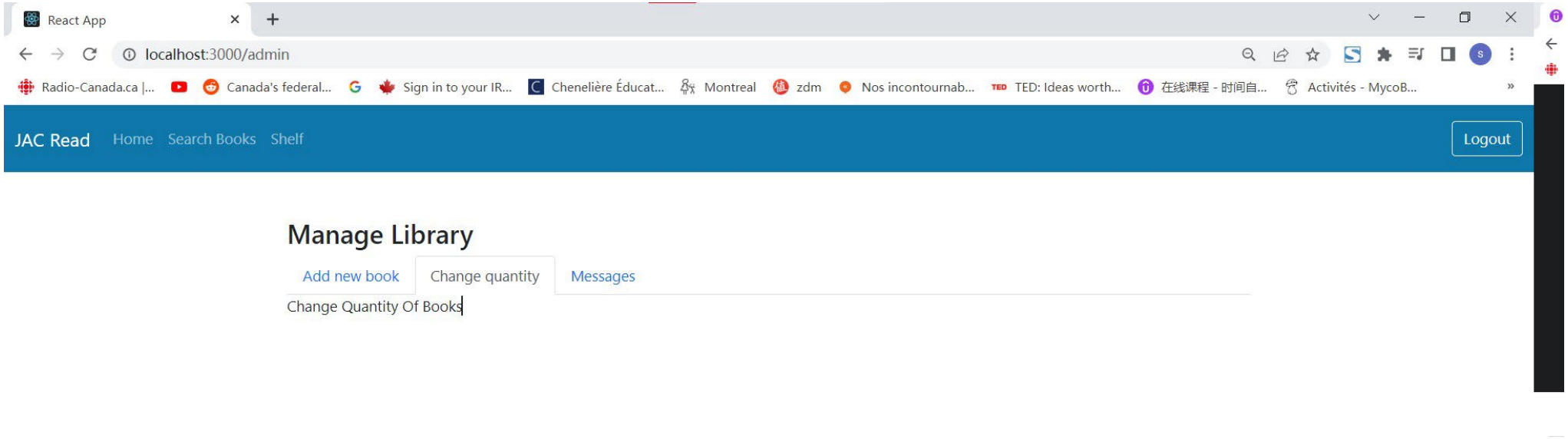


Step 2, add onClick listener to each tab.



Step 3, add click state judge to the tab content.

Open up our React app, refresh the page, and we can see that the tab still swaps between the two.
The only difference is we're calling different pieces of state now, and state is changing, which means when we start adding elements to these tabs, it'll call the useEffect each time.



4. REACT REDIRECT IMPLEMENTATION FOR NON-AMIN

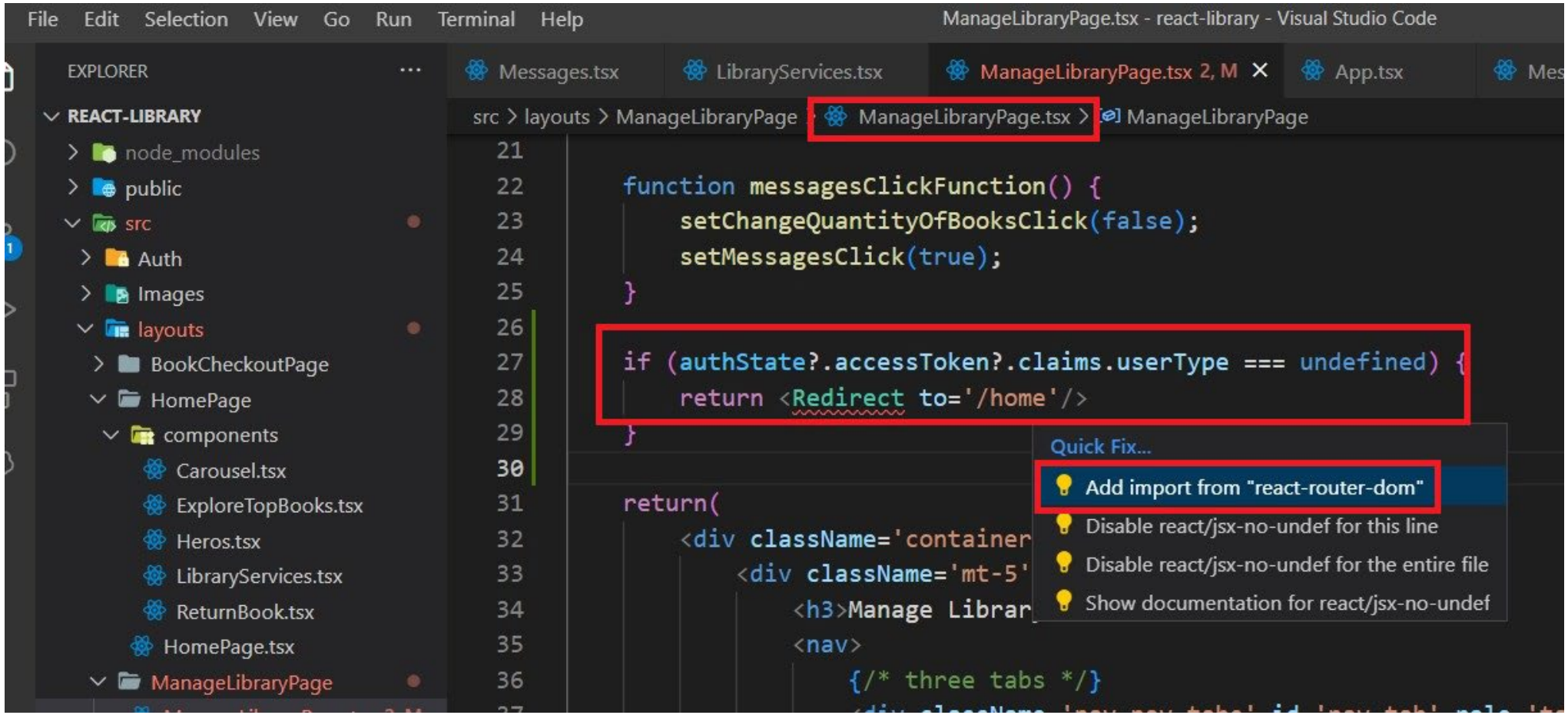
Role-based authentication

- We don't want anybody who's currently authenticated to be able to go to this Manage library page.
- And we currently have that set in our secure route where, hey, a user needs to be authenticated before they come to our component.
- But besides this authentication, we want to make sure that we're checking the user type or the role . That’s to say only the role of admin can visit this page.

Solution:

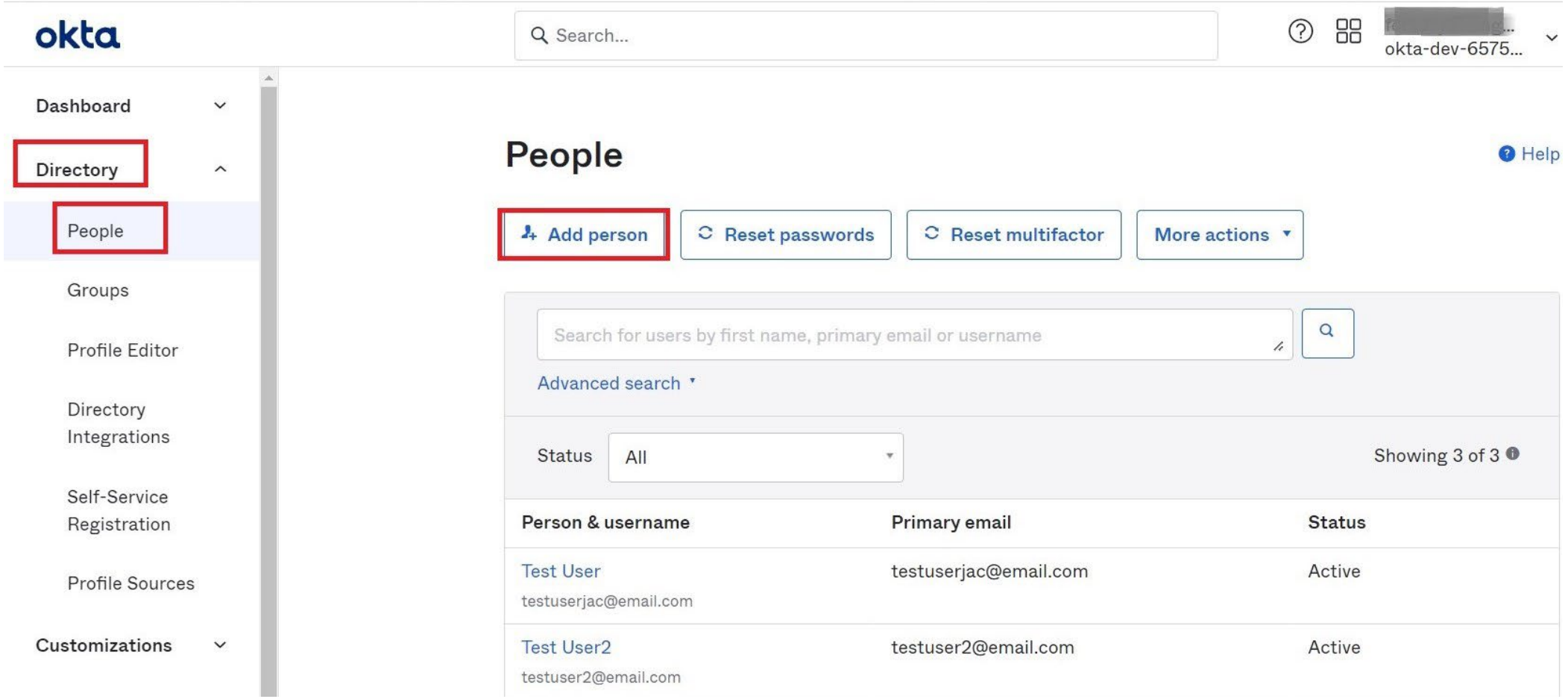
- add a new conditional statement on our manage library page that would check the user type of the user to see if they're an admin or not.
- Use okta to define the user type

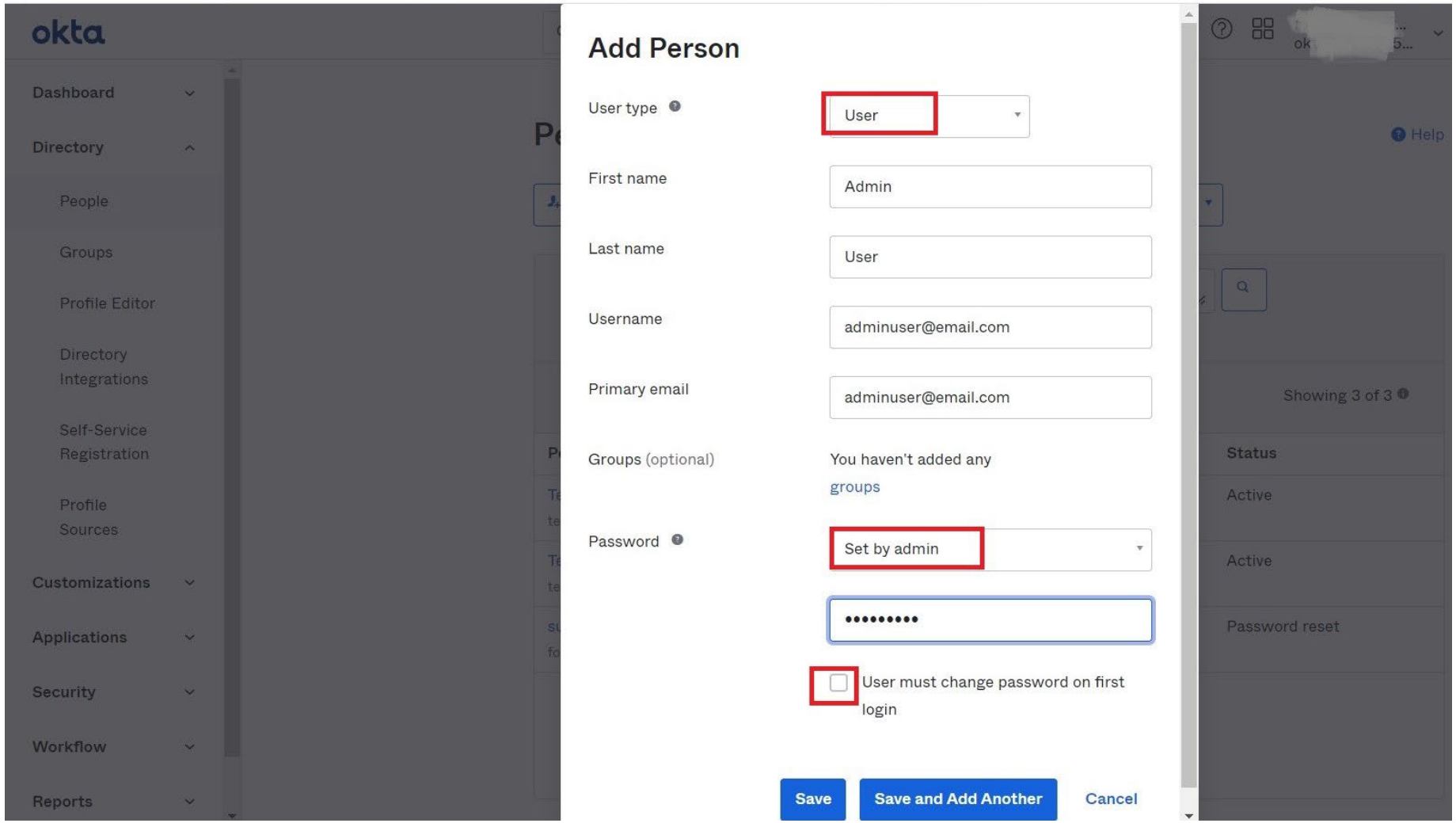
Step1, Go to ManageLlibraryPage.tsx, add a piece of code to judge if the user type is undefined.
if so, redirect to home page.
Click on the “quick fix” to import “react-router-dom” to fix the error when we input “Redirect”.



5. REACT CREATE ADMIN USER

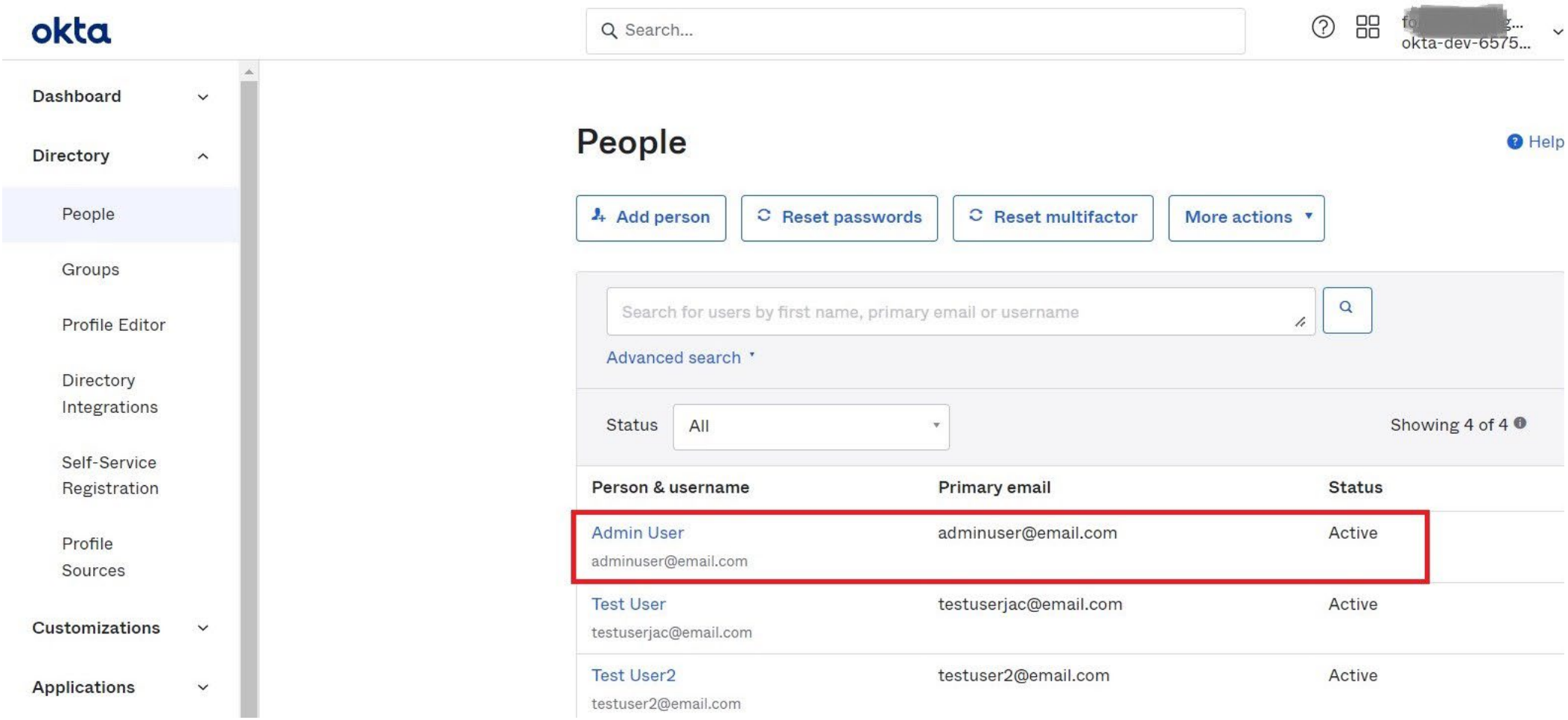
Step1, log into our okta account, choose “Directory”, “ People”, then click on “ add person”.



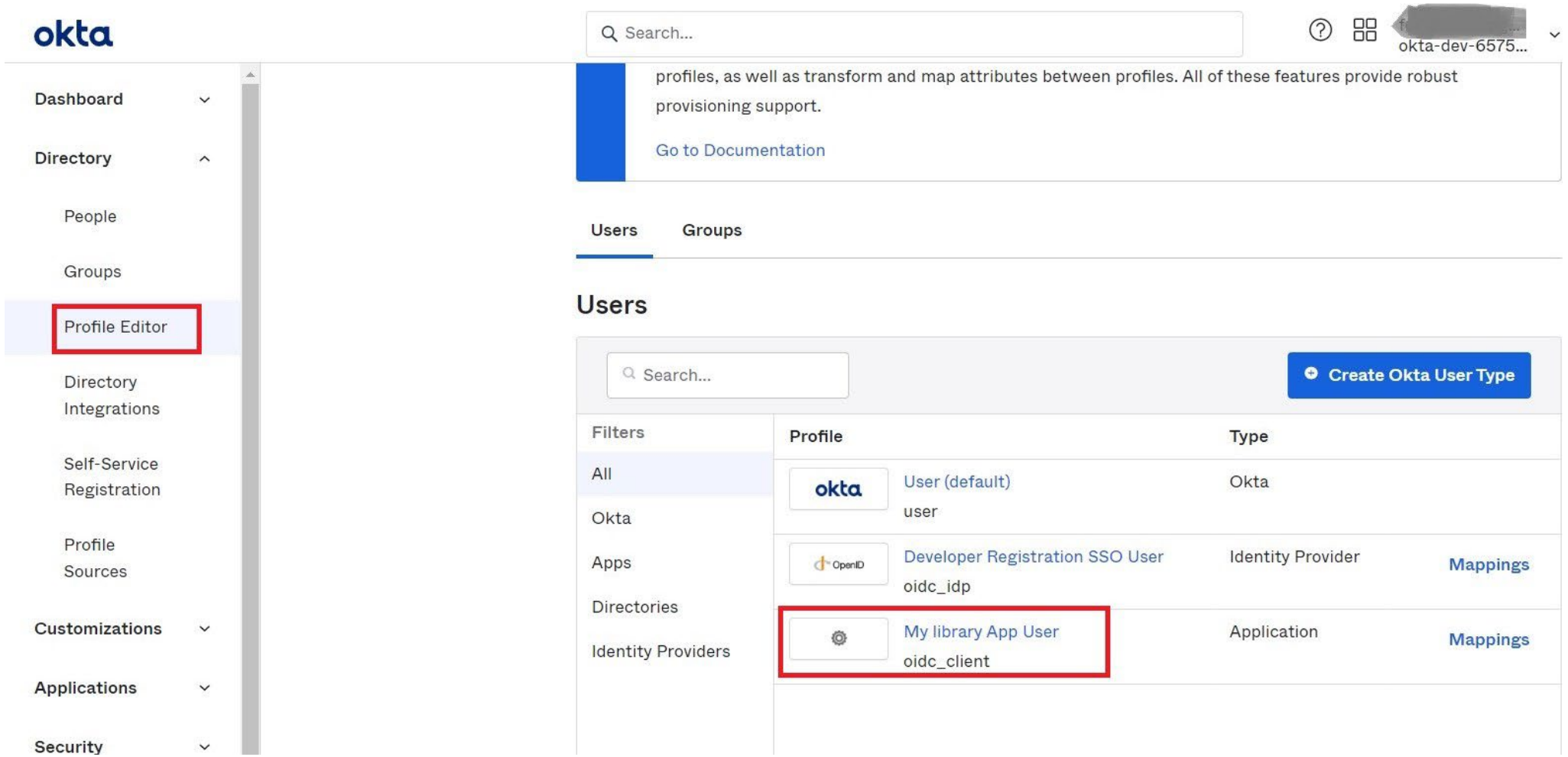


Make sure you input the same as the picture. Then click on “save” button.

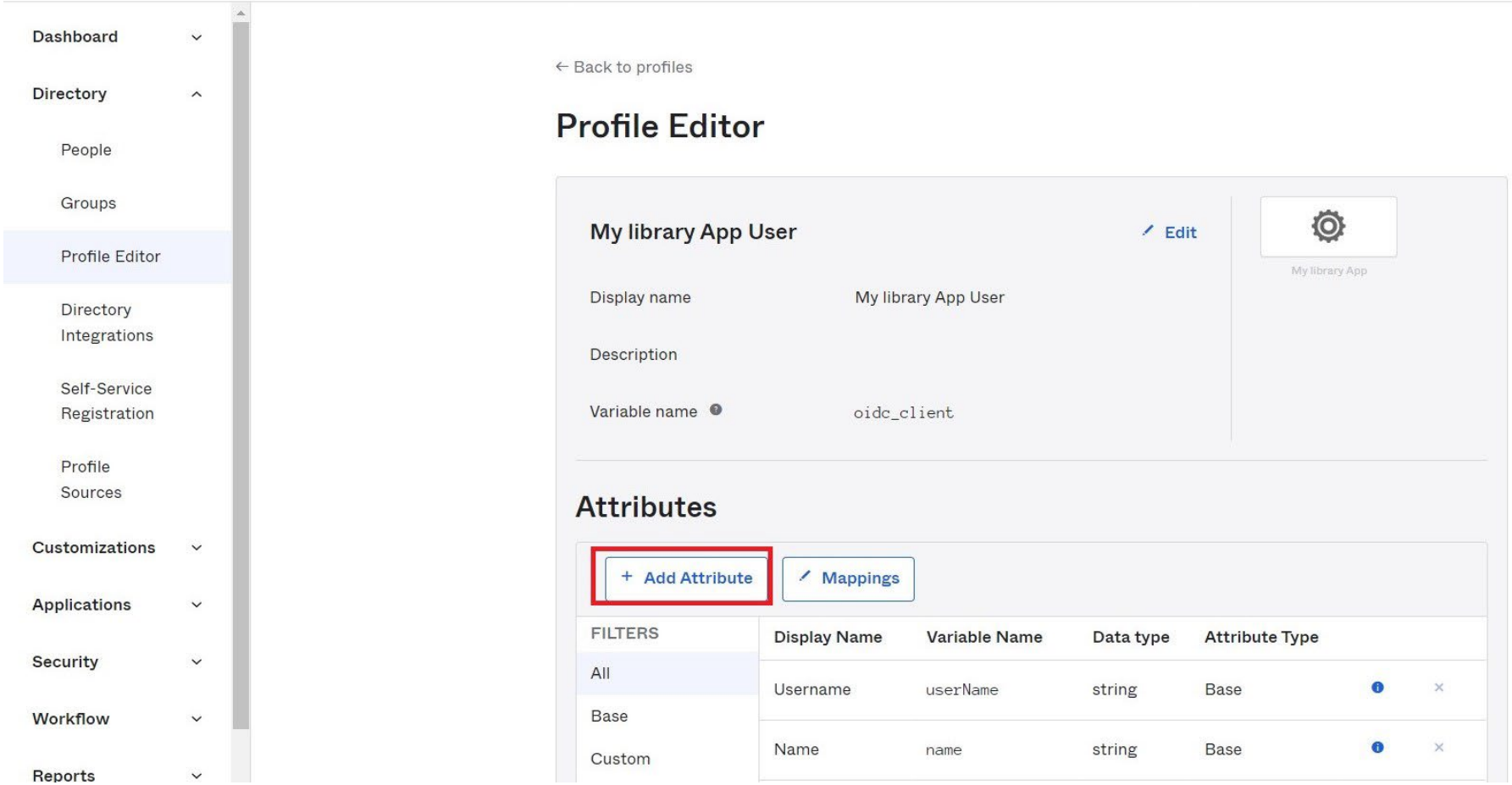
Our admin user will appear in the table.

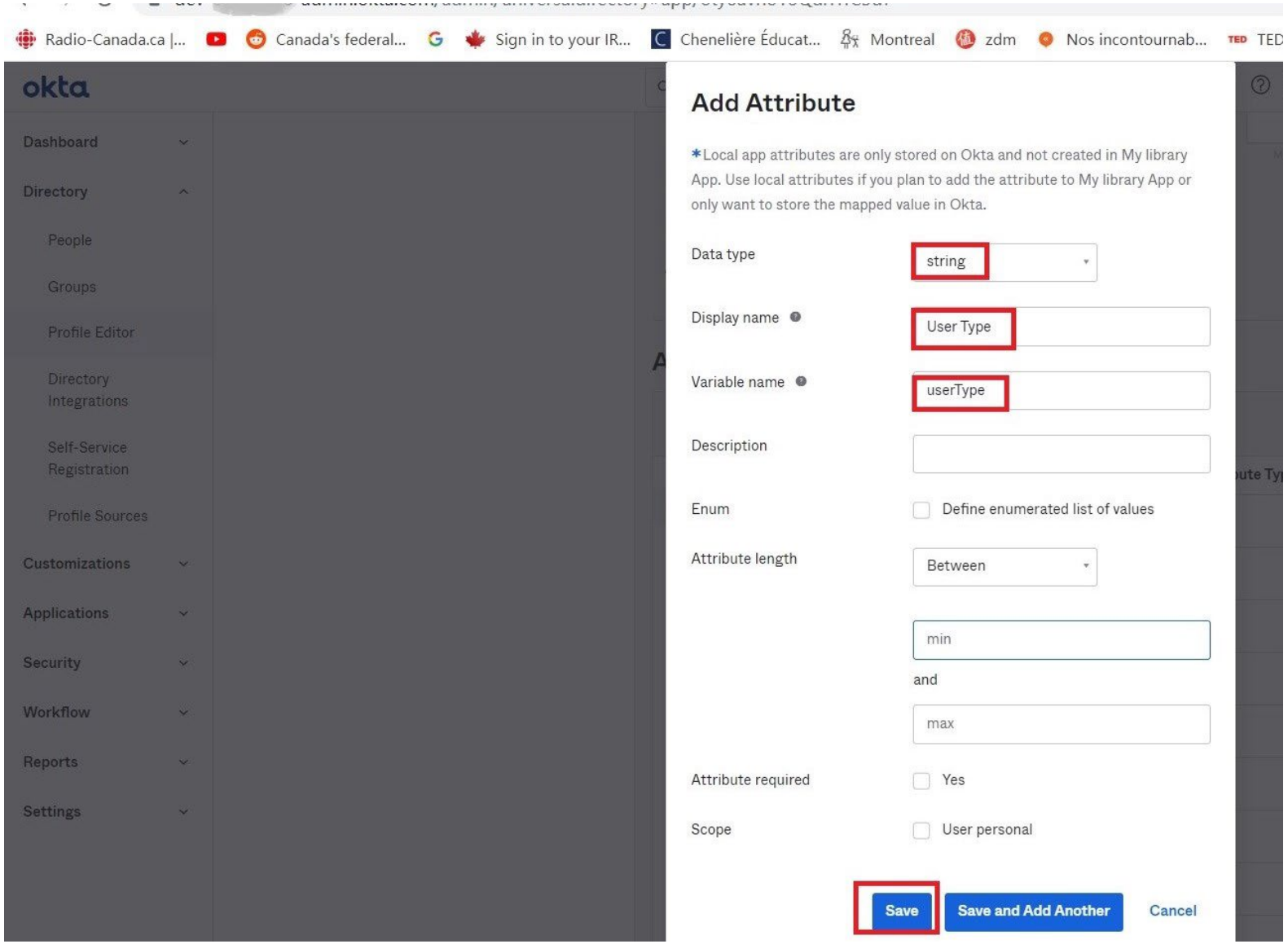


Step 2, add attribute。
let's go into directory, profile editor. Click on our app.



Click on “Add Attribute”.

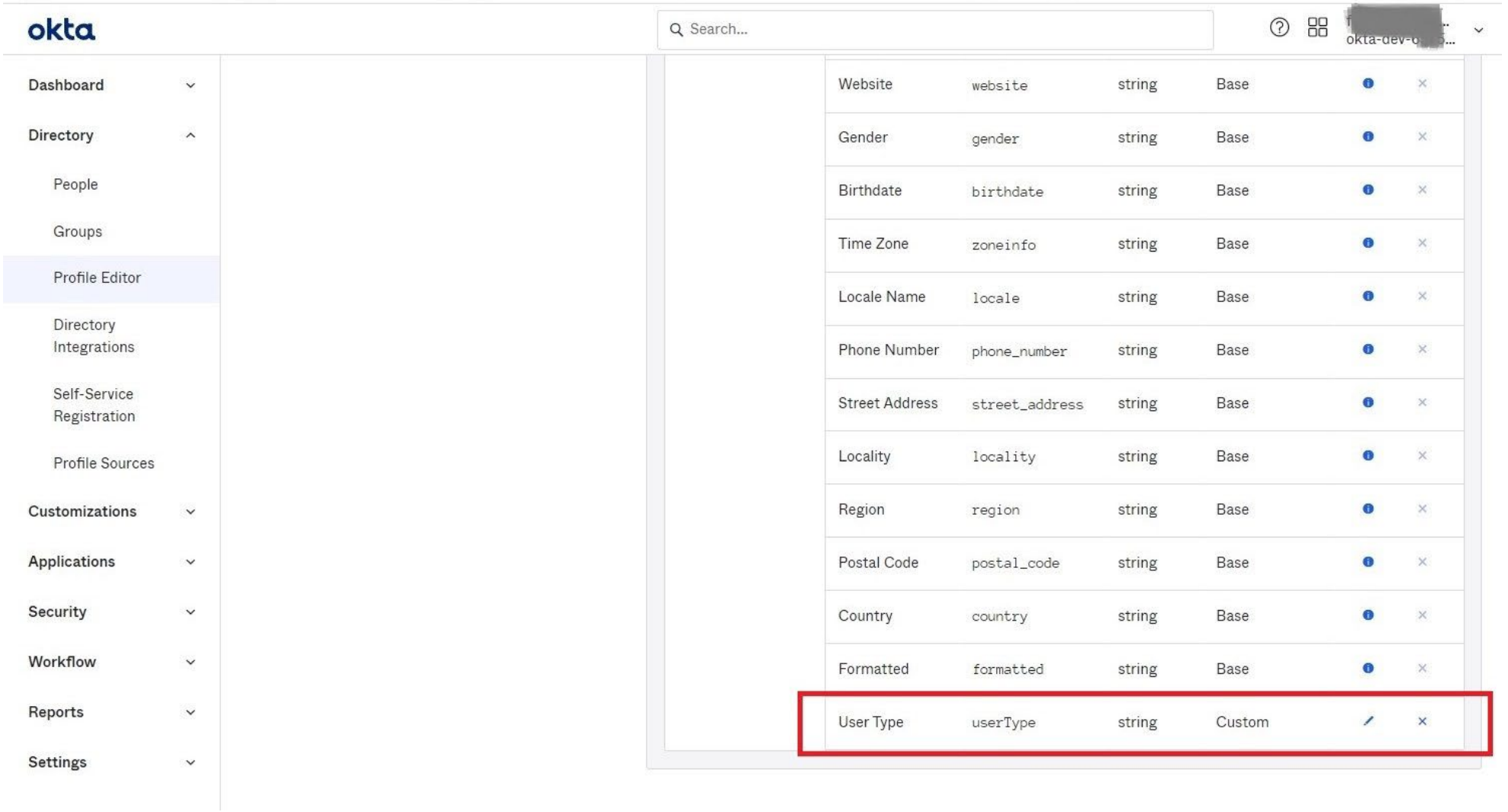




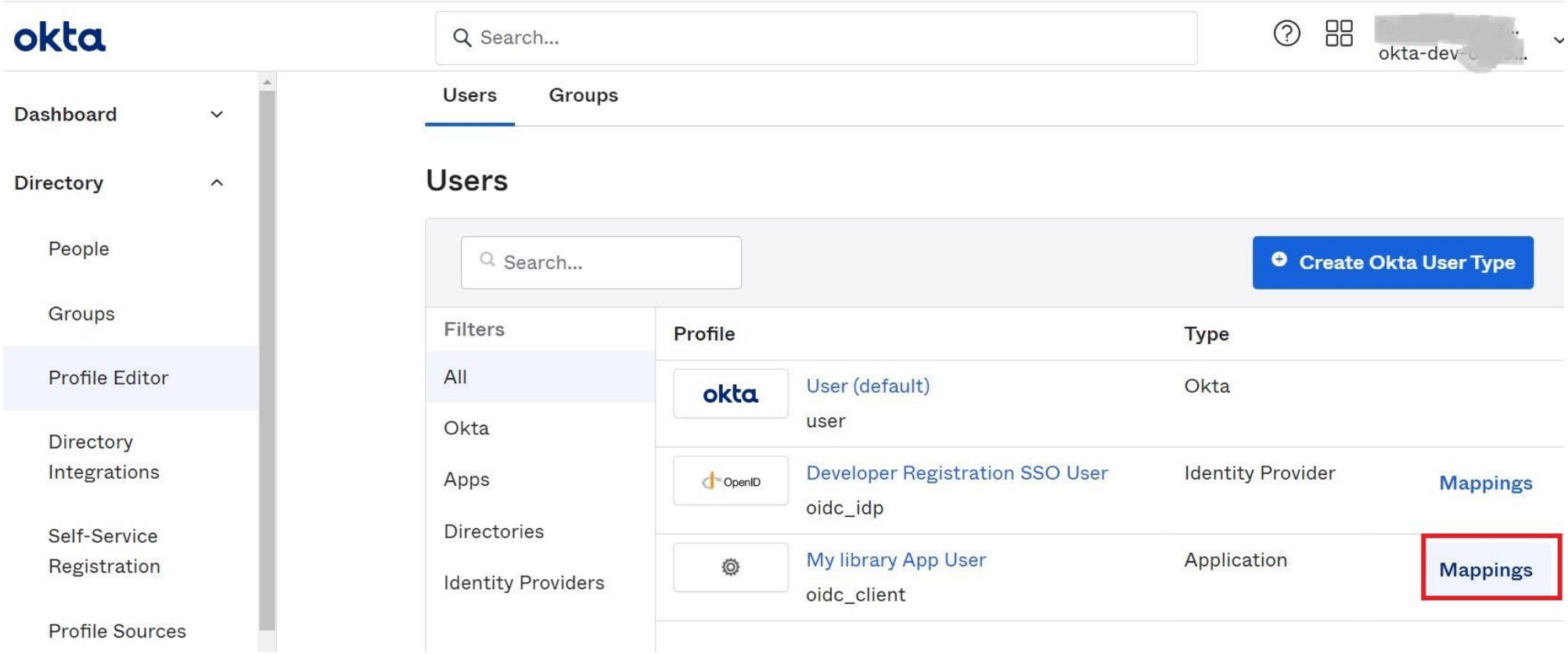
Make sure you input the same as the picture.

Then click on “save” button.

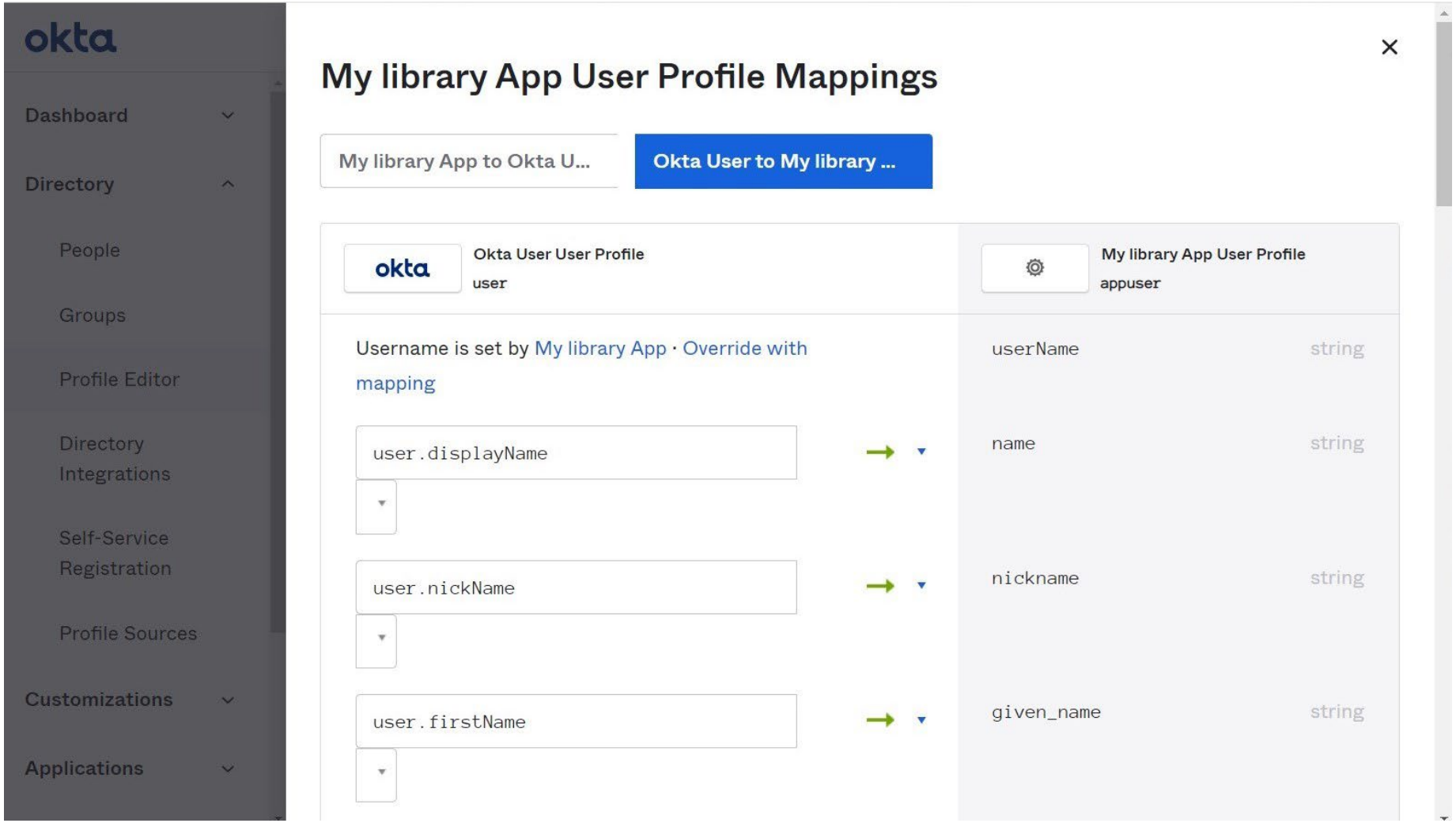
Then, scroll down the page, you will see our custom user Type is there.

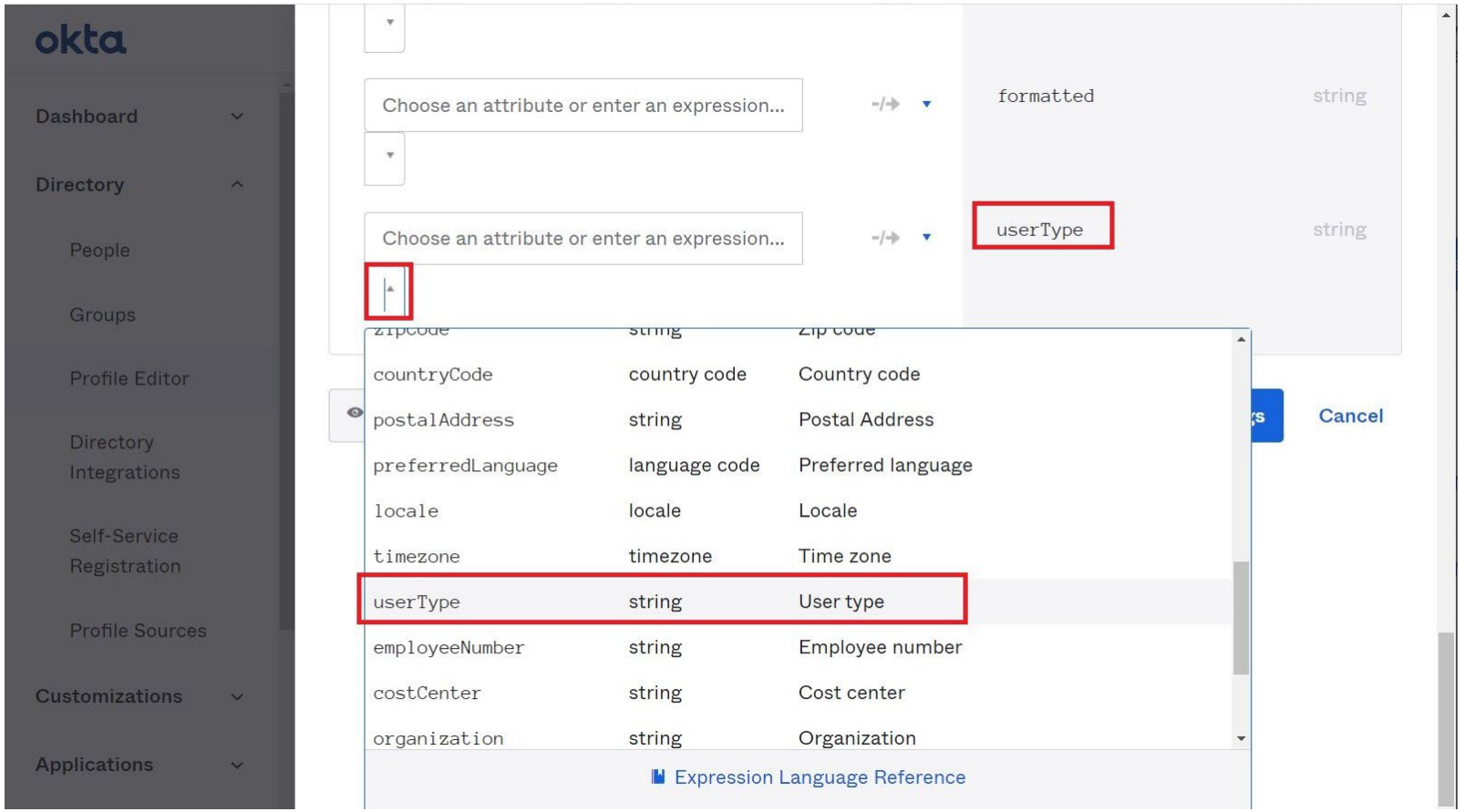


Step 3, mappings.
Now let's go back to our profile editor.
In my library app User Let's go ahead and click mappings.



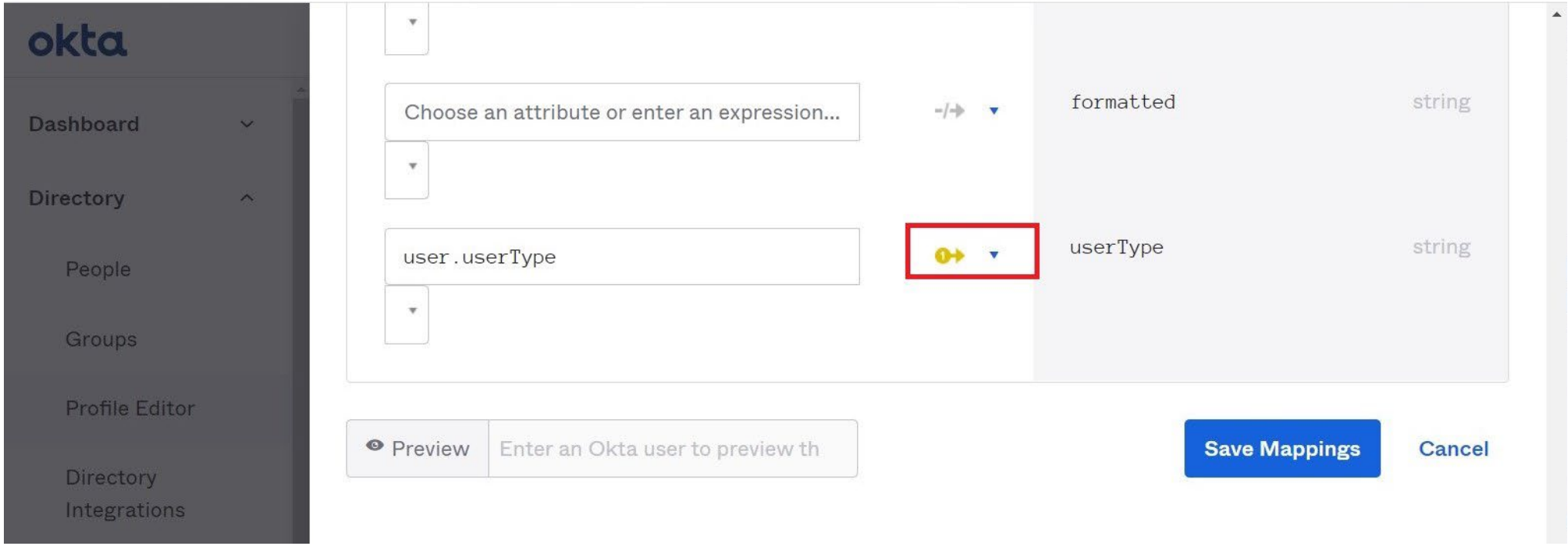
We will see this page.



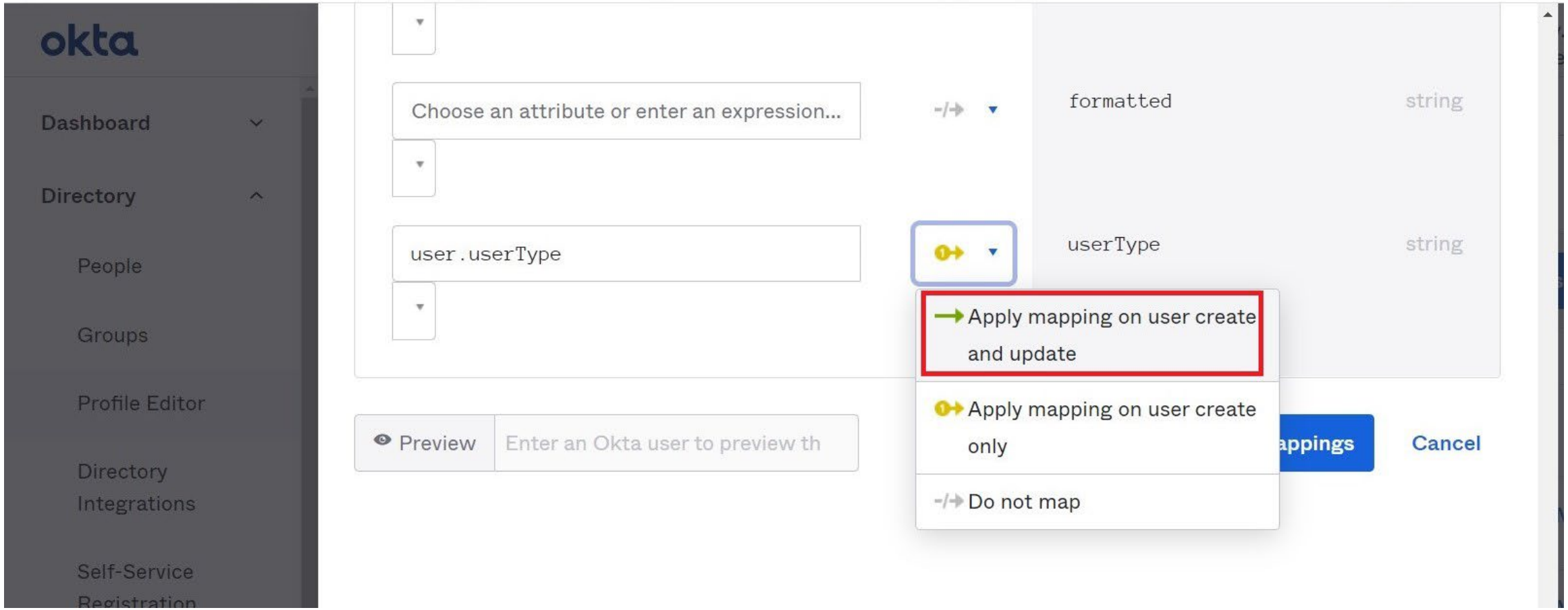


So let's go all the way to the bottom where we can see our new user type.
click the little arrow and scroll down until you can see user type.
type.

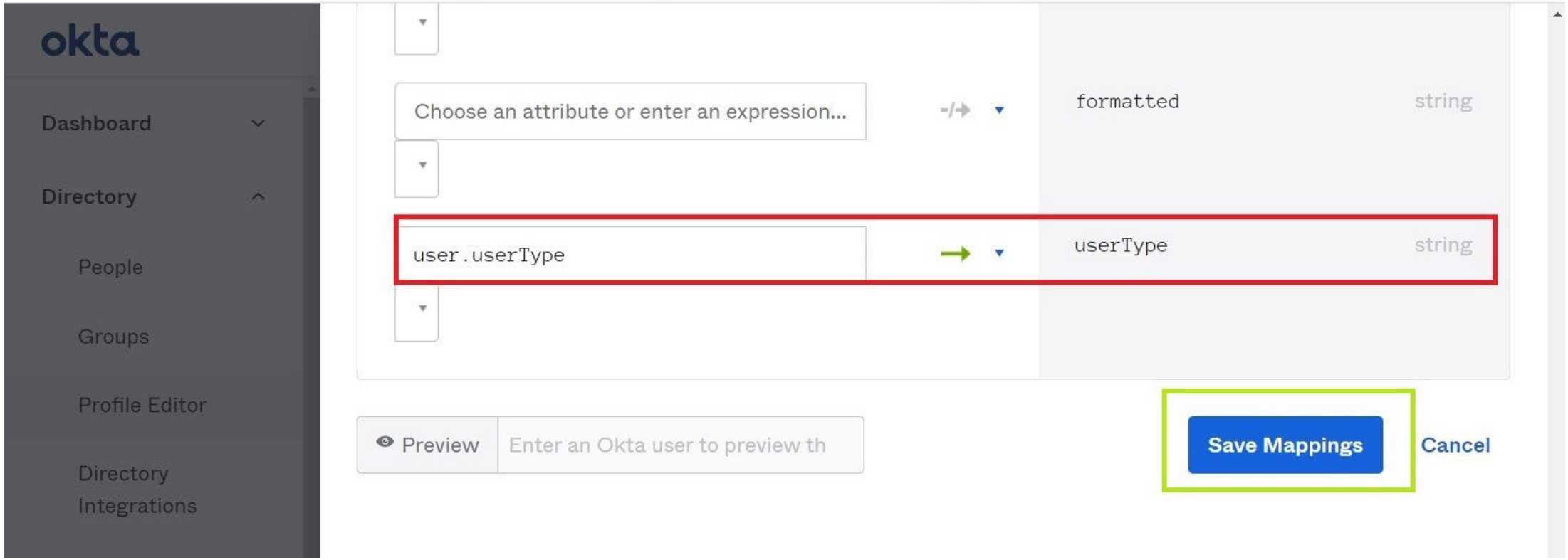
Click on the button beside the yellow arrow:



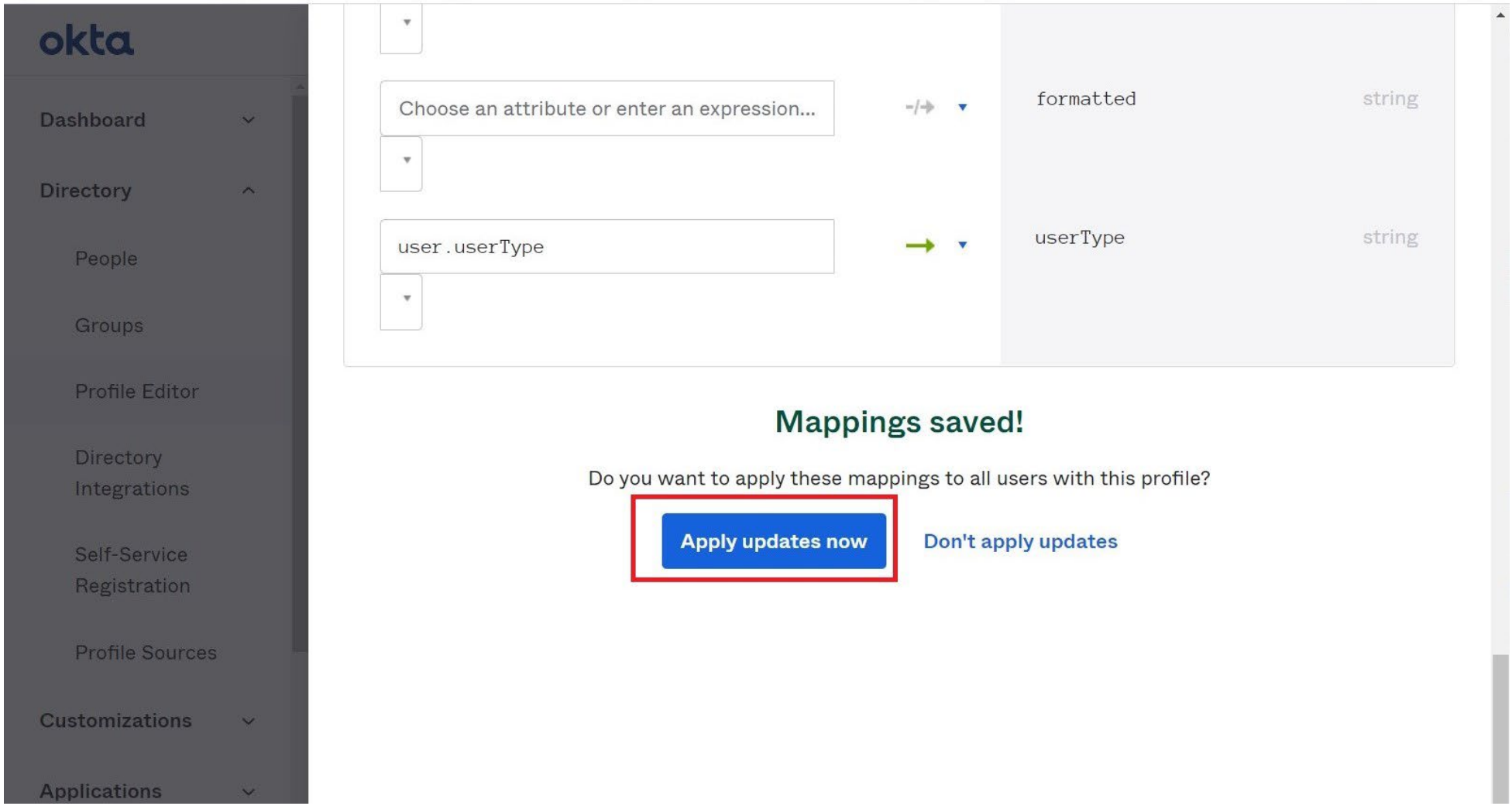
Choose this “Apply mapping on user create and update”.



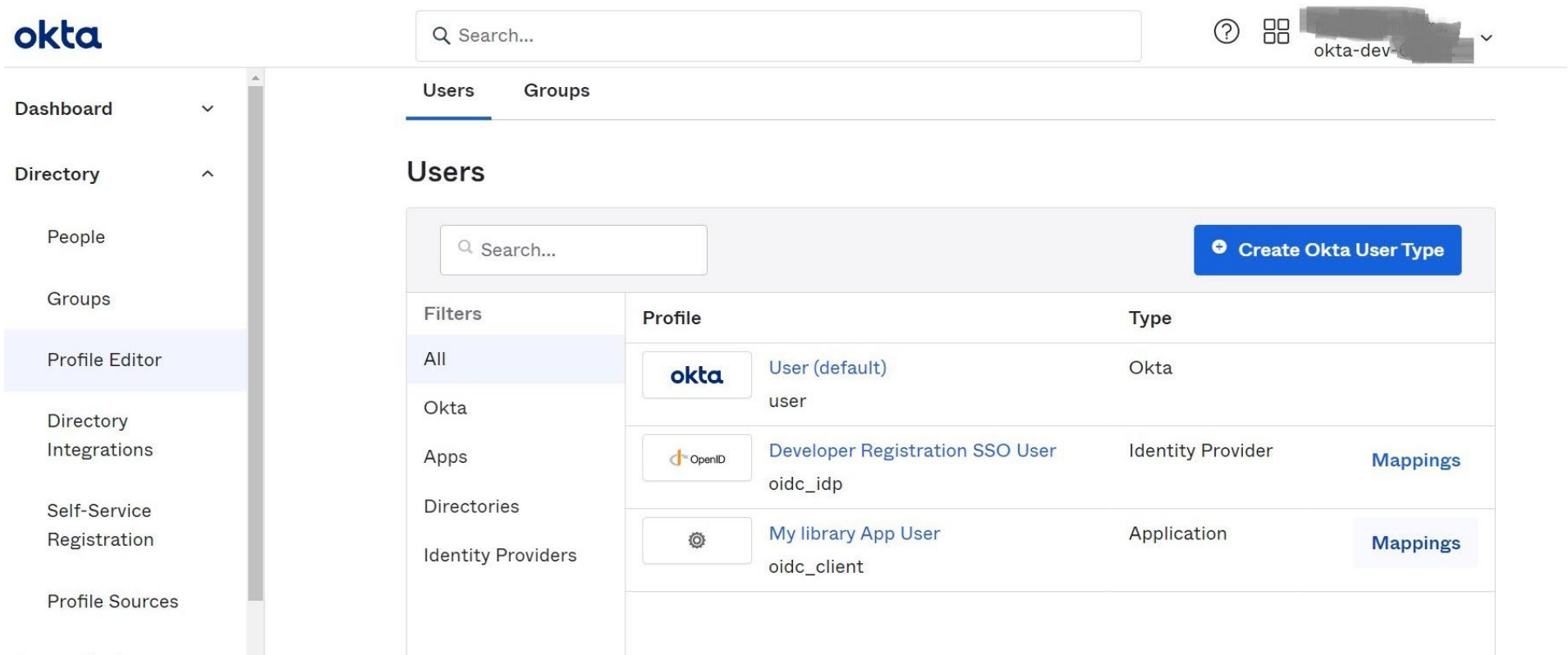
Make sure everything is the same as the red box, then click on “Save Mapping”.



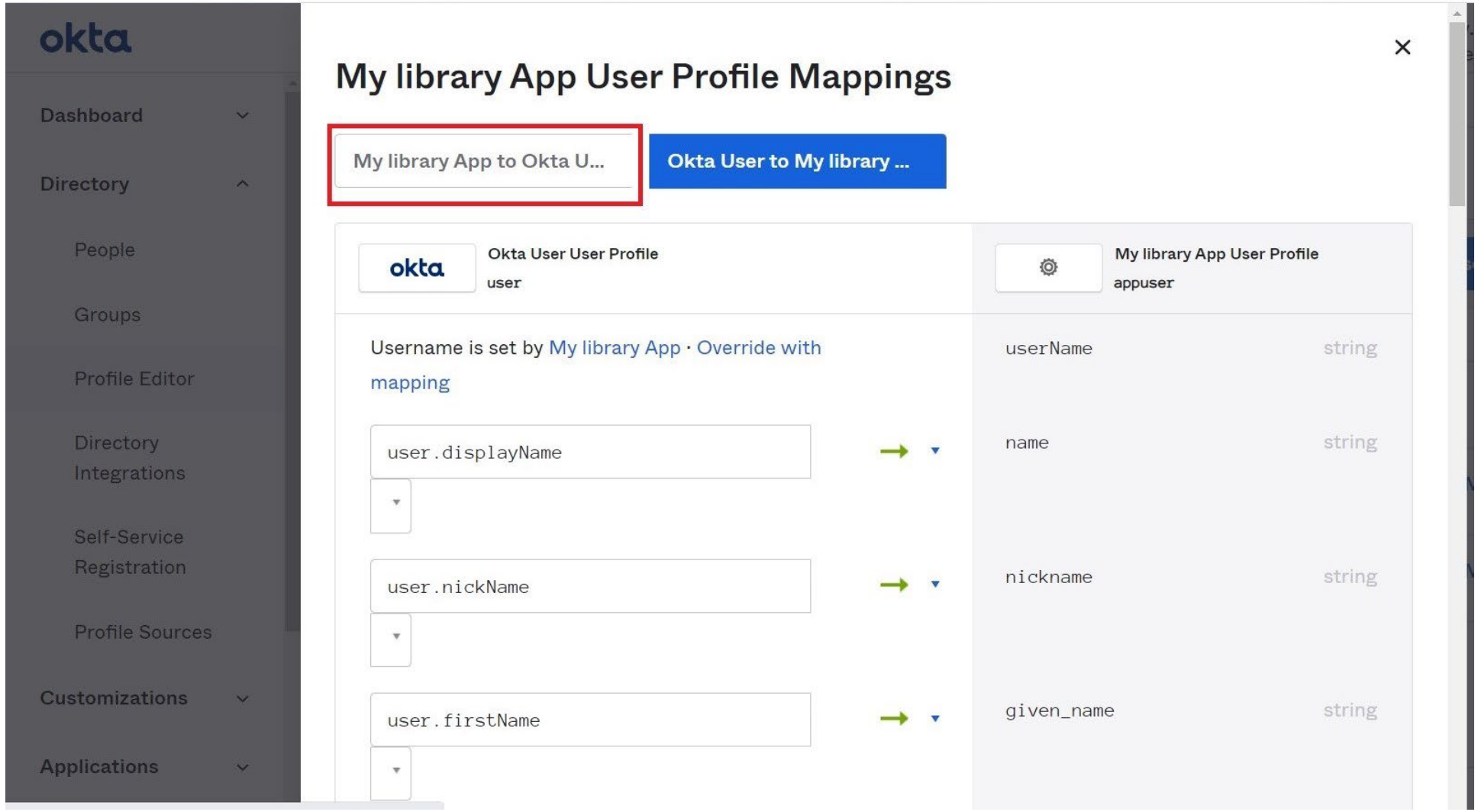
Then, click on the “Apply updates now ”.



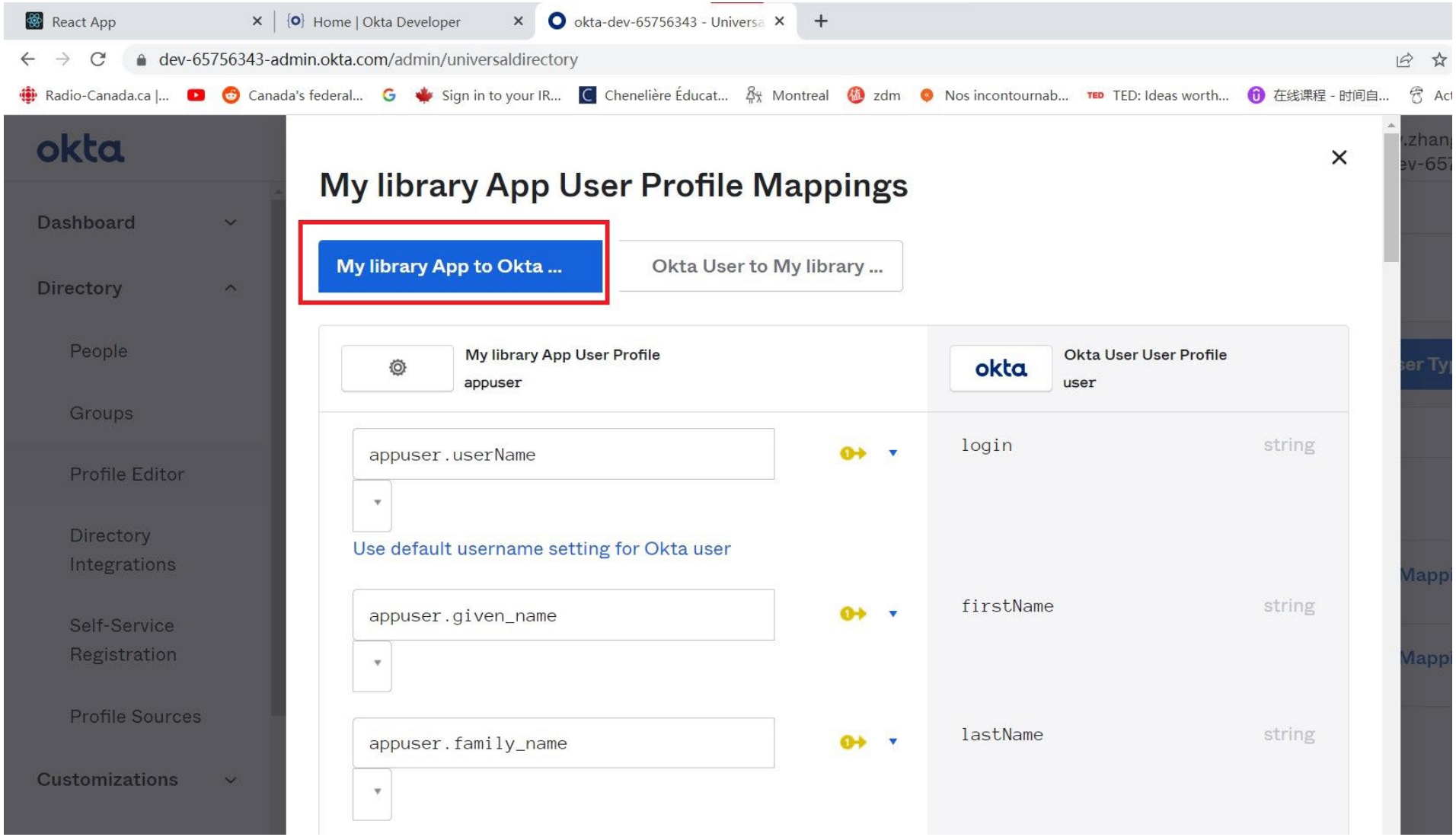
Click on “mapping” again.



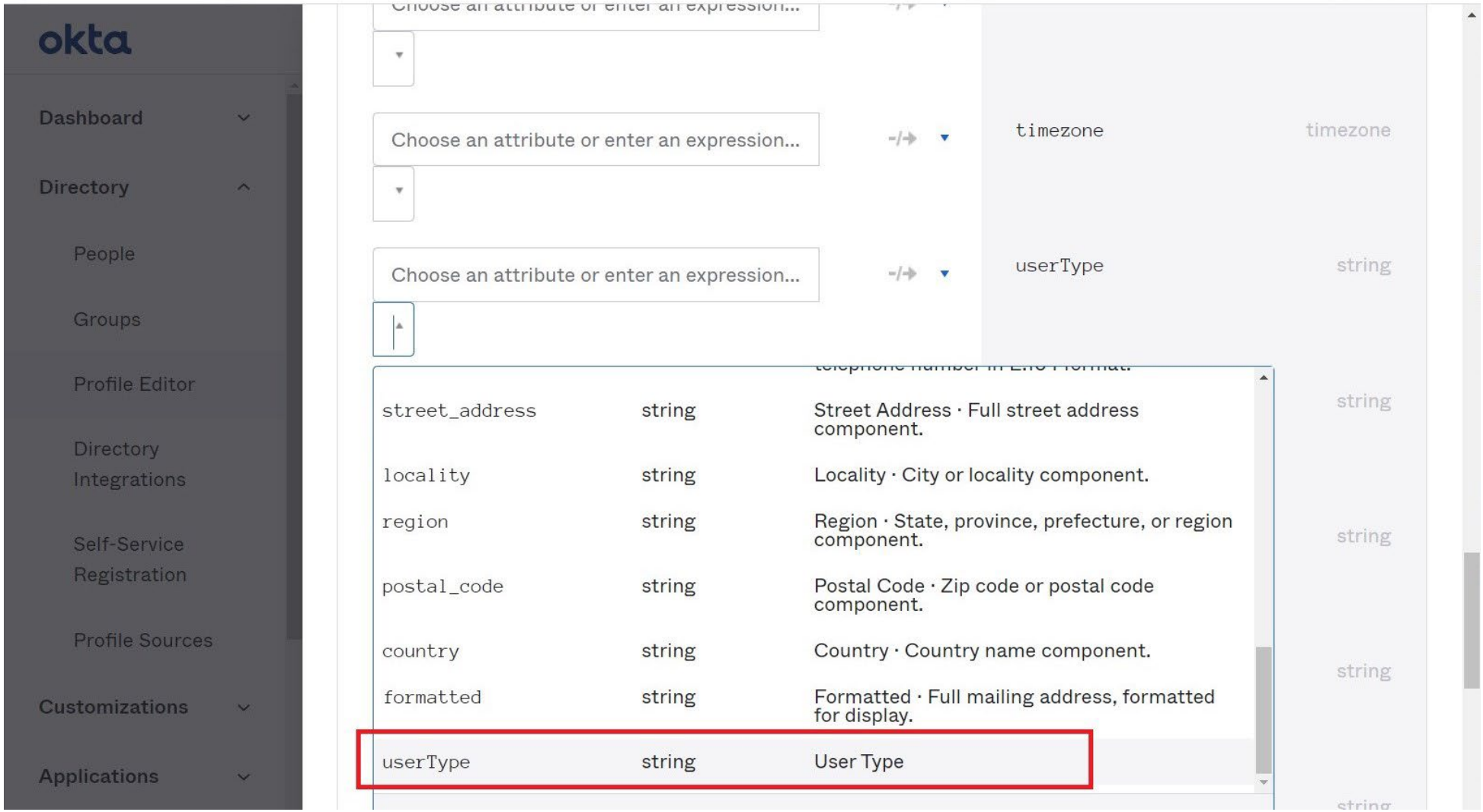
This time choose “My library App to Okta User Profile”.



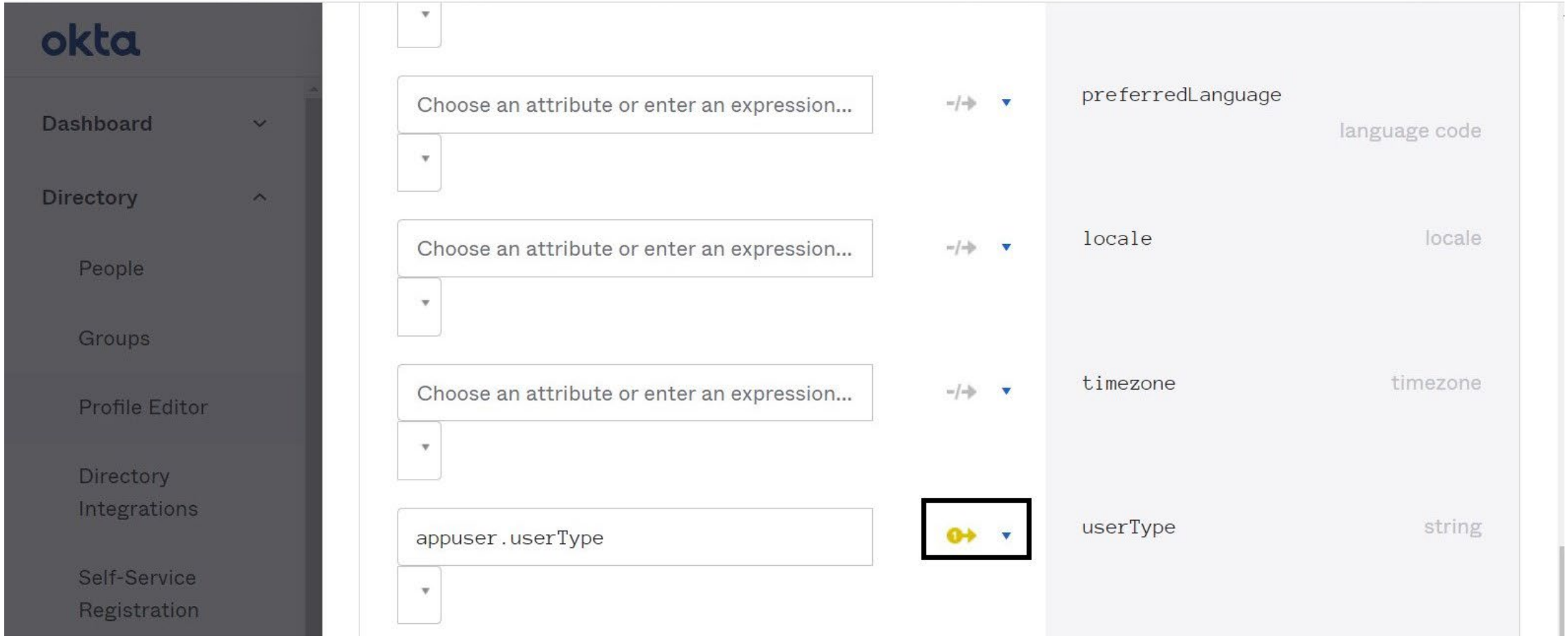
You will see the tab we have chosen become blue. Scroll down until we find user Type.



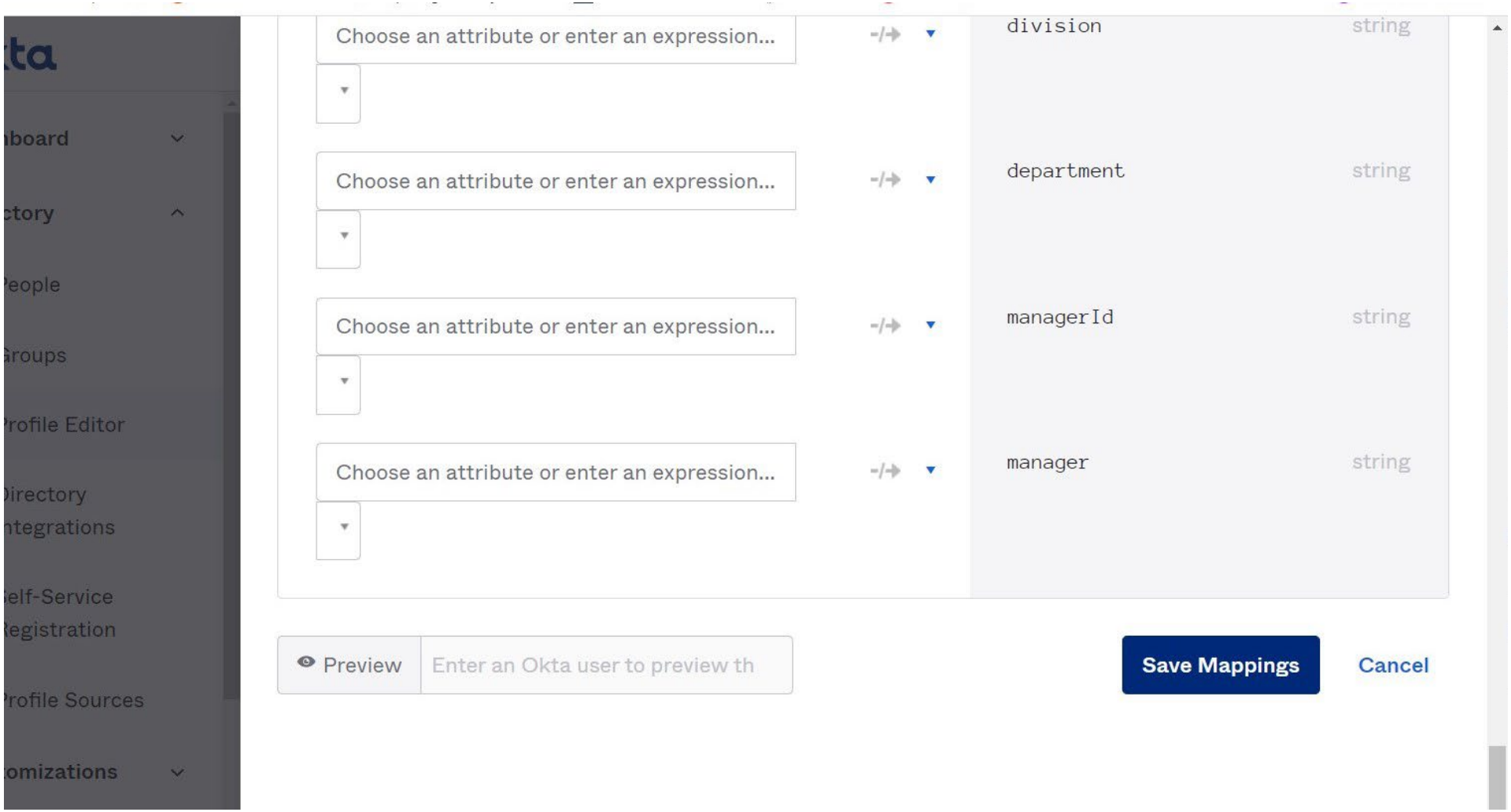
The same as the last step, choose as below:



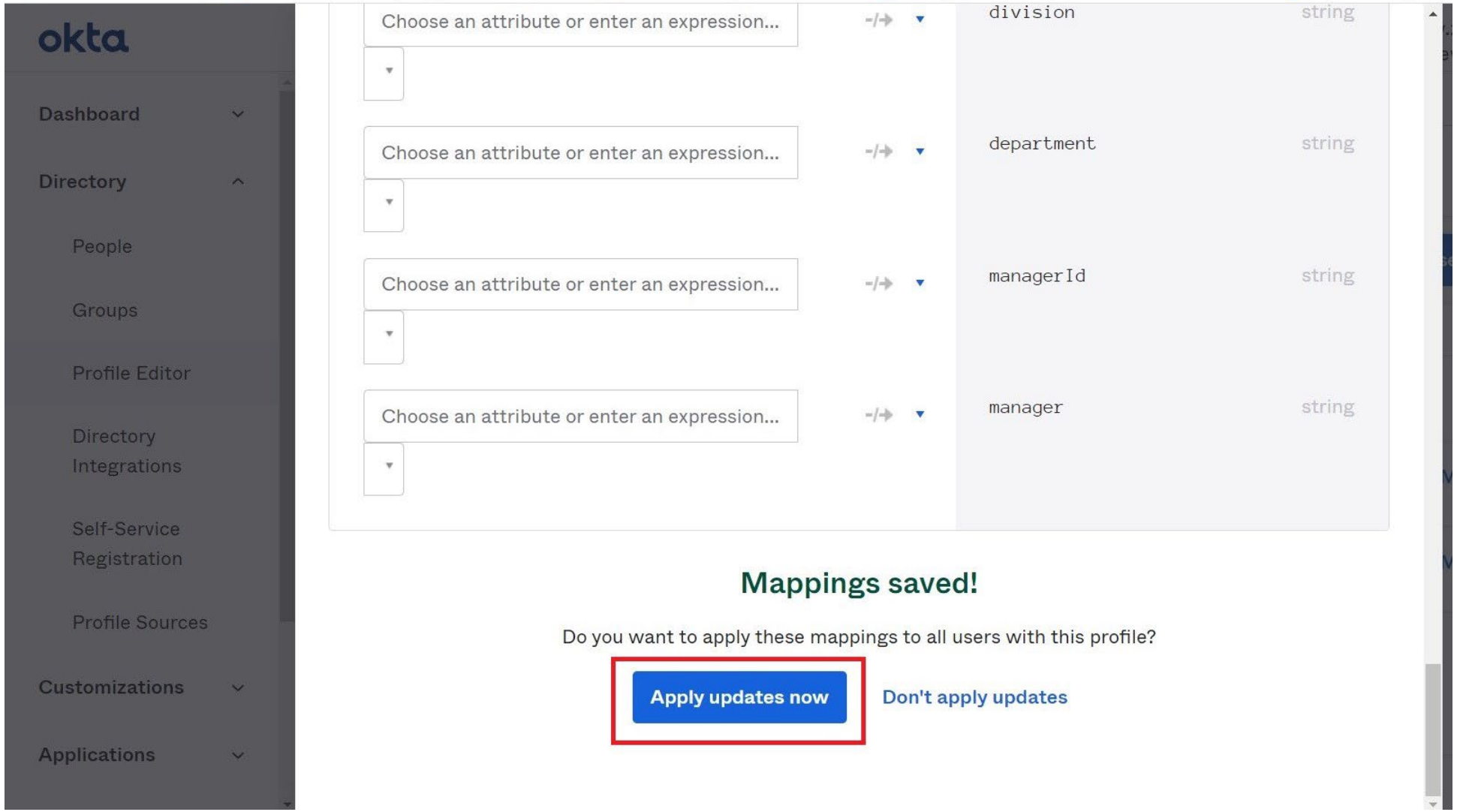
Here it's okay to leave it a yellow arrow.



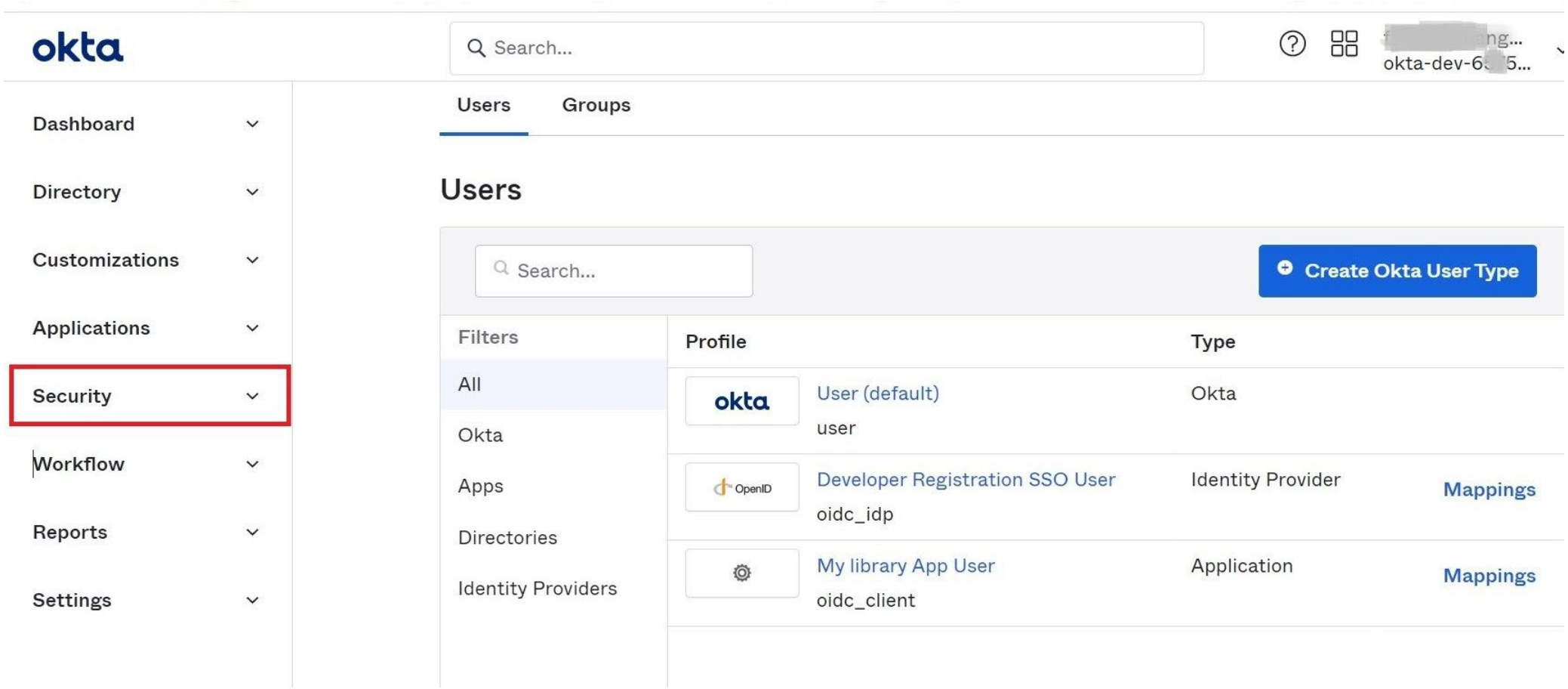
Click “Save Mapping”.



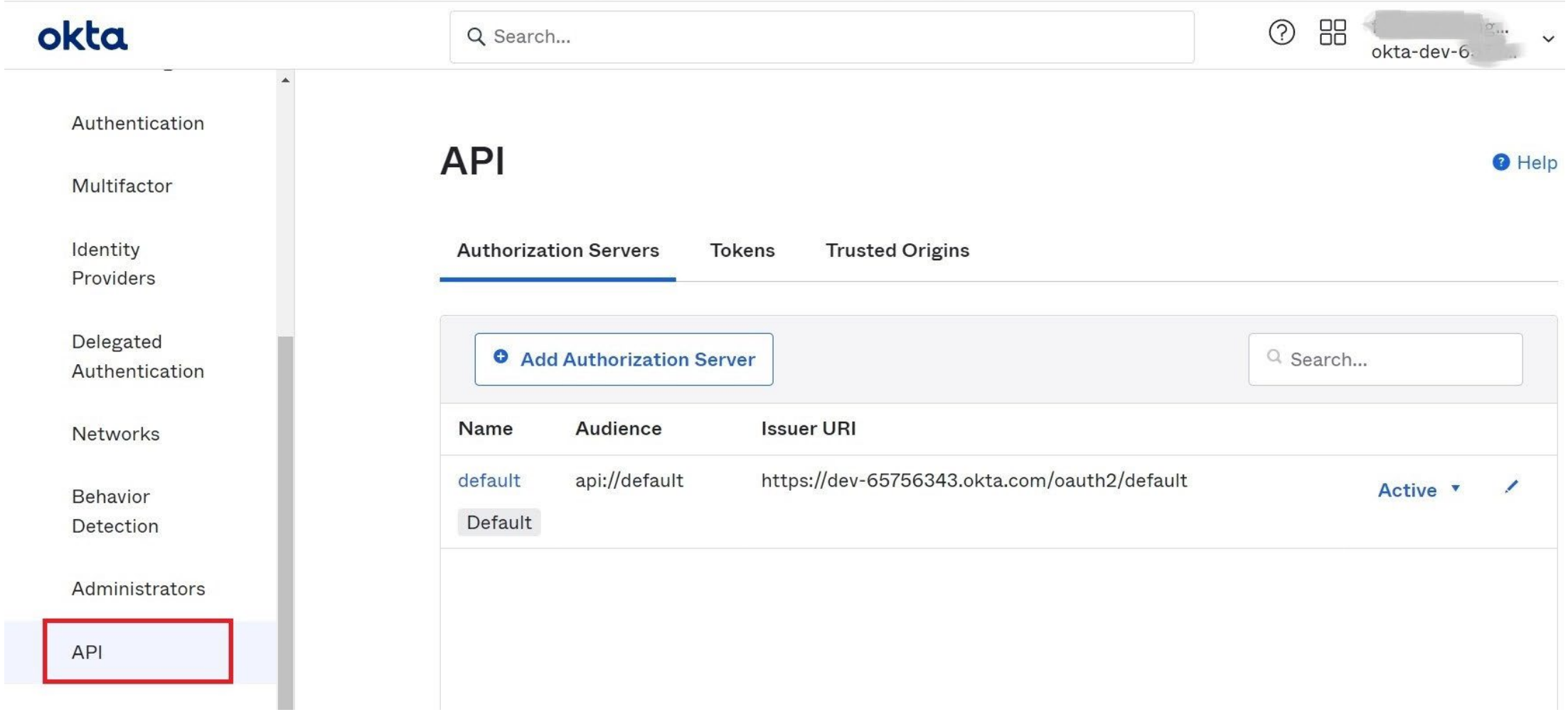
Then click on “Apply updates now”.



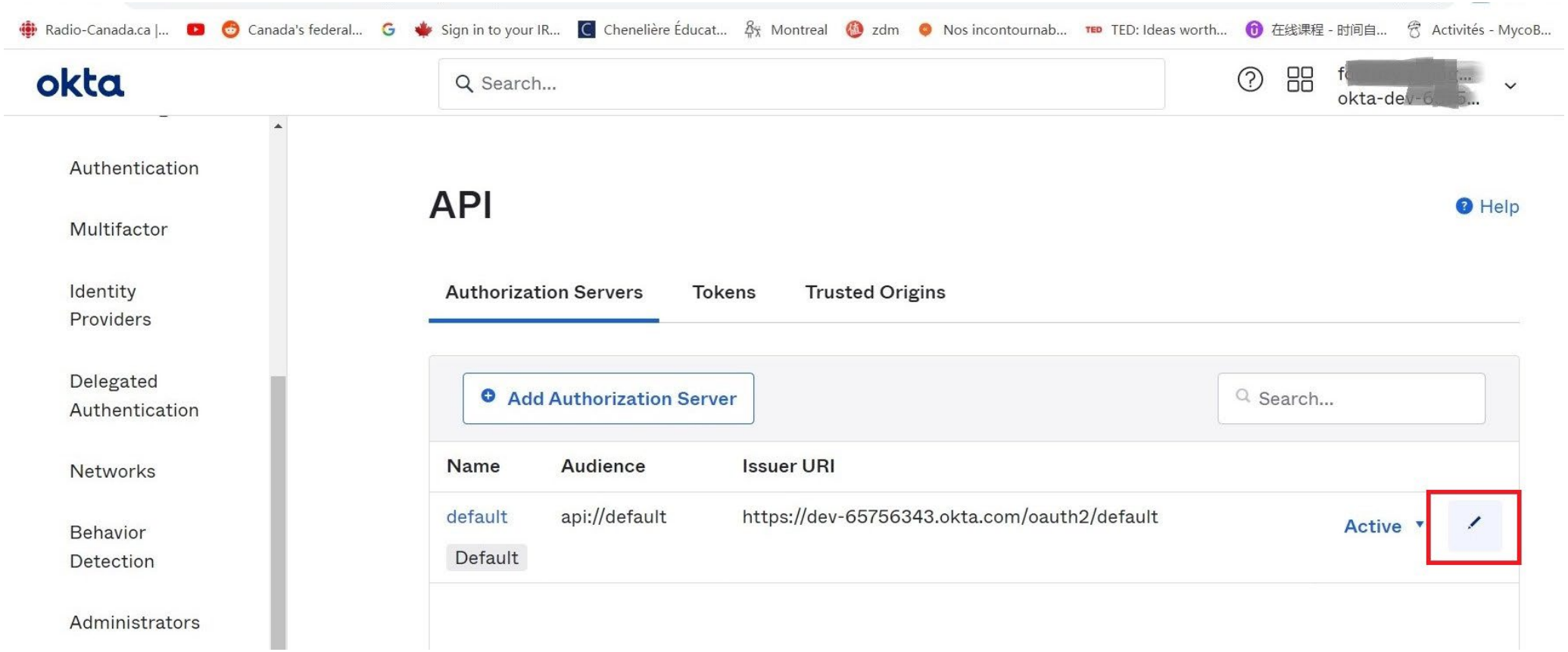
Step 4，configure claim in our API.
Go to security tab.



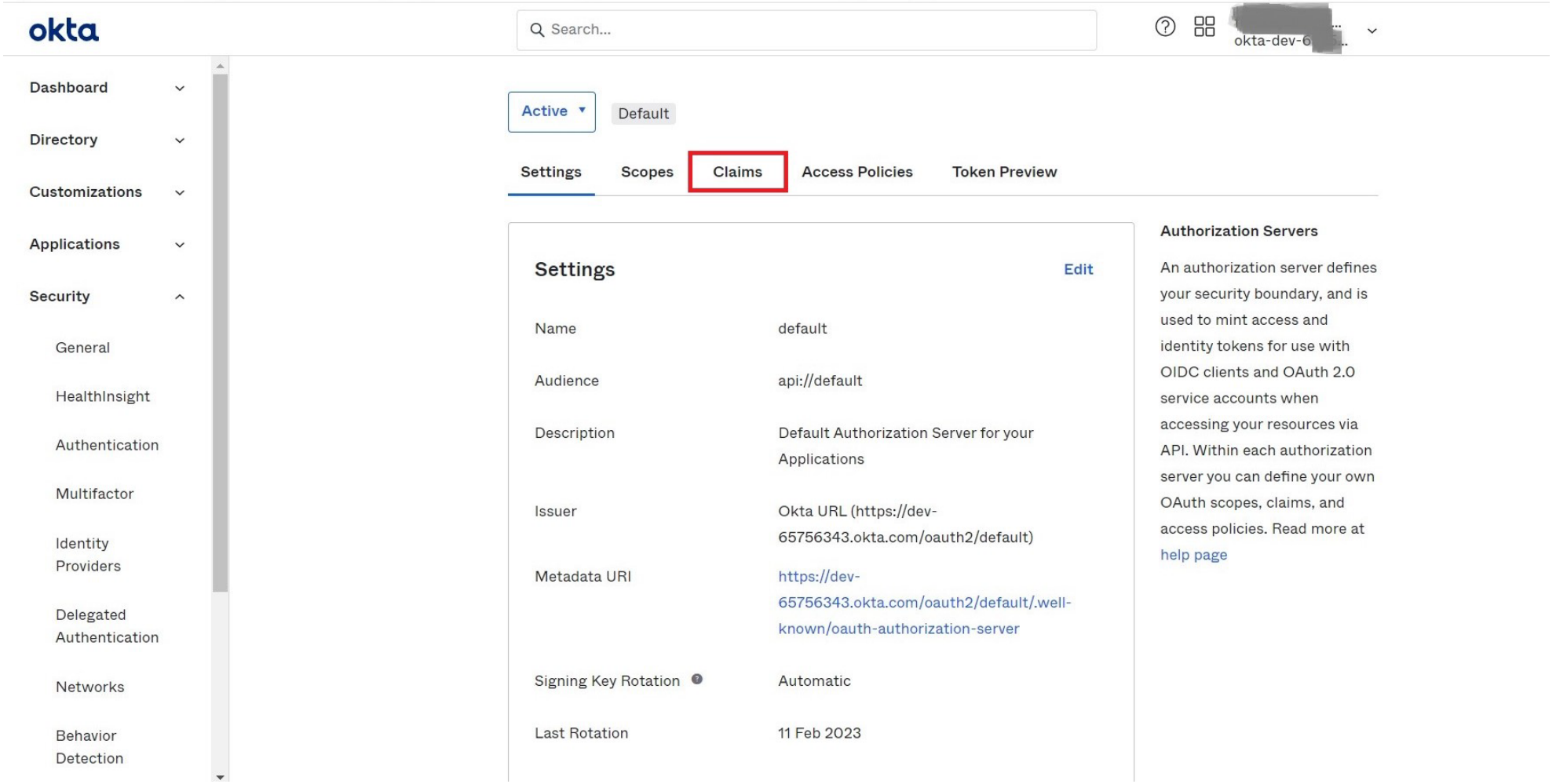
Scroll down until you can see API.



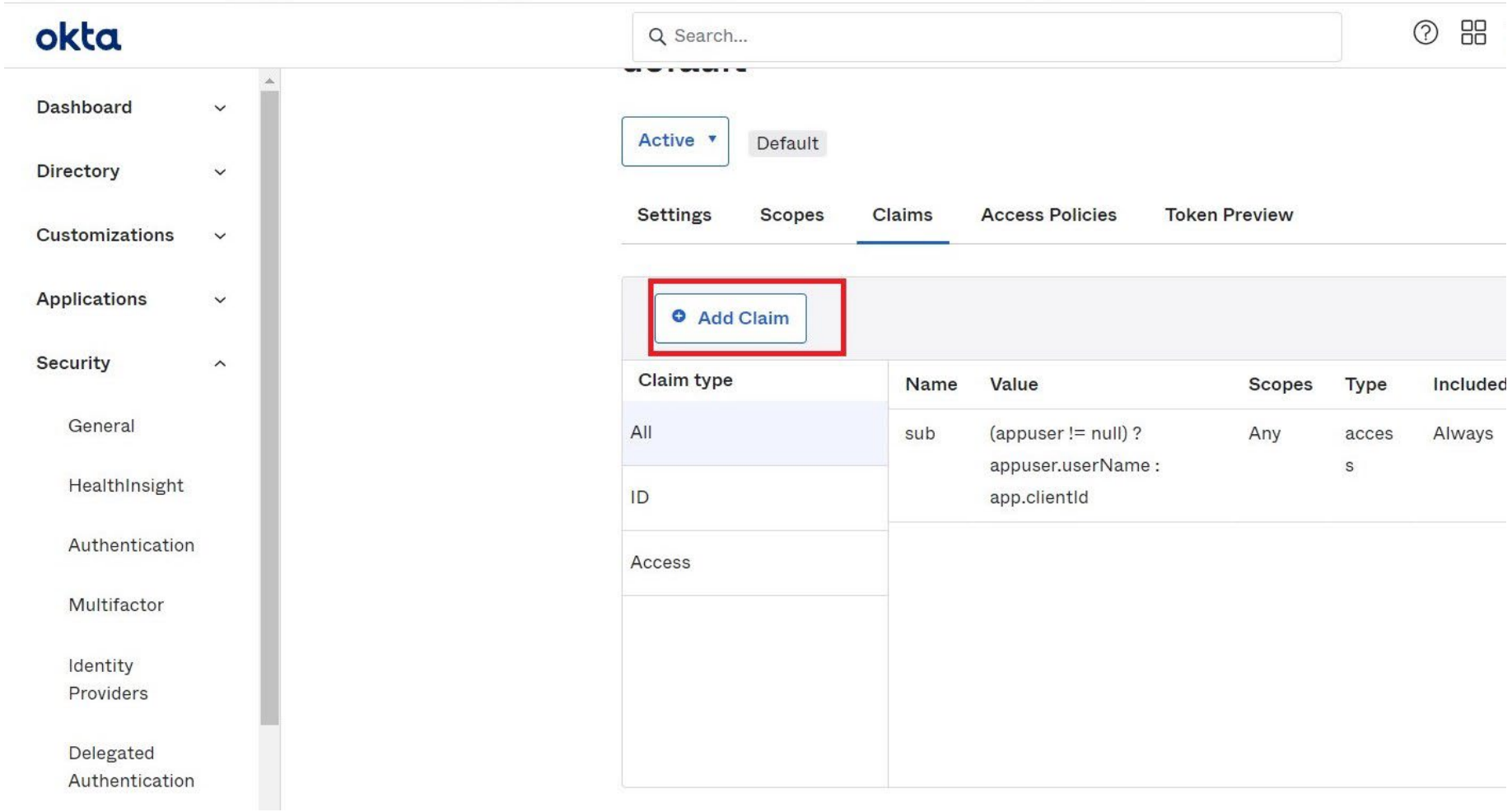
Click on “Edit”.

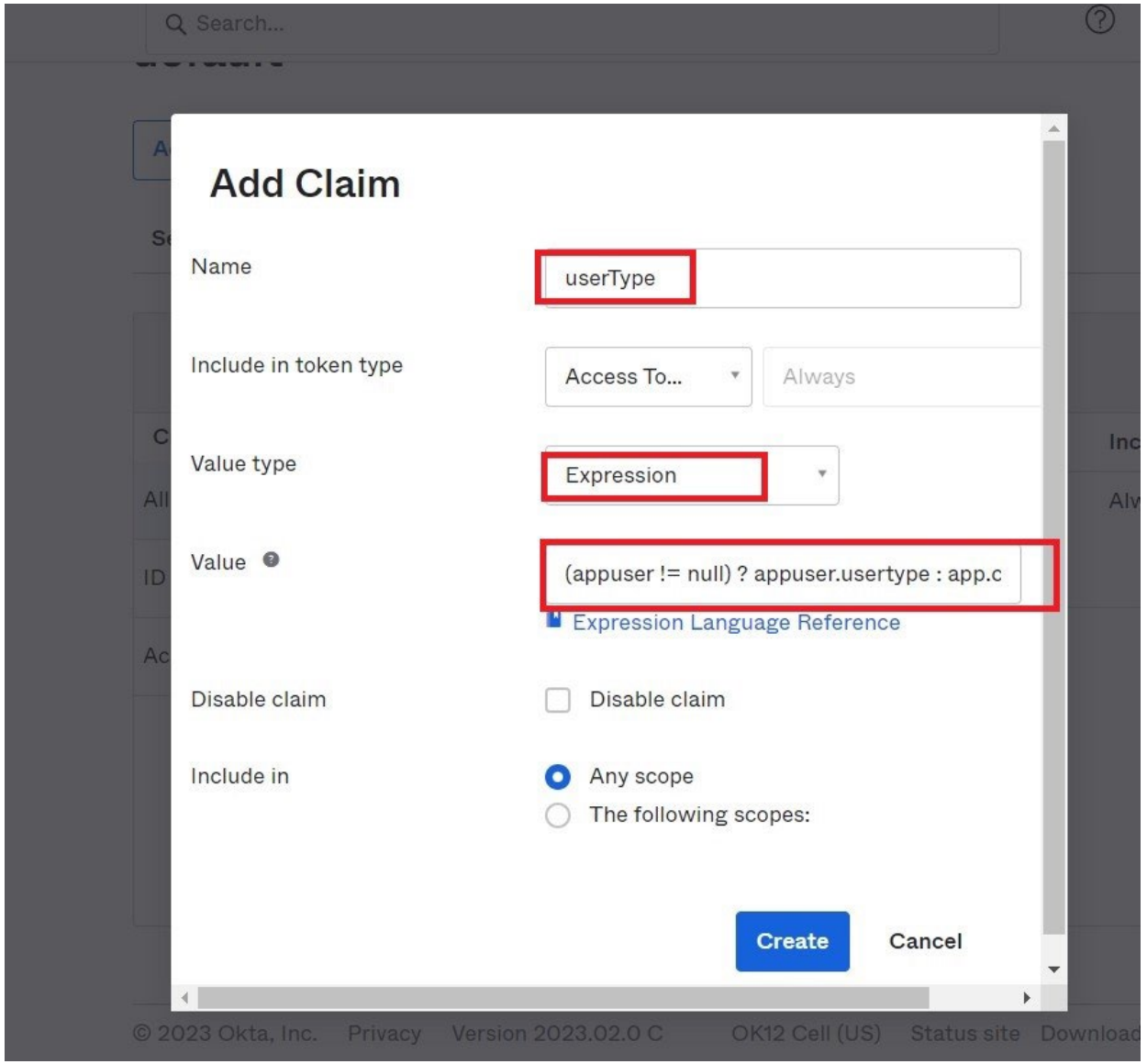


Click on the tab “Claims”.



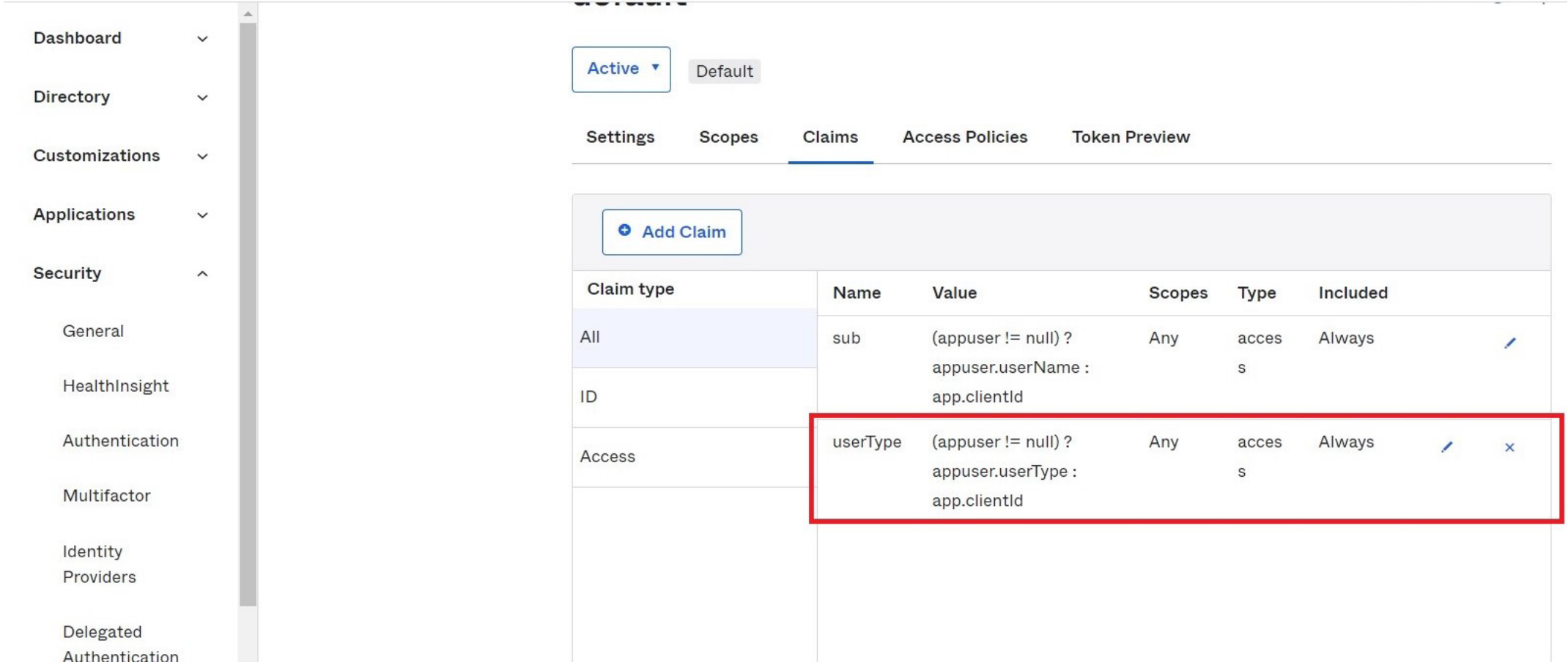
Click on “Add Claim”.





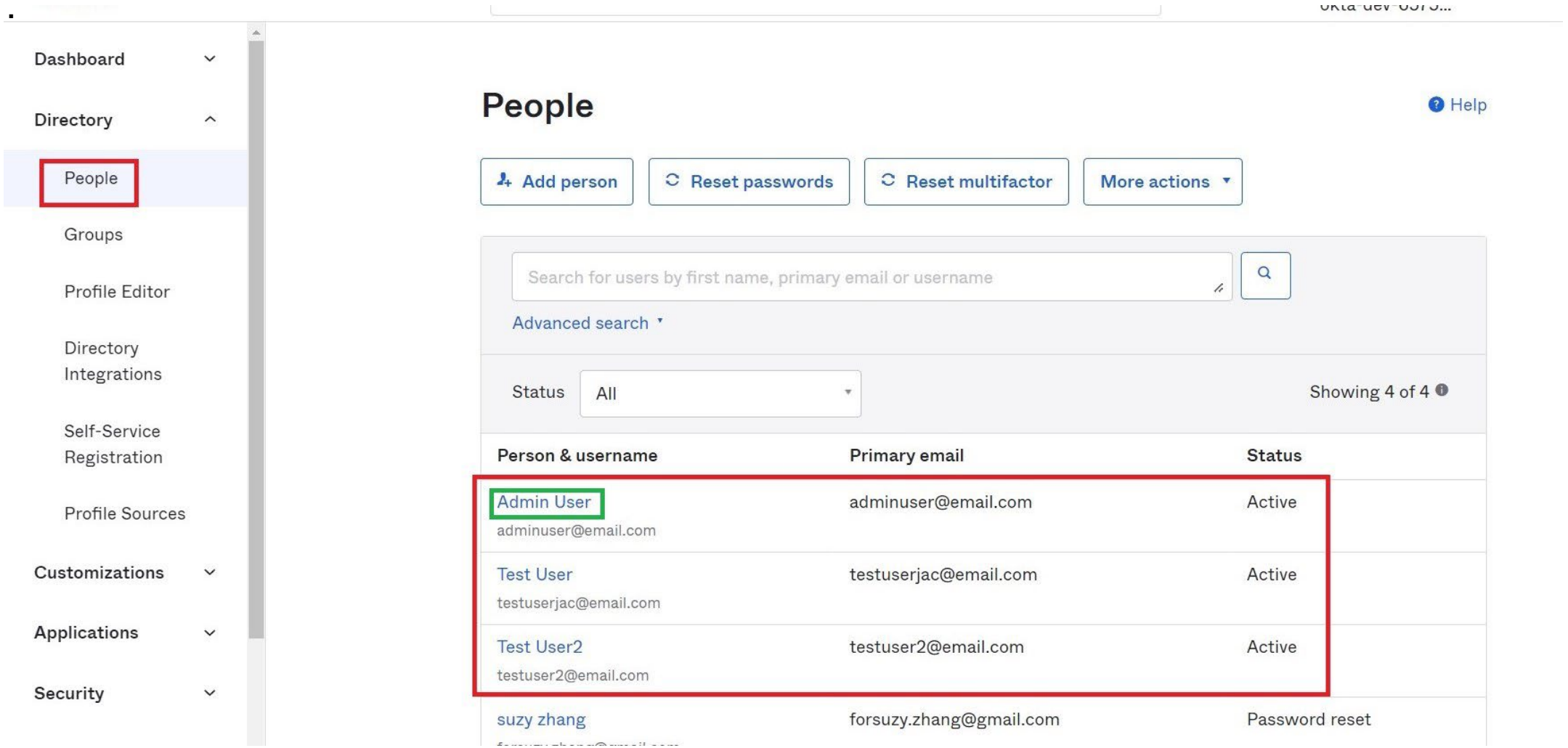
Make sure you do the same as the picture,
Value should be:
(appuser != null) ?
appuser.userType : app.clientId.
Then , click on “create”.

Here, we can get this page and our new created userType appeared.

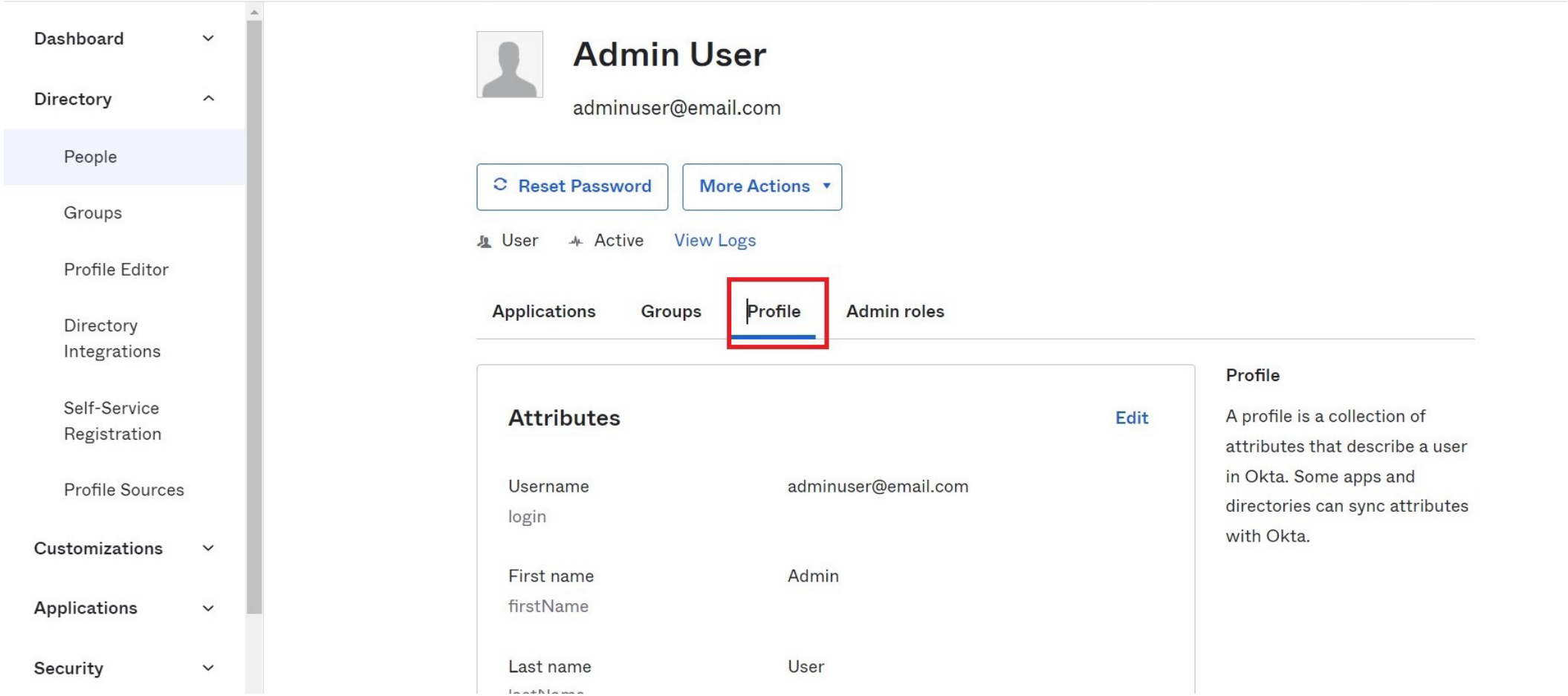


Now let’s go back to our directory and choose “people”.
Click on “admin user”.

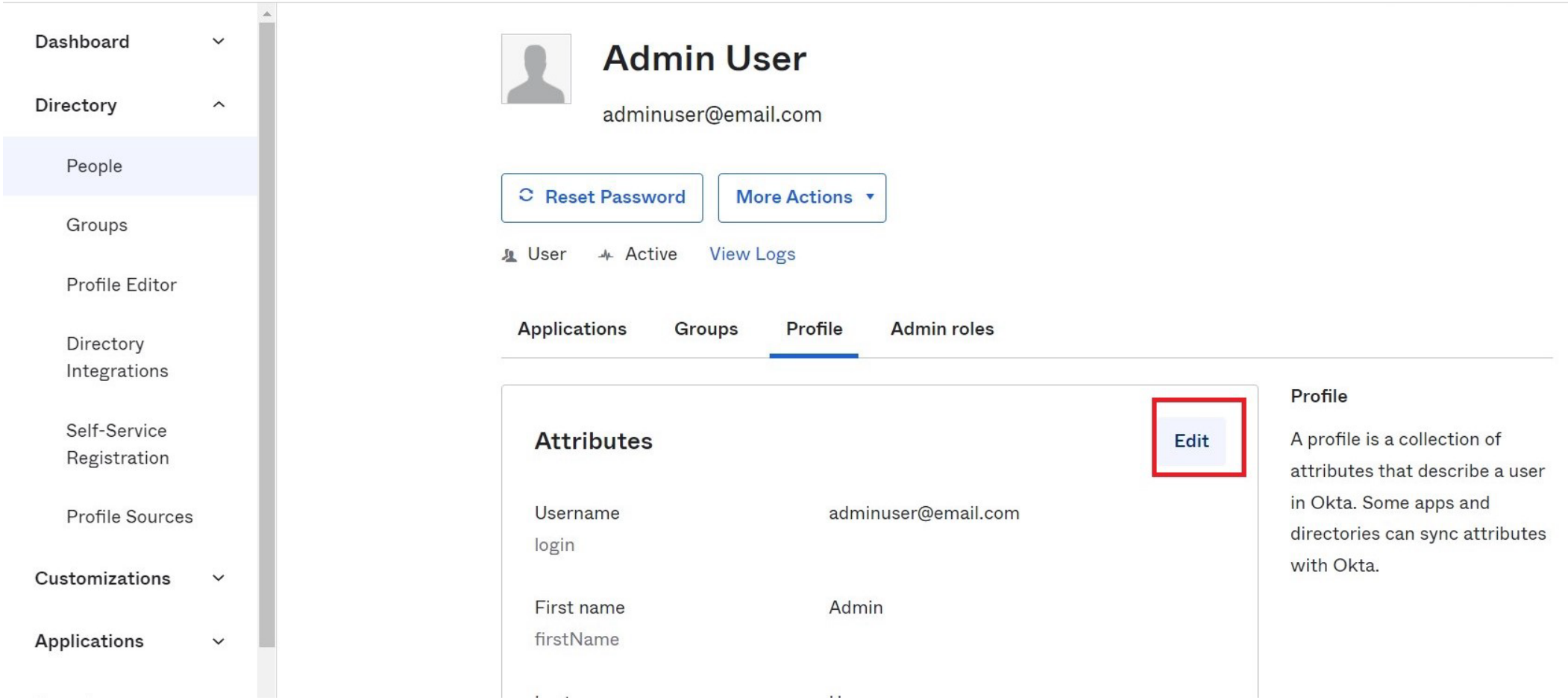
.



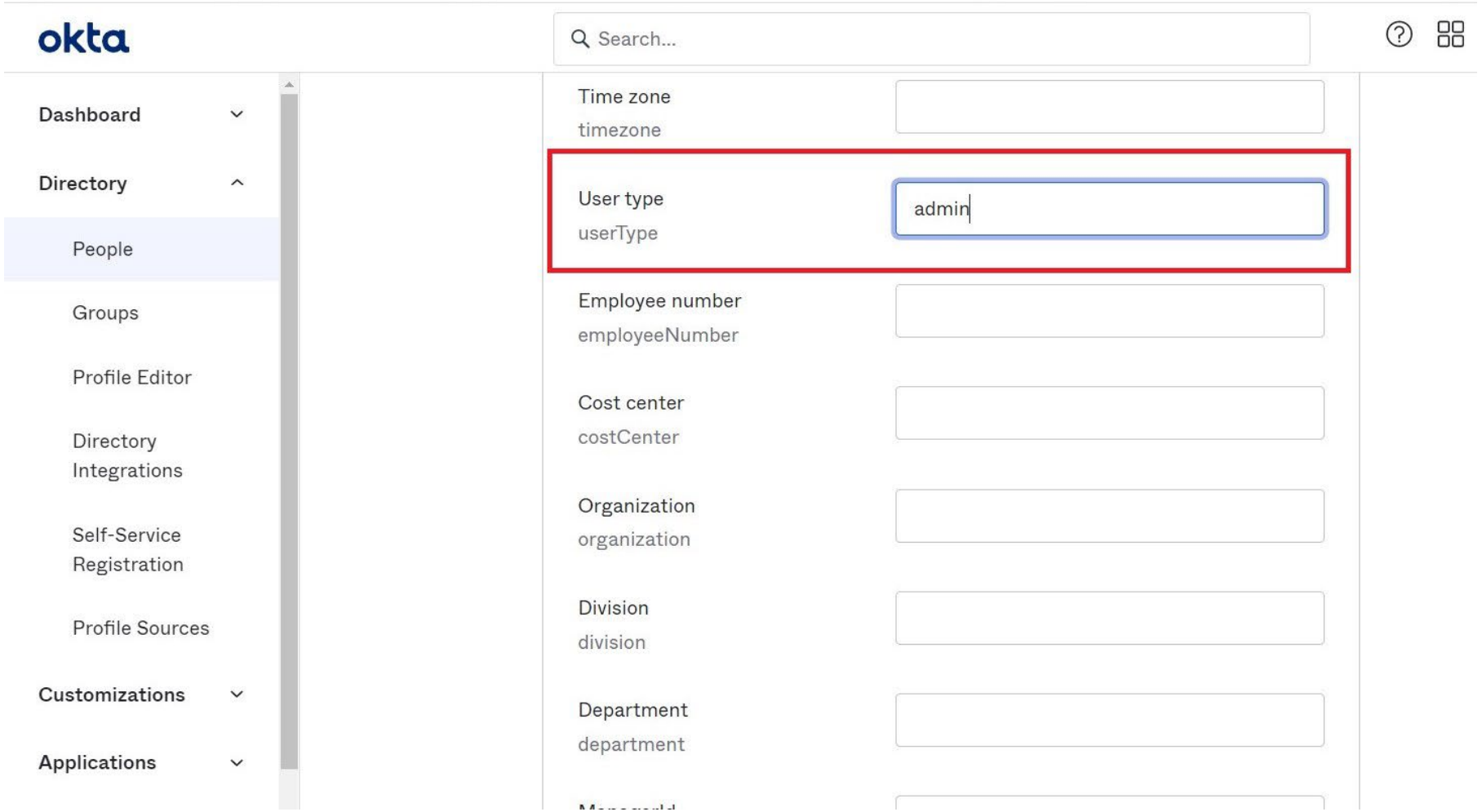
Let's go to profile.

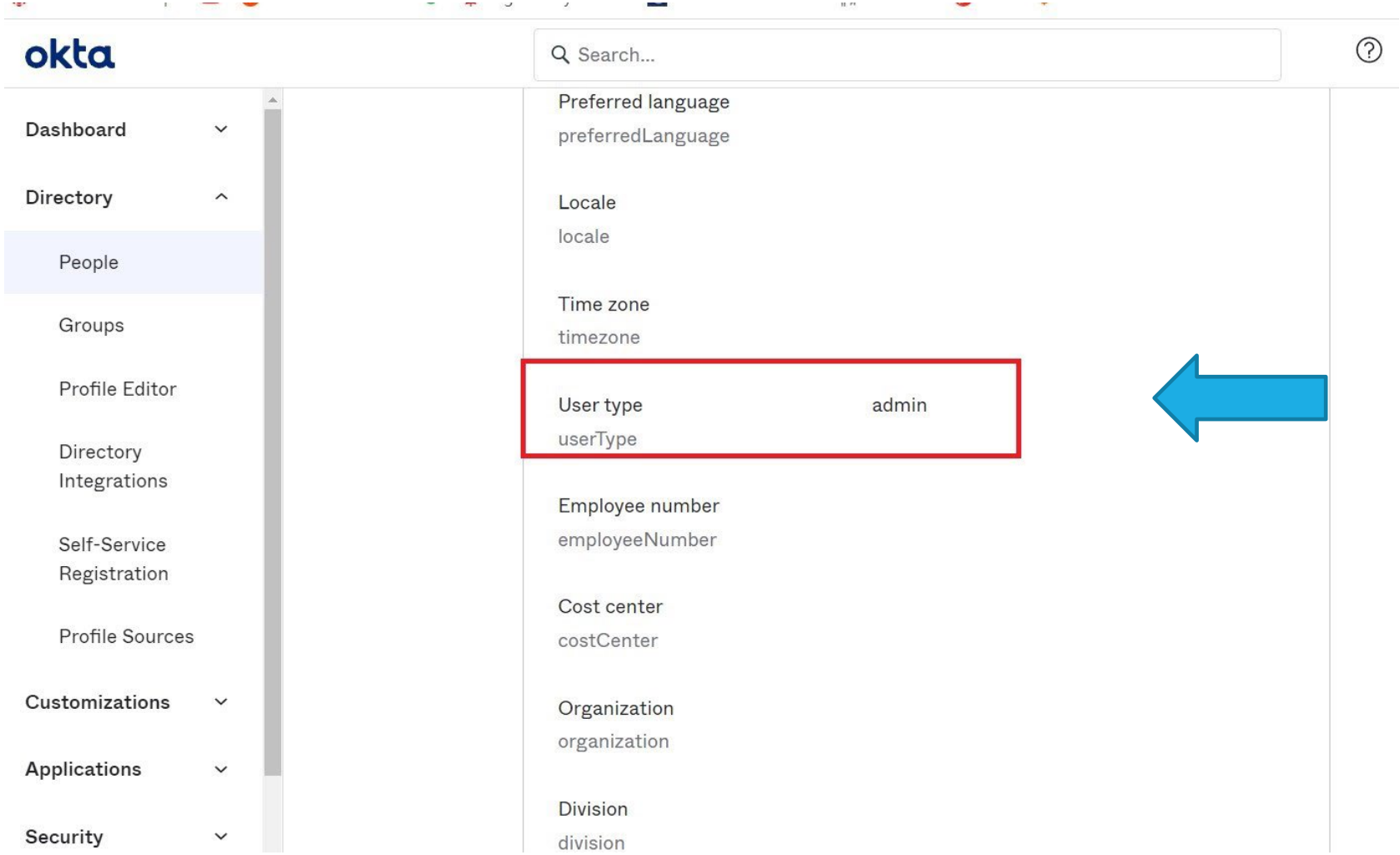


Click on “Edit”.



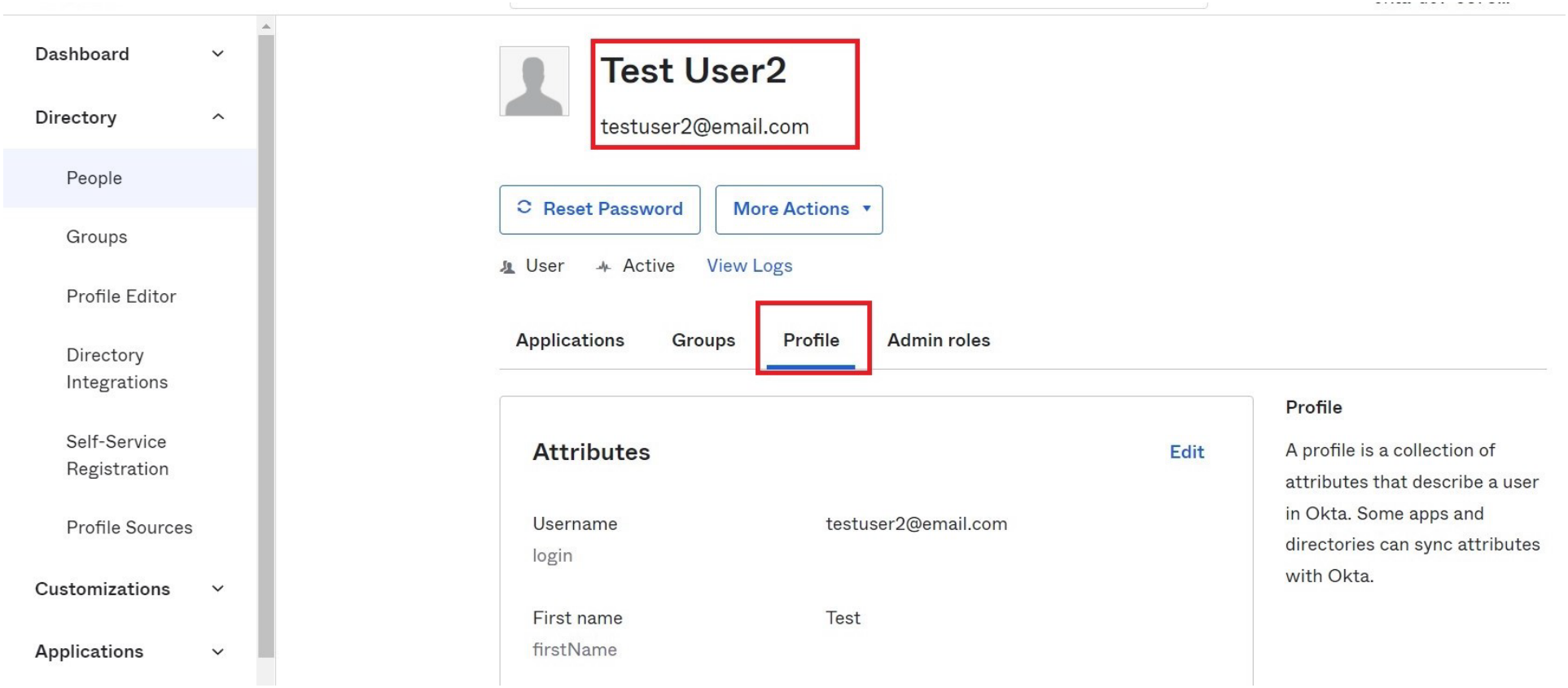
Input “admin” in User type box.



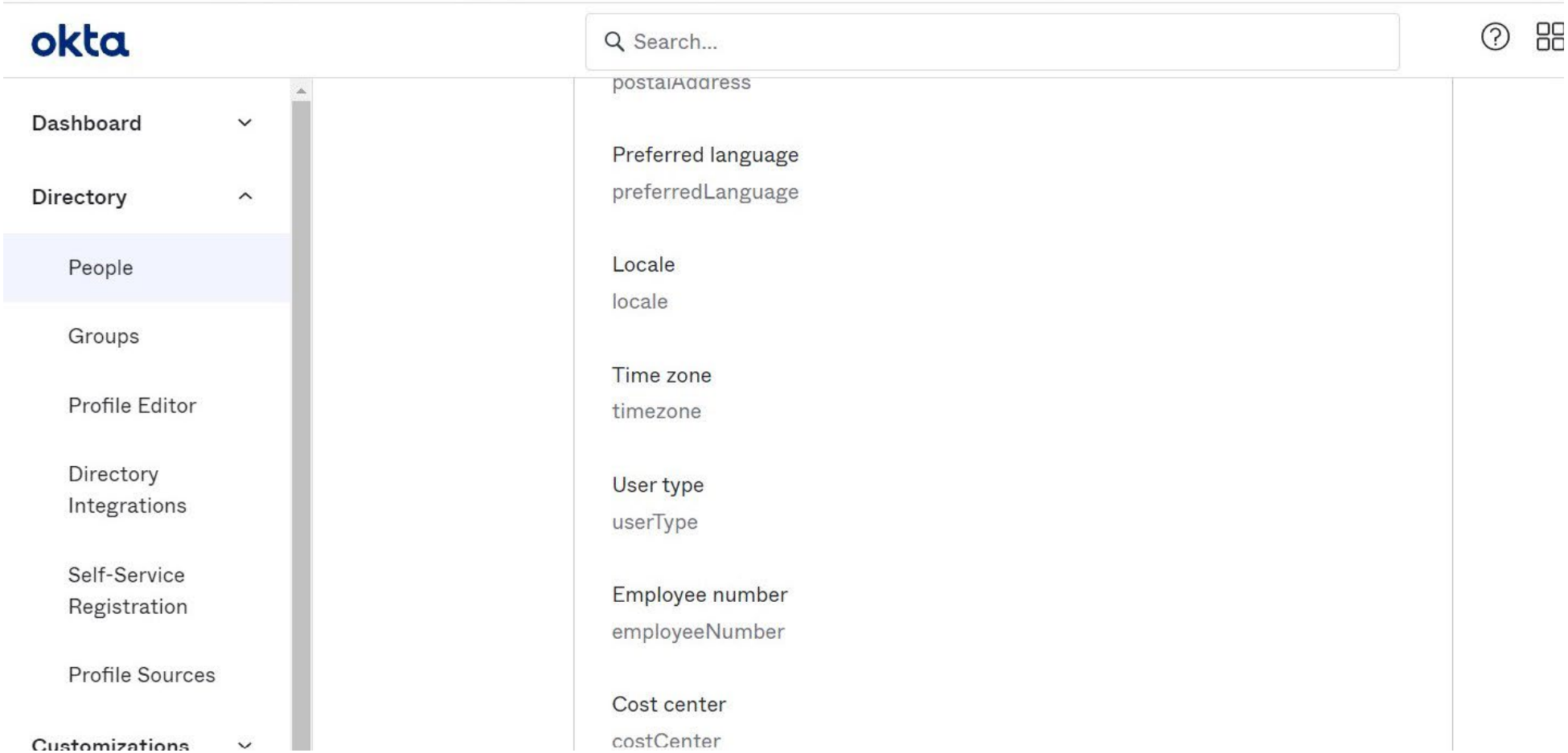


So if we go back to “people”, our admin user, profile.
We can see a user type of admin.

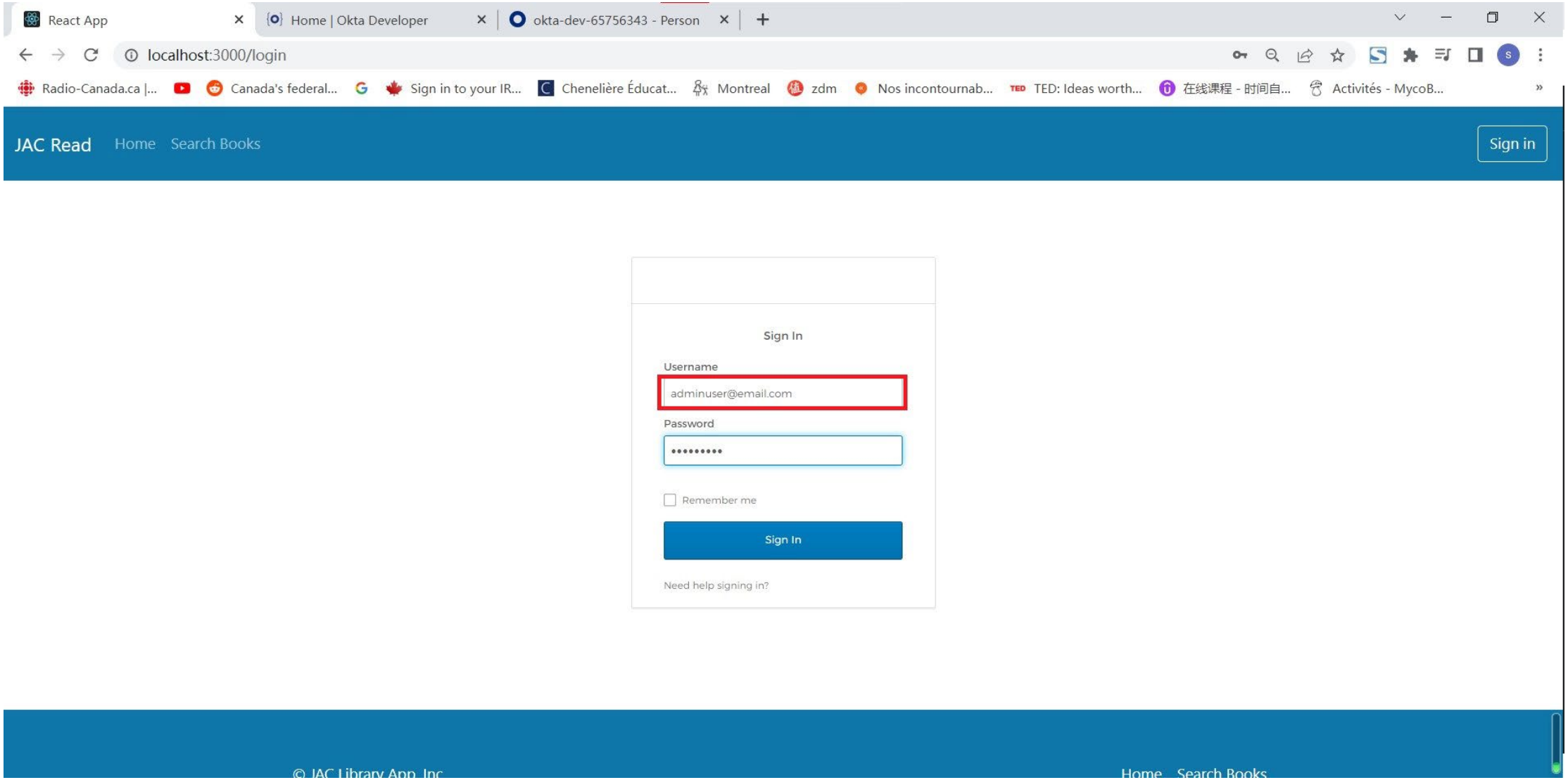
Go to our Test User2, “profile”,



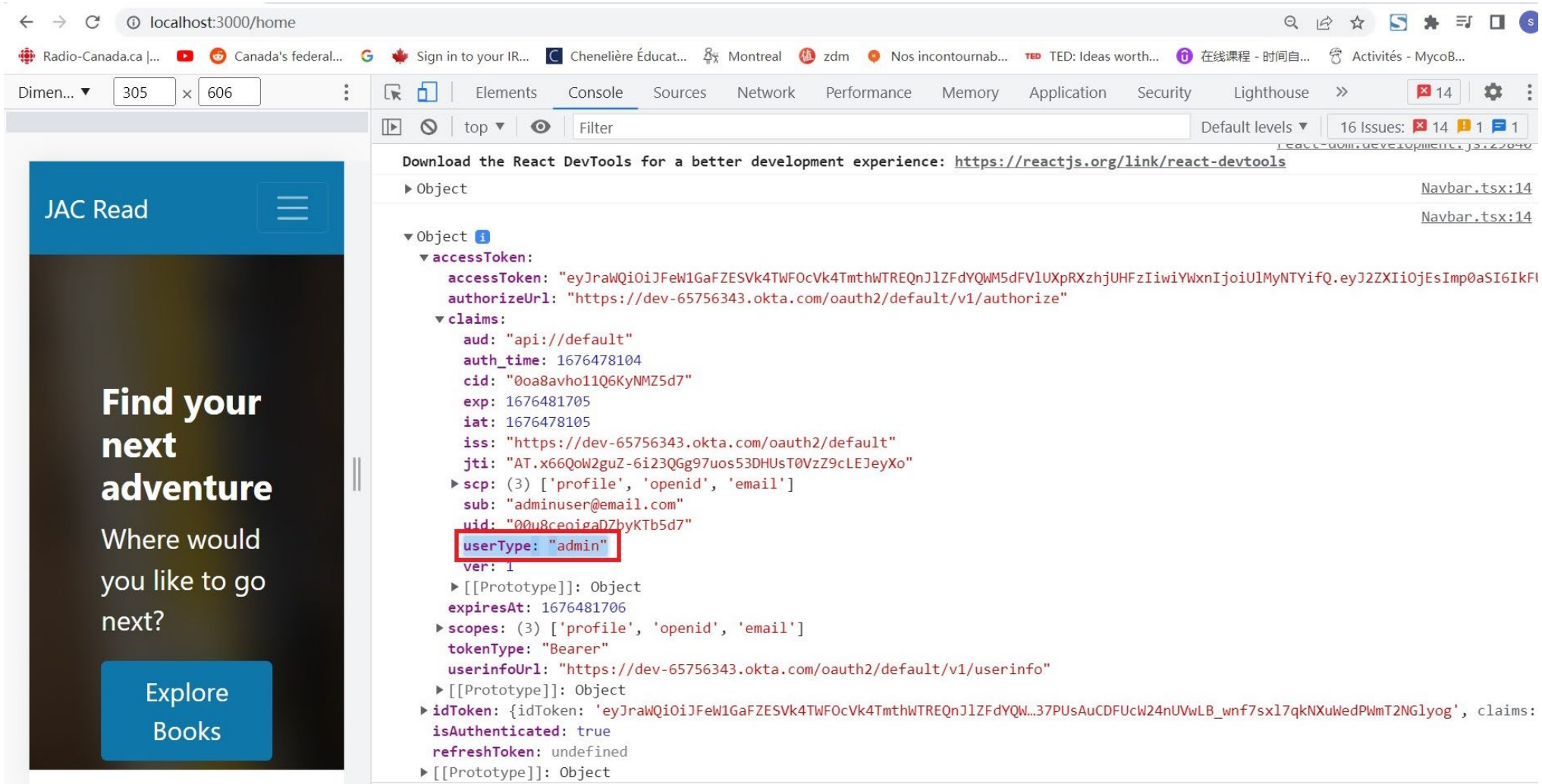
This user has nothing in admin.

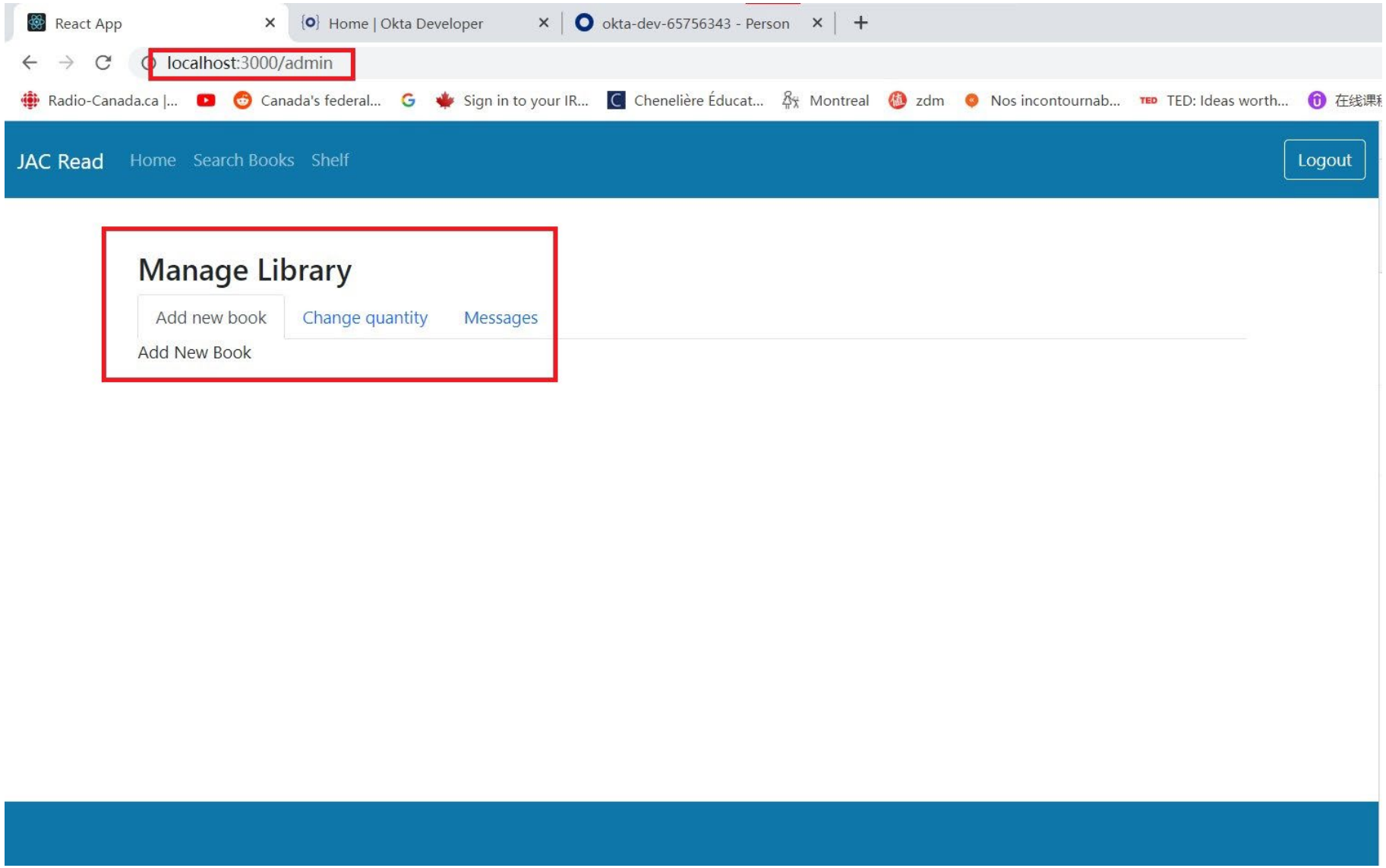


So now if we go to our application， sign in as our new admin user’s email.



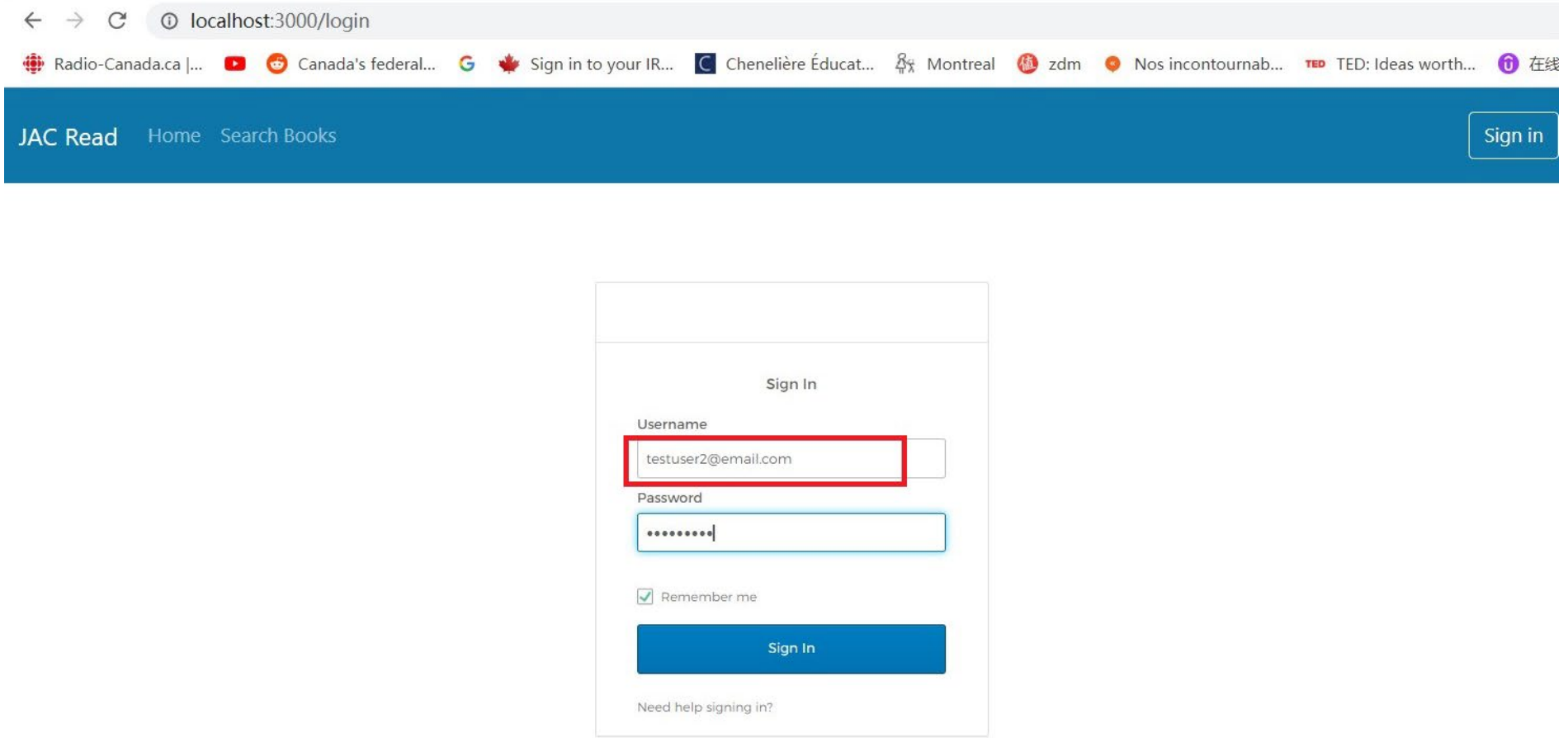
Right click, choose “Inspect”.We have our object for our admin user.
And now if we look at claims, we can see “userType:admin”.



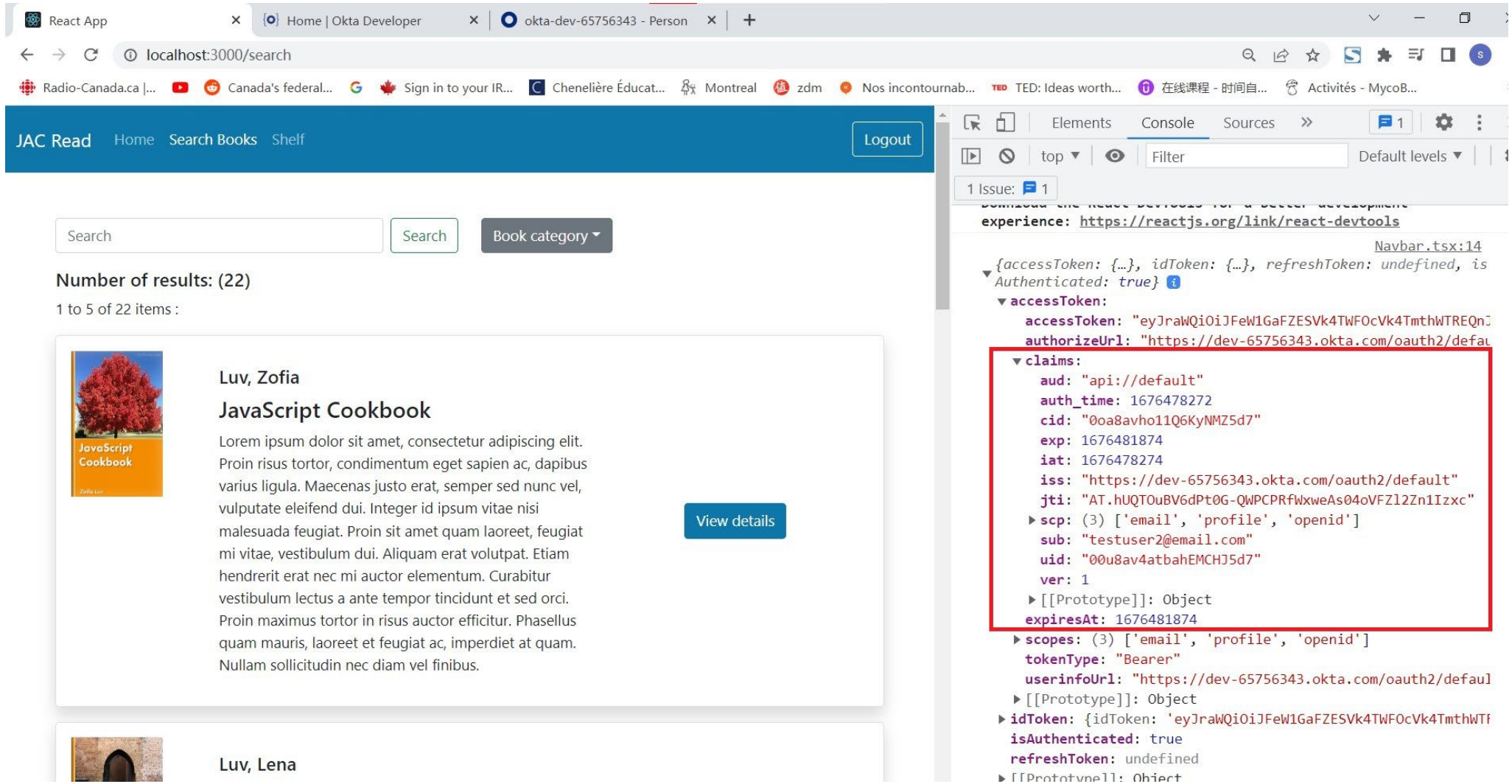


Now, with admin's log in, Visit http://localhost:3000/admin, it will show the manage library page.

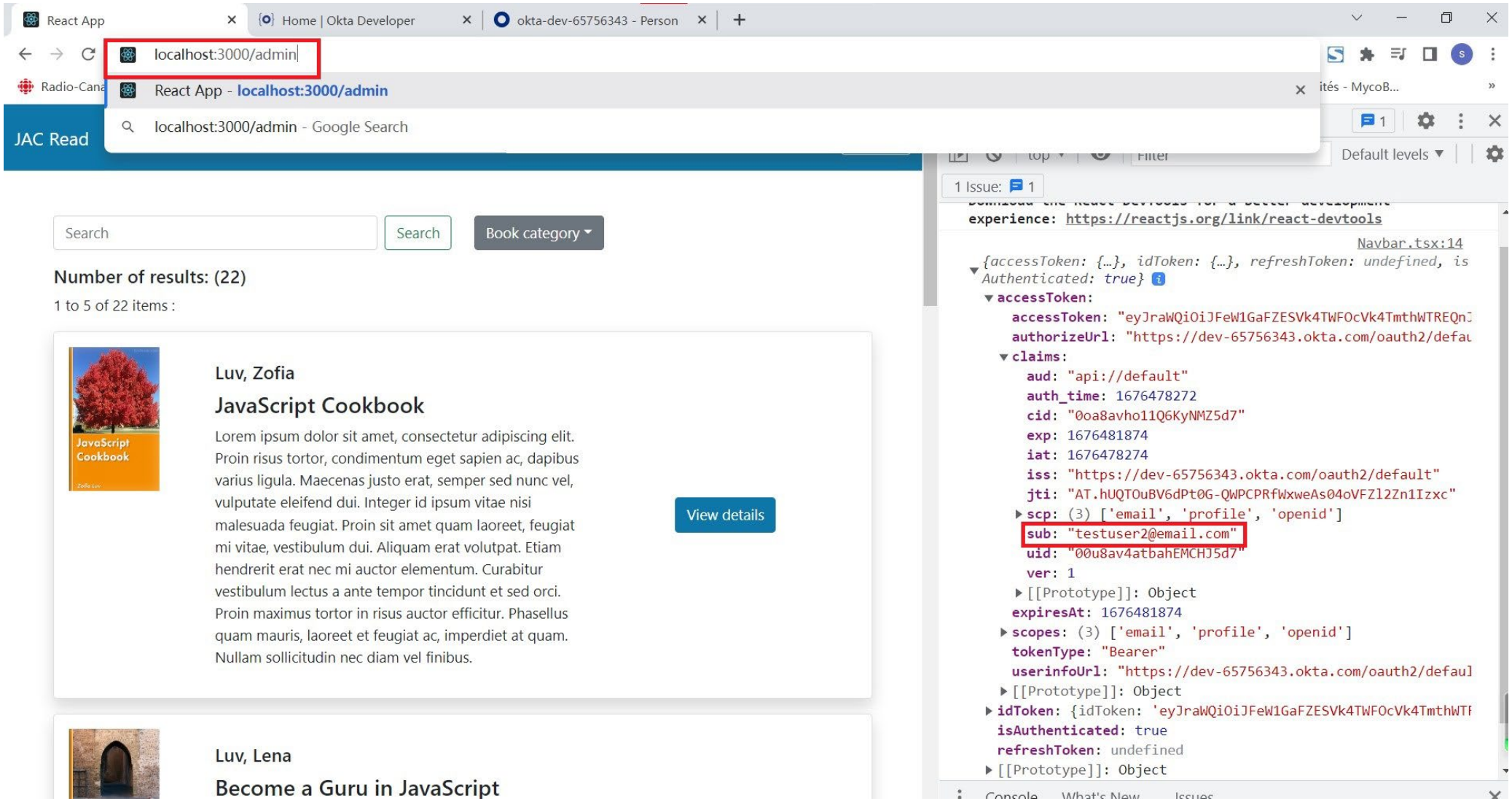
Now log in as “ testuser2 ”:



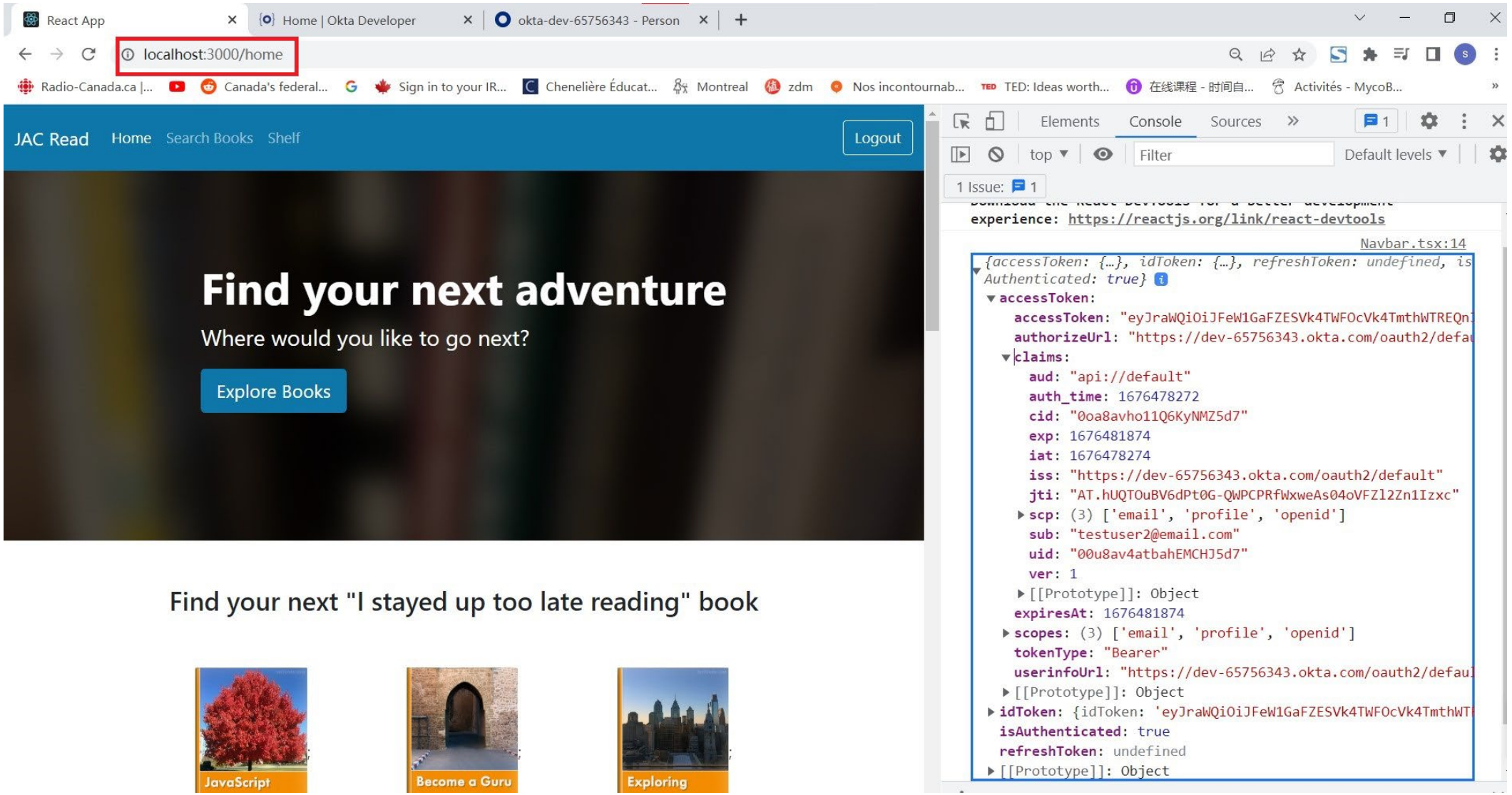
Open “Inspect”, and click on “claims”, you will see there is no user type.



Then input http://localhost:3000/admin,

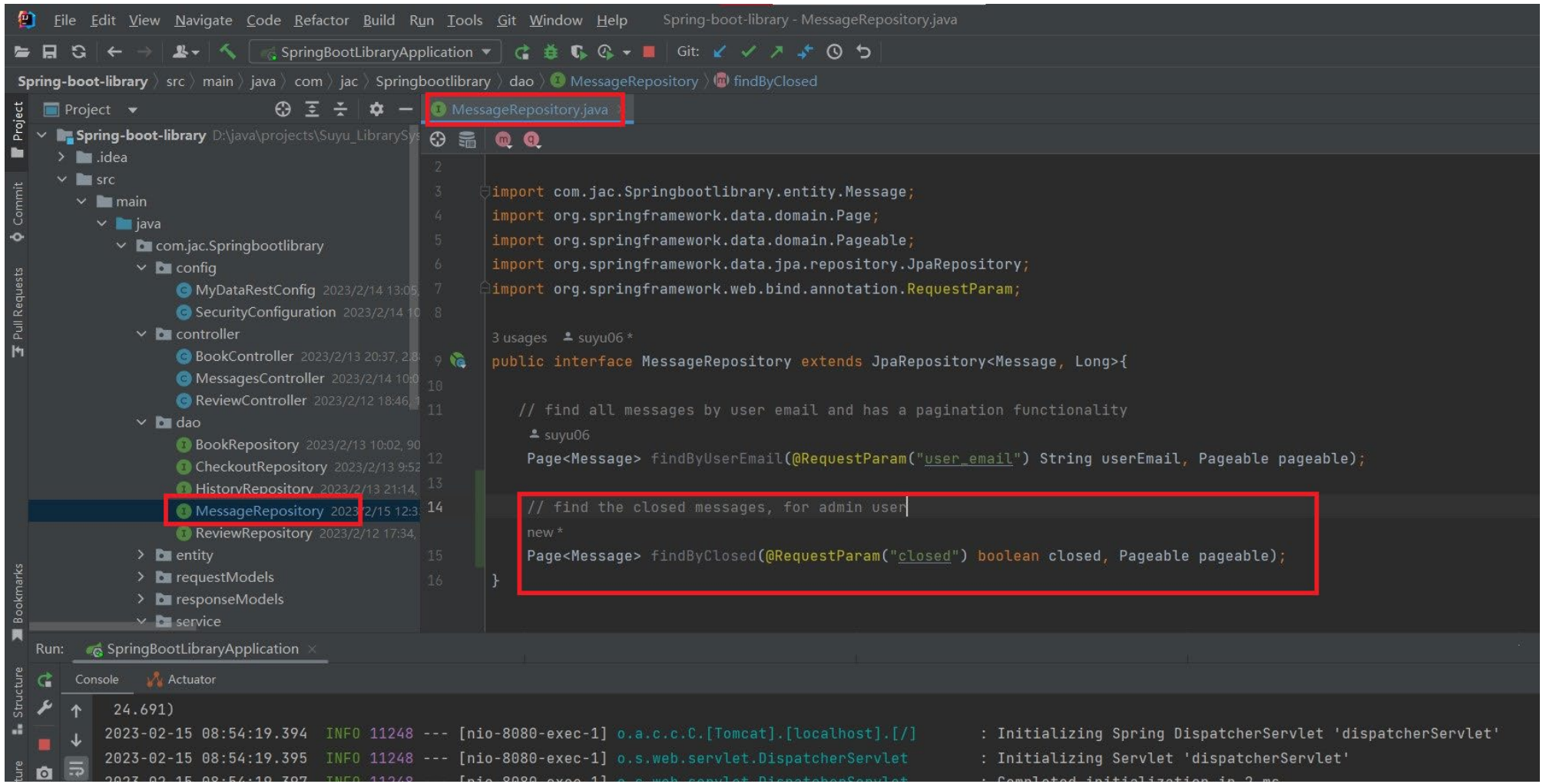


It will redirect us to the home page.

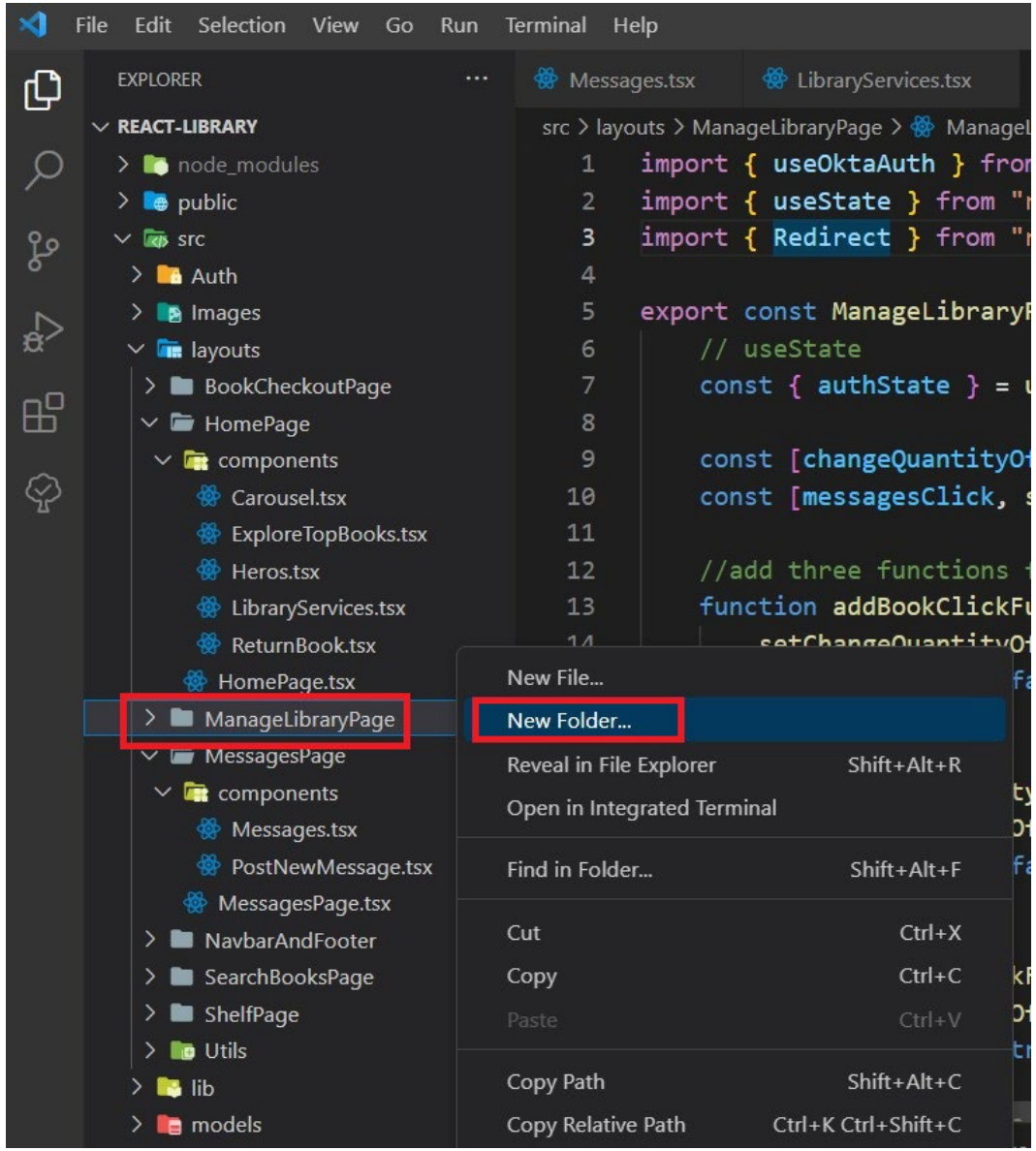


6. SPRINGBOOT OPEN TICKET ENDPOINT

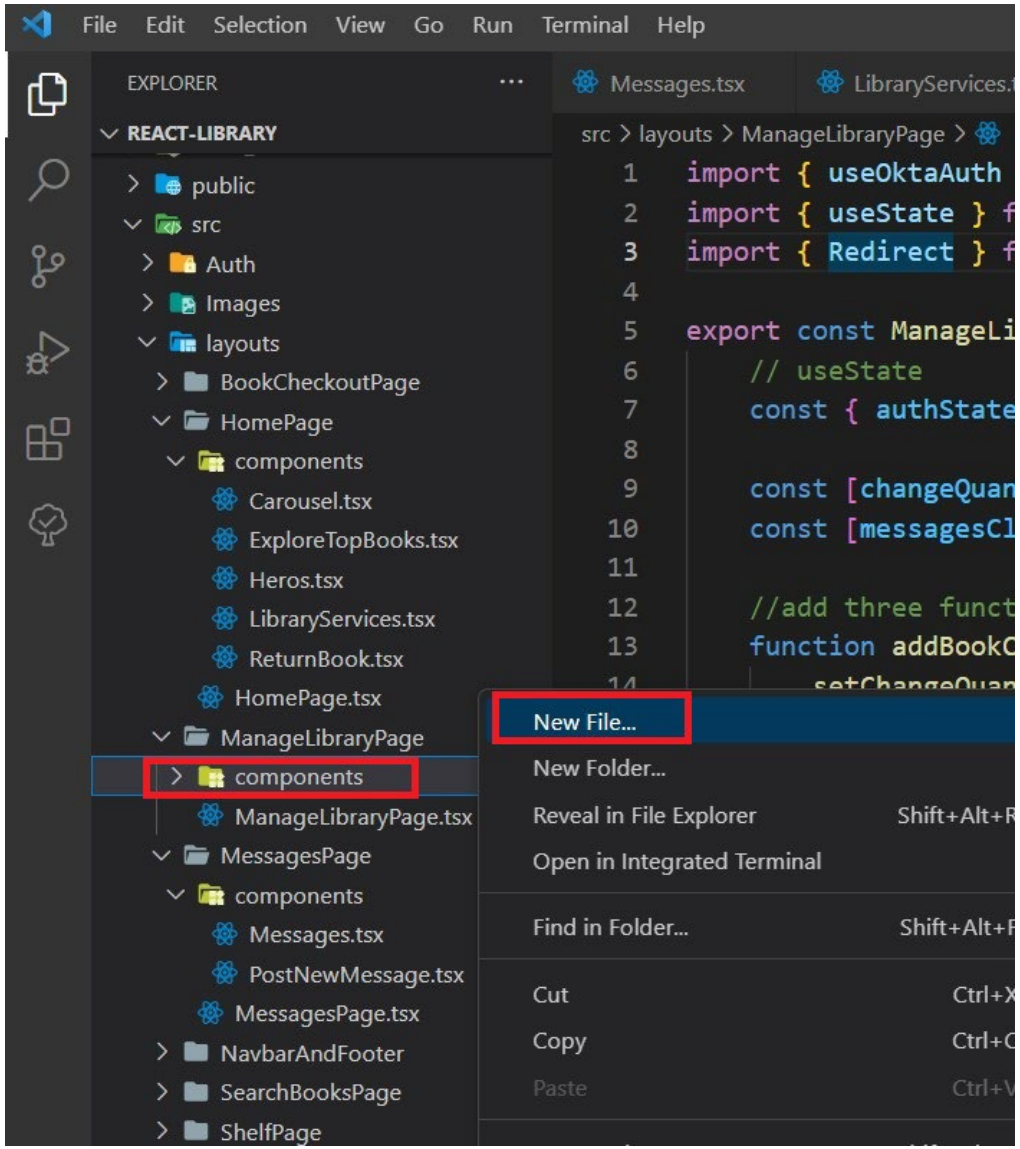
Go to dao package , in MessageRepository class ,add a findByClosed() interface function.



7. REACT ADMIN MESSAGES COMPONENT

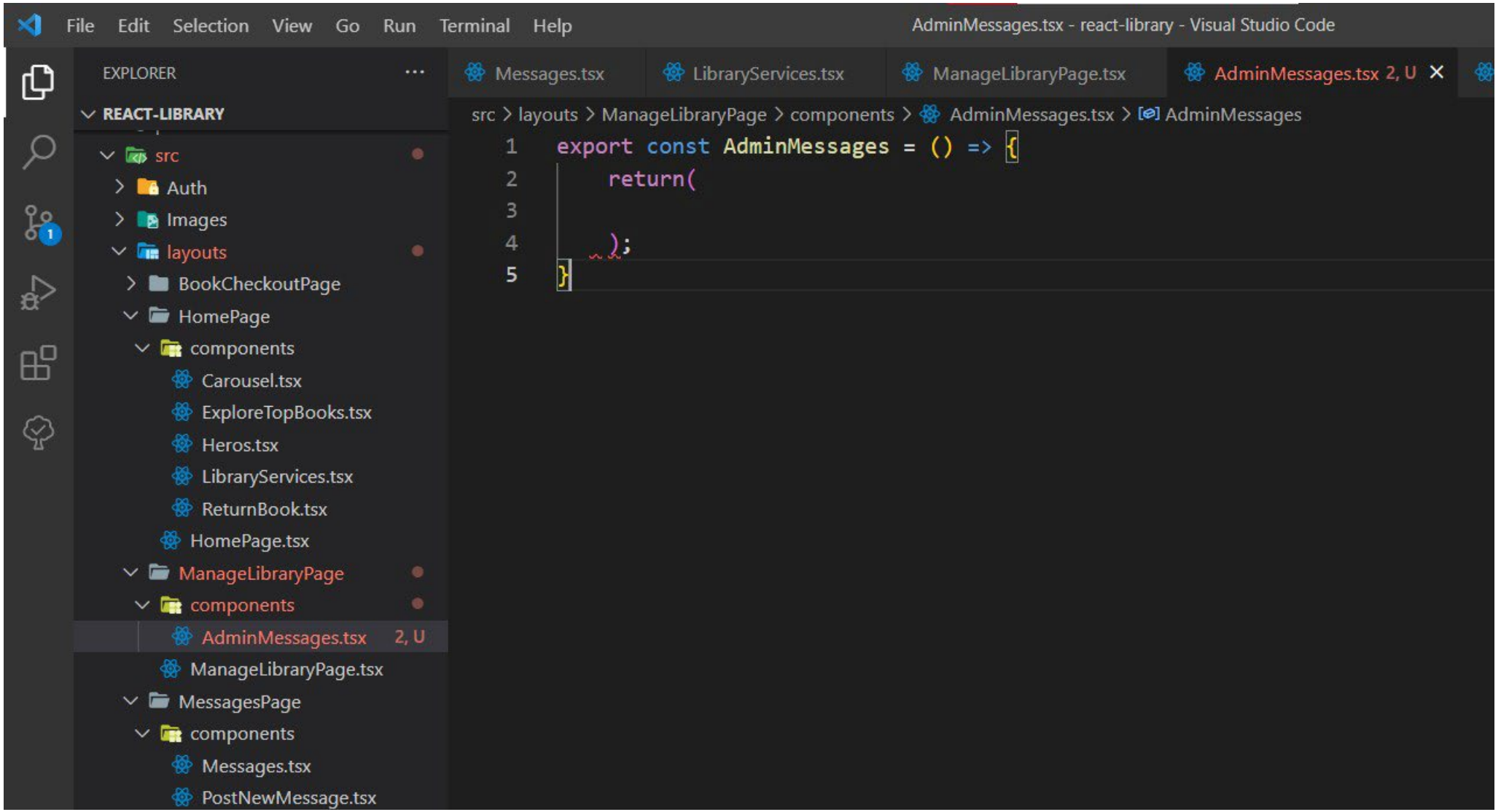


Step1, Let's go ahead and jump into our layouts, ManageLibraryPage,right click to create a new folder called Components.

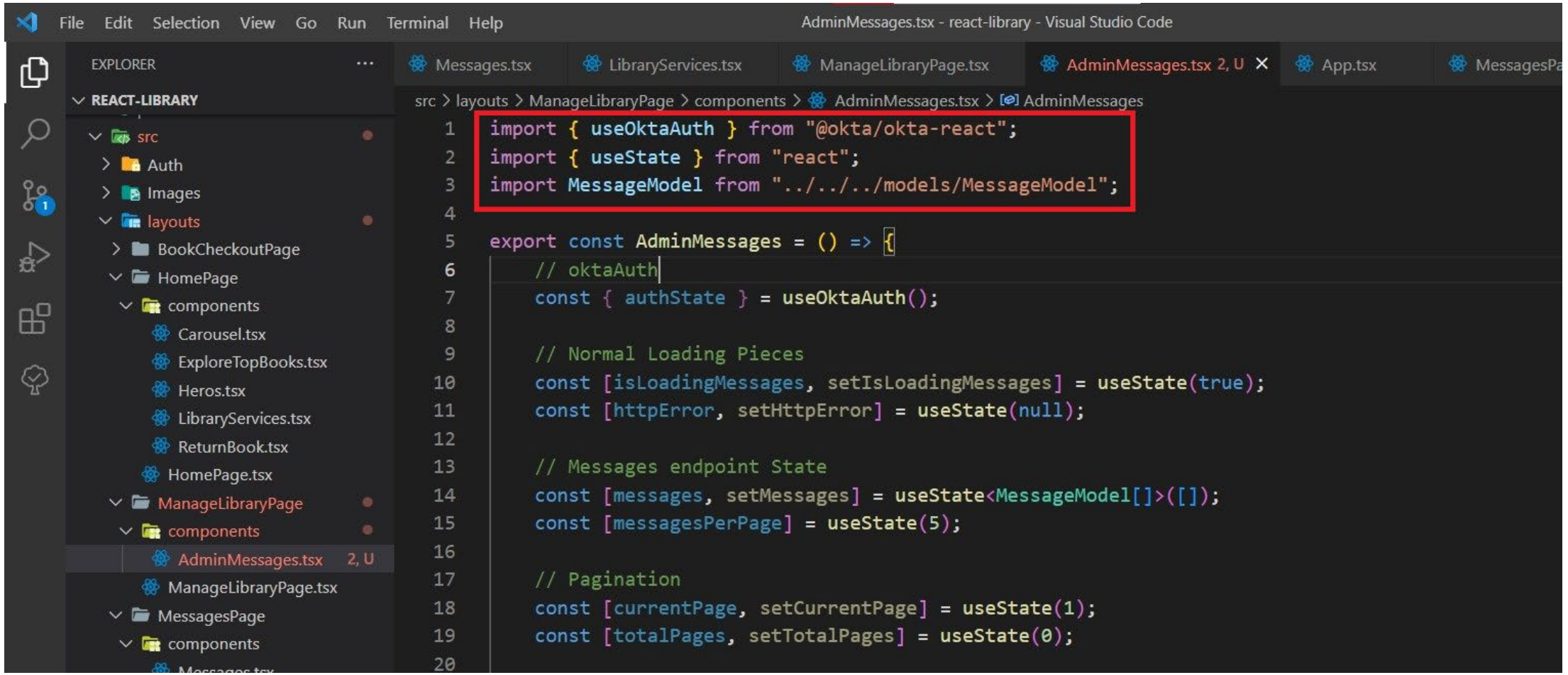


Step 2, And inside the components, reate a new file AdminMessages.tsx.

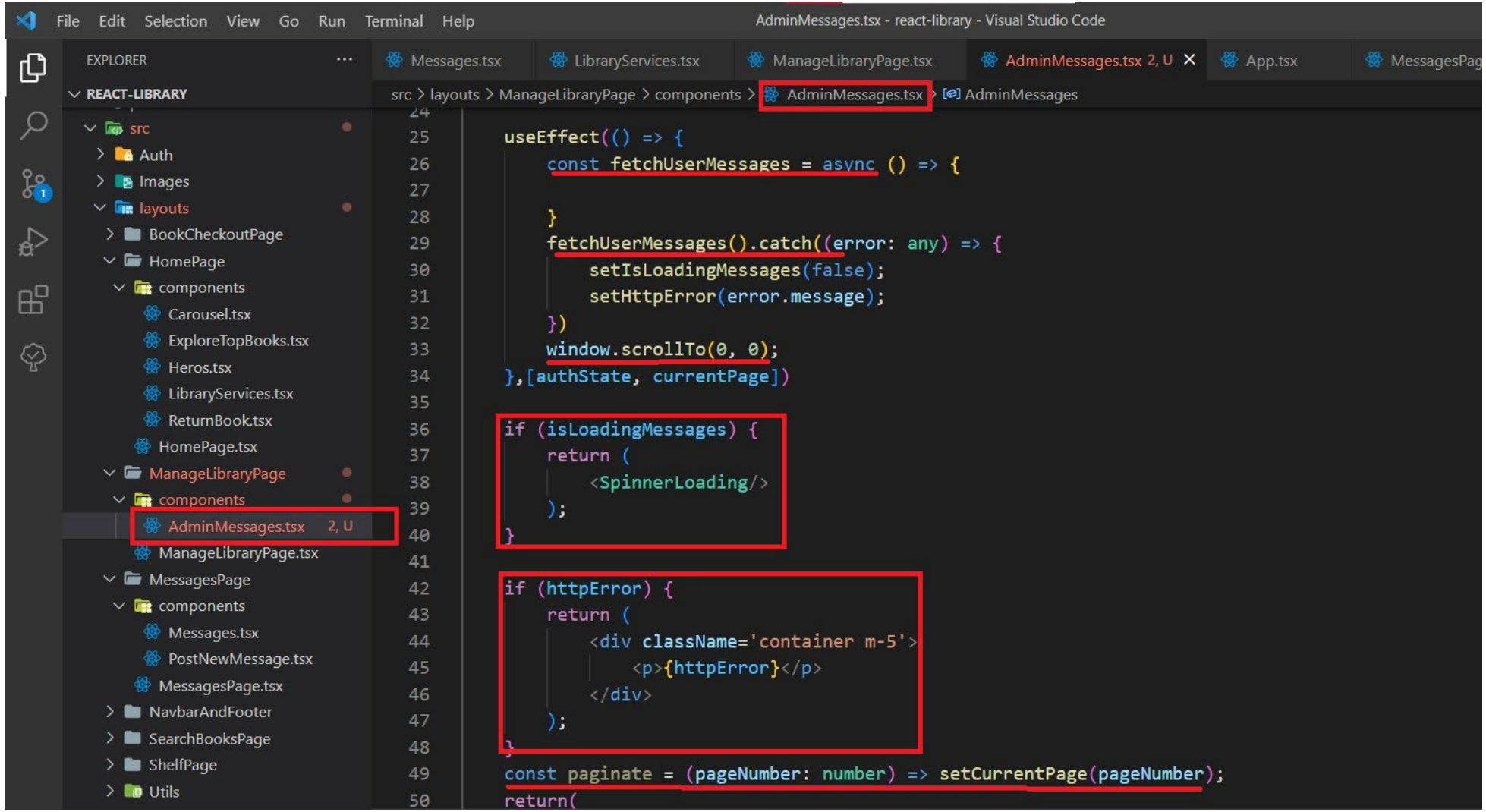
Step3, as always,create export constant AdminMessages() structure;



Step4， add useState and import useOktaAuth, useState and MessageModel.



Step5, as usual , create useEffect structure.



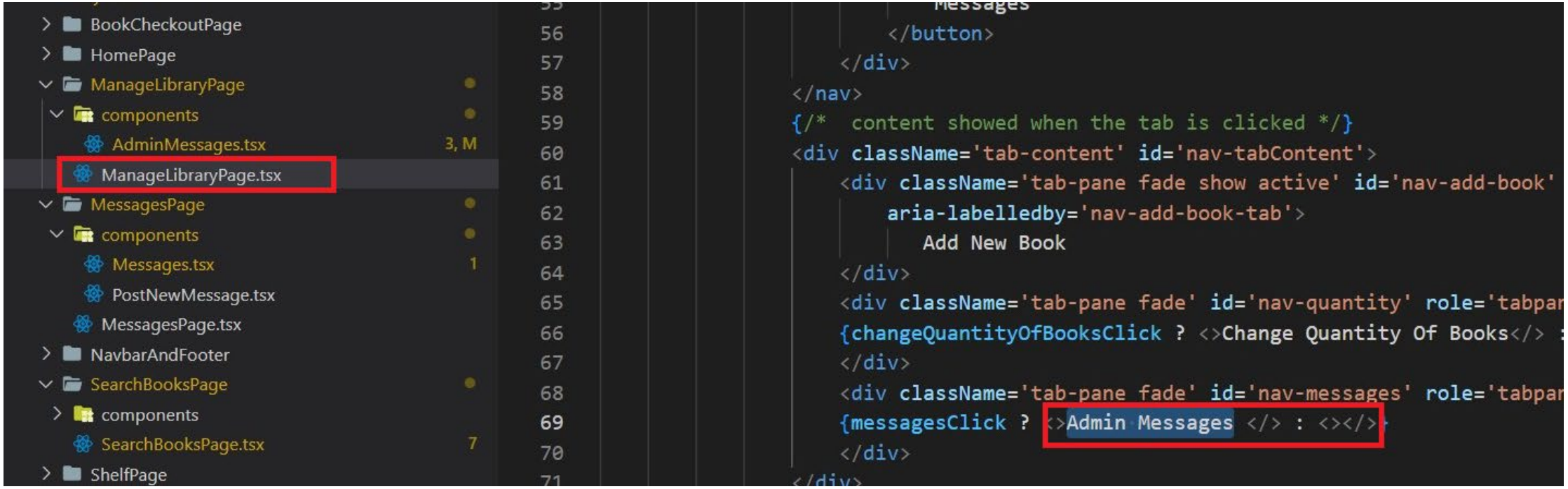
```
useEffect(() => {
    const fetchUserMessages = async () => {
        // only if user is authenticated
        if (authState && authState.isAuthenticated) {
            const url = `http://localhost:8080/api/messages/search/findByClosed
            /?closed=false&page=${currentPage - 1}&size=${messagesPerPage}`;
            const requestOptions = {
                method: 'GET',
                headers: {
                    Authorization: `Bearer ${authState.accessToken?.accessToken}`,
                    'Content-Type': 'application/json'
                }
            };
            //fetch() to get the data from the url
            const messagesResponse = await fetch(url, requestOptions);
            //meets error
            if (!messagesResponse.ok) {
                throw new Error('Something went wrong!');
            }
            // convert to JSON
            const messagesResponseJson = await messagesResponse.json();
            // save messages and total page number
            setMessages(messagesResponseJson._embedded.messages);
            setTotalPages(messagesResponseJson.page.totalPages);
        }
        //finish loadin process
        setIsLoadingMessages(false);
```

Step 6, fill in useEffect with async function.

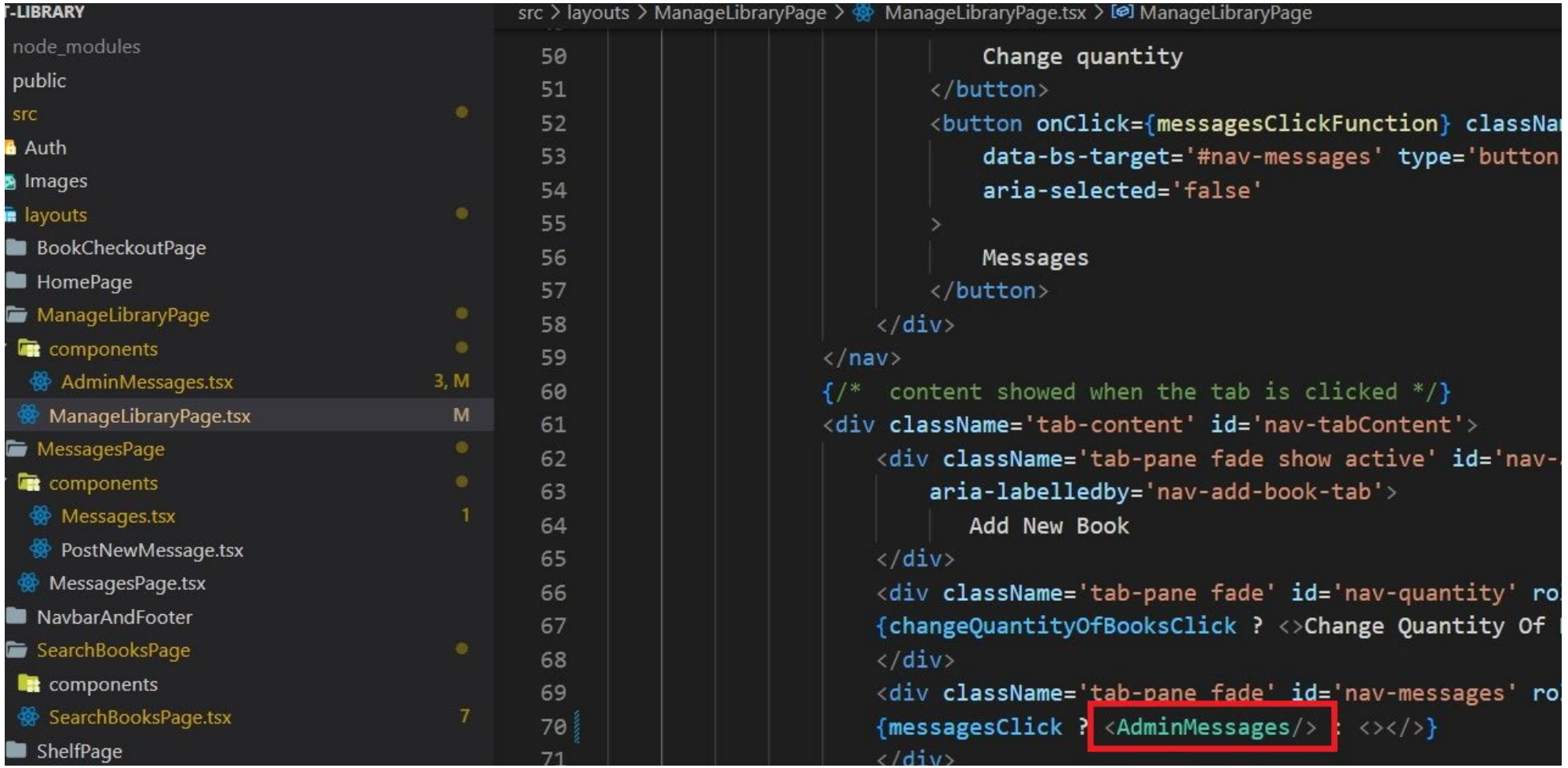
Step 7, fill in return part with Html/CSS code.

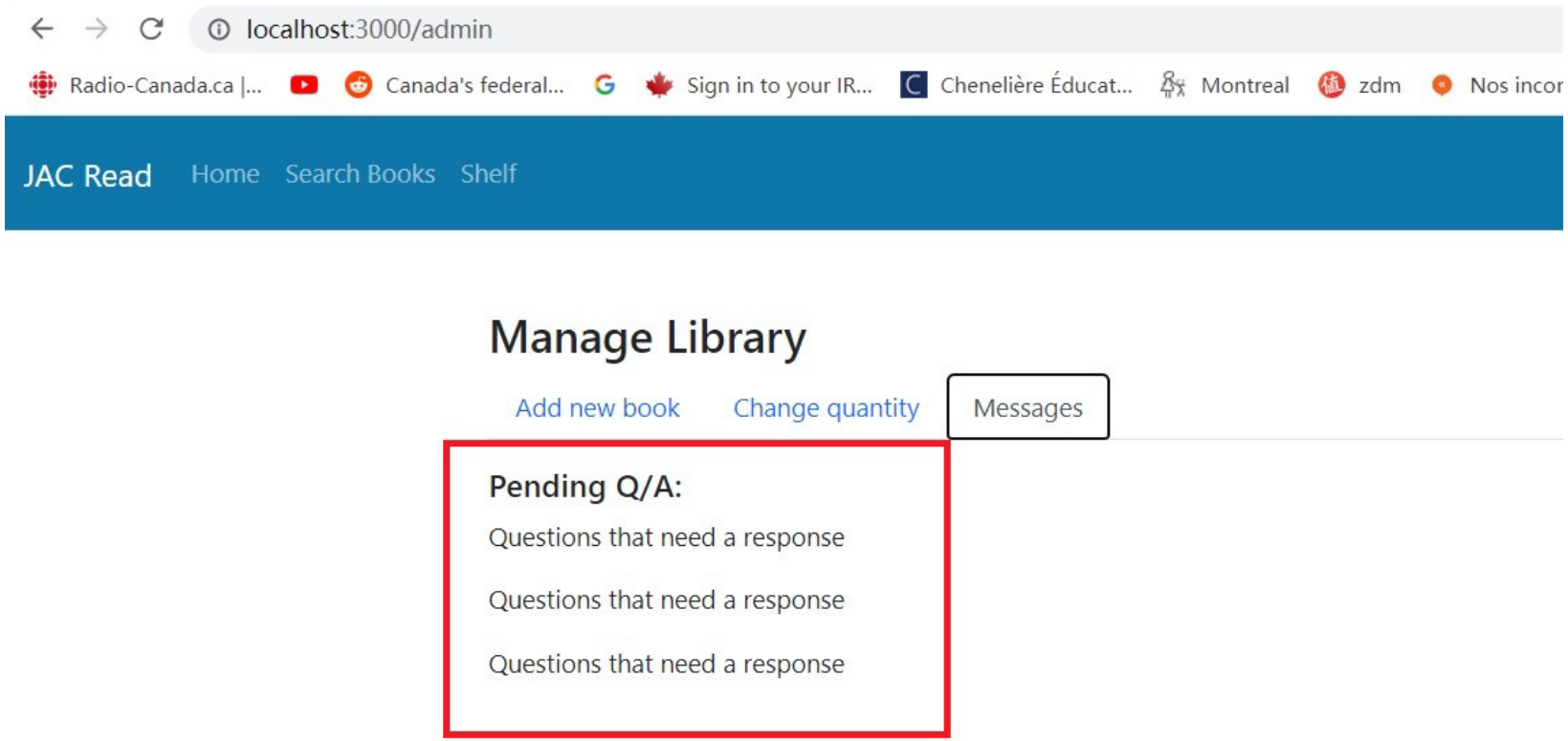
```
return(
    <div className='mt-3'>
        {/* if there is messages: */}
        {messages.length > 0 ?
            <>
                <h5>Pending Q/A: </h5>
                {messages.map(message => (
                    <p>Questions that need a response</p>
                ))}
            </>
            :
            //no messages
            <h5>No pending Q/A</h5>
        }
        {totalPages > 1 && <Pagination currentPage={currentPage} totalPages={totalPages} paginate={paginate}/>}
    </div>
);
```

Step 8, Open up our ManageLibraryPage, Scroll to the bottom.
And right here in our third tab, we can see “messages”.
Let's remove these words with new AdminMessages component.



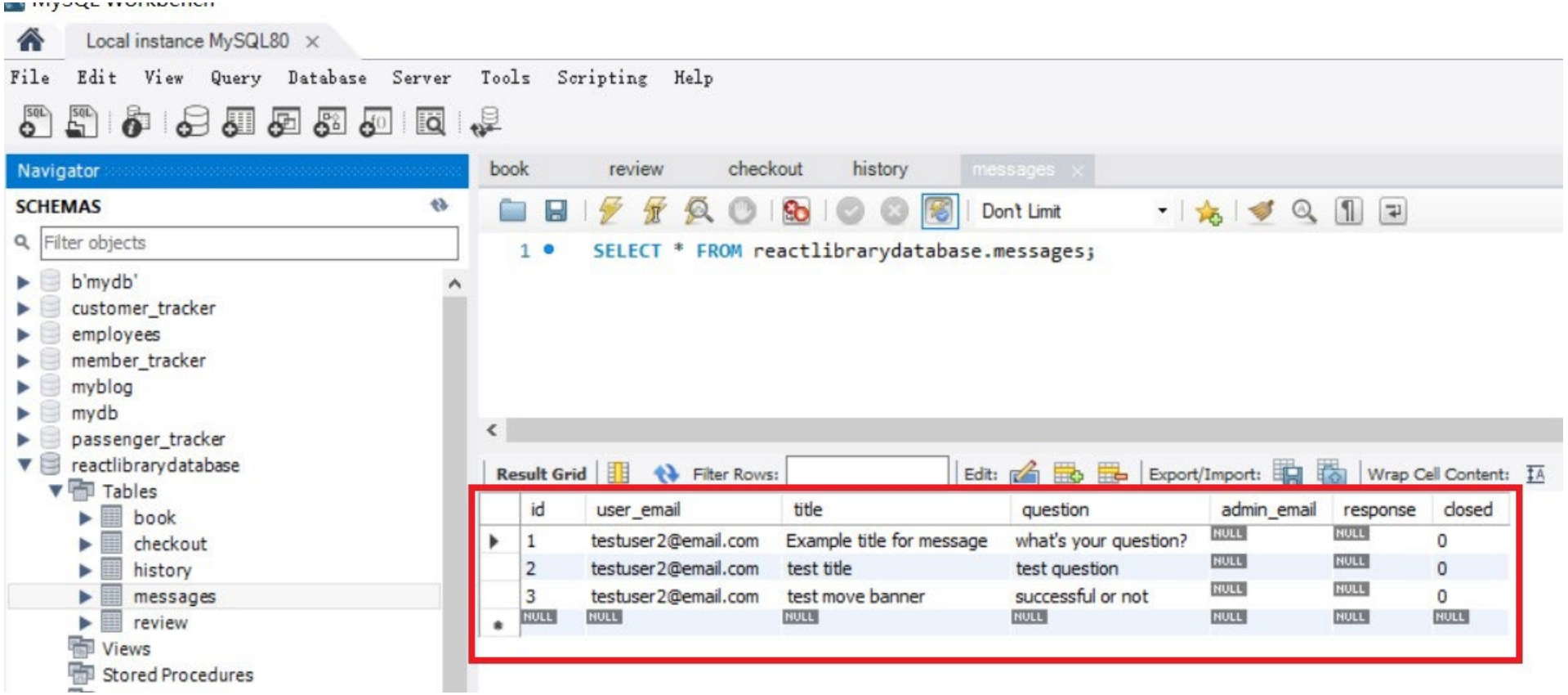
Like this below：



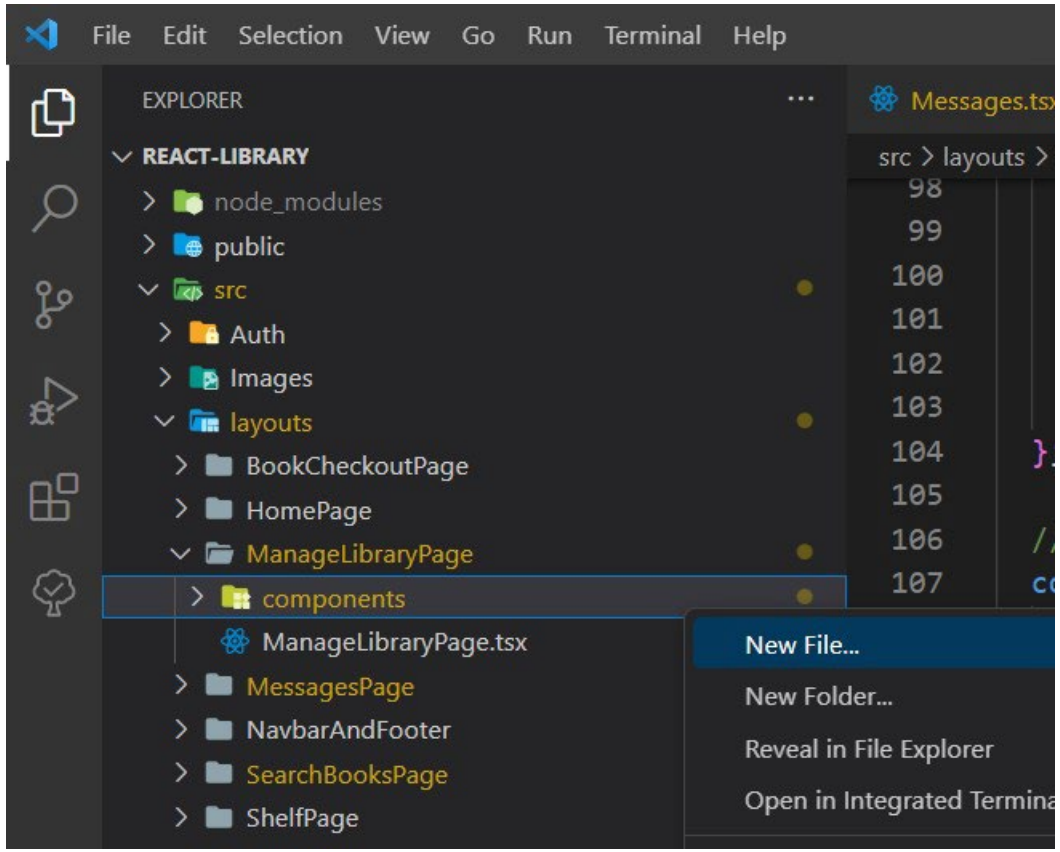


With admin user account logged in, click on messages tab, we can see there are three “questions needs to a response”.

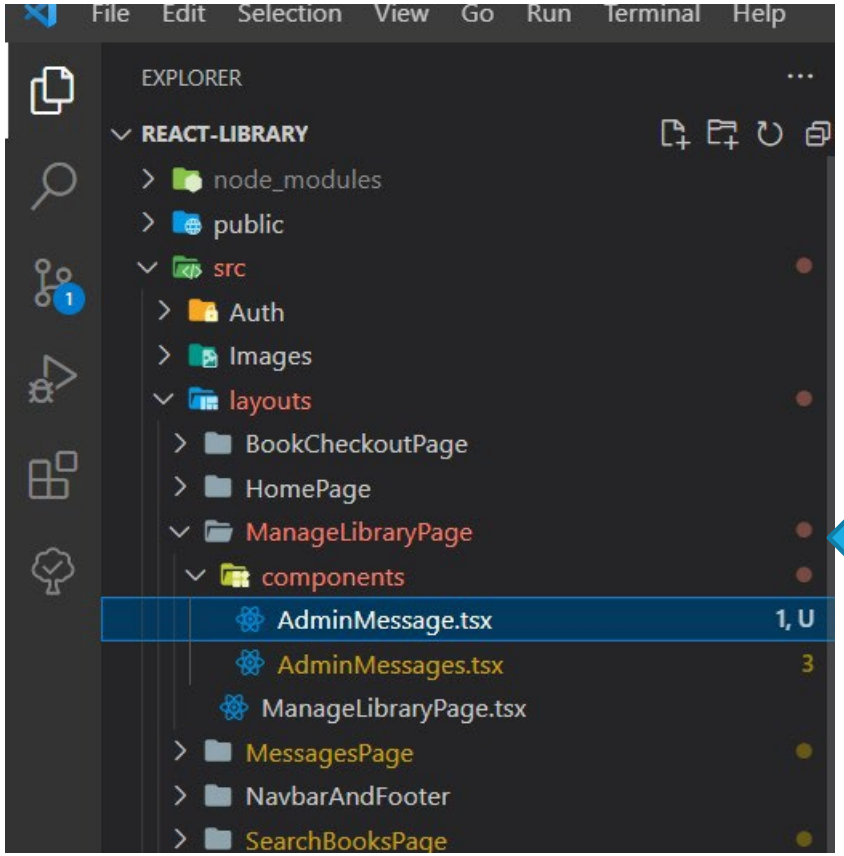
Check our table “messages”, you can see there are three questions without response.



8.REACT ADMIN MESSAGES COMPONENT

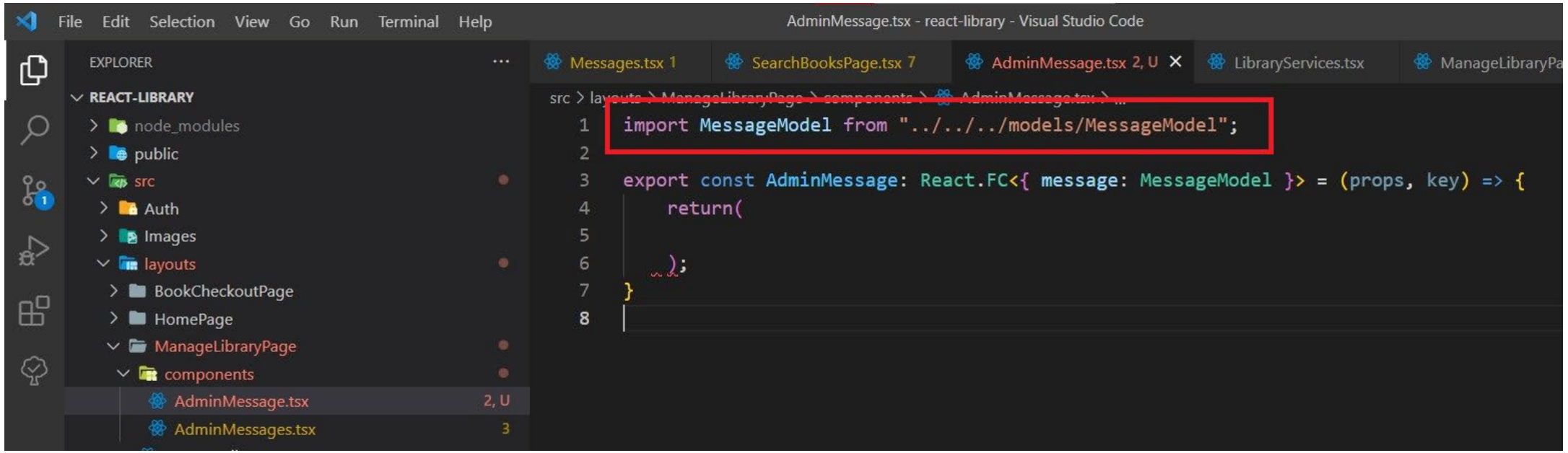


Step1 Let's go ahead and right click on our components within our ManageLibraryPage and say new file.

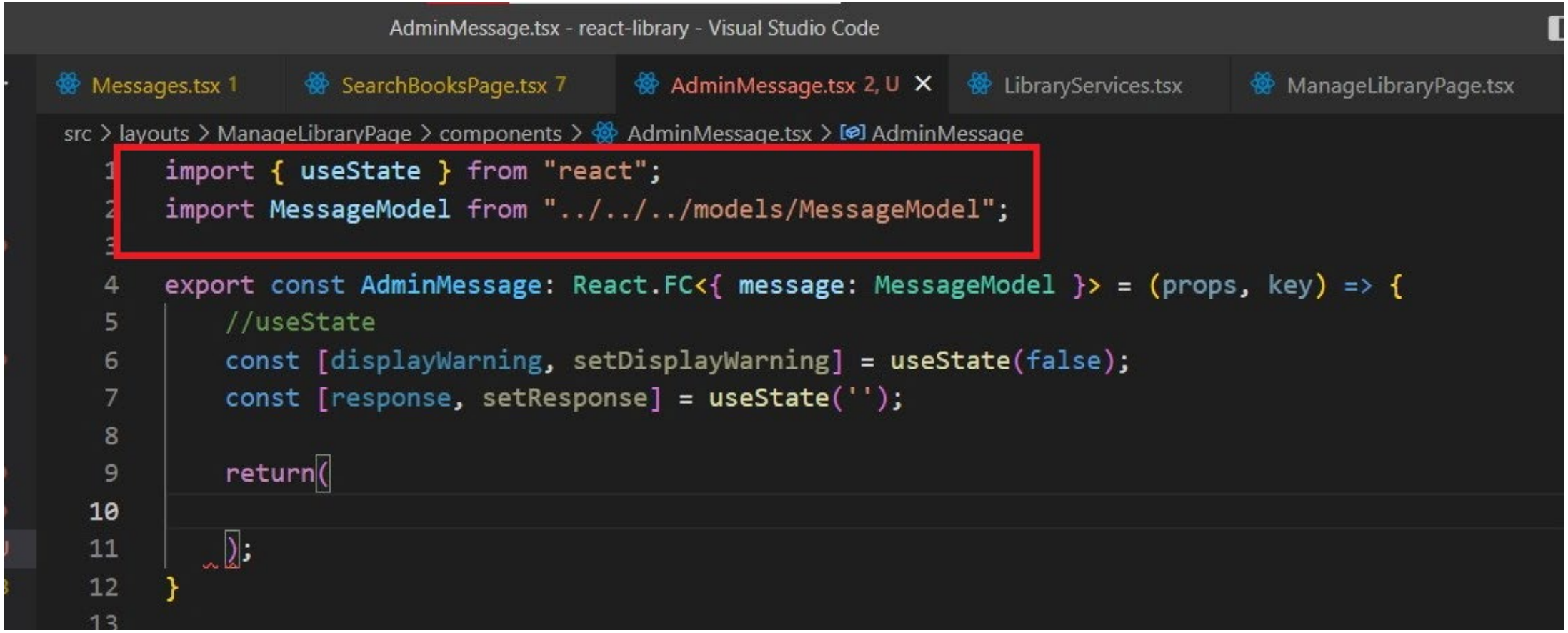


create our component AdminMessage.tsx.

Step 2, create export constant function ,take in a message of type message model as a props.

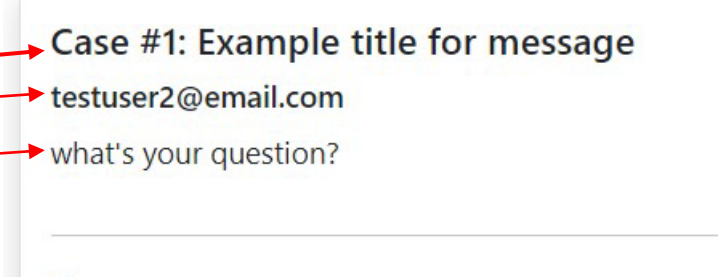


Step3, create useState.

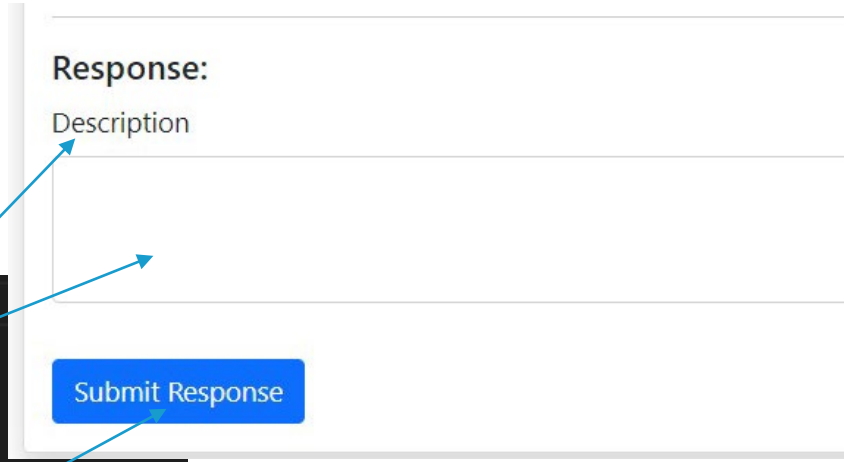


Step 4 return part : html/CSS part

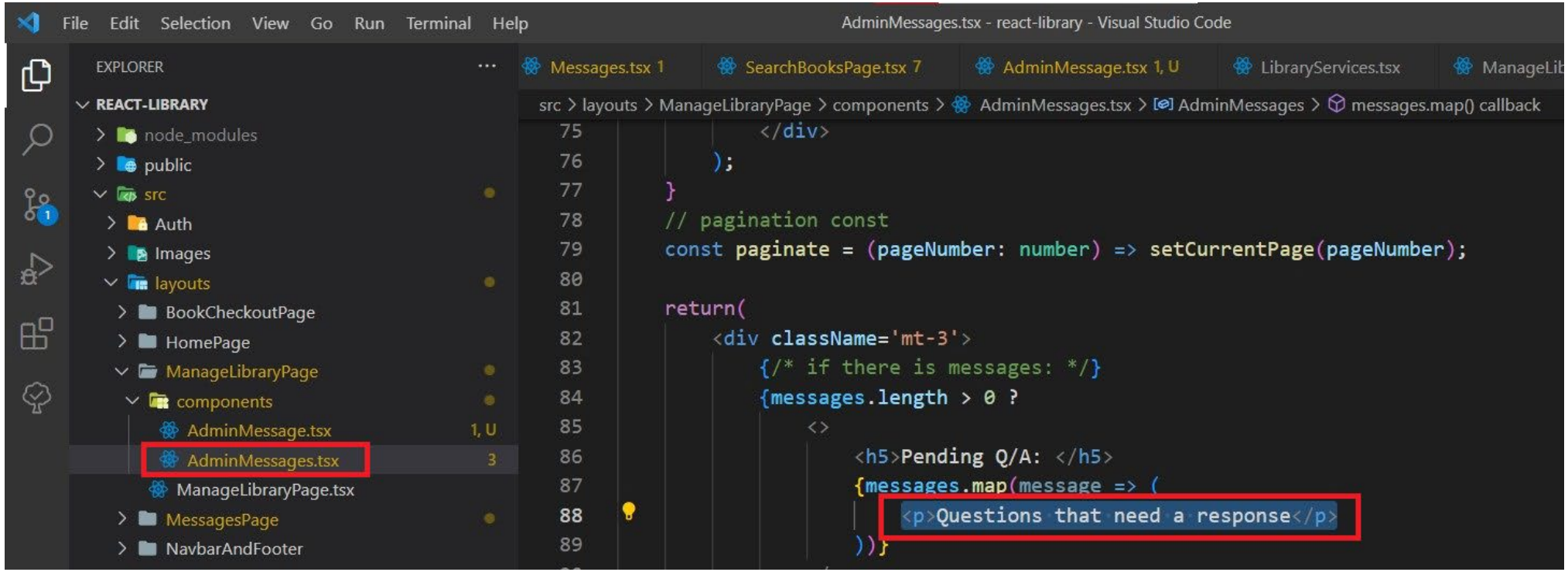
```
// html/CSS part
return(
    // go through each message by its id
    <div key={props.message.id}>
        <div className='card mt-2 shadow p-3 bg-body rounded'>
            <h5>Case #{props.message.id}: {props.message.title}</h5>
            <h6>{props.message.userEmail}</h6>
            <p>{props.message.question}</p>
            <hr/>
```



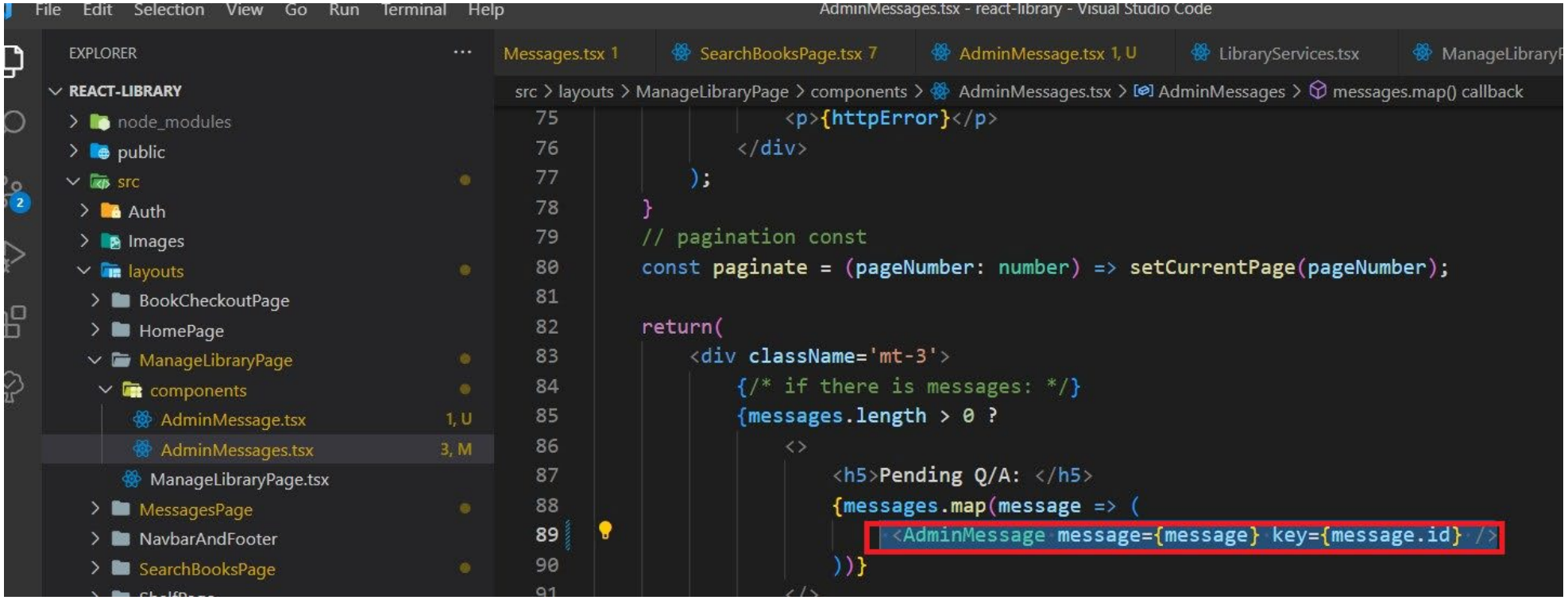
```jsx
<div>
    <h5>Response: </h5>
    <form action="PUT">
        {displayWarning &&
            <div className='alert alert-danger' role='alert'>
                All fields must be filled out.
            </div>
        }
        <div className='col-md-12 mb-3'>
            <label className='form-label'> Description </label>
            <textarea className='form-control' id='exampleFormControlTextarea1' rows={3}
                onChange={e => setResponse(e.target.value)} value={response}></textarea>
        </div>
        <div>
            <button type='button' className='btn btn-primary mt-3'>
                Submit Response
            </button>
        </div>
    </form>
</div>
```

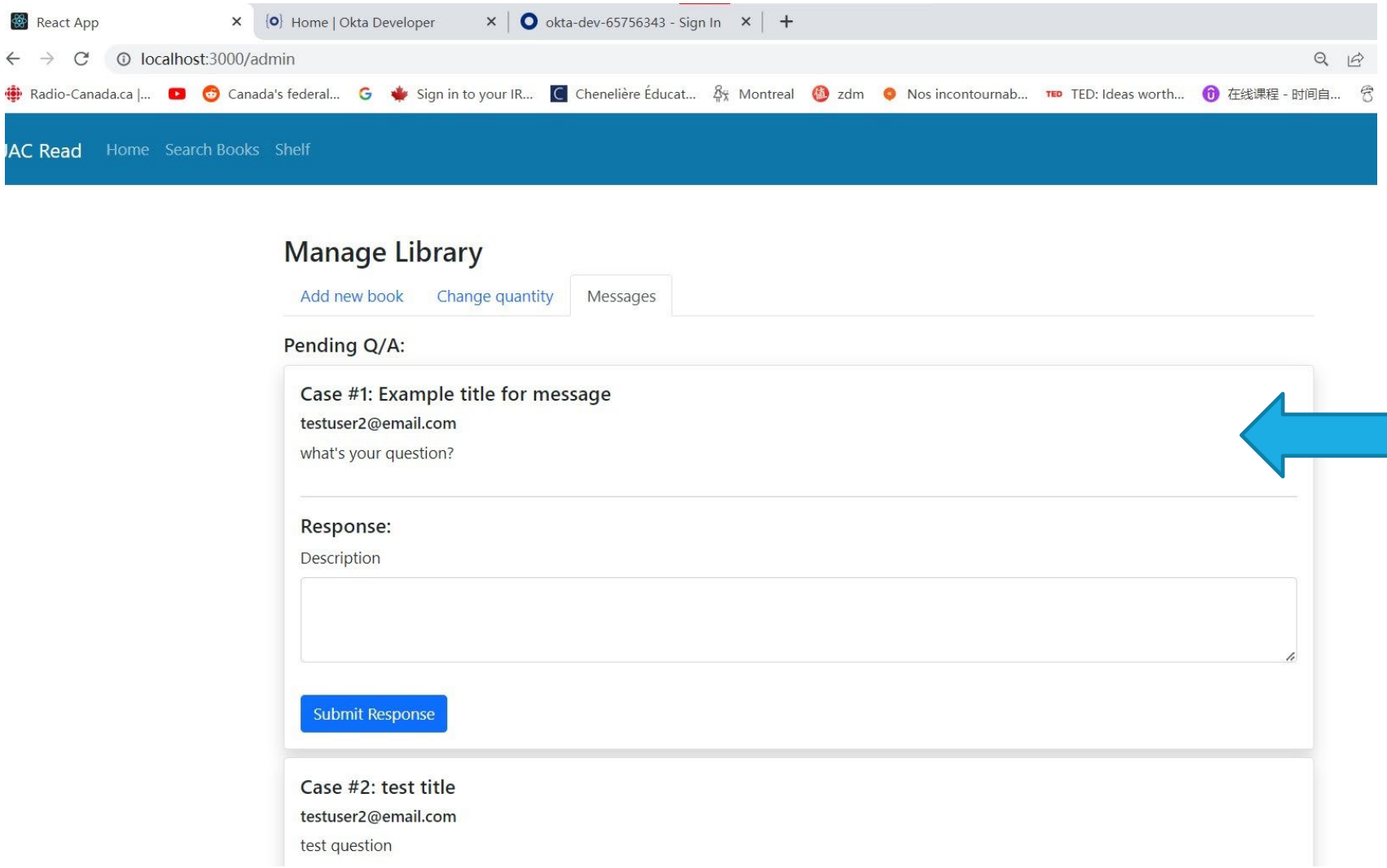


Step 5 Open up our AdminMessages.tsx, change AdminMessage p tag into our component.



Then the AdminMessage will be imported automatically.

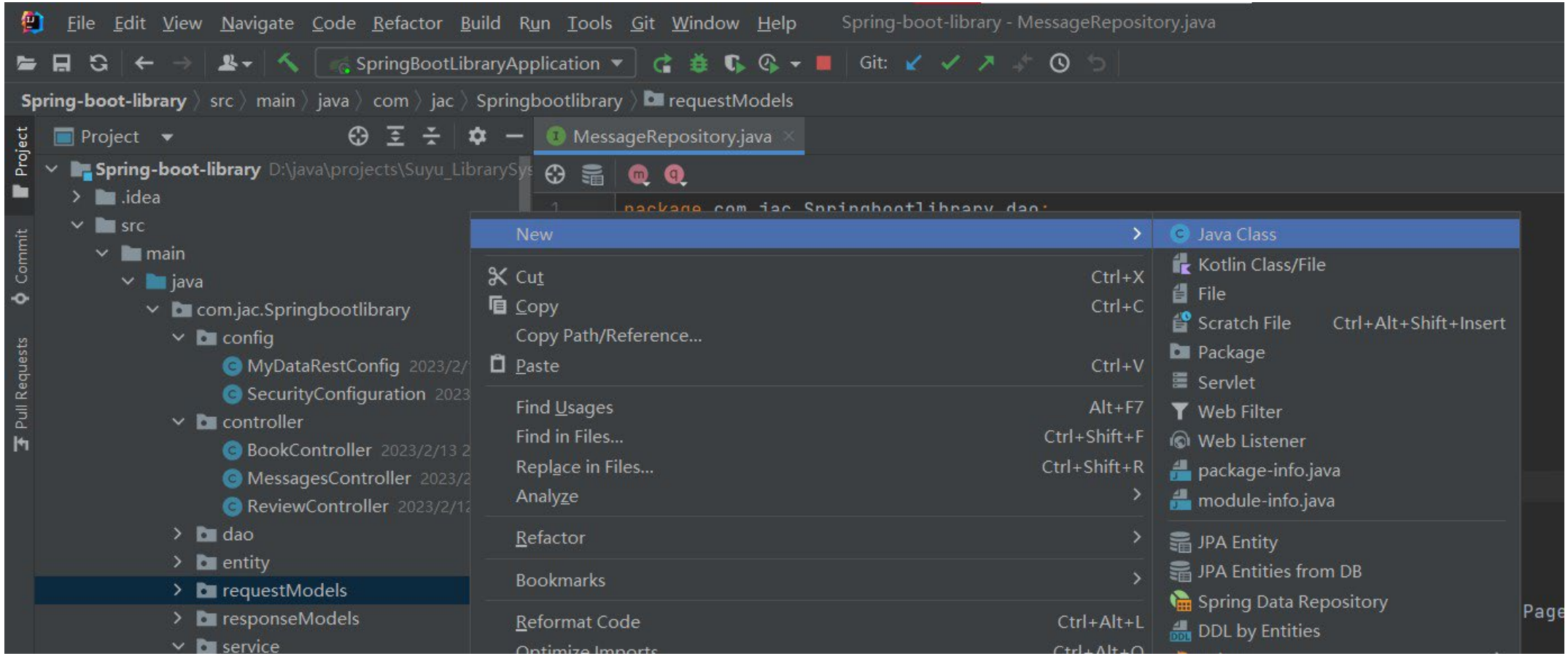




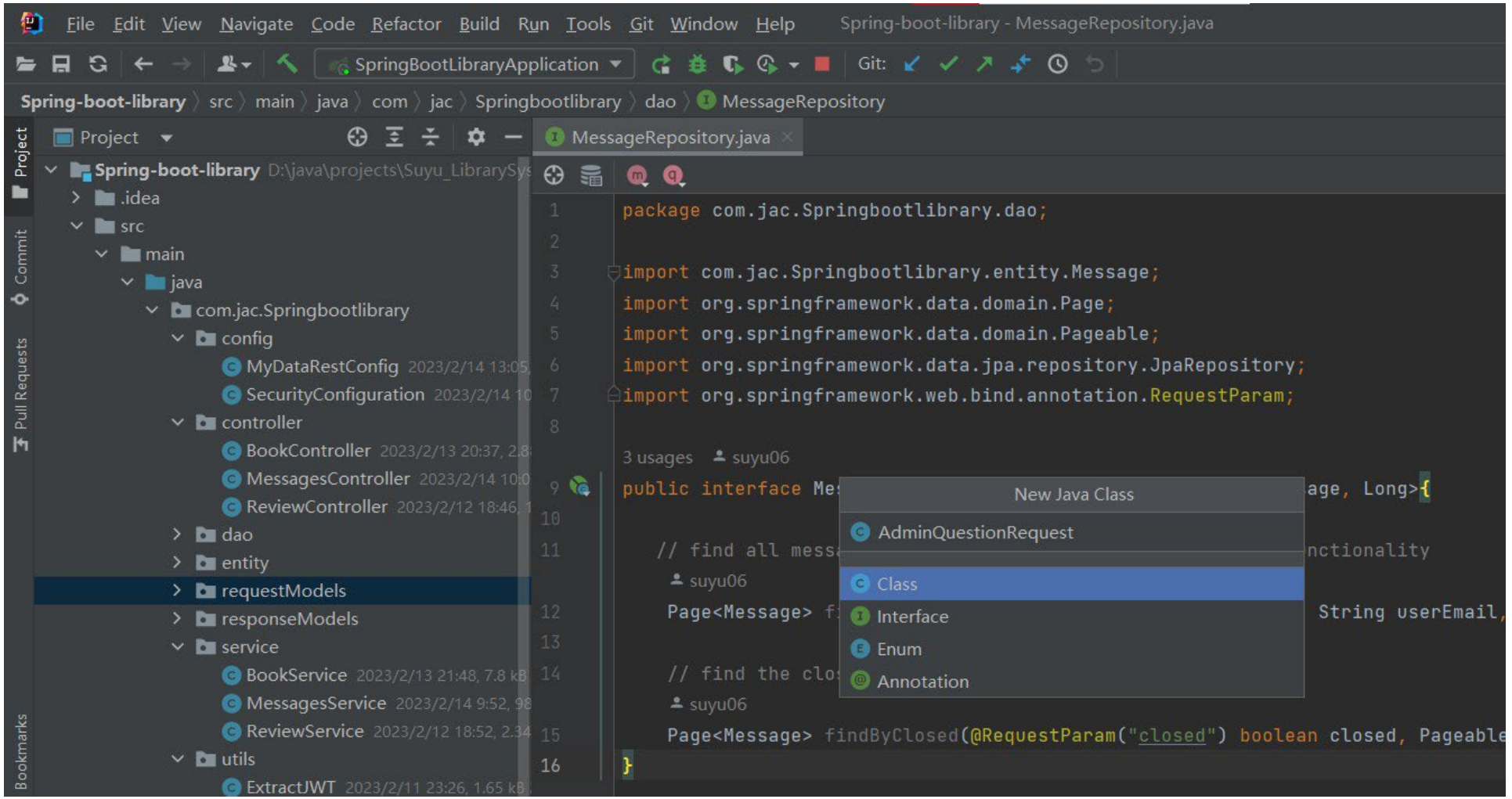
With admin user account logged in, click on messages tab, we can get all the questions that need a response.

9. SPRINGBOOT ADMIN QUESTION REQUEST

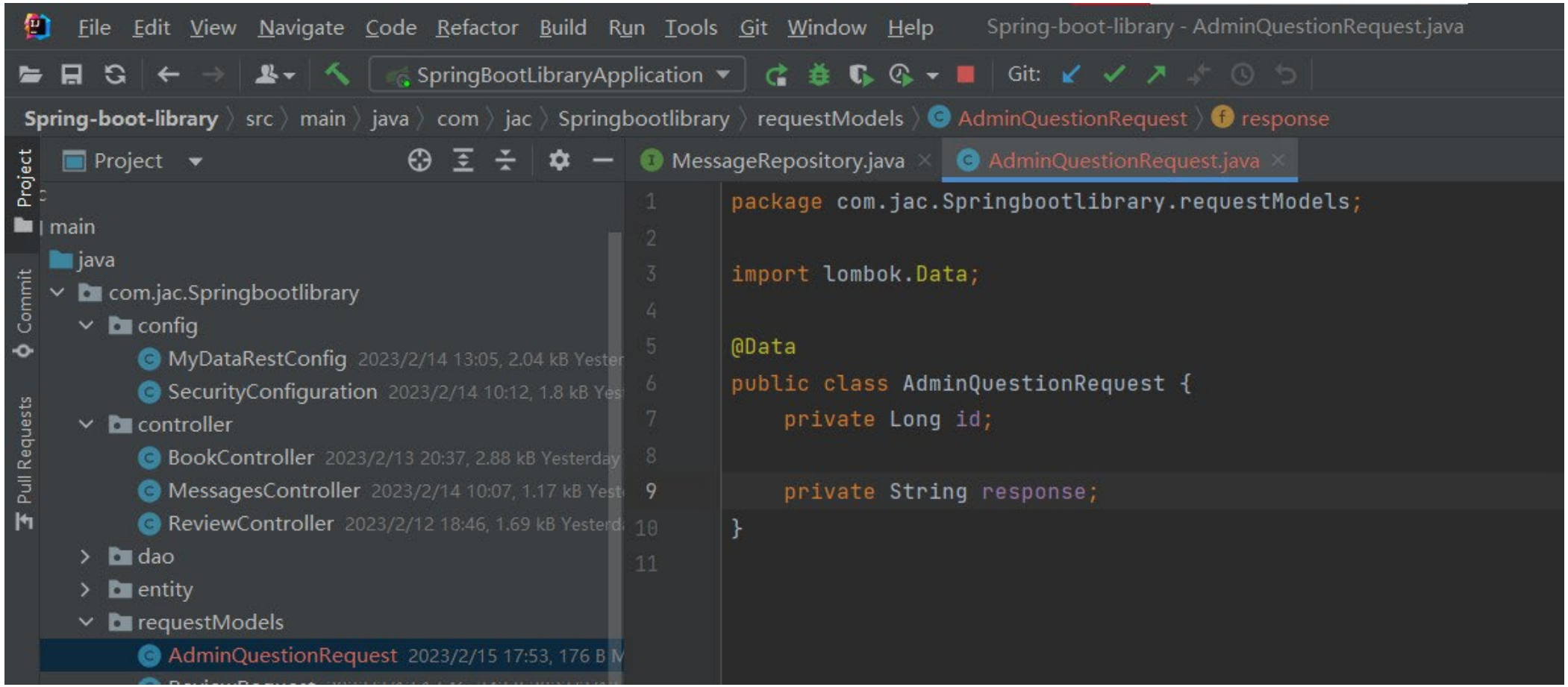
Step1 In requestModels package, create a new request model, named it AdminQuestionRequest.



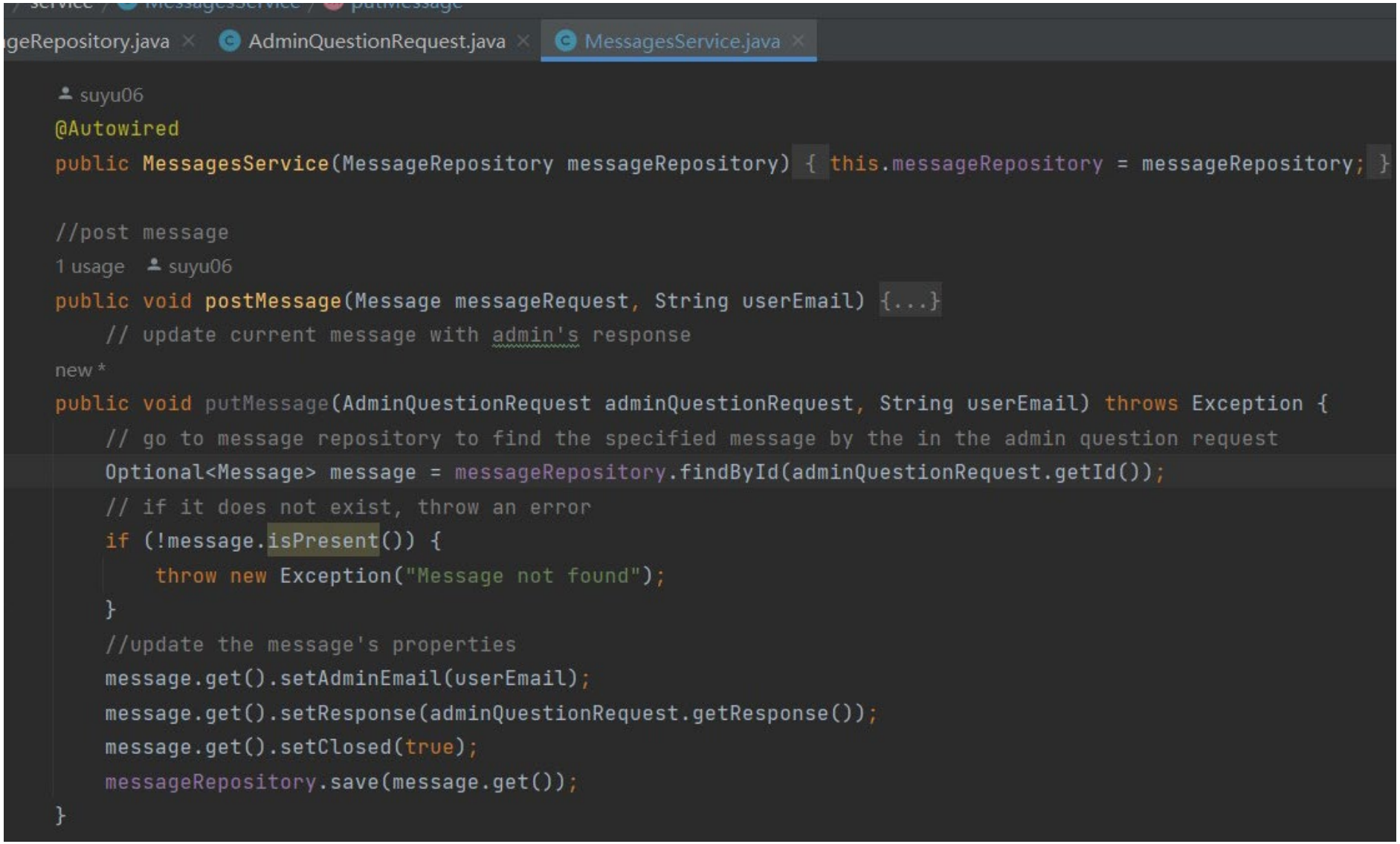
Choose “Class”, input the class name “AdminQuestionRequest”.

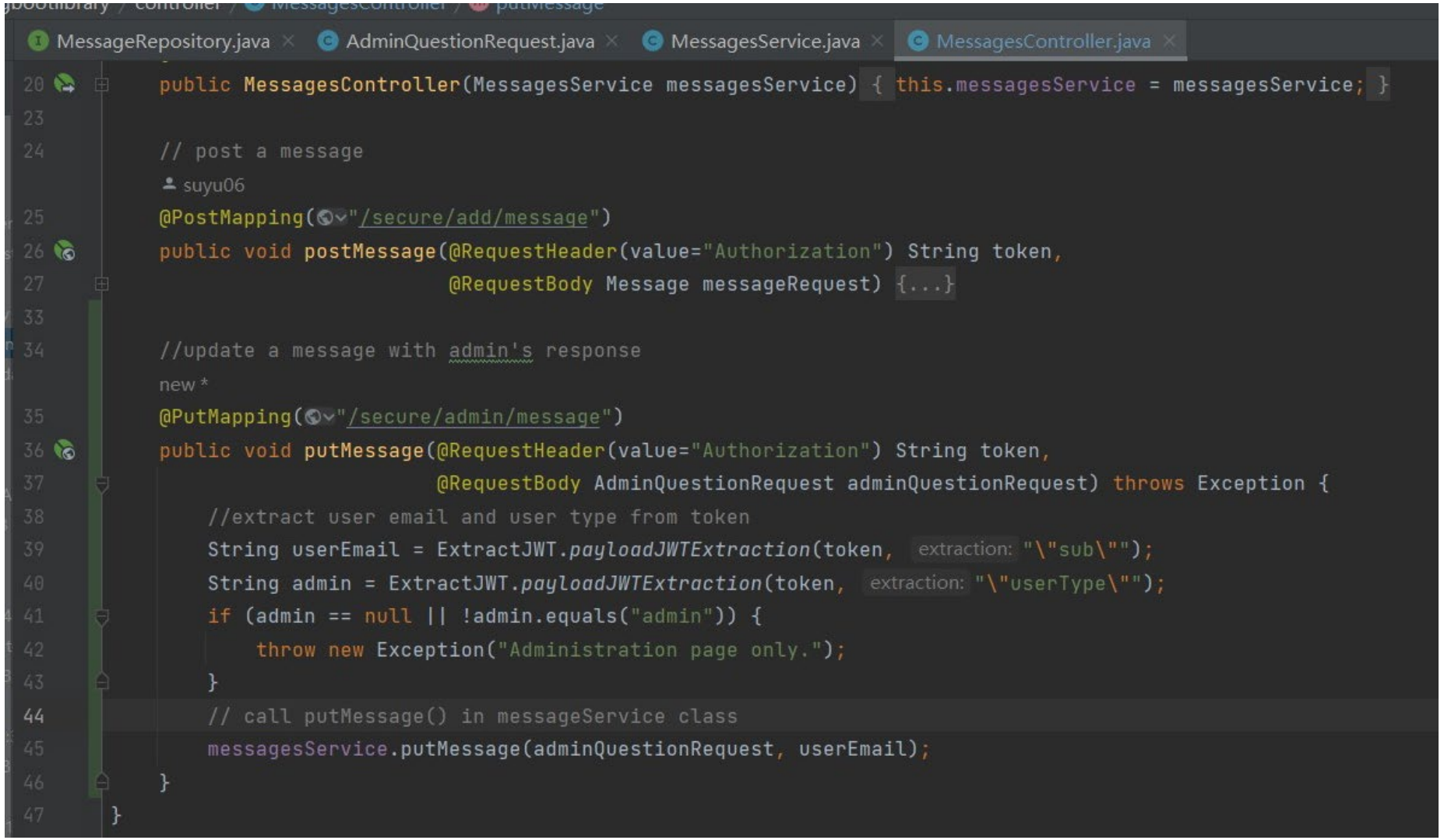


Step 2, add @Data annotation, and add its properites.



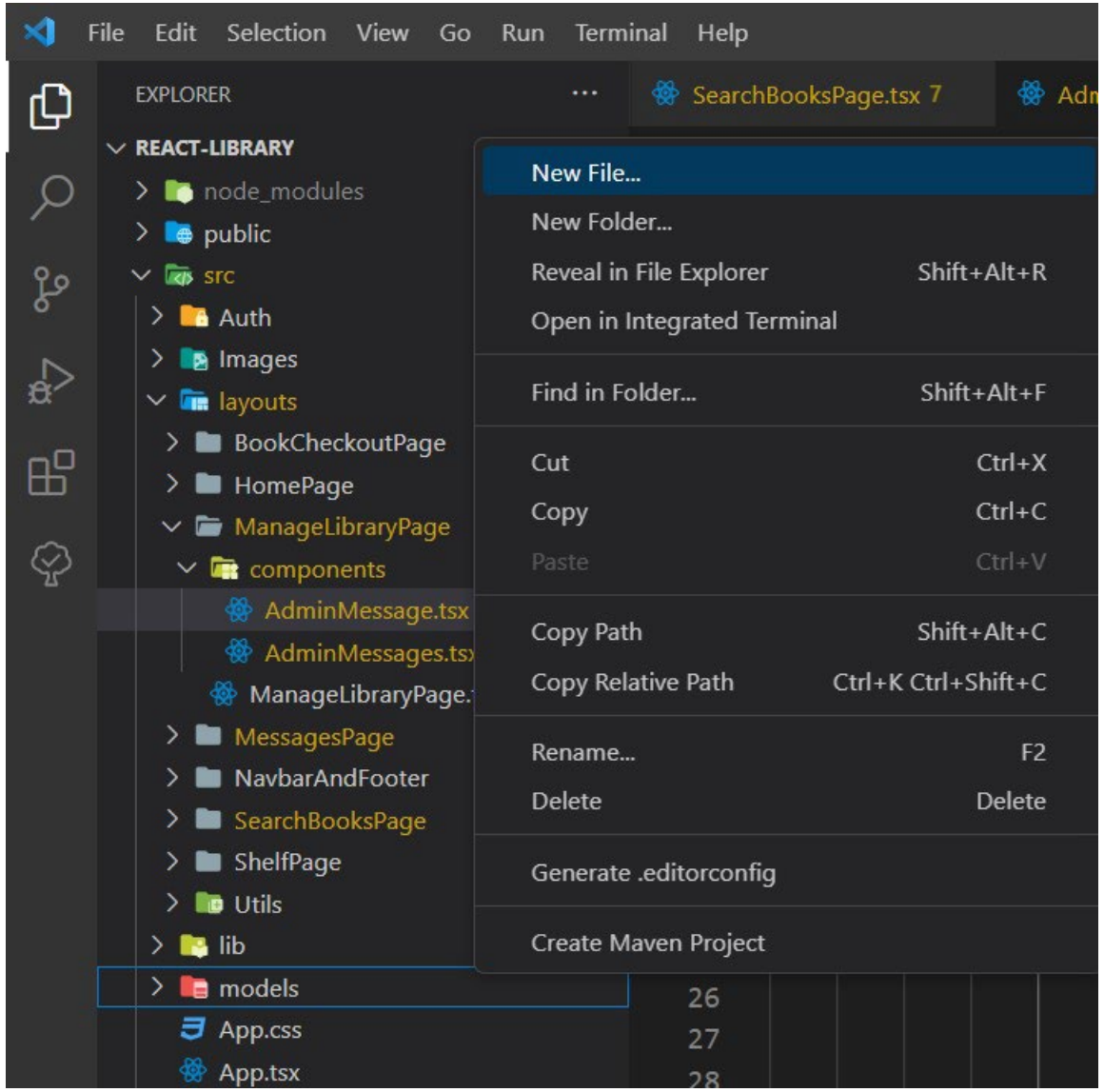
Step 3， Go to MessageService class， right underneath our public void post message, create a putMessage() to update our messages with admin’s response.



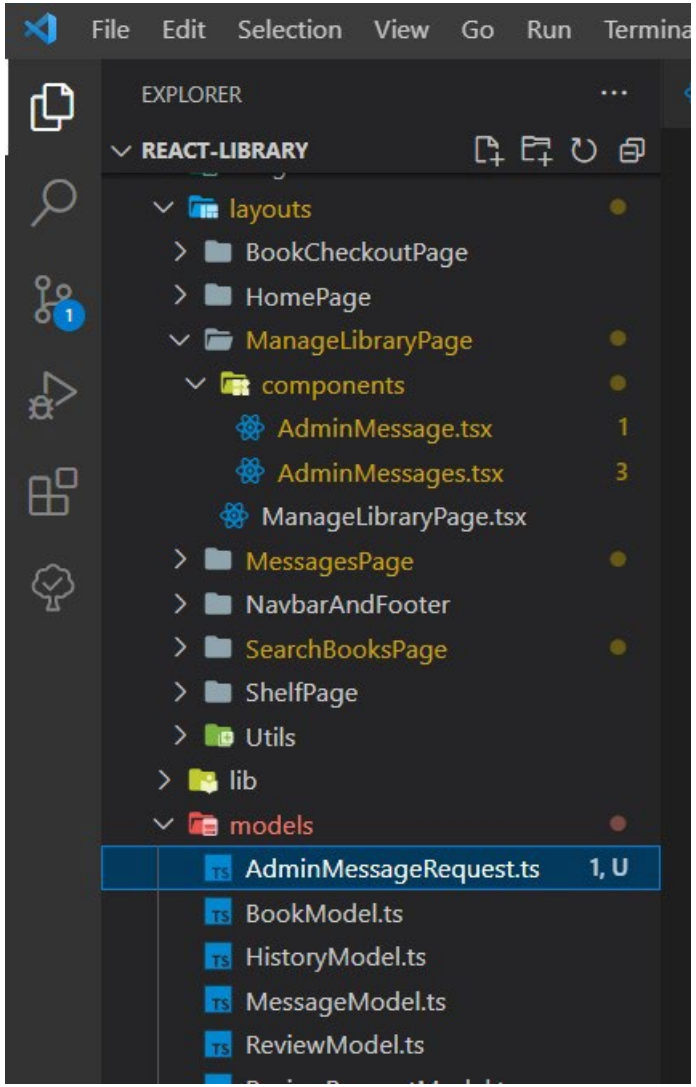


Step 4, Go to MessagesController class, add a putMapping API to our putMessage.

10. REACT ADMIN MESSAGES REQUEST

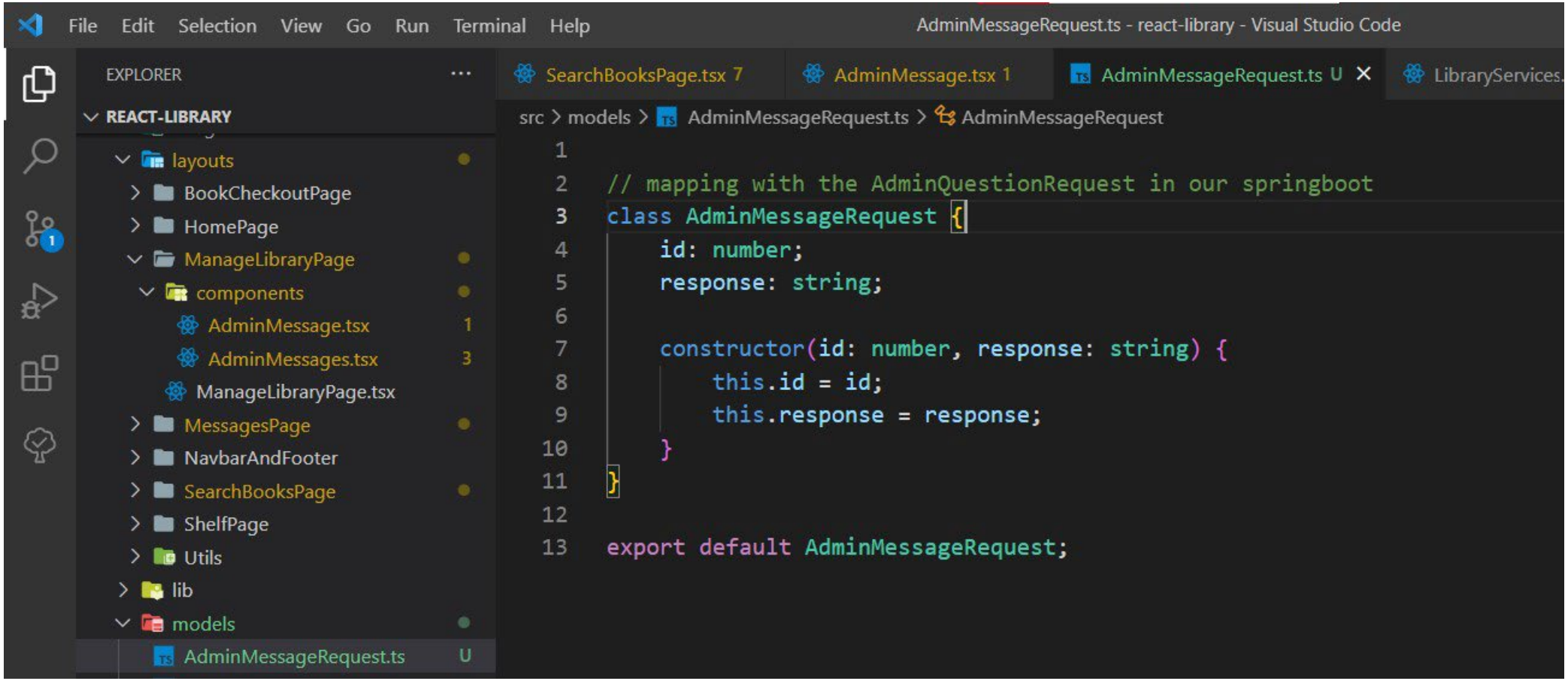


Step 1, So let's first go into our model's directory. And that's right click and say new file.



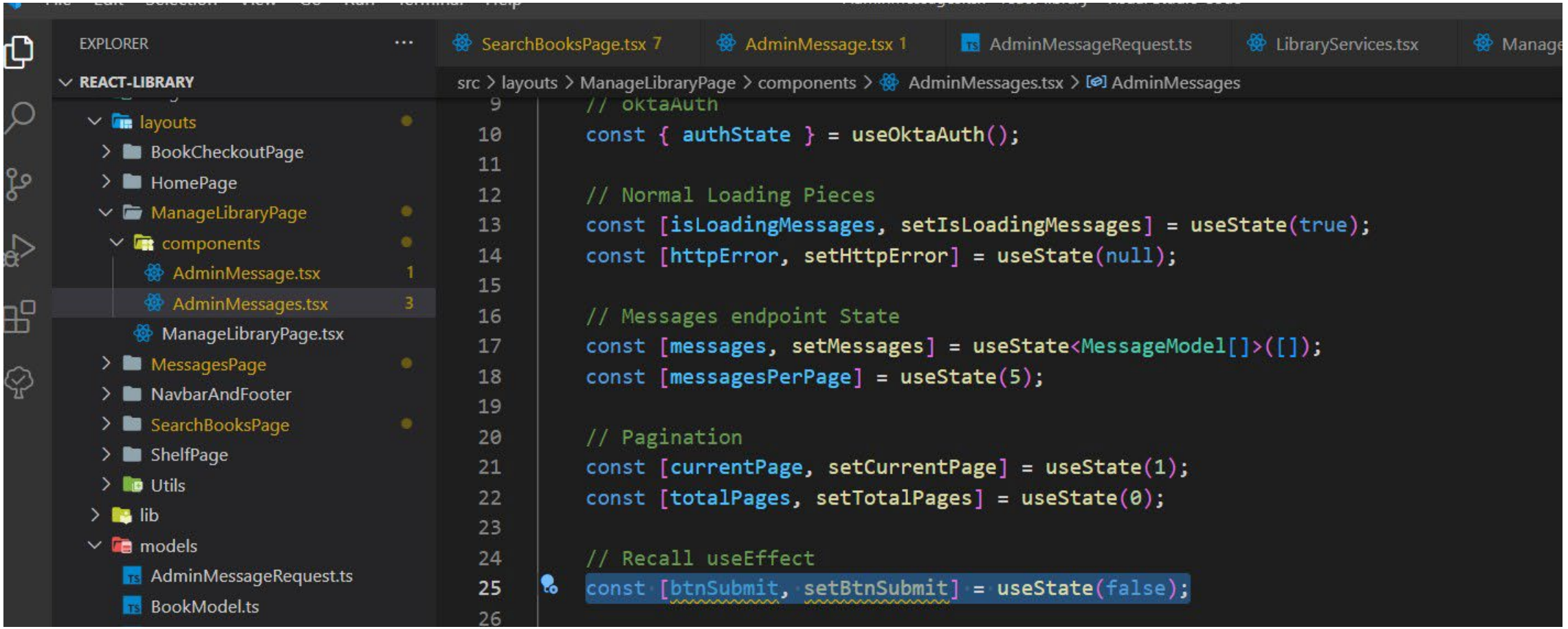
Name it AdminMessageRequest.ts.

Step 2, inside the class ,we will create two properties:
ID of type number and a response of type string.

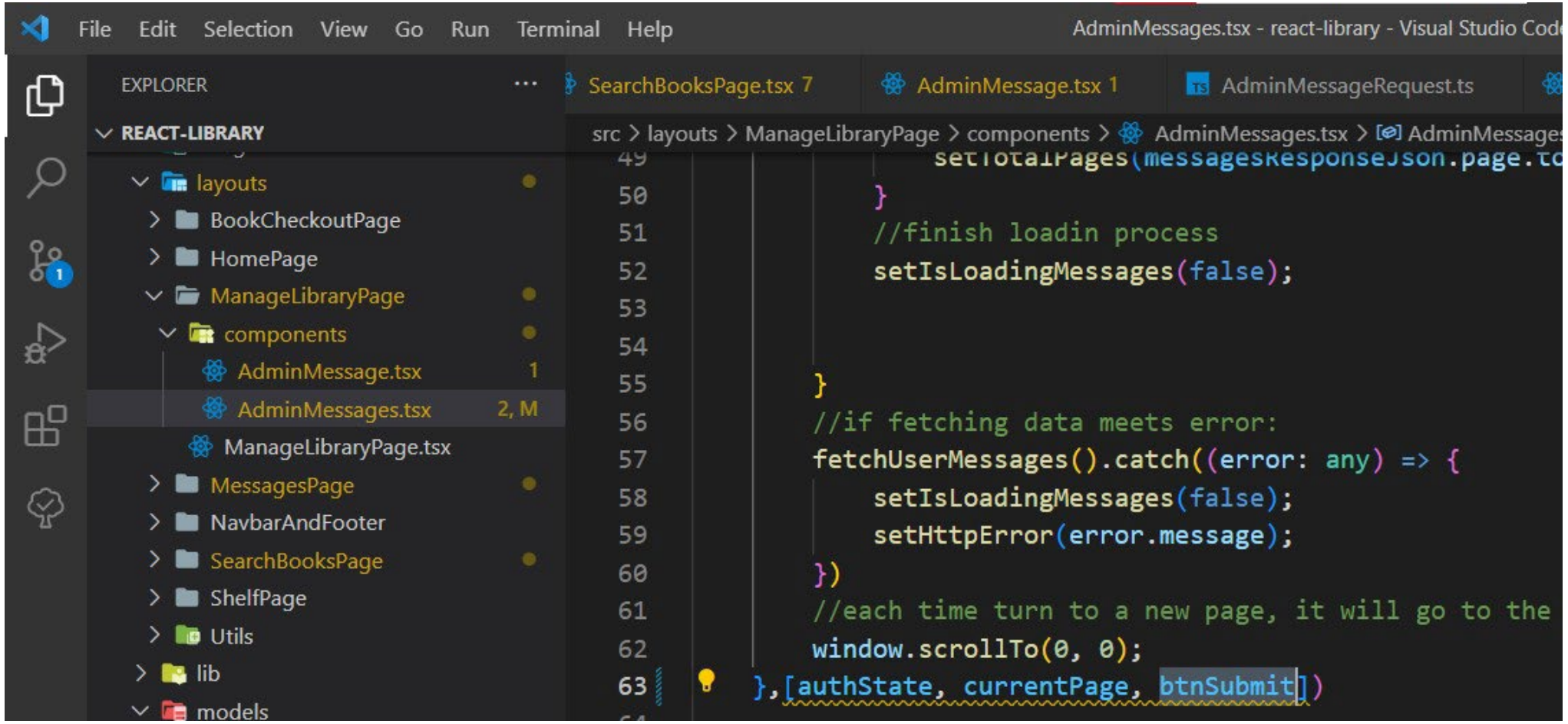


11.REACT SUBMIT RESPONSE FUNCTION

Step1 add const [btnSubmit, setBtnSubmit] = useState(false);



Step 2, add btnSubmit to the useState array.

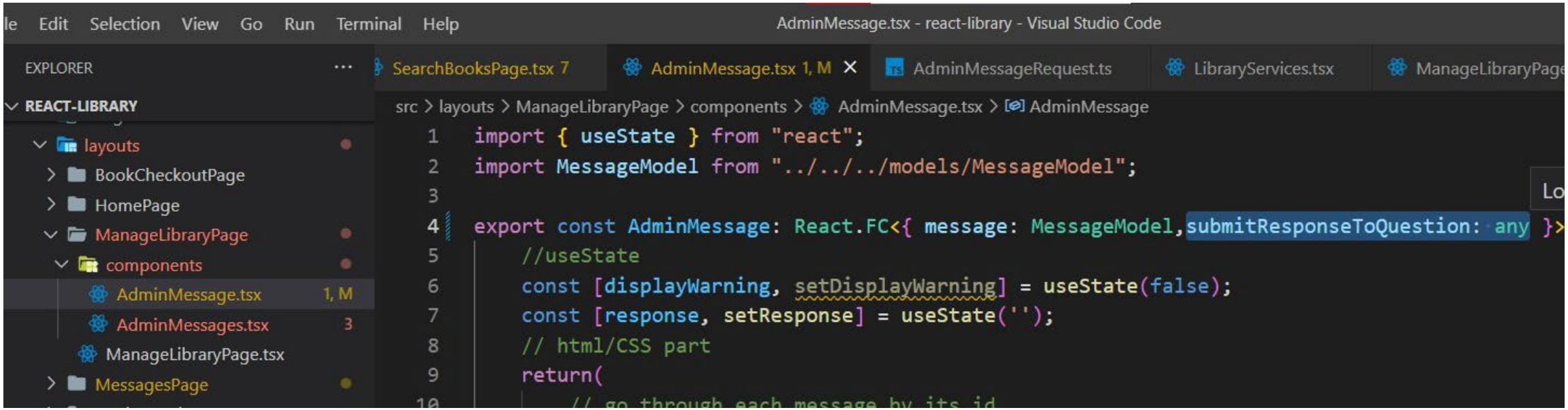


Step3, add an async function submitResponseToQuestion().

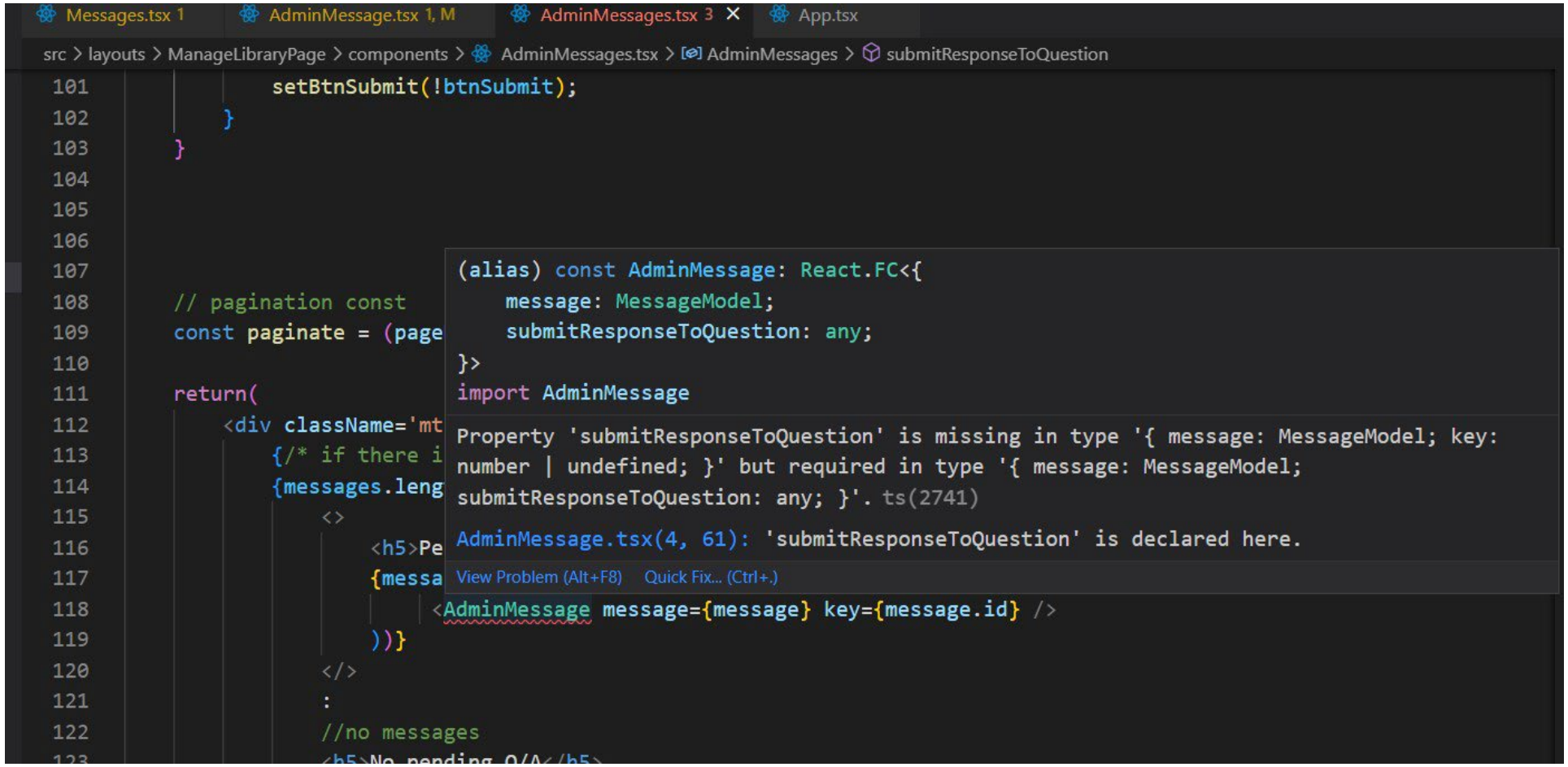
```
async function submitResponseToQuestion(id: number, response: string) {
    const url = `http://localhost:8080/api/messages/secure/admin/message`;
    // user is authenticated and both response request id and response exist:
    if (authState && authState?.isAuthenticated && id !== null && response !== '') {
        const messageAdminRequestModel: AdminMessageRequest = new AdminMessageRequest(id, response);
        const requestOptions = {
            method: 'PUT',
            headers: {
                Authorization: `Bearer ${authState?.accessToken?.accessToken}`,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(messageAdminRequestModel)
        };
        // fetch the data
        const messageAdminRequestModelResponse = await fetch(url, requestOptions);
        // if fetching meets error
        if (!messageAdminRequestModelResponse.ok) {
            throw new Error('Something went wrong!');
        }
        //reset the state of the btn
        setBtnSubmit(!btnSubmit);
```

12. REACT WRAP UP ADMIN RESOURCES

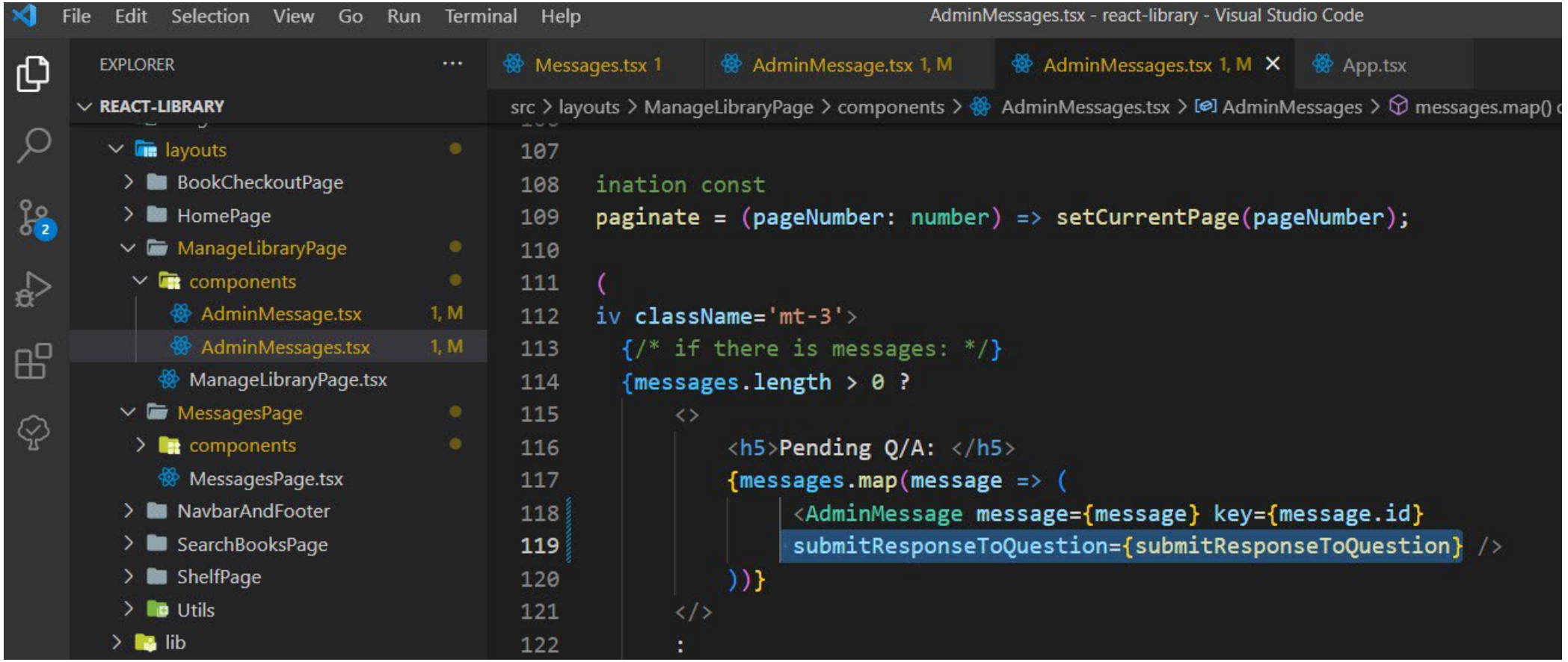
Step 1, in our AdminMessage.tsx, we need to pass in a new prop where this prop is going to be SubmitResponseToQuestion of type any.



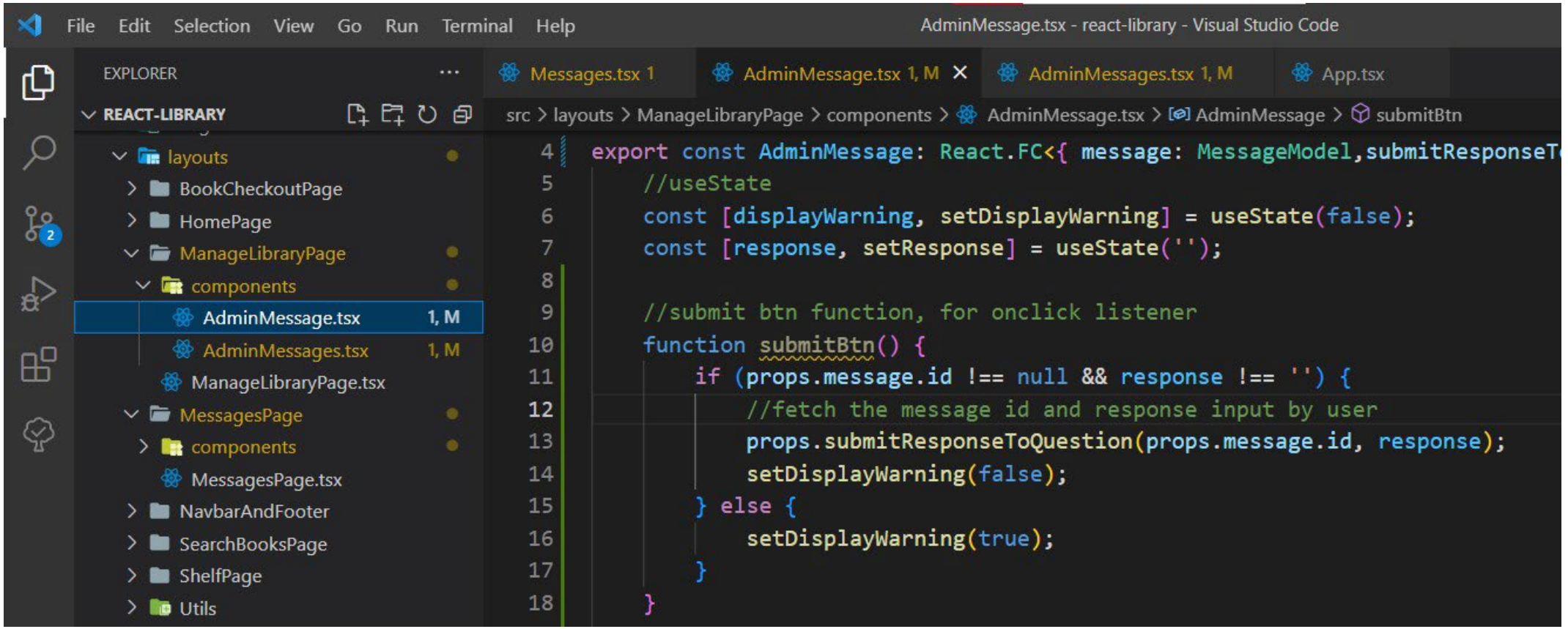
Step 2 Fix the error appeared for AdminMessages in AdminMessages.tsx,



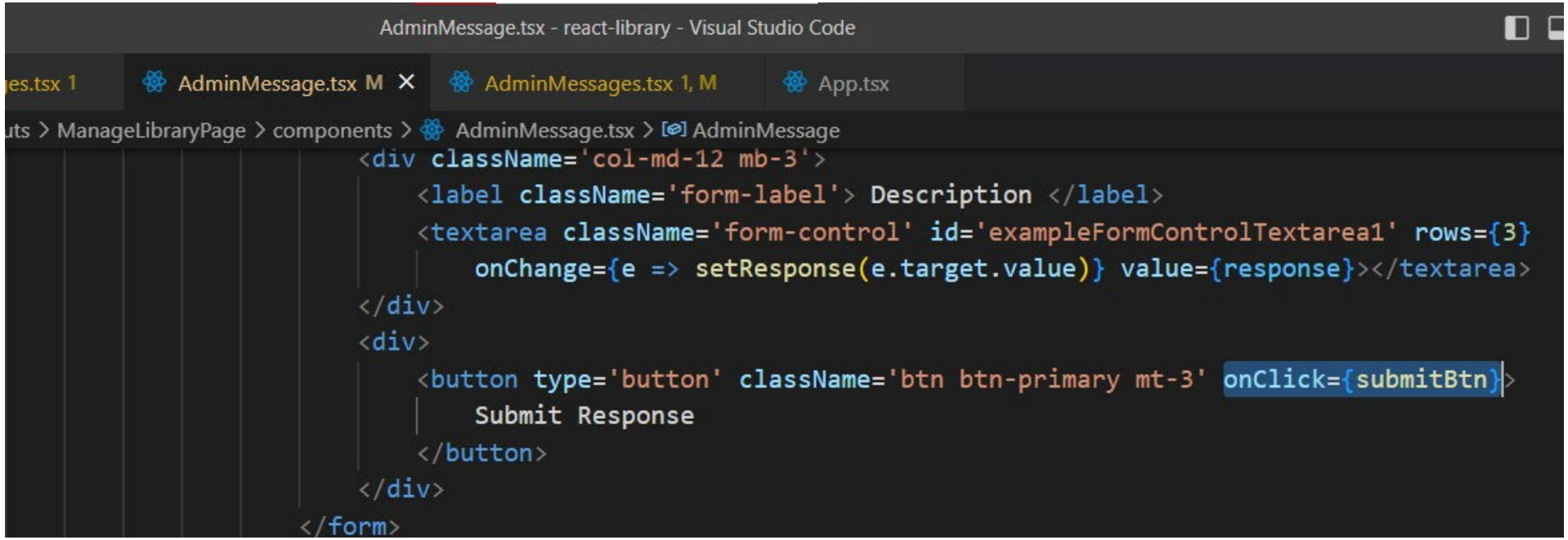
Add “submitResponseToQuestion={submitResponseToQuestion}” as a property to AdminMessage.

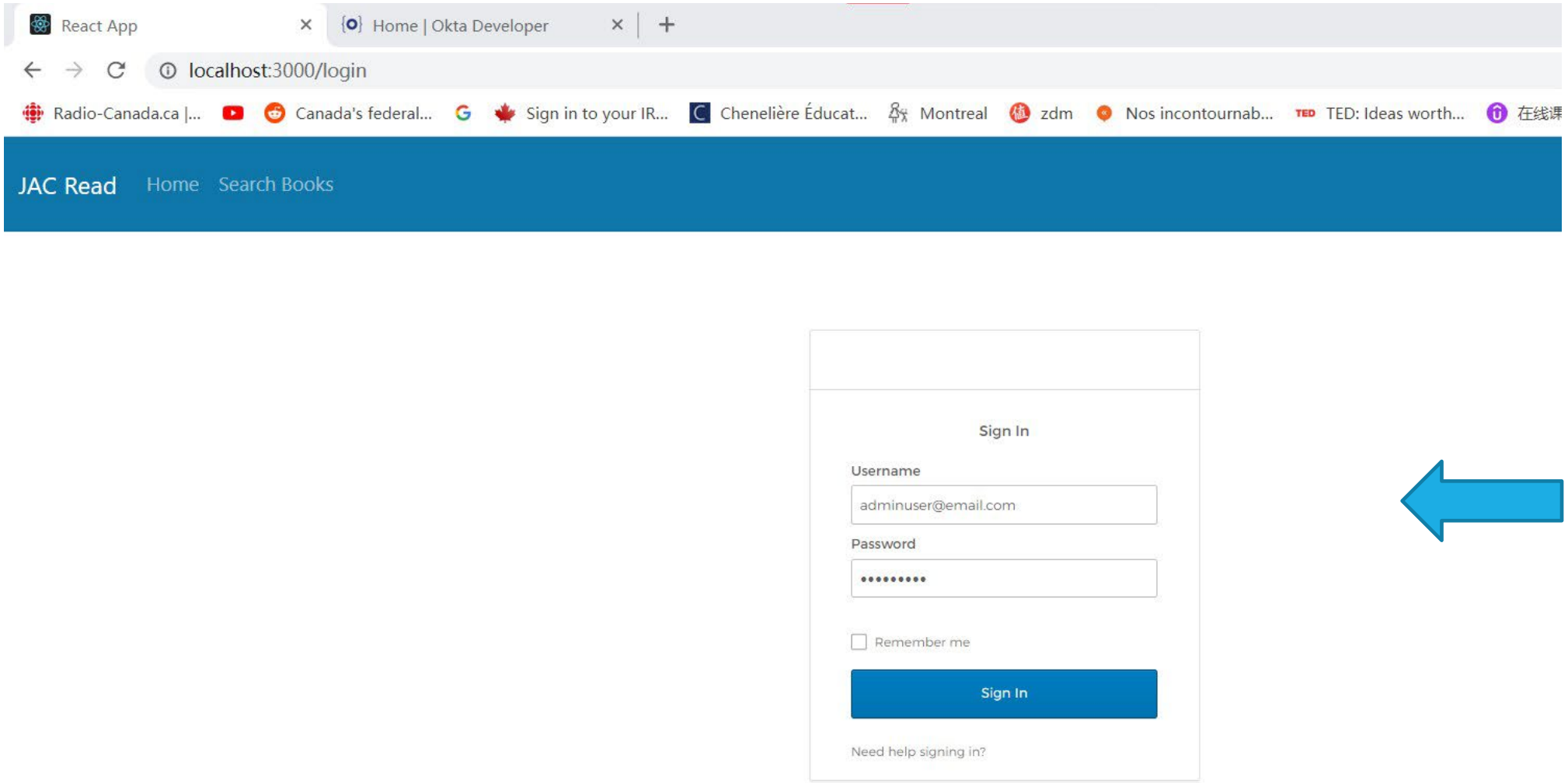


Step 3, AdminMessage.tsx,create submitBtn().



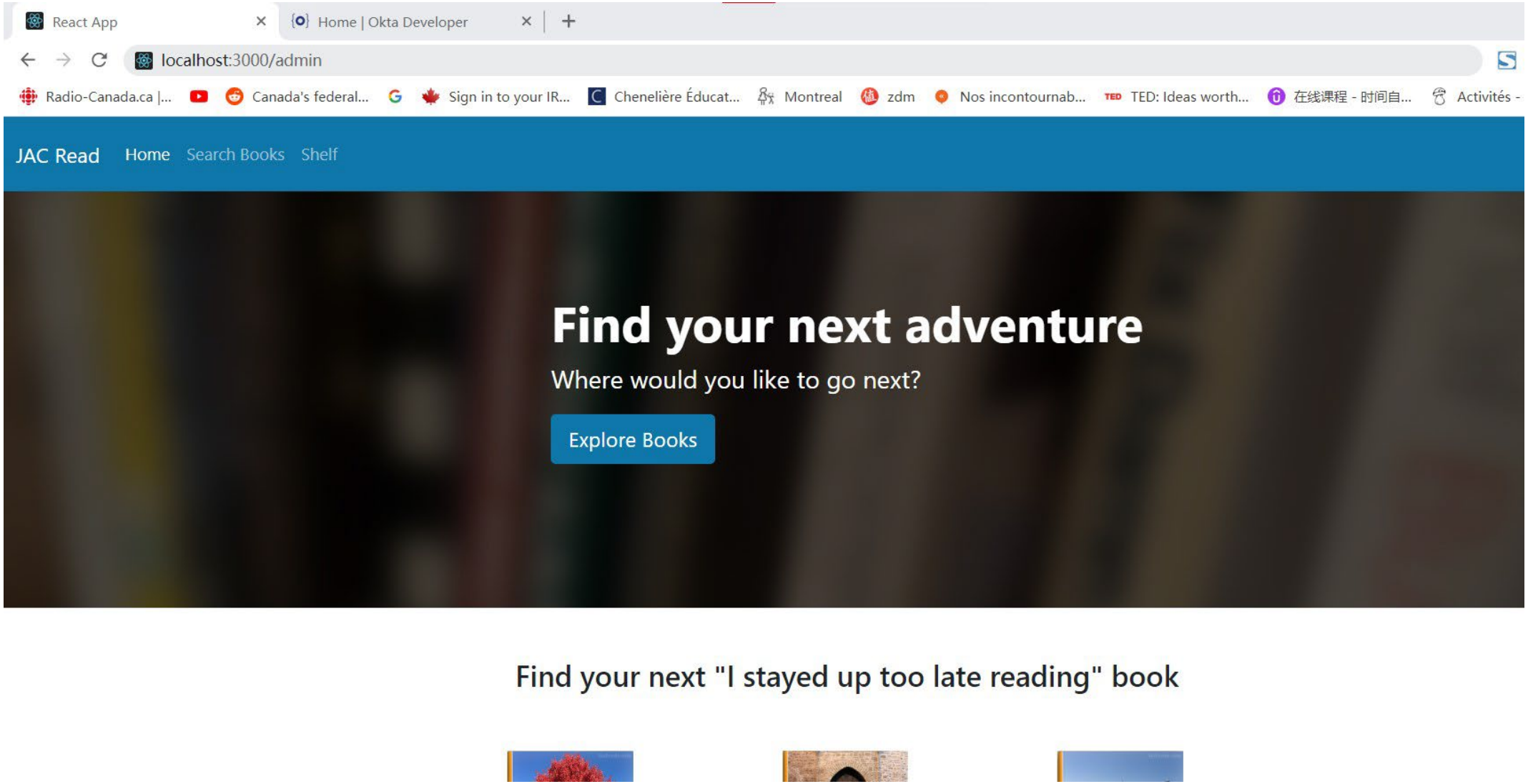
Step 4, add onClick listener to btn “submit Response”.



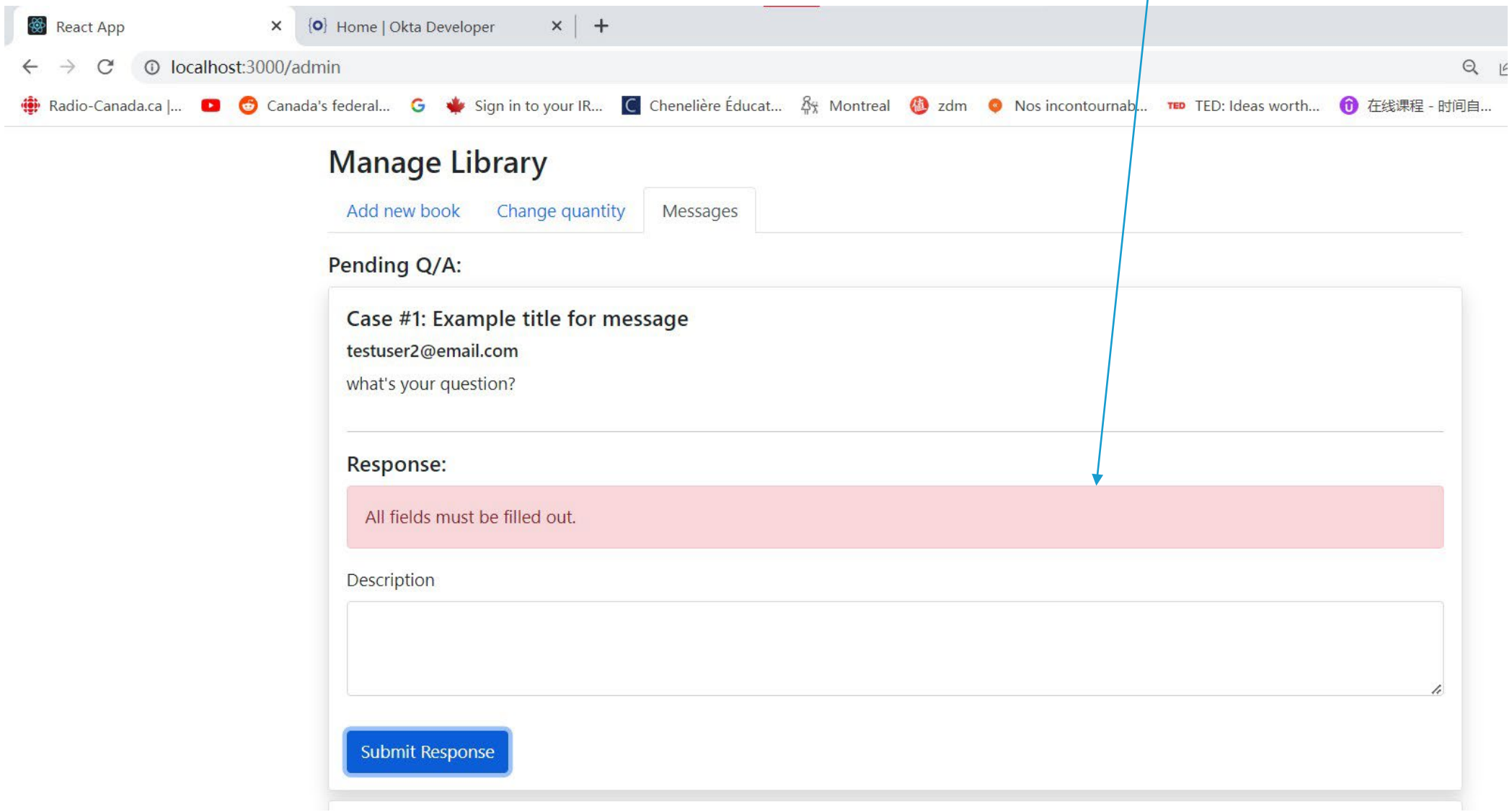


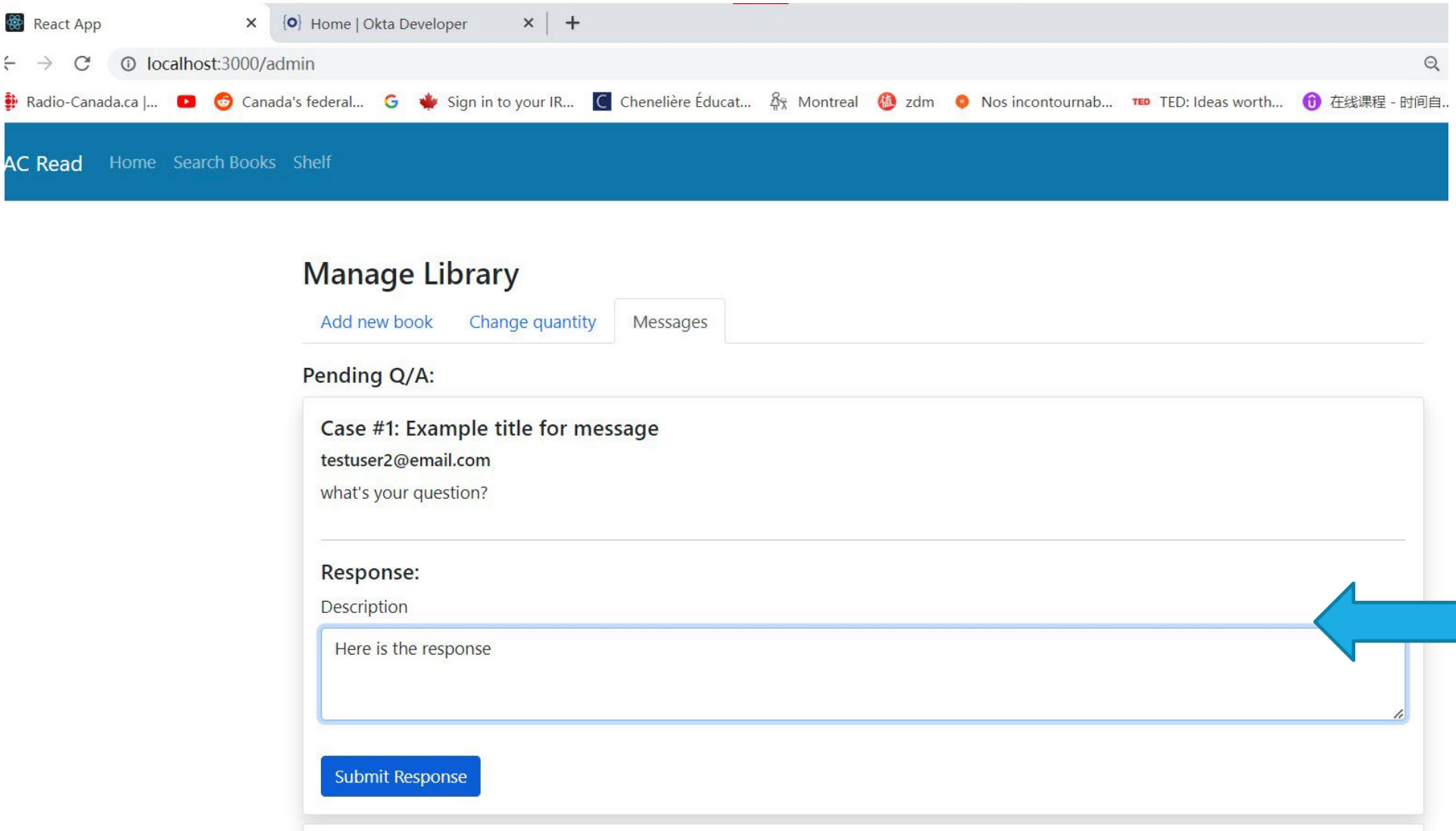
Log in as admin.

Go to http://localhost:3000/admin

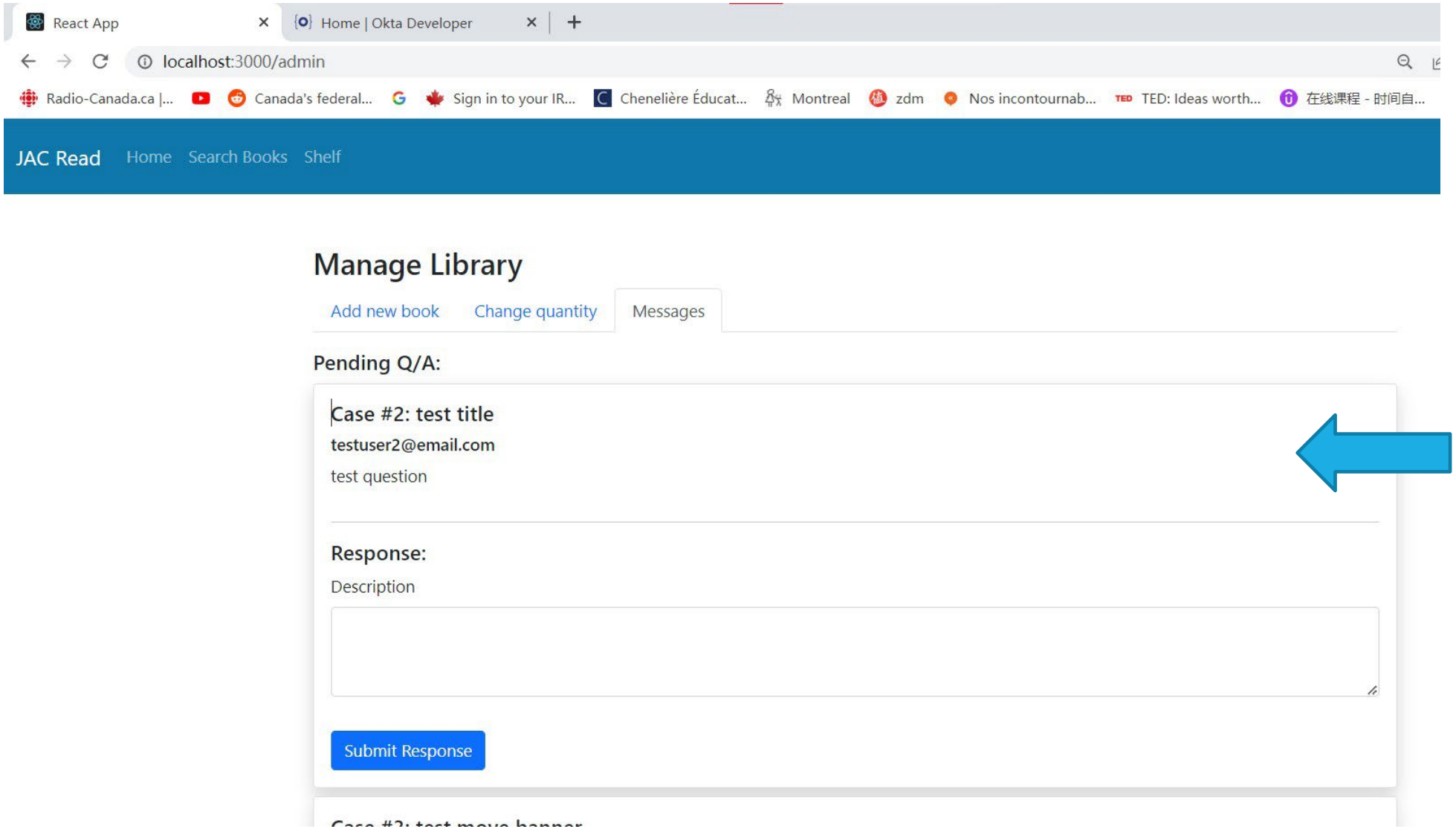


Click on “Messages” tab， type nothing , click “submit response” button, alert banner will appear.



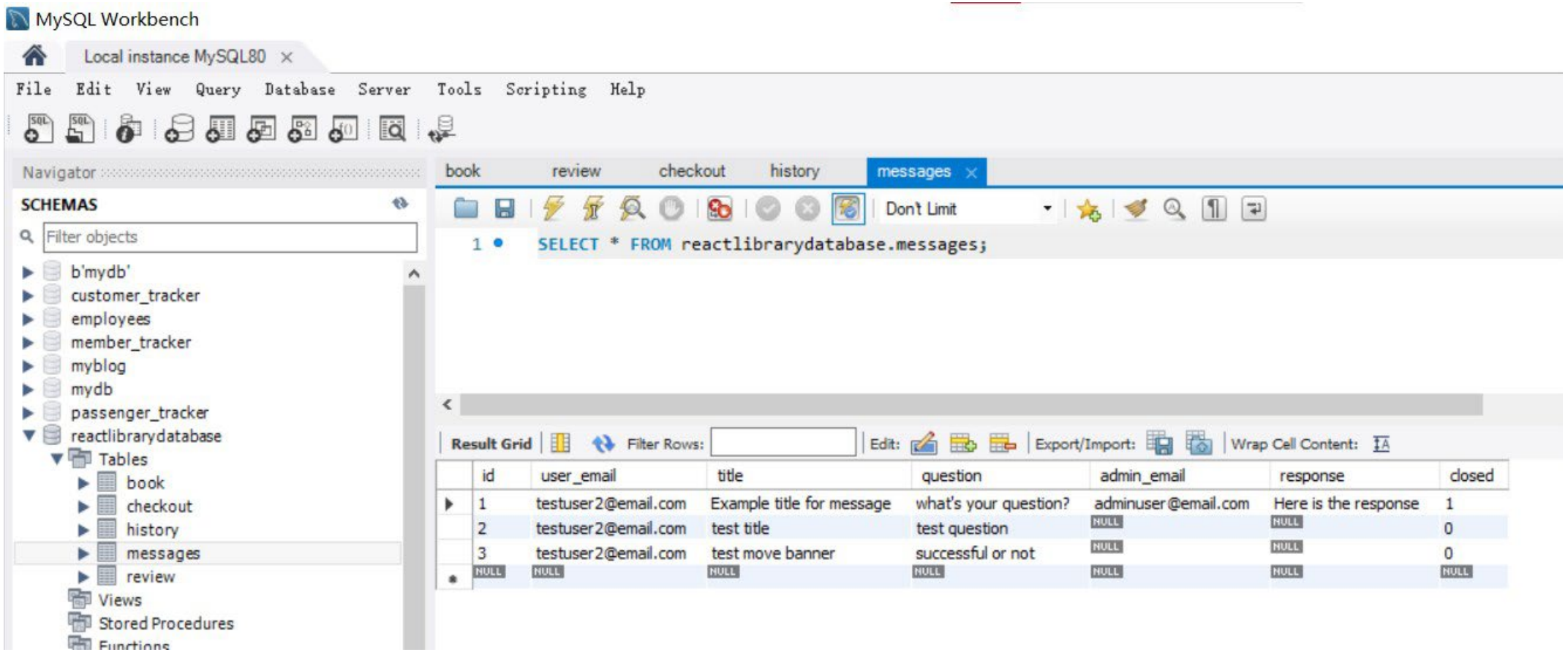


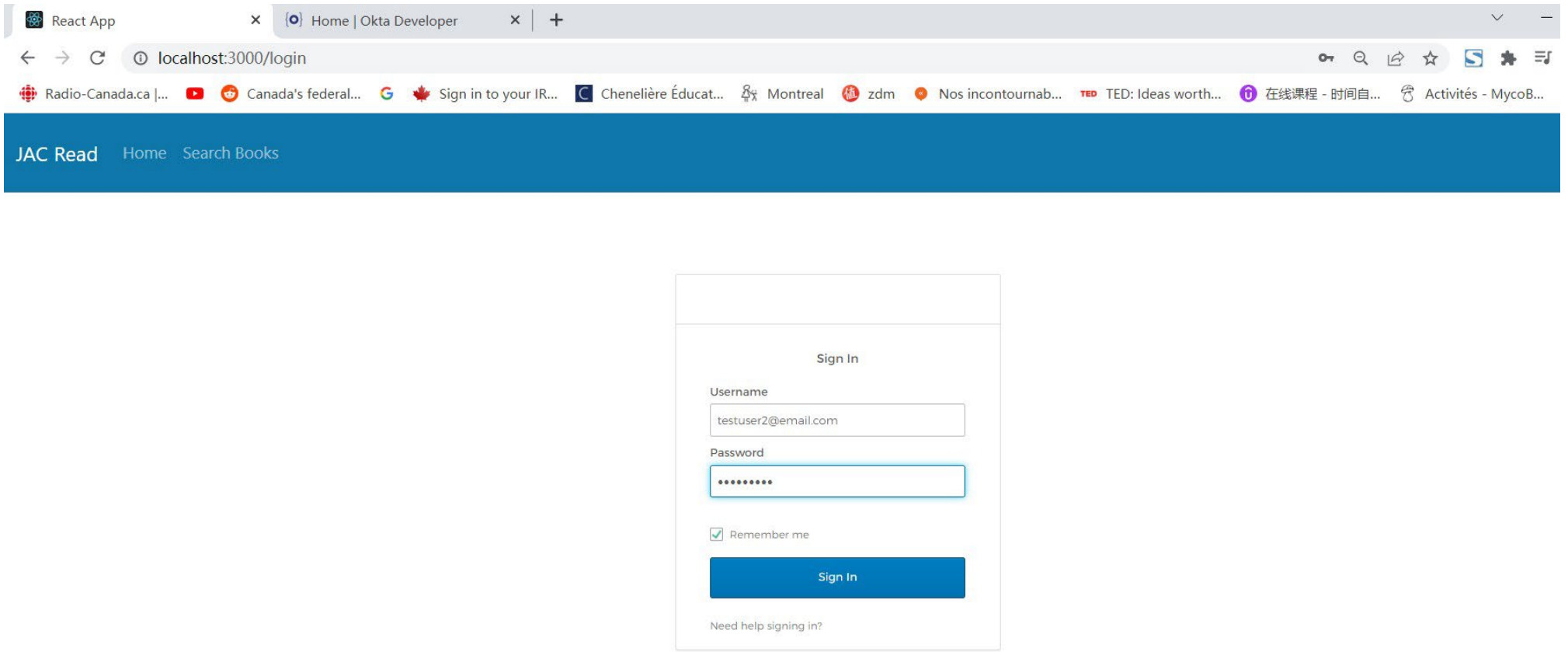
In description box for case #1, input some words, click on “submit response”.



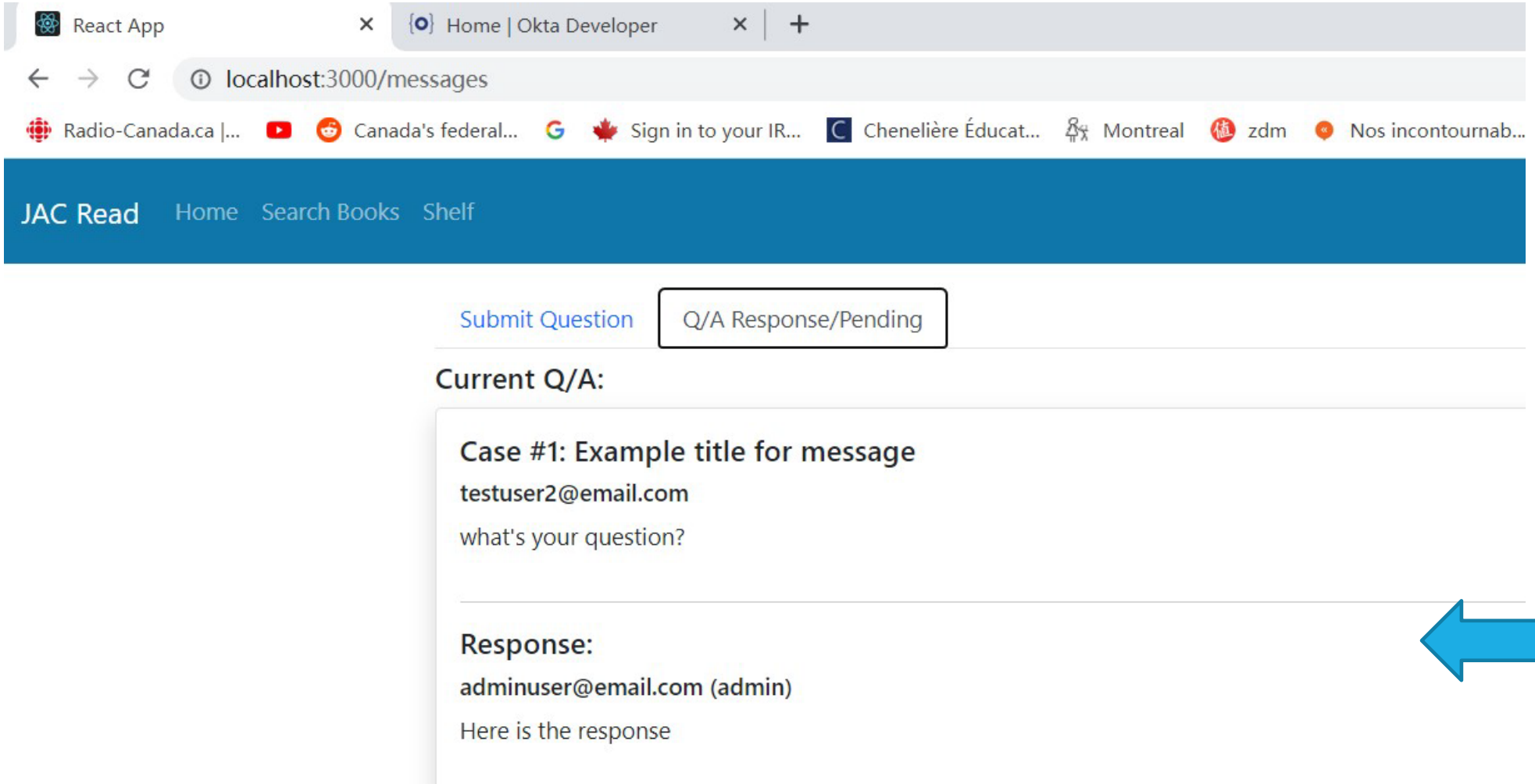
Case #1 will disappear. Because it now has a response, so it is no longer a pending question, but a closed question.

Now checkout our database, the first message's data has updated, now it has a response and been in the status of "closed".





Now, log in as testuser2.



Go to messages page, we will see, the response of Case #1 is showed there.

The other two questions still has no response.