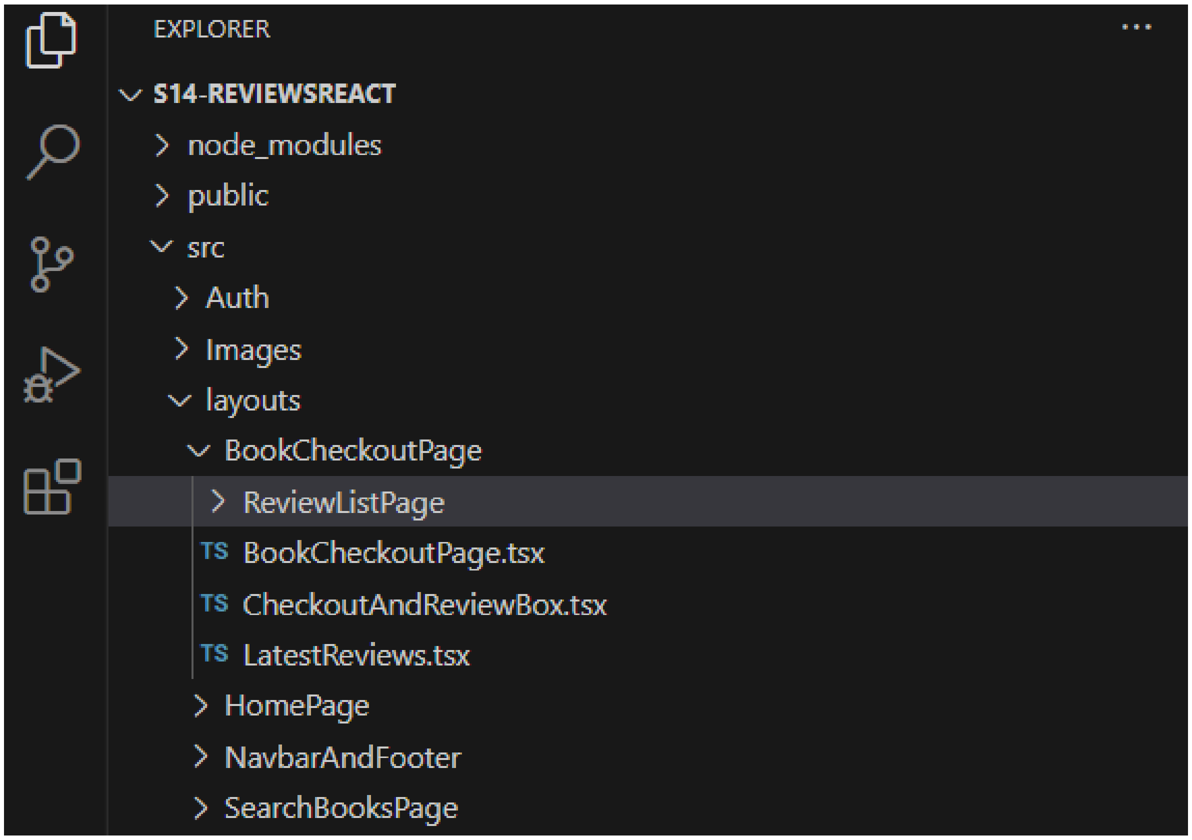
S15-Review List Page React

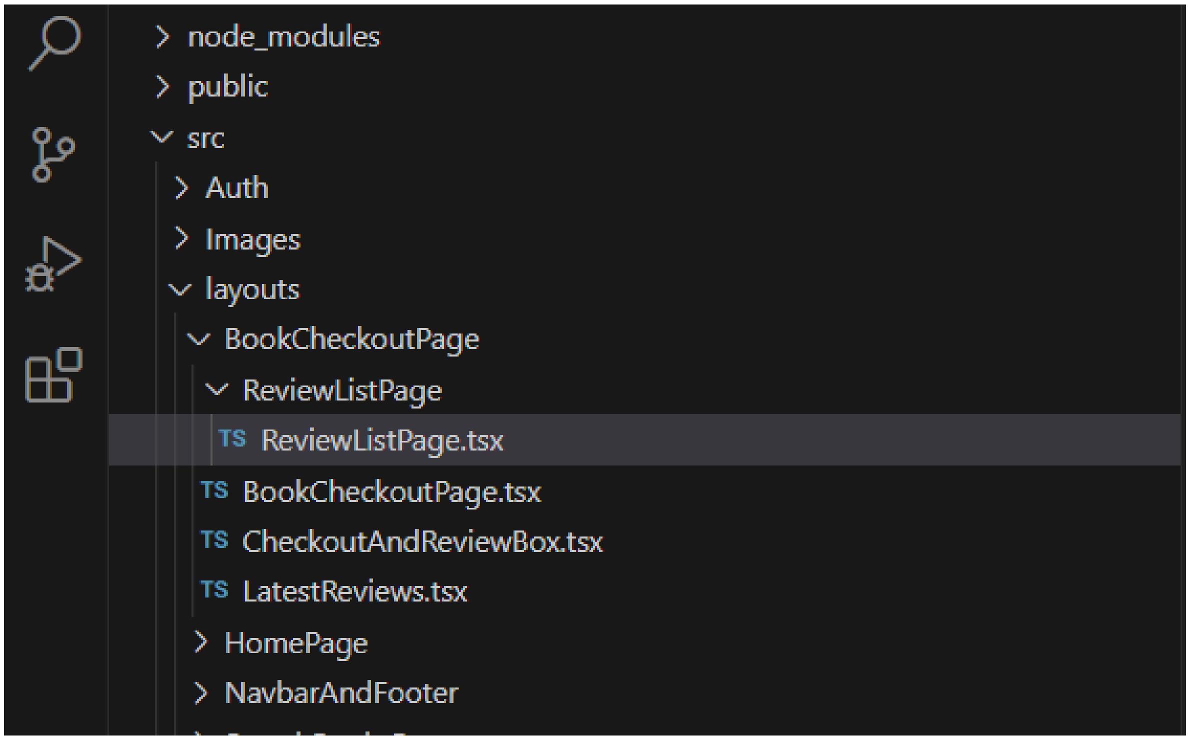
1. List Page and UseState and UseEffect

Step 1 Right-click on BookCheckoutPage folder and choose “new folder”. Name it ReviewListPage.



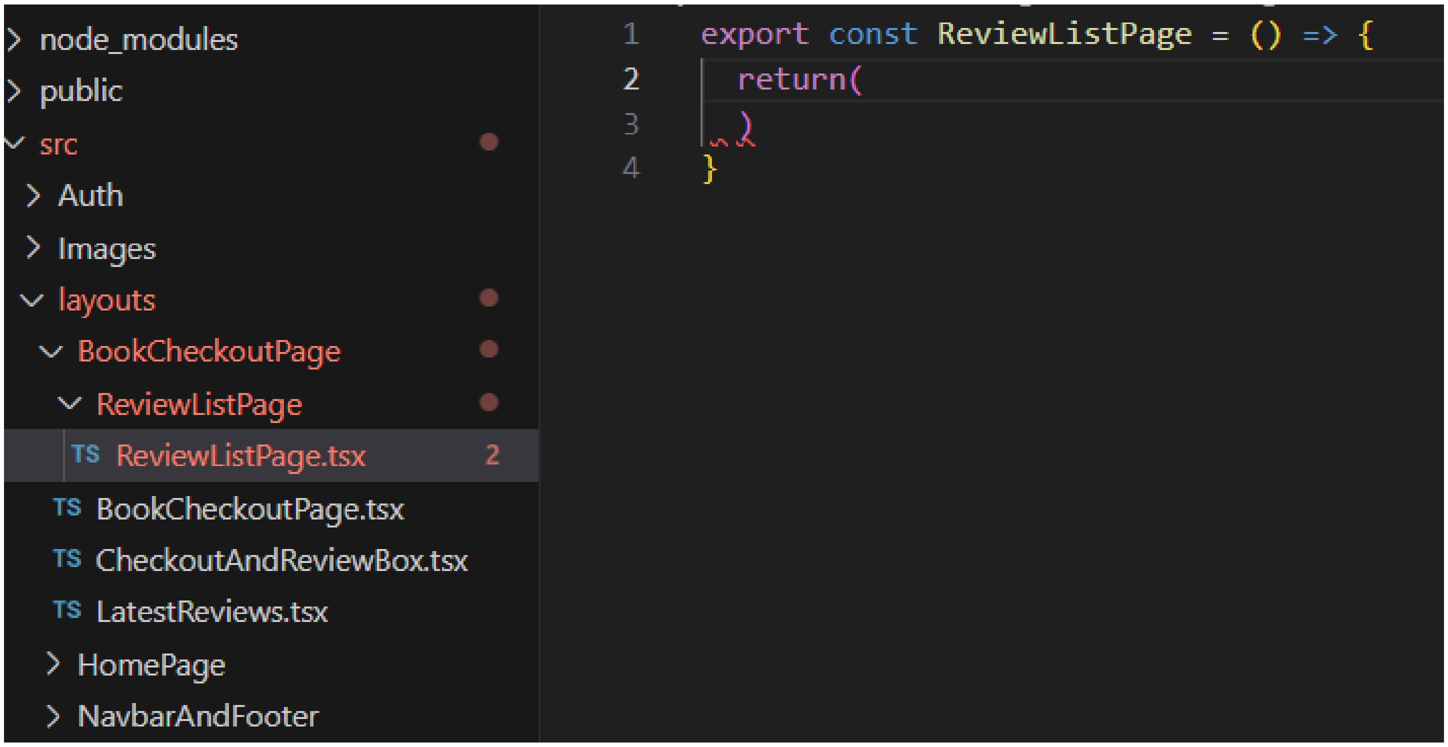
```
EXPLORER
S14-REVIEWSREACT
  node_modules
  public
  src
    Auth
    Images
    layouts
      BookCheckoutPage
        ReviewListPage
        BookCheckoutPage.tsx
        CheckoutAndReviewBox.tsx
        LatestReviews.tsx
      HomePage
      NavbarAndFooter
      SearchBooksPage
```

Step 2 Right-click on the “ReviewListPage” folder, and choose “New File”. Name it ReviewListPage.tsx.



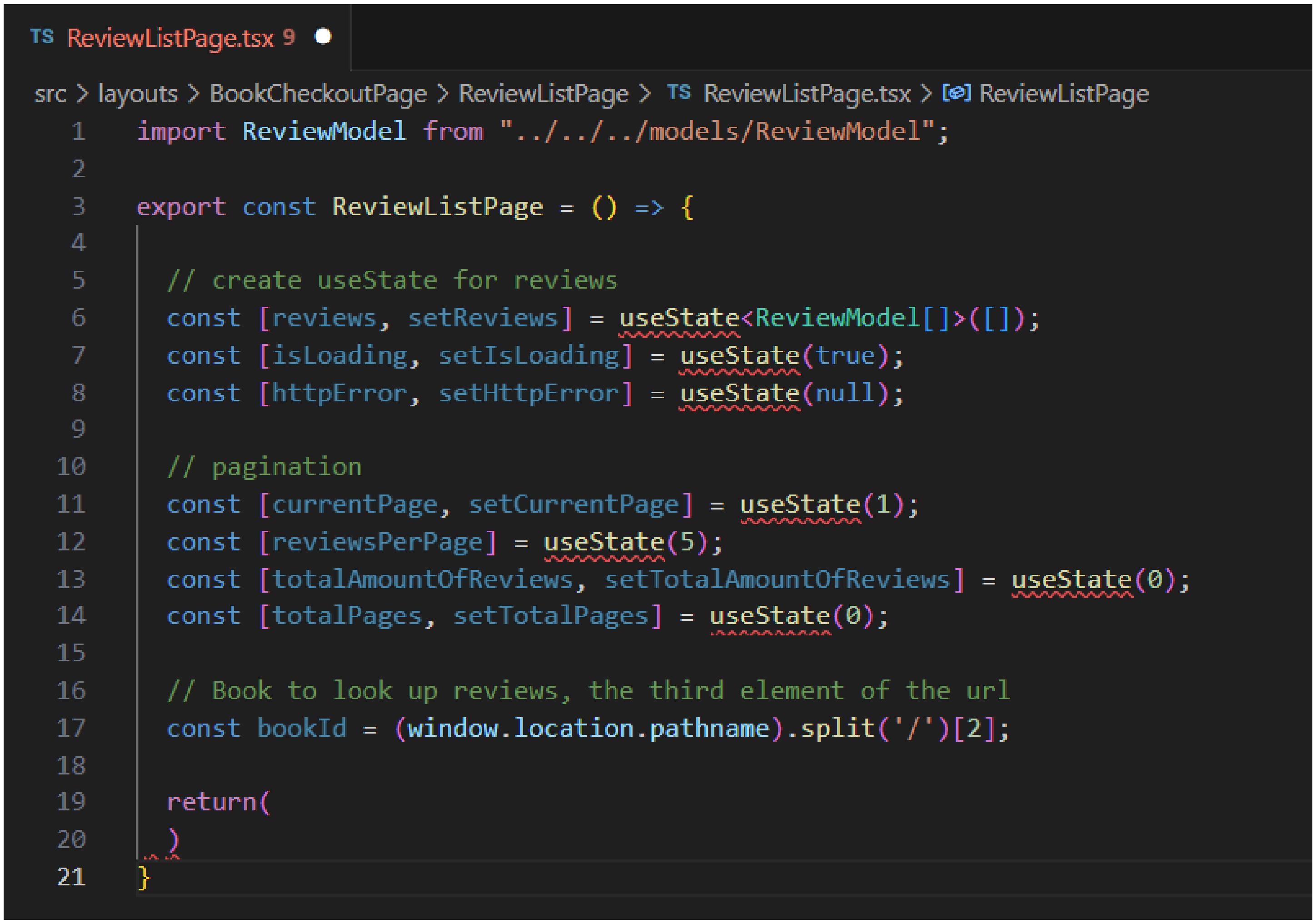
```
  node_modules
  public
  src
    Auth
    Images
    layouts
      BookCheckoutPage
        ReviewListPage
          ReviewListPage.tsx
        BookCheckoutPage.tsx
        CheckoutAndReviewBox.tsx
        LatestReviews.tsx
      HomePage
      NavbarAndFooter
```

Step 3 Create a structure export const in the file ReviewListPage.tsx.



```tsx
export const ReviewListPage = () => {
    return(
    )
}
```

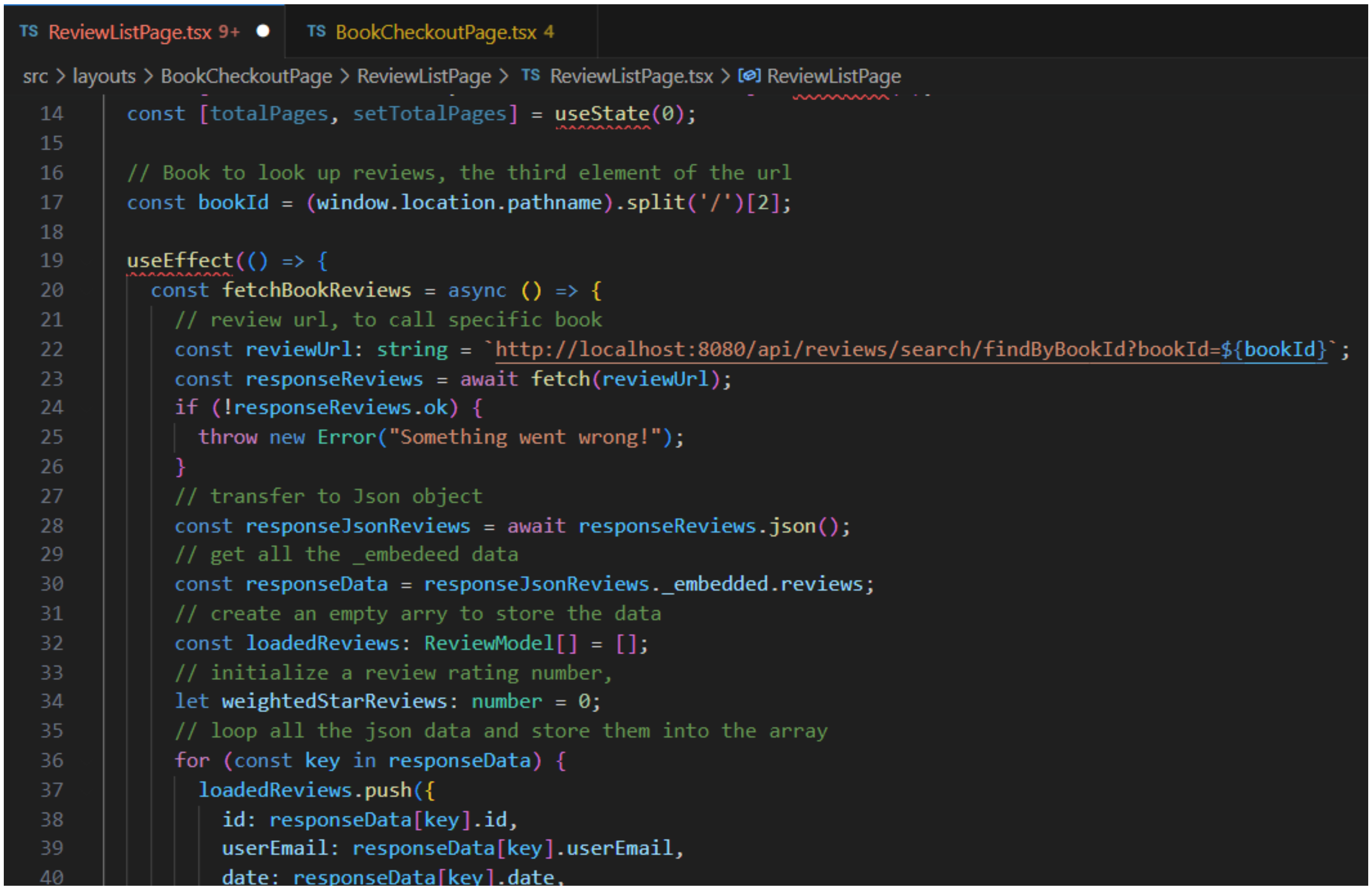
Step 4 Create useState for reviews , pagination book. And a constant for looking up reviews.



ReviewListPage.tsx 9
src > layouts > BookCheckoutPage > ReviewListPage > ReviewListPage.tsx > ReviewListPage

```tsx
import ReviewModel from "../../../models/ReviewModel";

export const ReviewListPage = () => {

    // create useState for reviews
    const [reviews, setReviews] = useState<ReviewModel[]>([]);
    const [isLoading, setIsLoading] = useState(true);
    const [httpError, setHttpError] = useState(null);

    // pagination
    const [currentPage, setCurrentPage] = useState(1);
    const [reviewsPerPage] = useState(5);
    const [totalAmountOfReviews, setTotalAmountOfReviews] = useState(0);
    const [totalPages, setTotalPages] = useState(0);

    // Book to look up reviews, the third element of the url
    const bookId = (window.location.pathname).split('/')[2];

    return(
    )
}
```

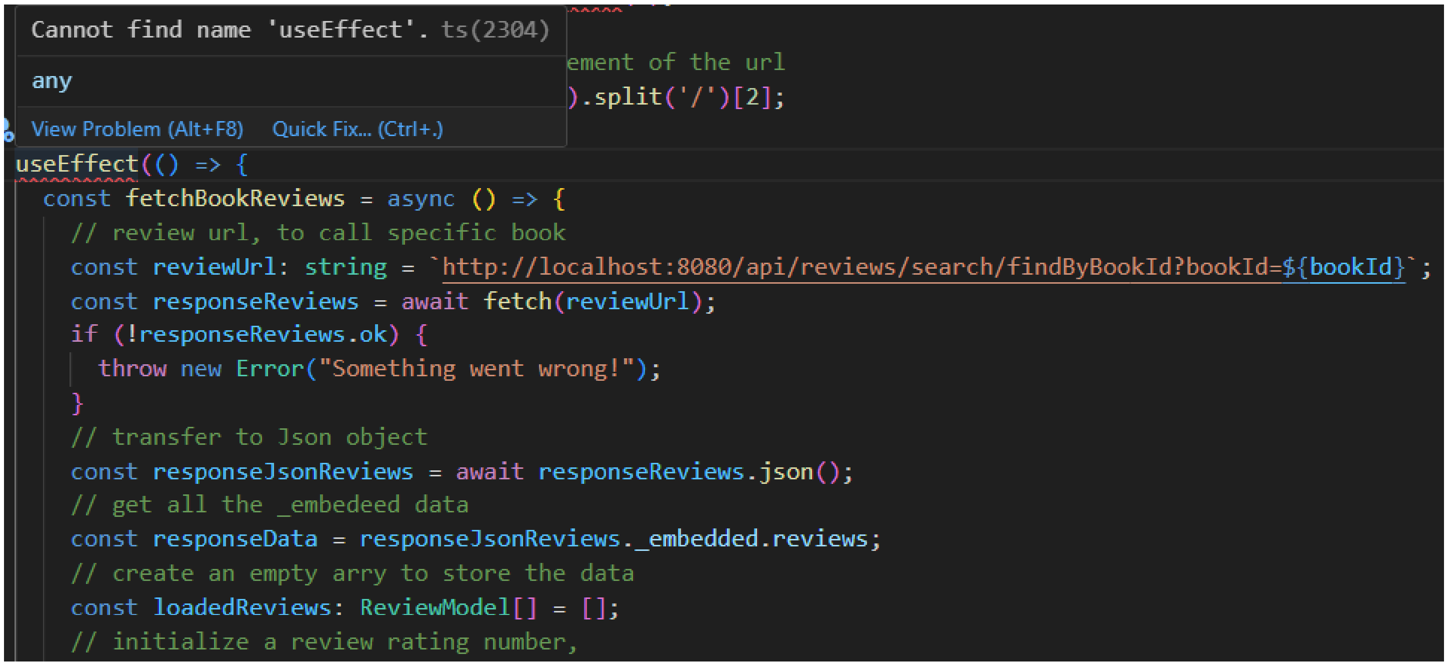
Step 5 Go to BookCheckoutPage.tsx, copy fetchBookReviews useEffect, and paste it to ReviewListPage.tsx.


TS ReviewListPage.tsx 9+
TS BookCheckoutPage.tsx 4
src > layouts > BookCheckoutPage > ReviewListPage > TS ReviewListPage.tsx > ReviewListPage

```
  const [totalPages, setTotalPages] = useState(0);

  // Book to look up reviews, the third element of the url
  const bookId = (window.location.pathname).split('/')[2];

  useEffect(() => {
    const fetchBookReviews = async () => {
      // review url, to call specific book
      const reviewUrl: string = `http://localhost:8080/api/reviews/search/findByBookId?bookId=${bookId}`;
      const responseReviews = await fetch(reviewUrl);
      if (!responseReviews.ok) {
        throw new Error("Something went wrong!");
      }
      // transfer to Json object
      const responseJsonReviews = await responseReviews.json();
      // get all the _embedeed data
      const responseData = responseJsonReviews._embedded.reviews;
      // create an empty arry to store the data
      const loadedReviews: ReviewModel[] = [];
      // initialize a review rating number,
      let weightedStarReviews: number = 0;
      // loop all the json data and store them into the array
      for (const key in responseData) {
        loadedReviews.push({
          id: responseData[key].id,
          userEmail: responseData[key].userEmail,
          date: responseData[key].date,
```

Step 6 Fix errors.

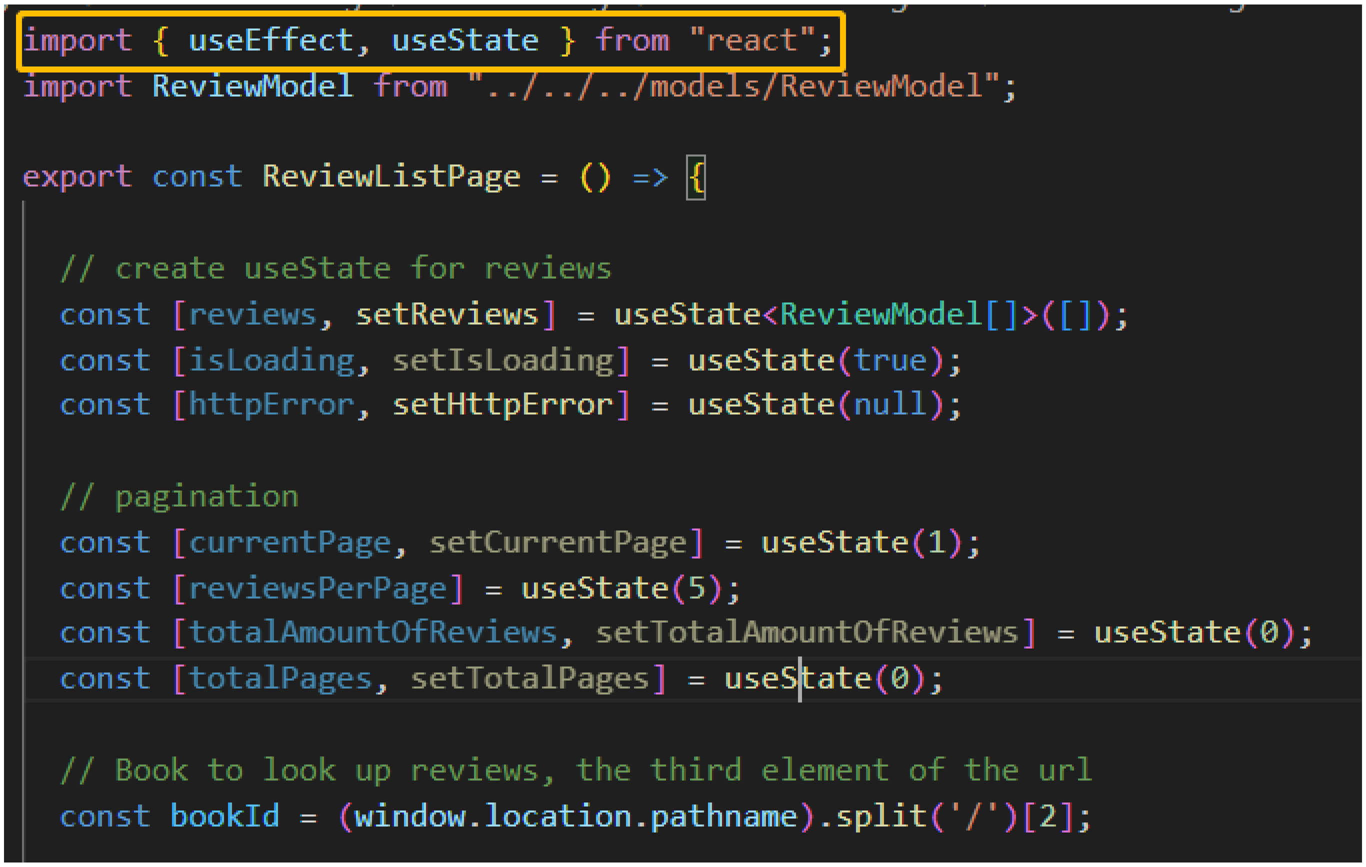
6.1 import useEffect. Click on “Quick Fix” and choose “update import from React”.



```
Cannot find name 'useEffect'. ts(2304)
any
```

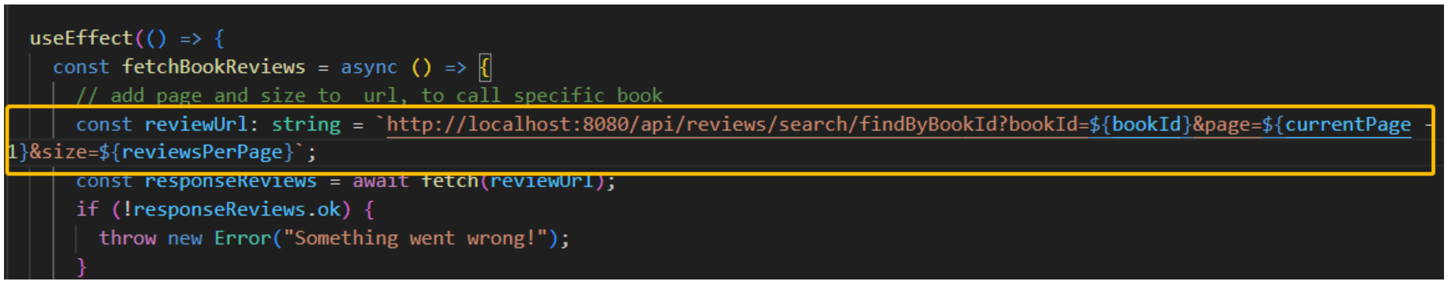
```
ement of the url
).split('/')[2];

useEffect(() => {
  const fetchBookReviews = async () => {
    // review url, to call specific book
    const reviewUrl: string = `http://localhost:8080/api/reviews/search/findByBookId?bookId=${bookId}`;
    const responseReviews = await fetch(reviewUrl);
    if (!responseReviews.ok) {
      throw new Error("Something went wrong!");
    }
    // transfer to Json object
    const responseJsonReviews = await responseReviews.json();
    // get all the _embedeed data
    const responseData = responseJsonReviews._embedded.reviews;
    // create an empty arry to store the data
    const loadedReviews: ReviewModel[] = [];
    // initialize a review rating number,
```

Look, useEffect is imported.



```
import { useEffect, useState } from "react";
import ReviewModel from "../../../models/ReviewModel";

export const ReviewListPage = () => {

  // create useState for reviews
  const [reviews, setReviews] = useState<ReviewModel[]>([]);
  const [isLoading, setIsLoading] = useState(true);
  const [httpError, setHttpError] = useState(null);

  // pagination
  const [currentPage, setCurrentPage] = useState(1);
  const [reviewsPerPage] = useState(5);
  const [totalAmountOfReviews, setTotalAmountOfReviews] = useState(0);
  const [totalPages, setTotalPages] = useState(0);

  // Book to look up reviews, the third element of the url
  const bookId = (window.location.pathname).split('/')[2];
```

6.2 Modify the url.

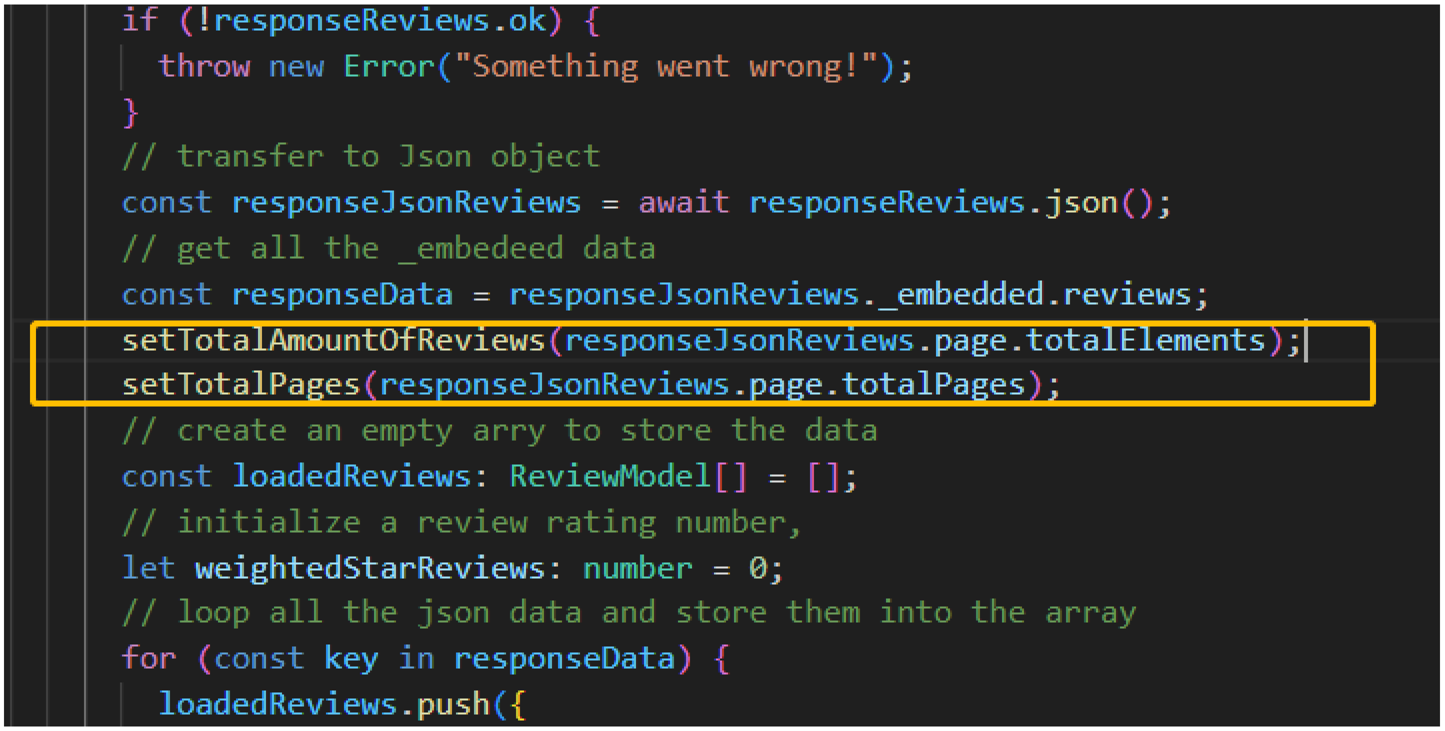


```
  useEffect(() => {
    const fetchBookReviews = async () => {
      // add page and size to  url, to call specific book
      const reviewUrl: string = `http://localhost:8080/api/reviews/search/findByBookId?bookId=${bookId}&page=${currentPage -
1}&size=${reviewsPerPage}`;
      const responseReviews = await fetch(reviewUrl);
      if (!responseReviews.ok) {
        throw new Error("Something went wrong!");
      }
```

Step 7 add:

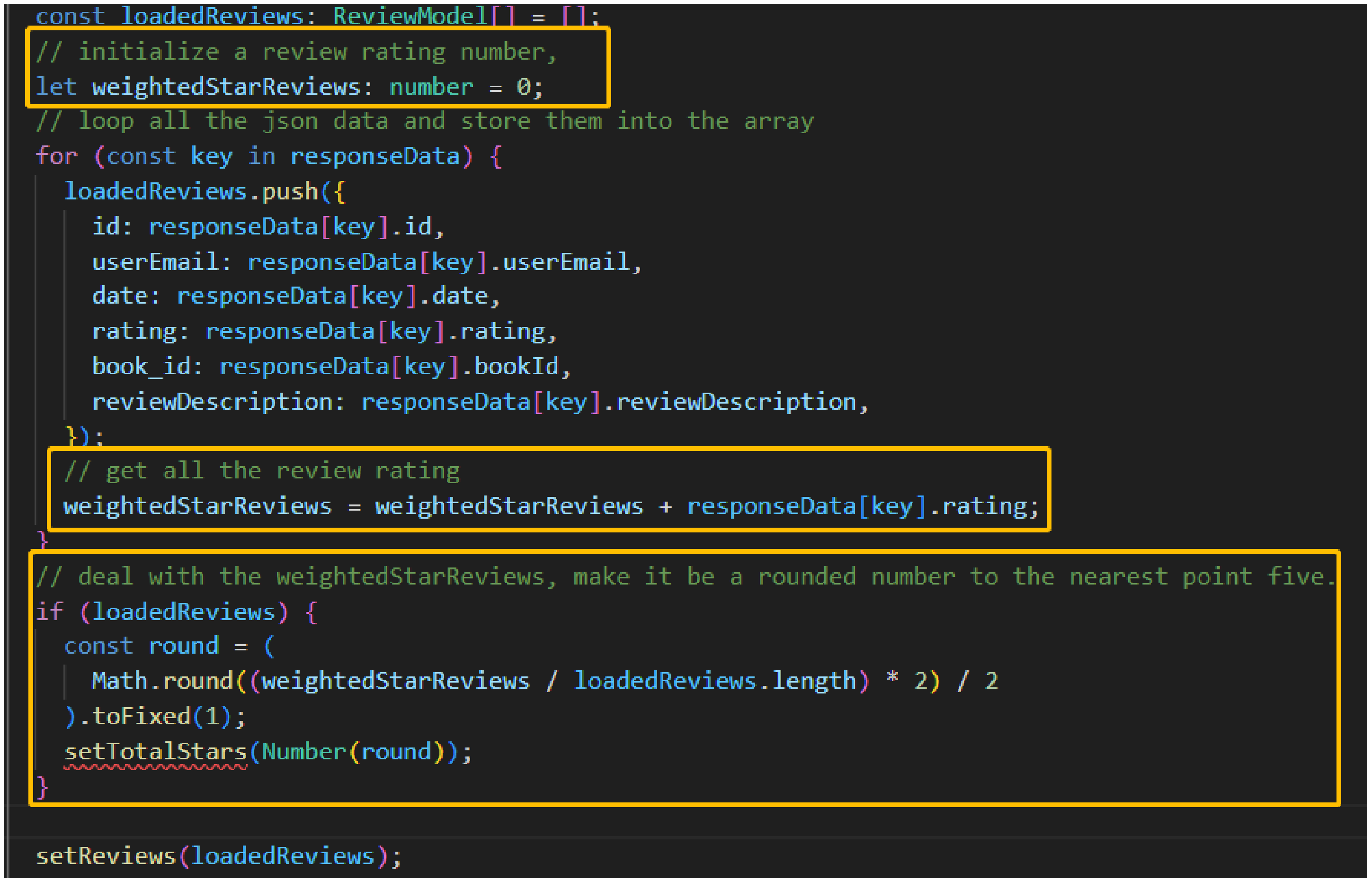
1. setTotalAmountOfReviews(responseJsonReviews.page.totalElements);

2. setTotalPages(responseJsonReviews.page.totalPages);



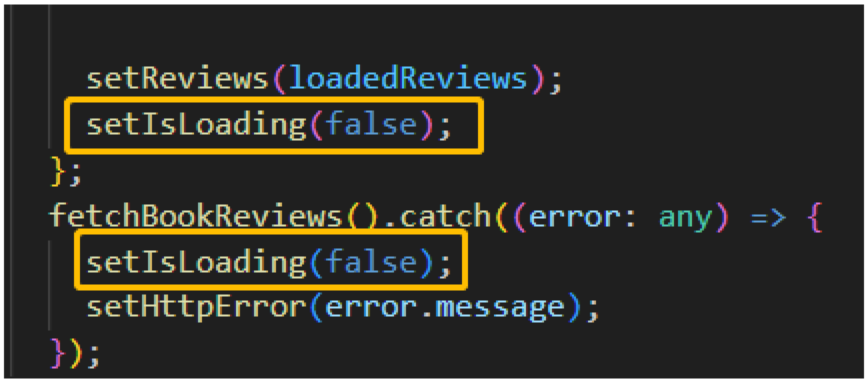
```
if (!responseReviews.ok) {
  throw new Error("Something went wrong!");
}
// transfer to Json object
const responseJsonReviews = await responseReviews.json();
// get all the _embedeed data
const responseData = responseJsonReviews._embedded.reviews;
setTotalAmountOfReviews(responseJsonReviews.page.totalElements);
setTotalPages(responseJsonReviews.page.totalPages);
// create an empty arry to store the data
const loadedReviews: ReviewModel[] = [];
// initialize a review rating number,
let weightedStarReviews: number = 0;
// loop all the json data and store them into the array
for (const key in responseData) {
  loadedReviews.push({
```

Step 8 Remove 3 parts of weightedStarReviews related code



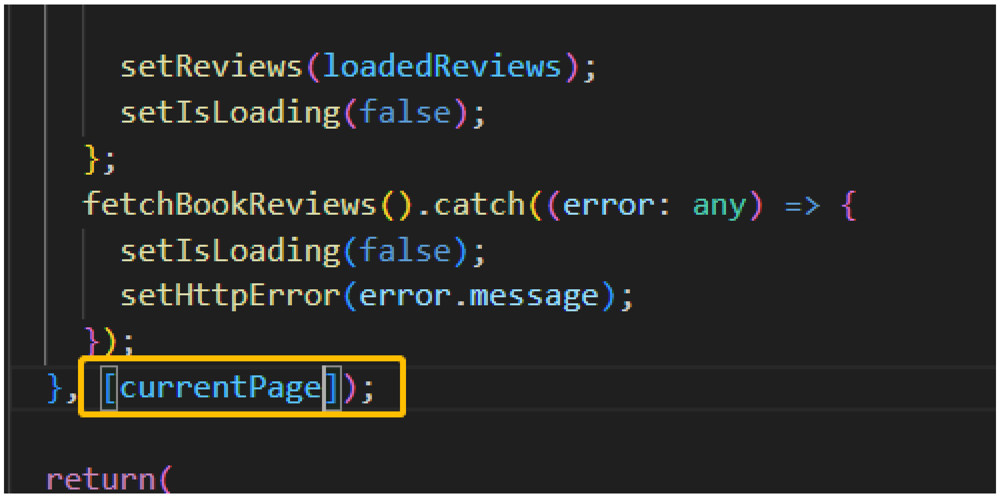
```
const loadedReviews: ReviewModel[] = [];
// initialize a review rating number,
let weightedStarReviews: number = 0;
// loop all the json data and store them into the array
for (const key in responseData) {
  loadedReviews.push({
    id: responseData[key].id,
    userEmail: responseData[key].userEmail,
    date: responseData[key].date,
    rating: responseData[key].rating,
    book_id: responseData[key].bookId,
    reviewDescription: responseData[key].reviewDescription,
  });
  // get all the review rating
  weightedStarReviews = weightedStarReviews + responseData[key].rating;
}
// deal with the weightedStarReviews, make it be a rounded number to the nearest point five.
if (loadedReviews) {
  const round = (
    Math.round((weightedStarReviews / loadedReviews.length) * 2) / 2
  ).toFixed(1);
  setTotalStars(Number(round));
}

setReviews(loadedReviews);
```

Step 9 Modify setIsLoadingReview to setIsLoading



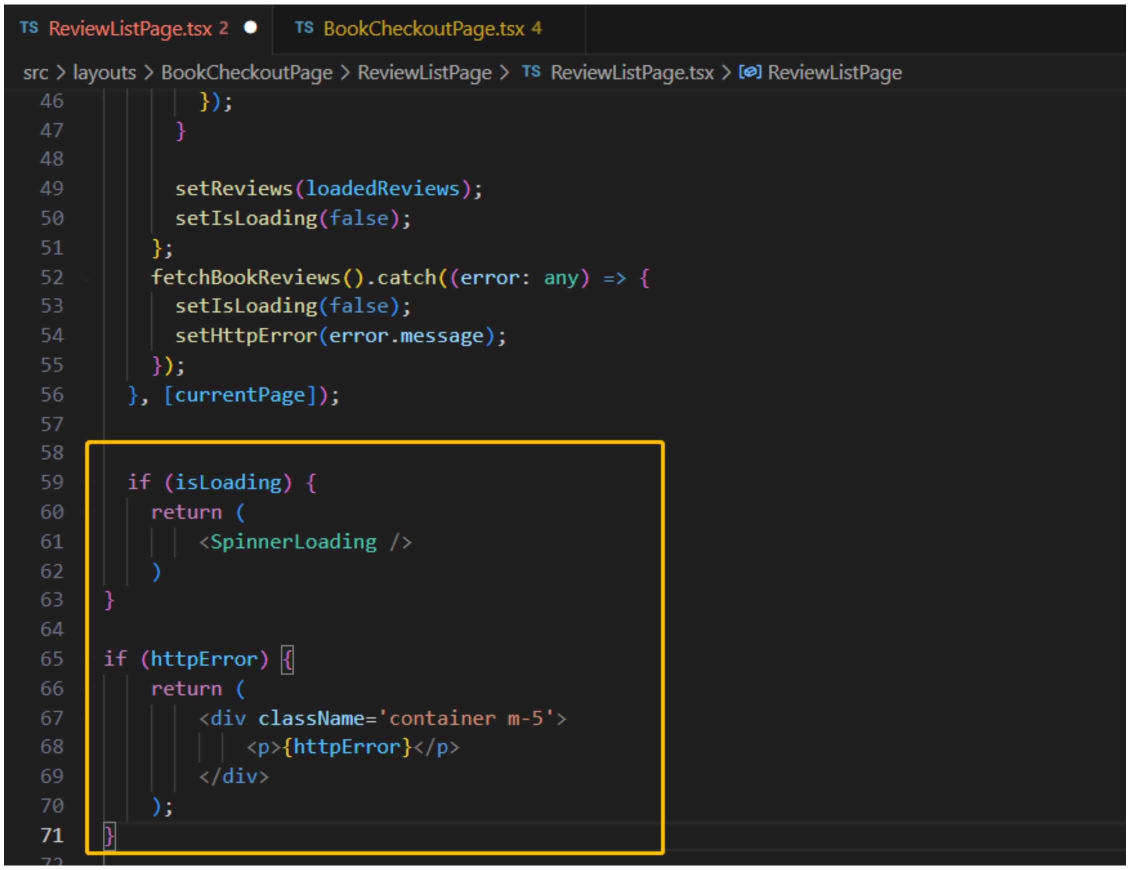
```
    setReviews(loadedReviews);
    setIsLoading(false);
  };
  fetchBookReviews().catch((error: any) => {
    setIsLoading(false);
    setHttpError(error.message);
  });
```

Step 10 Modify the state change factor for the useEffect.



```
      setReviews(loadedReviews);
      setIsLoading(false);
    };
    fetchBookReviews().catch((error: any) => {
      setIsLoading(false);
      setHttpError(error.message);
    });
  }, [currentPage]);

  return(
```

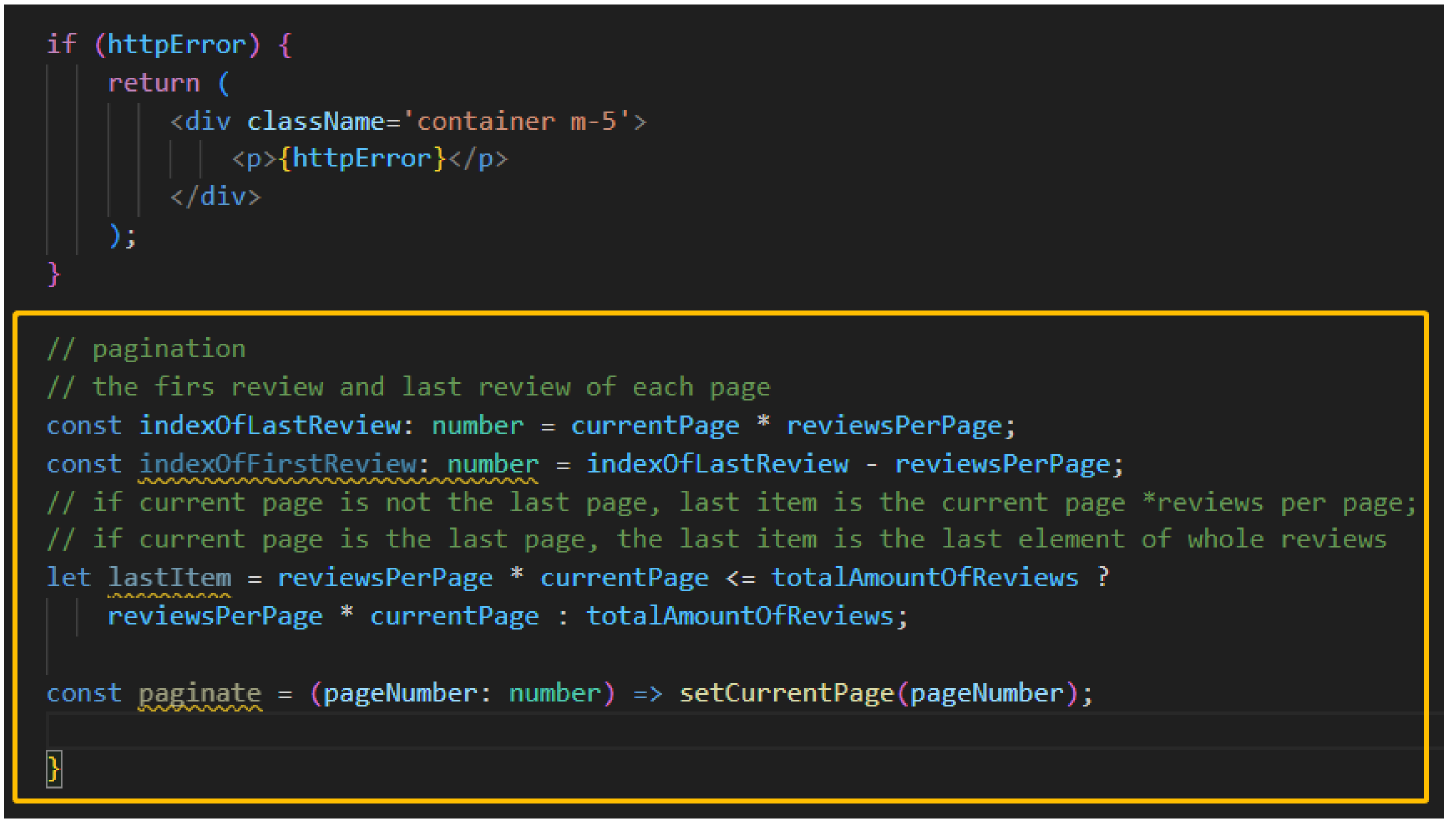
Step 11 Add spinnerLoading and httpError part of the code



TS ReviewListPage.tsx 2
TS BookCheckoutPage.tsx 4
src > layouts > BookCheckoutPage > ReviewListPage > TS ReviewListPage.tsx > ReviewListPage

```
                });
            }

            setReviews(loadedReviews);
            setIsLoading(false);
        };
        fetchBookReviews().catch((error: any) => {
            setIsLoading(false);
            setHttpError(error.message);
        });
    }, [currentPage]);


    if (isLoading) {
        return (
            <SpinnerLoading />
        )
}

if (httpError) {
        return (
            <div className='container m-5'>
                <p>{httpError}</p>
            </div>
        );
}
```

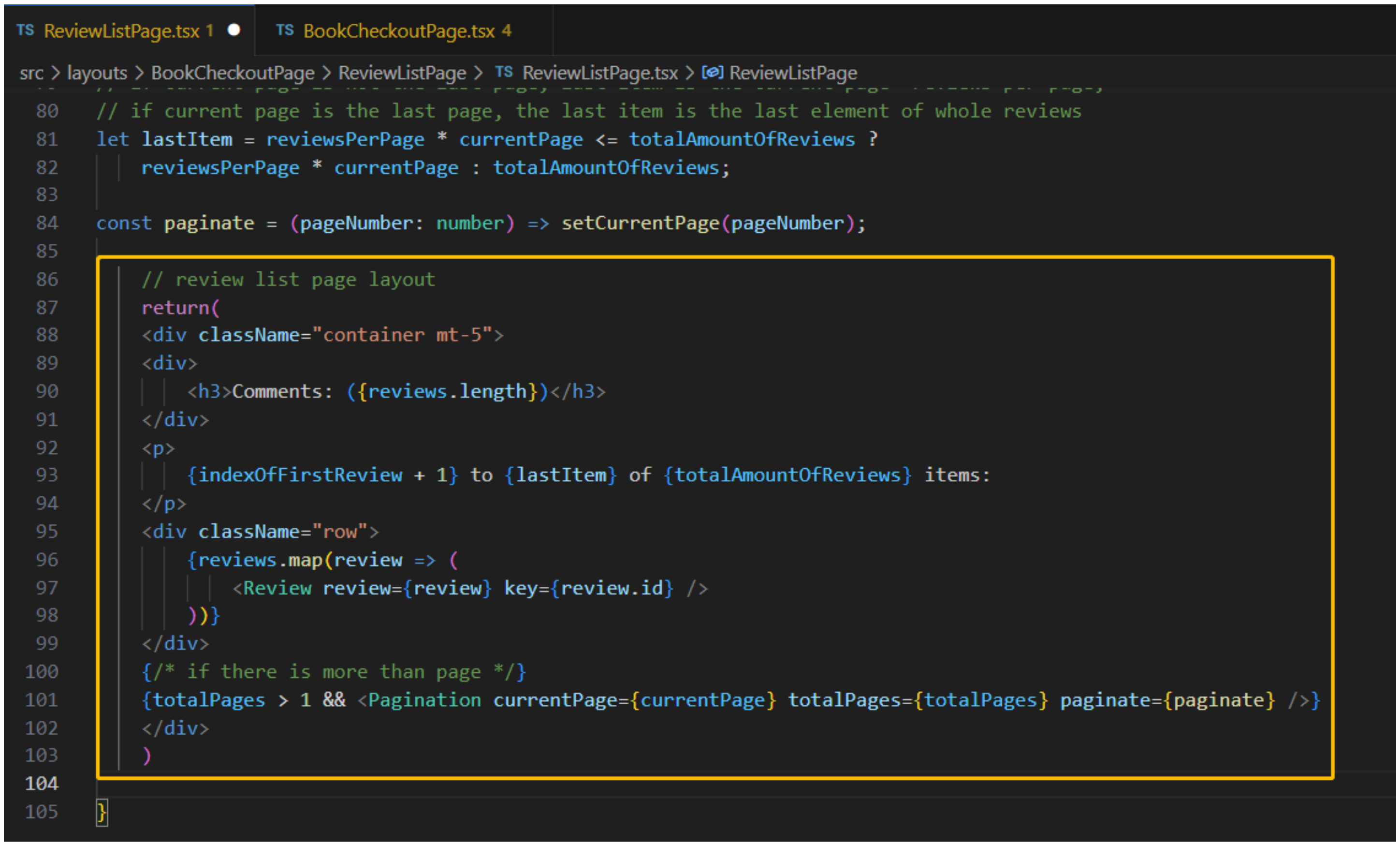
Step 12 Create code about pagination.



```
if (httpError) {
    return (
        <div className='container m-5'>
            <p>{httpError}</p>
        </div>
    );
}

// pagination
// the firs review and last review of each page
const indexOfLastReview: number = currentPage * reviewsPerPage;
const indexOfFirstReview: number = indexOfLastReview - reviewsPerPage;
// if current page is not the last page, last item is the current page *reviews per page;
// if current page is the last page, the last item is the last element of whole reviews
let lastItem = reviewsPerPage * currentPage <= totalAmountOfReviews ?
    reviewsPerPage * currentPage : totalAmountOfReviews;

const paginate = (pageNumber: number) => setCurrentPage(pageNumber);

}
```

2. LIST PAGE HTML/CSS

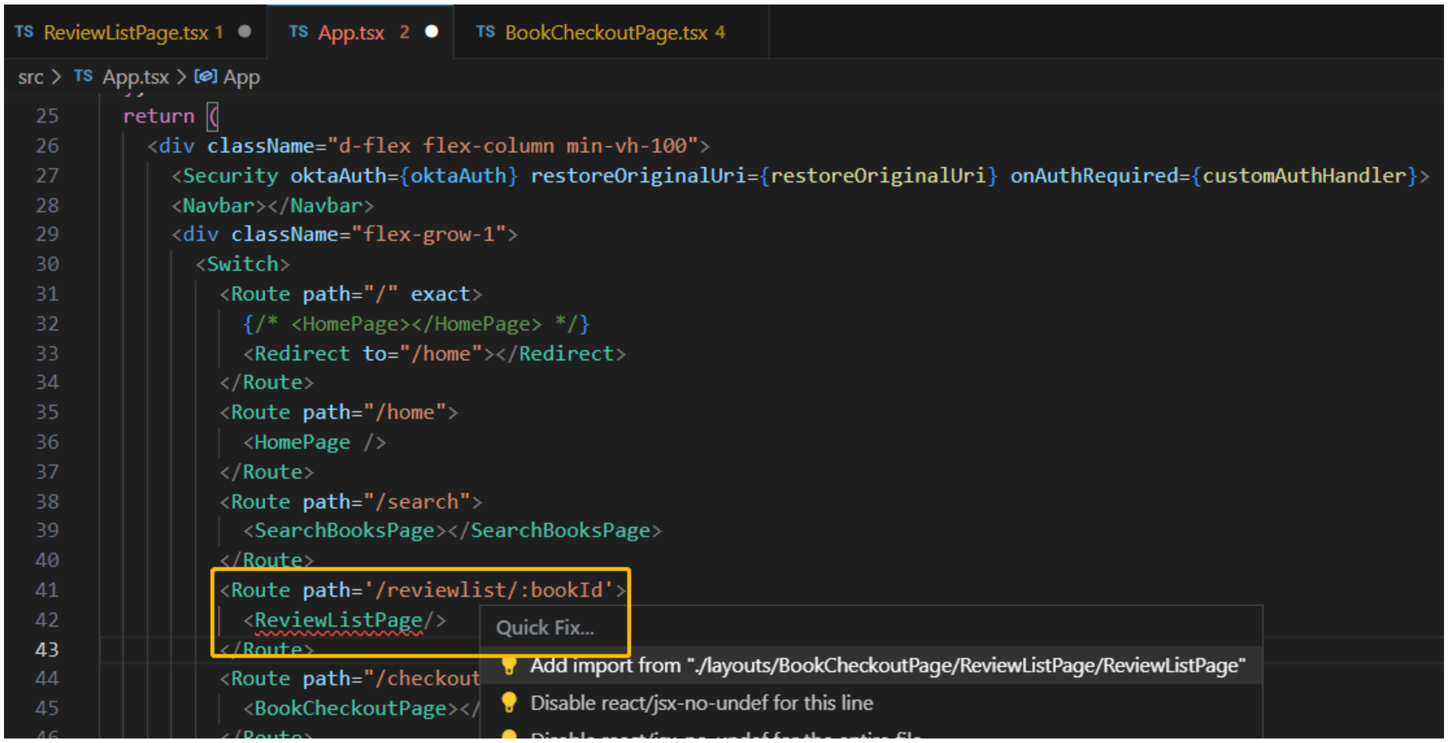
Fill return part with the HTML/CSS code:


TS ReviewListPage.tsx 1
TS BookCheckoutPage.tsx 4
src > layouts > BookCheckoutPage > ReviewListPage > TS ReviewListPage.tsx > ReviewListPage

```
// if current page is the last page, the last item is the last element of whole reviews
let lastItem = reviewsPerPage * currentPage <= totalAmountOfReviews ?
    reviewsPerPage * currentPage : totalAmountOfReviews;

const paginate = (pageNumber: number) => setCurrentPage(pageNumber);

    // review list page layout
    return(
    <div className="container mt-5">
    <div>
        <h3>Comments: ({reviews.length})</h3>
    </div>
    <p>
        {indexOfFirstReview + 1} to {lastItem} of {totalAmountOfReviews} items:
    </p>
    <div className="row">
        {reviews.map(review => (
            <Review review={review} key={review.id} />
        ))}
    </div>
    {/* if there is more than page */}
    {totalPages > 1 && <Pagination currentPage={currentPage} totalPages={totalPages} paginate={paginate} />}
    </div>
    )

}
```

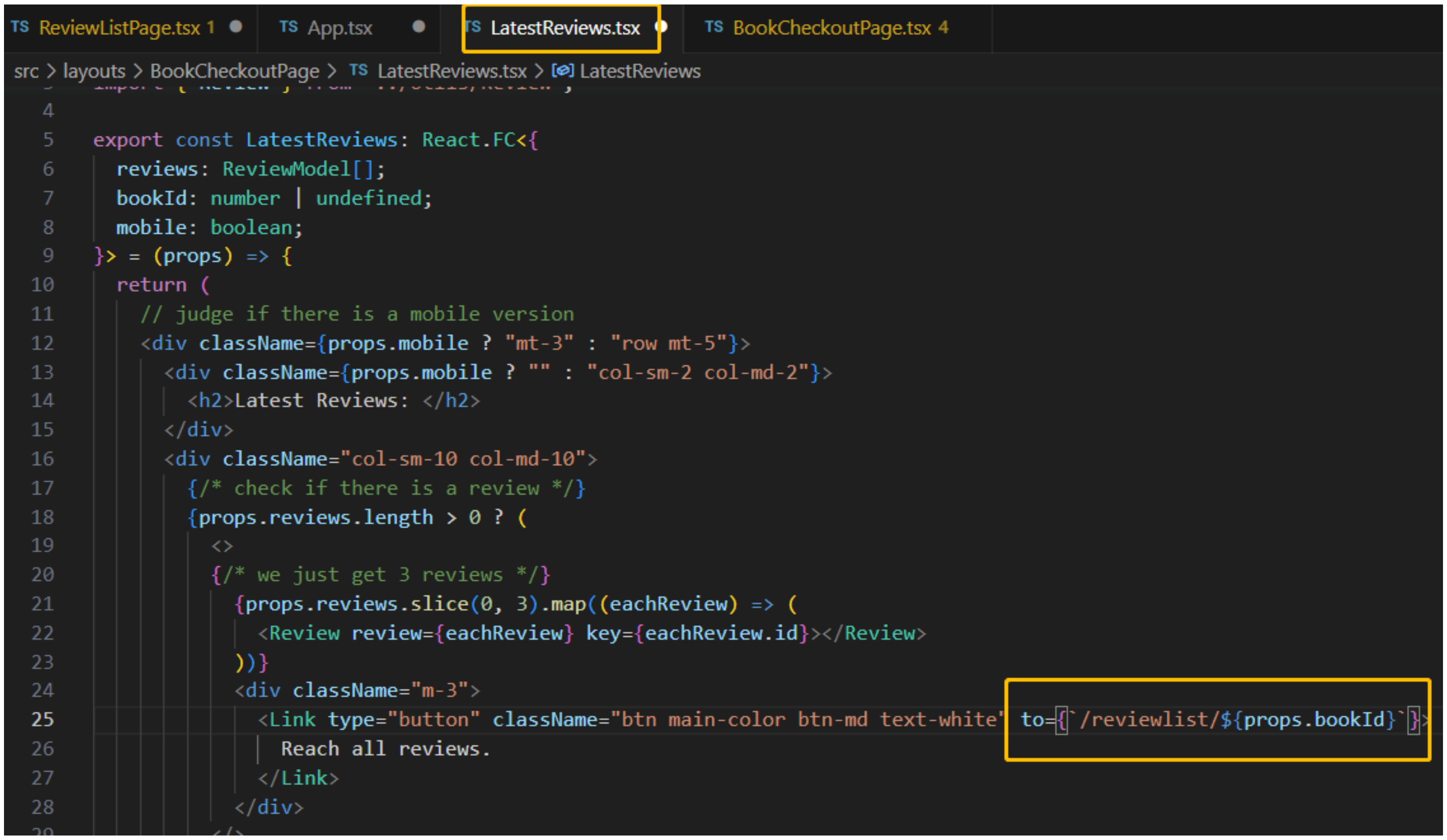
3. LIST PAGE ROUTING

Step 1 go to app.tsx, add route for ReviewListPage, and add import for the component.


TS ReviewListPage.tsx 1
TS App.tsx 2
TS BookCheckoutPage.tsx 4
src > TS App.tsx > App
Quick Fix...
Add import from "./layouts/BookCheckoutPage/ReviewListPage/ReviewListPage"
Disable react/jsx-no-undef for this line

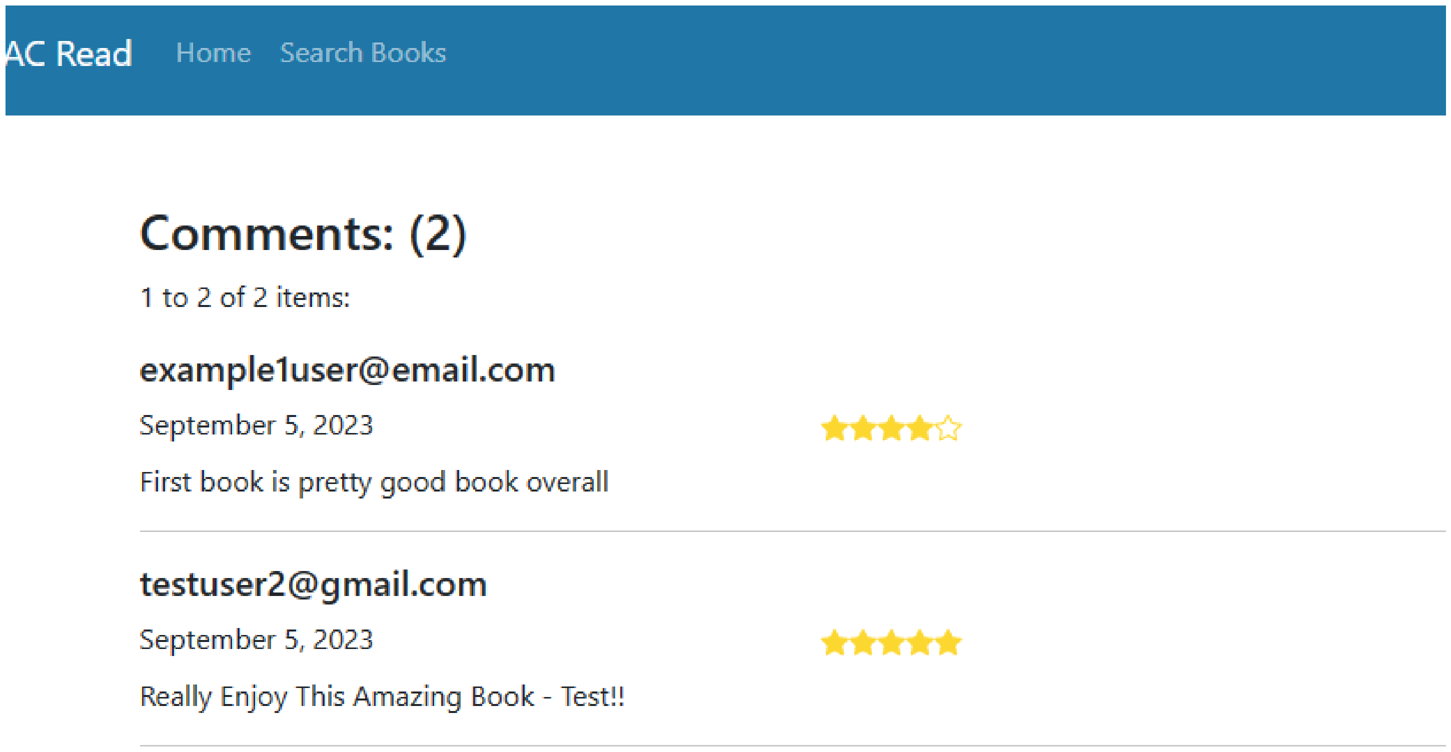
```
  return (
    <div className="d-flex flex-column min-vh-100">
      <Security oktaAuth={oktaAuth} restoreOriginalUri={restoreOriginalUri} onAuthRequired={customAuthHandler}>
      <Navbar></Navbar>
      <div className="flex-grow-1">
        <Switch>
          <Route path="/" exact>
            {/* <HomePage></HomePage> */}
            <Redirect to="/home"></Redirect>
          </Route>
          <Route path="/home">
            <HomePage />
          </Route>
          <Route path="/search">
            <SearchBooksPage></SearchBooksPage>
          </Route>
          <Route path='/reviewlist/:bookId'>
            <ReviewListPage/>
          </Route>
          <Route path="/checkout
            <BookCheckoutPage></
```

Step 2 In the LatestReviews.tsx, add a redirect path to the “Reach all reviews” button. It will redirect us to the review list page of a specified book.


TS ReviewListPage.tsx 1
TS App.tsx
TS LatestReviews.tsx
TS BookCheckoutPage.tsx 4
src > layouts > BookCheckoutPage > TS LatestReviews.tsx > LatestReviews

```

export const LatestReviews: React.FC<{
  reviews: ReviewModel[];
  bookId: number | undefined;
  mobile: boolean;
}> = (props) => {
  return (
    // judge if there is a mobile version
    <div className={props.mobile ? "mt-3" : "row mt-5"}>
      <div className={props.mobile ? "" : "col-sm-2 col-md-2"}>
        <h2>Latest Reviews: </h2>
      </div>
      <div className="col-sm-10 col-md-10">
        {/* check if there is a review */}
        {props.reviews.length > 0 ? (
          <>
          {/* we just get 3 reviews */}
            {props.reviews.slice(0, 3).map((eachReview) => (
              <Review review={eachReview} key={eachReview.id}></Review>
            ))}
            <div className="m-3">
              <Link type="button" className="btn main-color btn-md text-white" to={`/reviewlist/${props.bookId}`}>
                Reach all reviews.
              </Link>
            </div>
```

Click on the “Reach all reviews” button will go to the review list page.



Comments: (2)

1 to 2 of 2 items:

example1user@email.com

September 5, 2023 ★★★★☆

First book is pretty good book overall

testuser2@gmail.com

September 5, 2023 ★★★★★

Really Enjoy This Amazing Book - Test!!