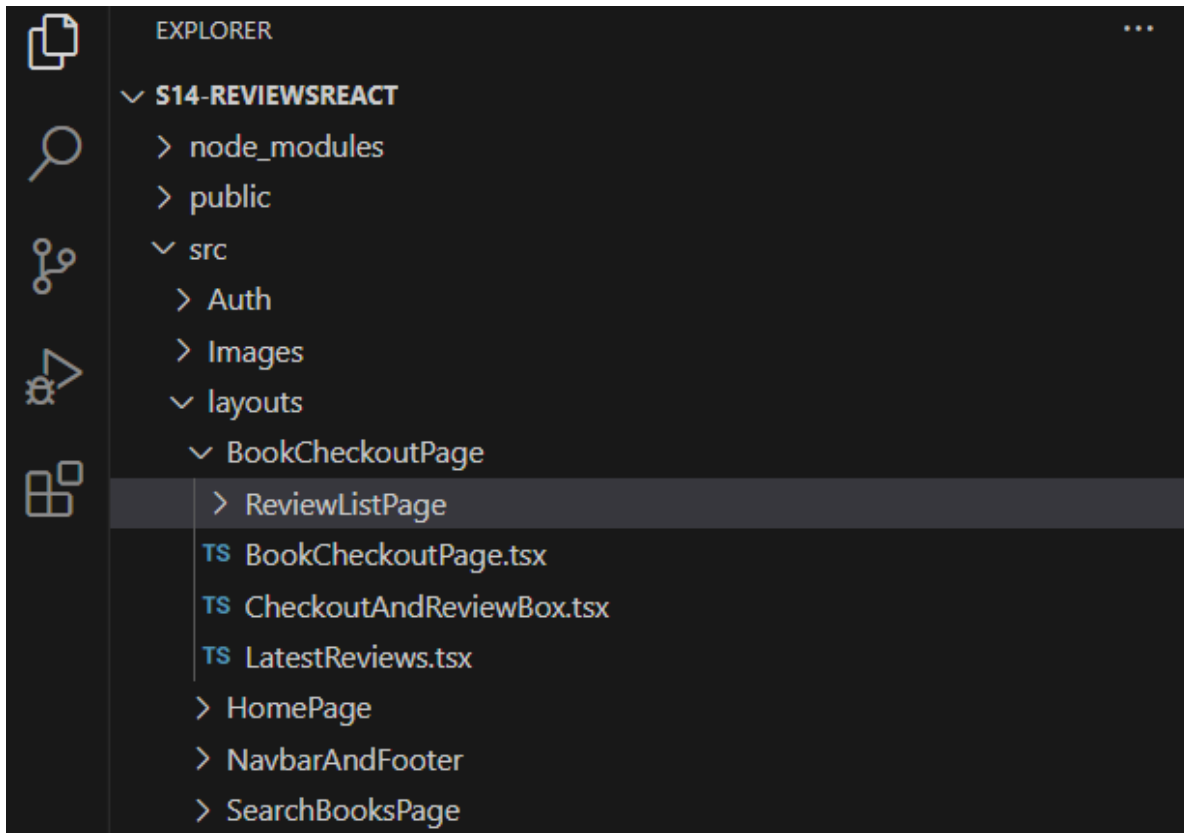


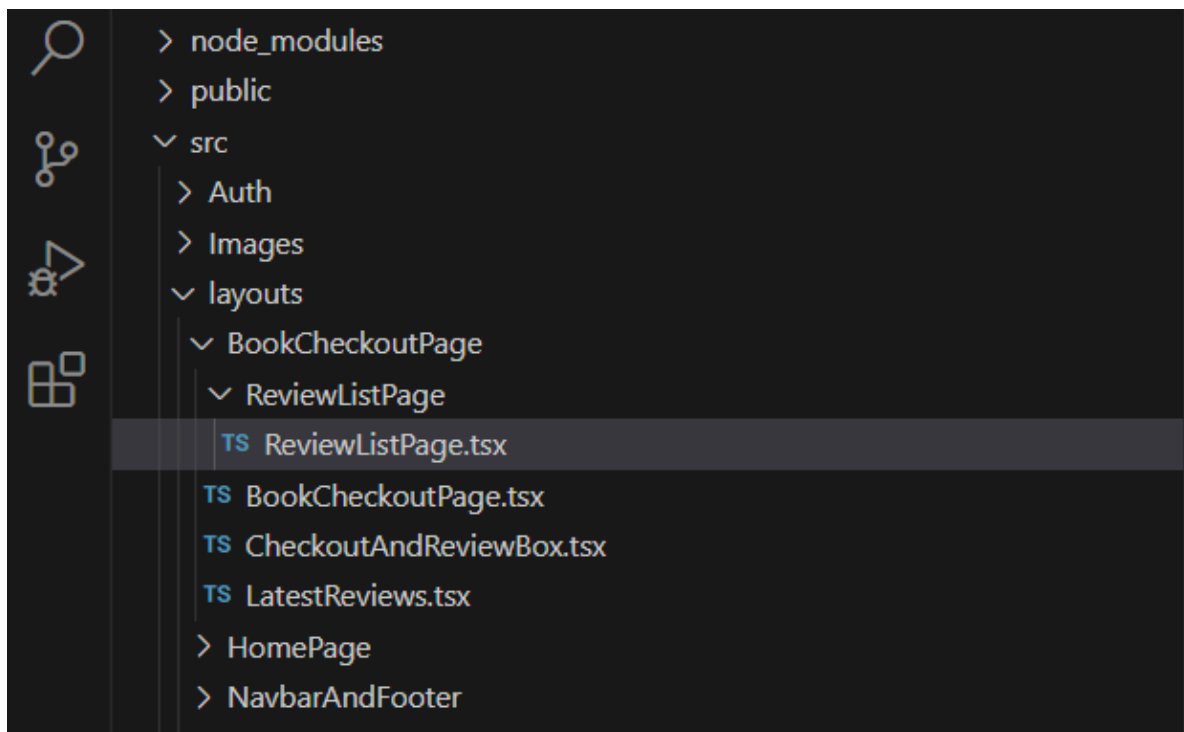
S15-Review List Page React

1. List Page and UseState and UseEffect

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



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



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

```
> node_modules
> public
✓ src
  > Auth
  > Images
  ✓ layouts
    ✓ BookCheckoutPage
      ✓ ReviewListPage
        TS ReviewListPage.tsx 2
      TS BookCheckoutPage.tsx
      TS CheckoutAndReviewBox.tsx
      TS LatestReviews.tsx
    > HomePage
    > NavbarAndFooter
```

```
1  export const ReviewListPage = () => {
2    return(
3
4  }
```

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

```
TS ReviewListPage.tsx 9 ●
src > layouts > BookCheckoutPage > ReviewListPage > TS ReviewListPage.tsx > [🔗] ReviewListPage
1  import ReviewModel from "../../models/ReviewModel";
2
3  export const ReviewListPage = () => {
4
5    // create useState for reviews
6    const [reviews, setReviews] = useState<ReviewModel[]>([]);
7    const [isLoading, setIsLoading] = useState(true);
8    const [httpError, setHttpError] = useState(null);
9
10   // pagination
11   const [currentPage, setCurrentPage] = useState(1);
12   const [reviewsPerPage] = useState(5);
13   const [totalAmountOfReviews, setTotalAmountOfReviews] = useState(0);
14   const [totalPages, setTotalPages] = useState(0);
15
16   // Book to look up reviews, the third element of the url
17   const bookId = (window.location.pathname).split('/')[2];
18
19   return(
20   )
21 }
```

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
14   const [totalPages, setTotalPages] = useState(0);
15
16   // Book to look up reviews, the third element of the url
17   const bookId = (window.location.pathname).split('/')[2];
18
19   useEffect(() => {
20     const fetchBookReviews = async () => {
21       // review url, to call specific book
22       const reviewUrl: string = `http://localhost:8080/api/reviews/search/findByBookId?bookId=${bookId}`;
23       const responseReviews = await fetch(reviewUrl);
24       if (!responseReviews.ok) {
25         throw new Error("Something went wrong!");
26       }
27       // transfer to Json object
28       const responseJsonReviews = await responseReviews.json();
29       // get all the _embedded data
30       const responseData = responseJsonReviews._embedded.reviews;
31       // create an empty array to store the data
32       const loadedReviews: ReviewModel[] = [];
33       // initialize a review rating number,
34       let weightedStarReviews: number = 0;
35       // loop all the json data and store them into the array
36       for (const key in responseData) {
37         loadedReviews.push({
38           id: responseData[key].id,
39           userEmail: responseData[key].userEmail,
40           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
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
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 _embedded data
    const responseData = responseJsonReviews._embedded.reviews;
    // create an empty array 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}&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 _embedded data
const responseData = responseJsonReviews._embedded.reviews;
setTotalAmountOfReviews(responseJsonReviews.page.totalElements);
setTotalPages(responseJsonReviews.page.totalPages);
// create an empty array 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
46     });
47   }
48
49   setReviews(loadedReviews);
50   setIsLoading(false);
51 };
52 fetchBookReviews().catch((error: any) => {
53   setIsLoading(false);
54   setHttpError(error.message);
55 });
56 }, [currentPage]);
57
58
59 if (isLoading) {
60   return (
61     <SpinnerLoading />
62   )
63 }
64
65 if (httpError) {
66   return (
67     <div className='container m-5'>
68       <p>{httpError}</p>
69     </div>
70   );
71 }

```

Step 12 Create code about pagination.

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

// pagination
// the first 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);
}
```

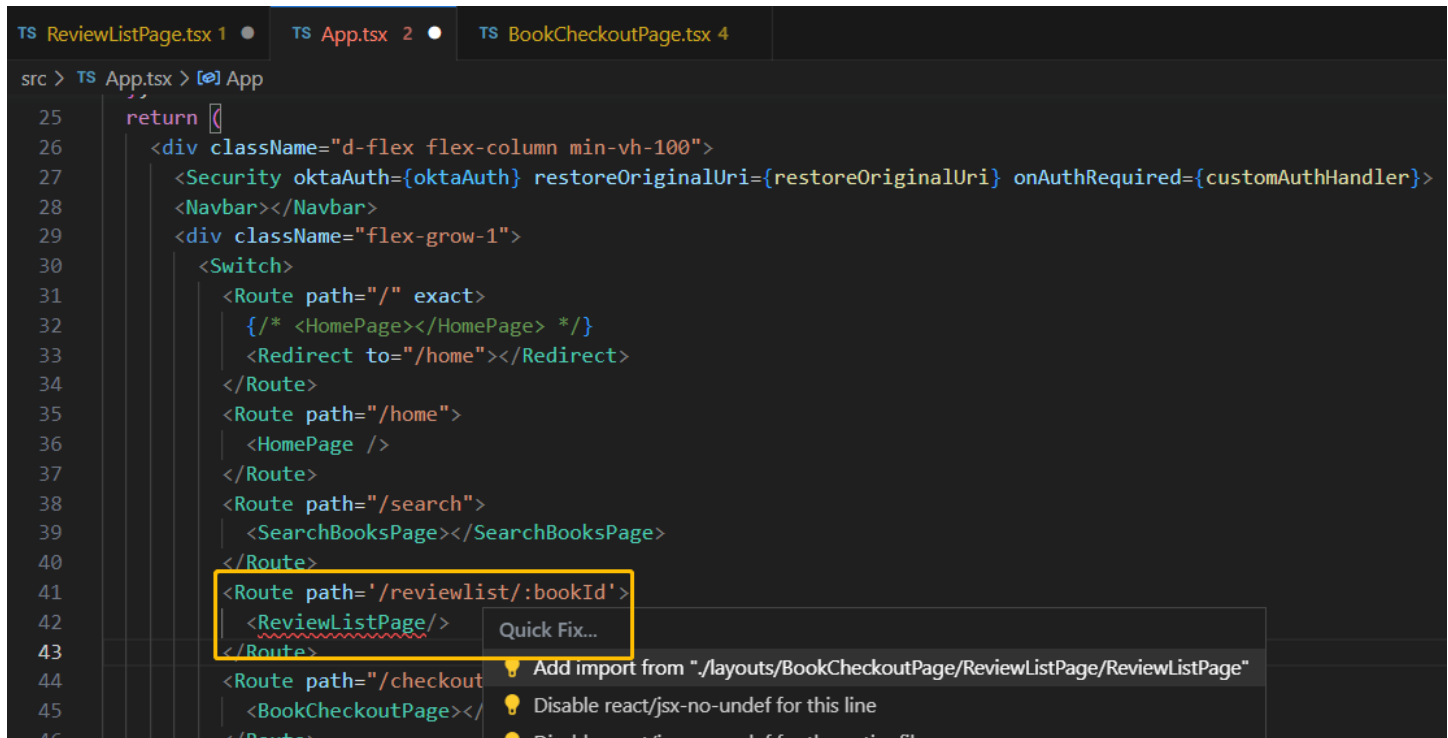
2. LIST PAGE HTML/CSS

Fill return part with the HTML/CSS code:

```
TS ReviewListPage.tsx 1 • TS BookCheckoutPage.tsx 4
src > layouts > BookCheckoutPage > ReviewListPage > TS ReviewListPage.tsx > [0] ReviewListPage
80 // if current page is the last page, the last item is the last element of whole reviews
81 let lastItem = reviewsPerPage * currentPage <= totalAmountOfReviews ?
82   reviewsPerPage * currentPage : totalAmountOfReviews;
83
84 const paginate = (pageNumber: number) => setCurrentPage(pageNumber);
85
86 // review list page layout
87 return(
88   <div className="container mt-5">
89     <div>
90       <h3>Comments: ({reviews.length})</h3>
91     </div>
92     <p>
93       {indexOfFirstReview + 1} to {lastItem} of {totalAmountOfReviews} items:
94     </p>
95     <div className="row">
96       {reviews.map(review => (
97         <Review review={review} key={review.id} />
98       ))}
99     </div>
100     { /* if there is more than page */ }
101     {totalPages > 1 && <Pagination currentPage={currentPage} totalPages={totalPages} paginate={paginate} />}
102   </div>
103 )
104
105 }
```

3. LIST PAGE ROUTING

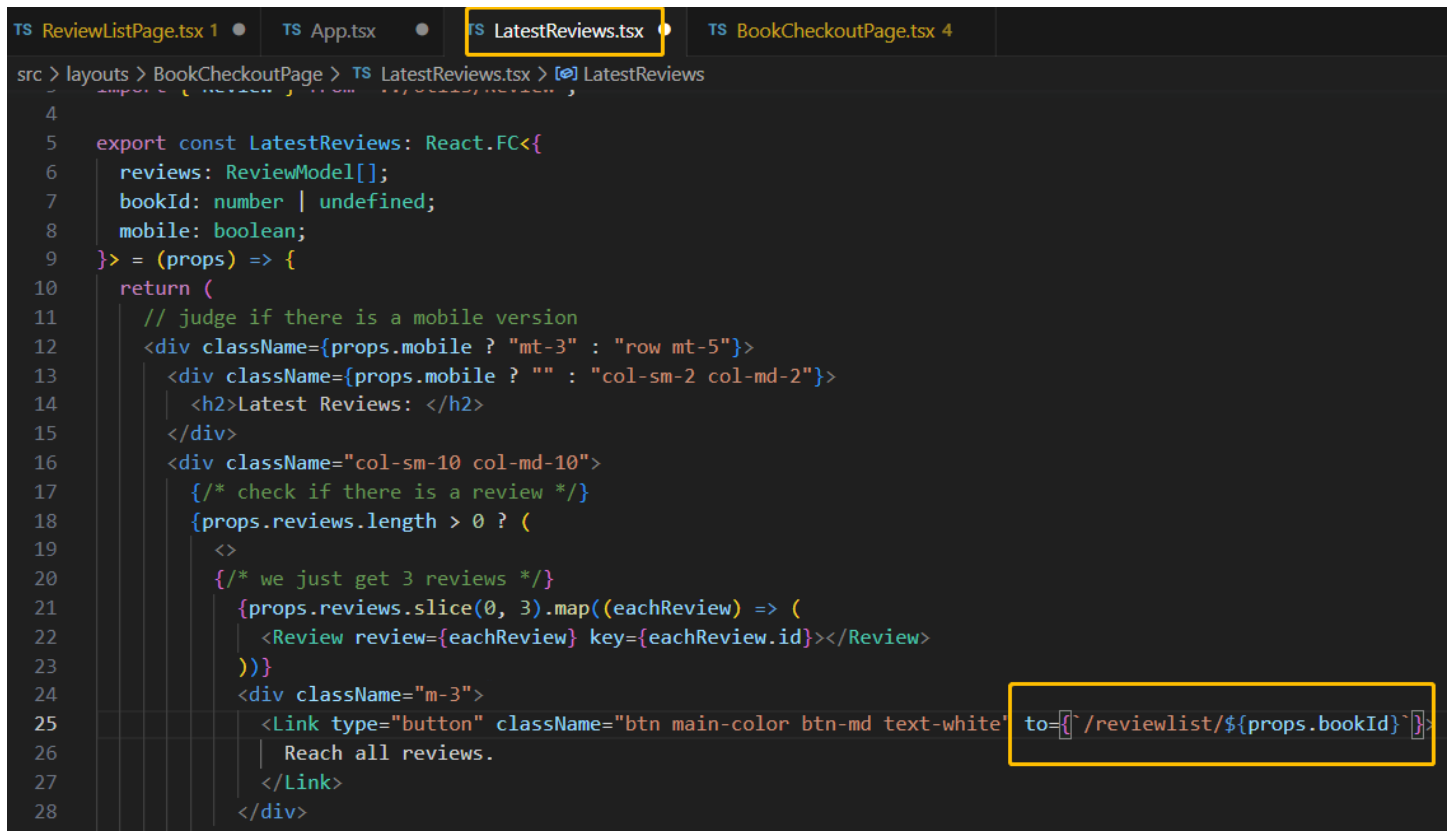
Step 1 go to app.tsx, add route for ReviewListPage, and add import for the component.



The screenshot shows the `App.tsx` file in VS Code. The `src > TS App.tsx` tab is active. The code is a React component that returns a `div` with a `Switch` for routing. A new route is being added at line 41: `<Route path='/reviewlist/:bookId'>`. The `<ReviewListPage />` line is highlighted with a yellow box, and a "Quick Fix..." menu is open, suggesting to "Add import from './layouts/BookCheckoutPage/ReviewListPage/ReviewListPage'".

```
src > TS App.tsx > [?] App
25   return [
26     <div className="d-flex flex-column min-vh-100">
27       <Security oktaAuth={oktaAuth} restoreOriginalUri={restoreOriginalUri} onAuthRequired={customAuthHandler}>
28       <Navbar></Navbar>
29       <div className="flex-grow-1">
30         <Switch>
31           <Route path="/" exact>
32             { /* <HomePage></HomePage> */ }
33             <Redirect to="/home"></Redirect>
34           </Route>
35           <Route path="/home">
36             <HomePage />
37           </Route>
38           <Route path="/search">
39             <SearchBooksPage></SearchBooksPage>
40           </Route>
41           <Route path='/reviewlist/:bookId'>
42             <ReviewListPage />
43           </Route>
44           <Route path="/checkout">
45             <BookCheckoutPage></BookCheckoutPage>
46           </Route>
```

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.



The screenshot shows the `LatestReviews.tsx` file in VS Code. The `src > layouts > BookCheckoutPage > TS LatestReviews.tsx` tab is active. The code defines a `LatestReviews` component. A `Link` component is being added at line 25, with the `to` prop highlighted in a yellow box: `to={['/reviewlist/${props.bookId}']}`.

```
src > layouts > BookCheckoutPage > TS LatestReviews.tsx > [?] LatestReviews
4
5   export const LatestReviews: React.FC<{
6     reviews: ReviewModel[];
7     bookId: number | undefined;
8     mobile: boolean;
9   }> = (props) => {
10     return (
11       // judge if there is a mobile version
12       <div className={props.mobile ? "mt-3" : "row mt-5"}>
13         <div className={props.mobile ? "" : "col-sm-2 col-md-2"}>
14           <h2>Latest Reviews: </h2>
15         </div>
16         <div className="col-sm-10 col-md-10">
17           { /* check if there is a review */ }
18           {props.reviews.length > 0 ? (
19             <>
20               { /* we just get 3 reviews */ }
21               {props.reviews.slice(0, 3).map((eachReview) => (
22                 <Review review={eachReview} key={eachReview.id}></Review>
23               ))}
24             <div className="m-3">
25               <Link type="button" className="btn main-color btn-md text-white" to={['/reviewlist/${props.bookId}']}>
26                 Reach all reviews.
27               </Link>
28             </div>
29           ) : null}
```


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!!
