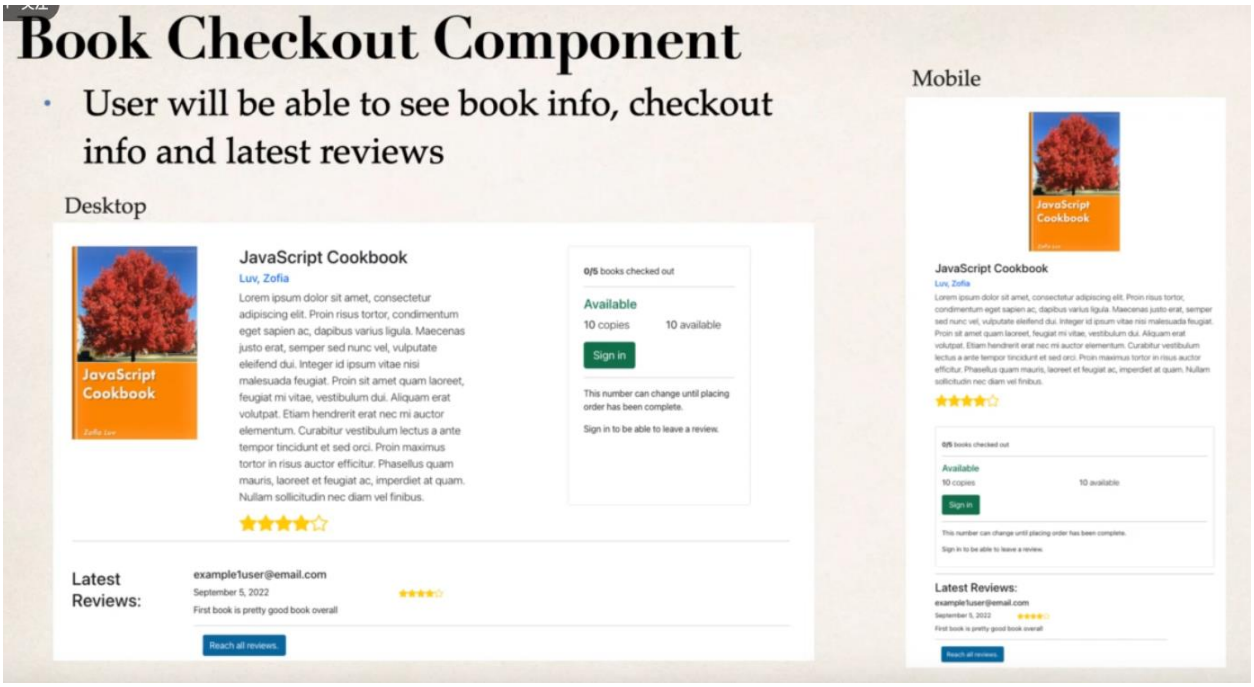
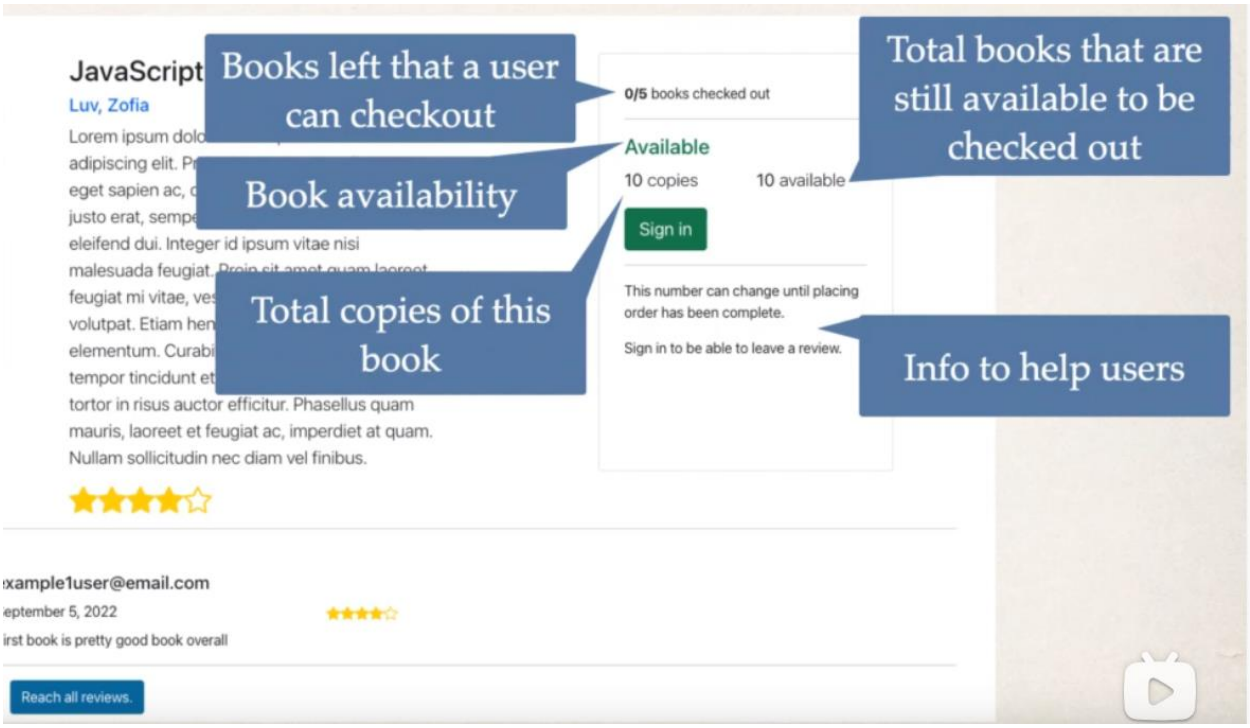
S06 Book Checkout

BOOK CHECKOUT COMPONENT OVERVIEW

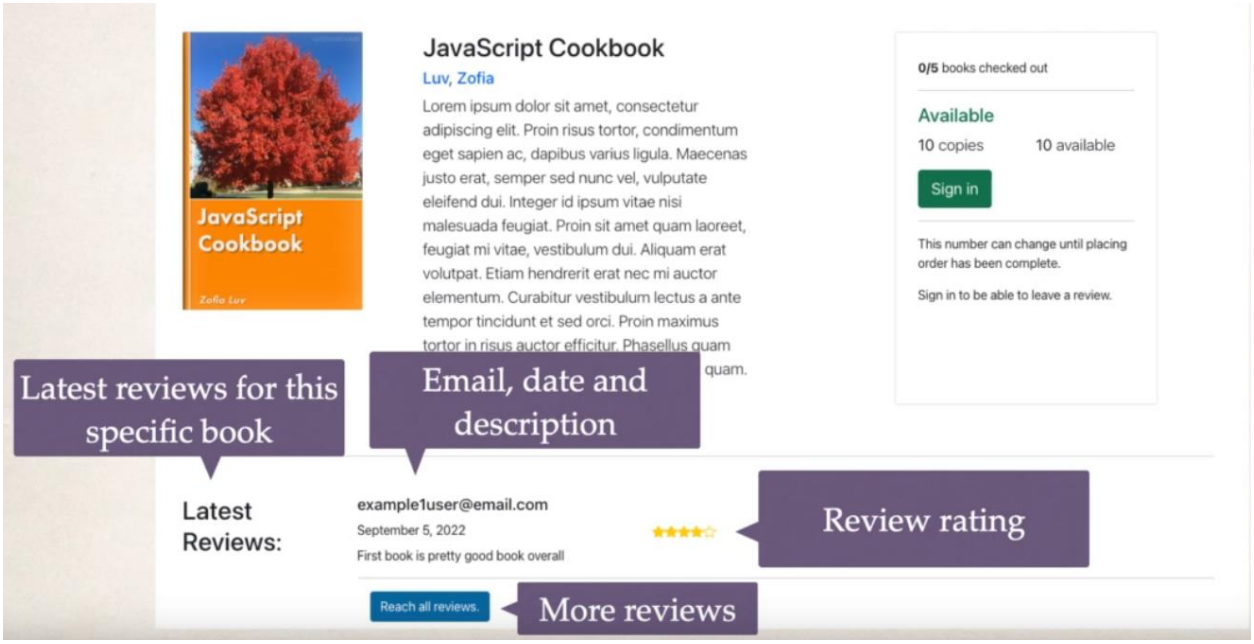


Book info Module:

Checkout info Module:



Latest review Module:



Book Checkout Component

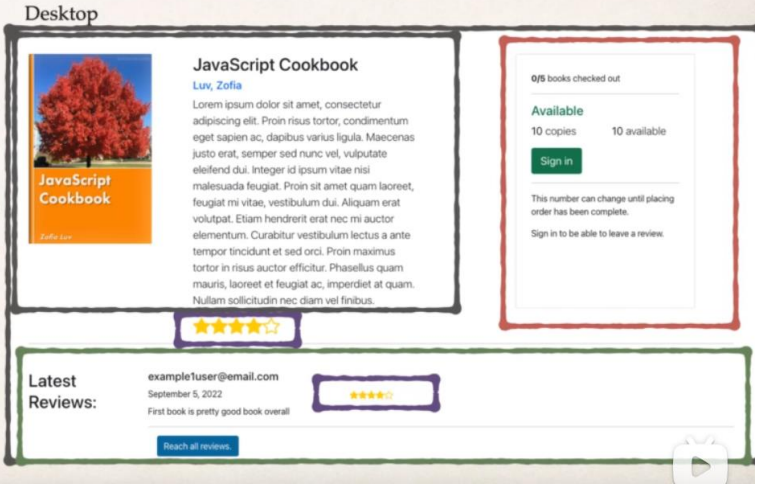
- Four new TypeScript components

BookCheckoutPage.tsx

CheckoutAndReviewBox.tsx

LatestReviews.tsx

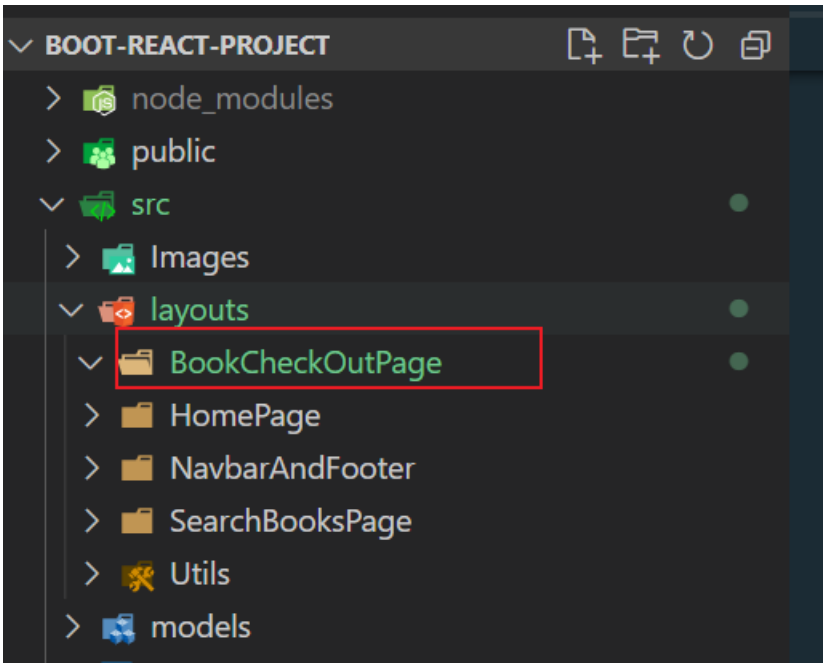
StarsReview.tsx



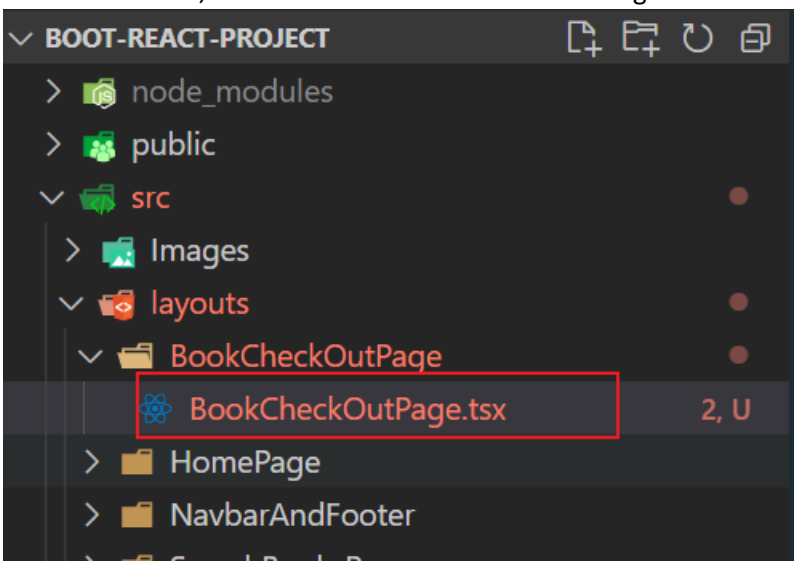
PART A.

BOOK CHECKOUT PAGE IMPLEMENTATION

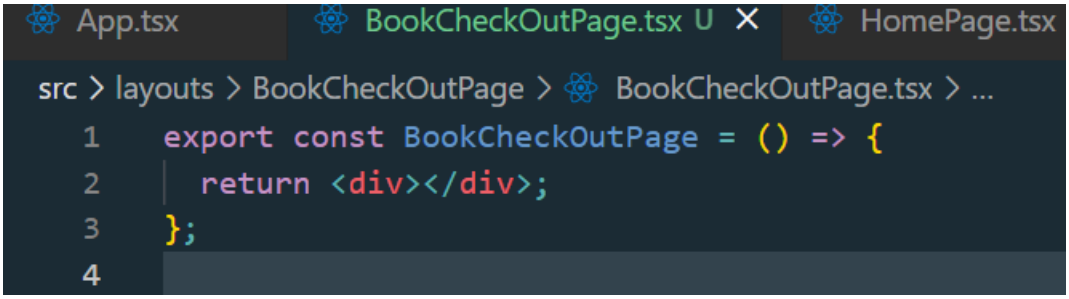
Under “layouts” folder, create a new folder , name it “BookCheckoutPage”.



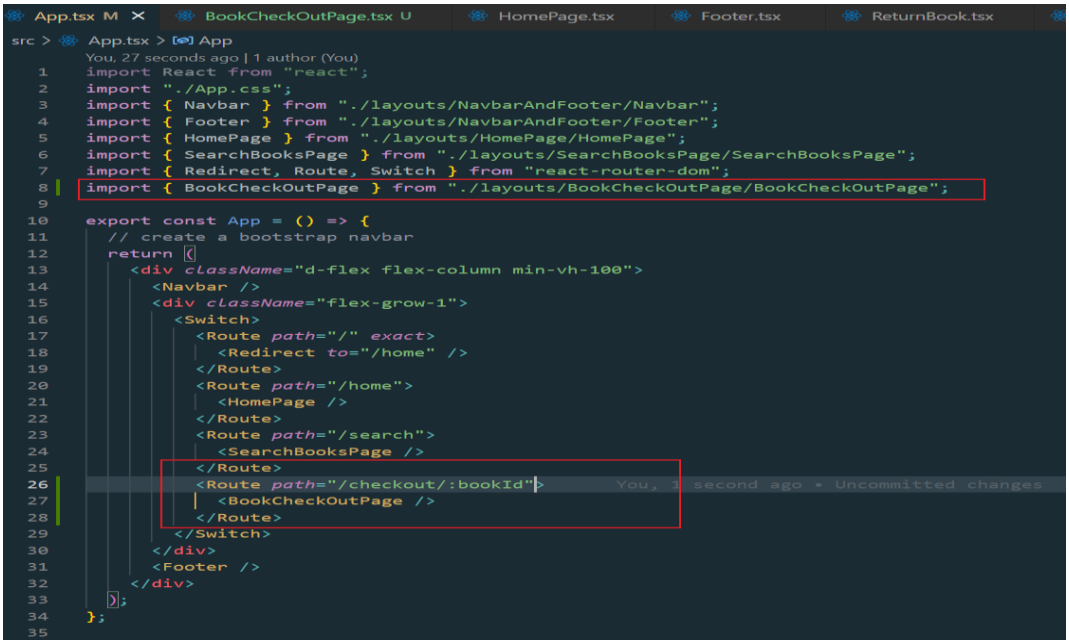
Inside this folder, create a new file “BookCheckOutPage.tsx”



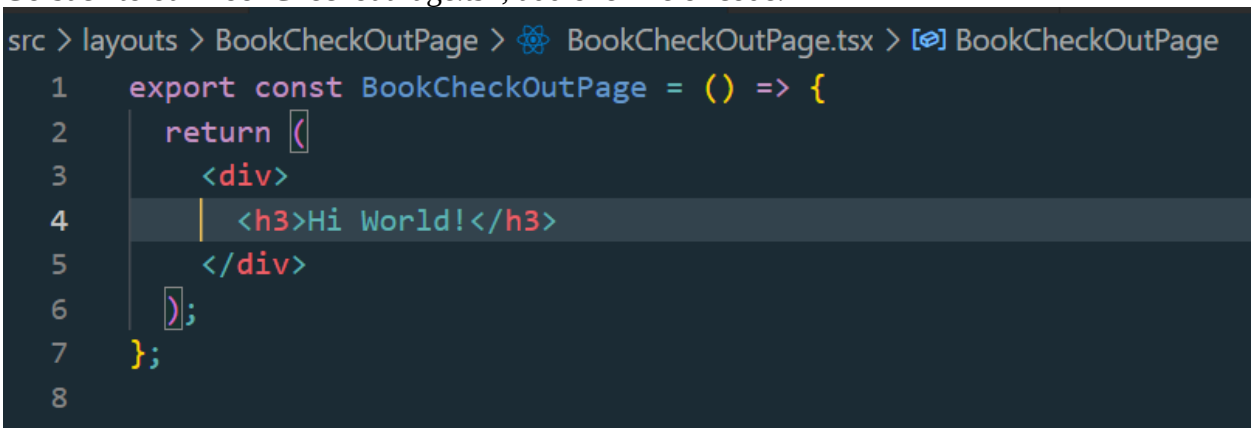
Write the code like this:



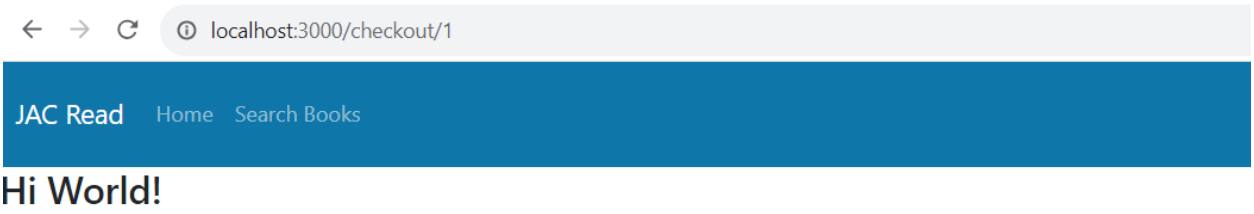
Go to App.tsx file, between Switch tag ,add a route to our BookCheckoutPage, and then the BookCheckoutPage component will automatically imported on the top of the page.



Go back to our BookCheckoutPage.tsx, add one line of code.

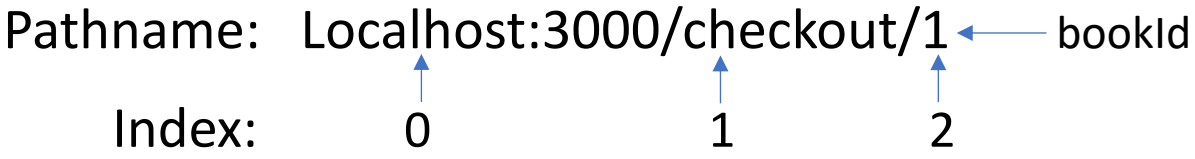


Run the application, open our browser, and visit http://localhost:3000/checkout/1, we will get the page like this: We have Nav bar, our h3 content and the footer, which means that our route works

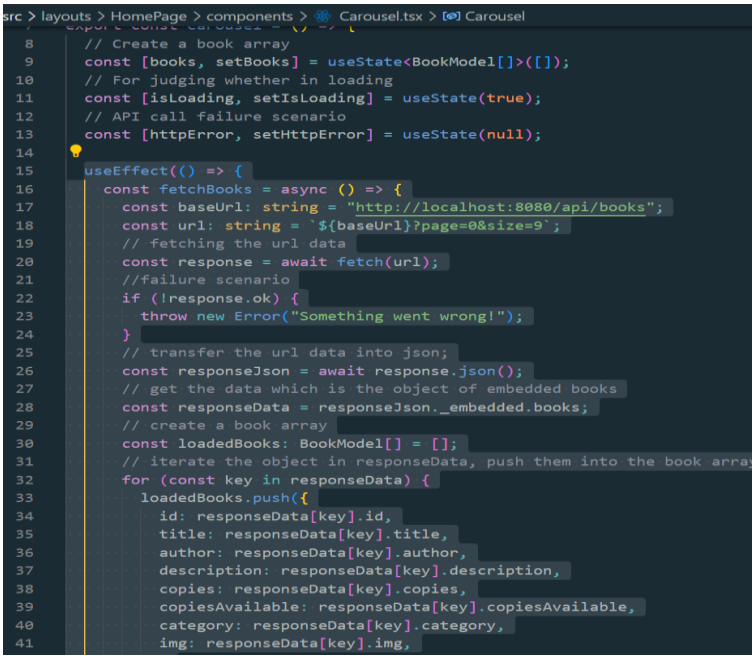


In the BookCheckoutPage.tsx, we first create UseState for book object, loading process , httpError and the book Id.

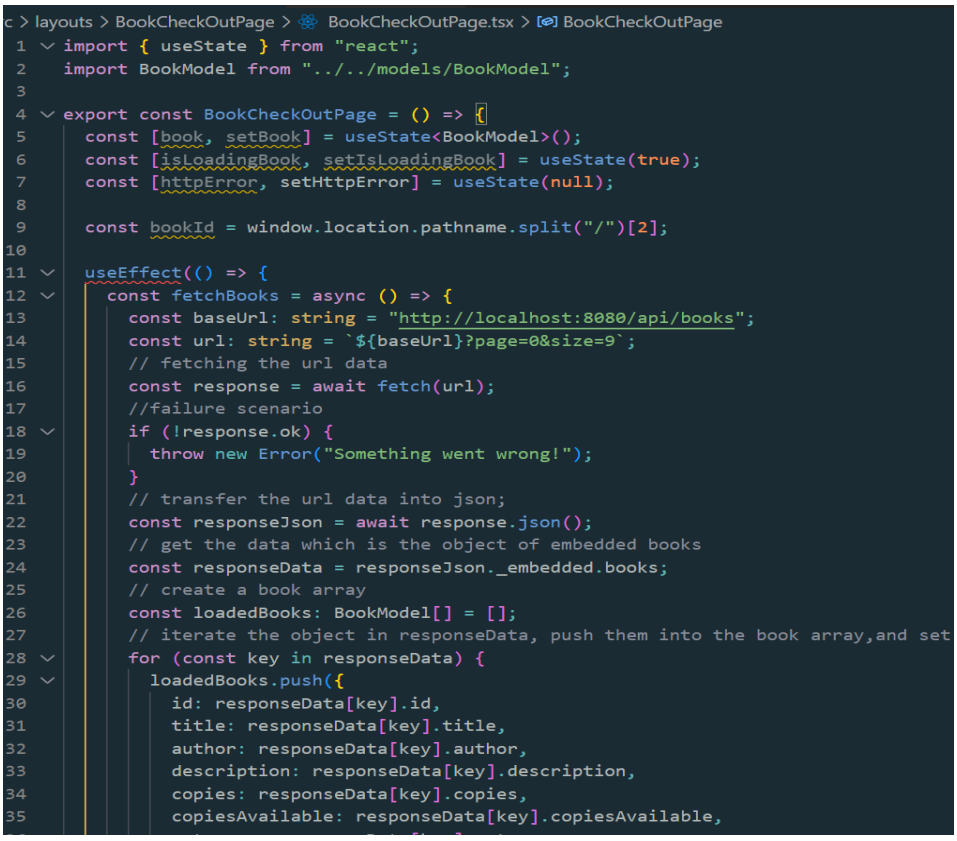
```tsx
import { useState } from "react";
import BookModel from "../../models/BookModel";

export const BookCheckOutPage = () => {
  const [book, setBook] = useState<BookModel>();
  const [isLoadingBook, setIsLoadingBook] = useState(true);
  const [httpError, setHttpError] = useState(null);

  const bookId = window.location.pathname.split("/")[2];

  return (
    <div>
      <h3>Hi World!</h3>
    </div>
  );
};
```



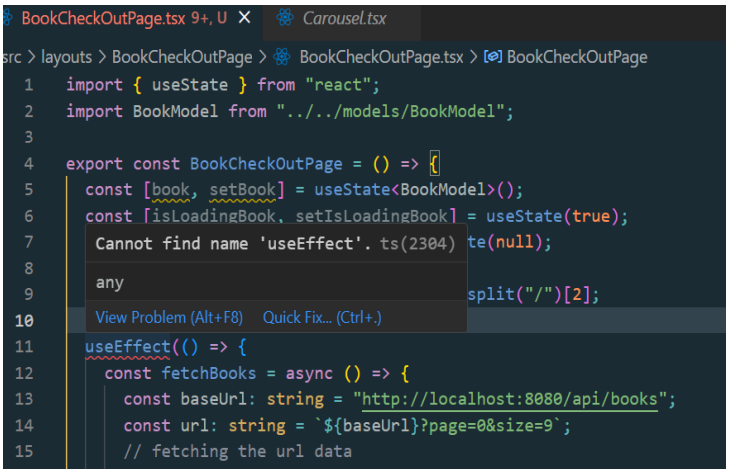
Create the useEffect method. Remember the useEffect method we created in the Carousel component? We can copy that part of code. The only difference is, here we are dealing with one single book not a book array.



Back to BookCheckoutPage.tsx, paste the code we copied and put it just under the constant code part.



Begin to fix the red errors. Put the cursor on the red tilde, we will get the error description and a quick fix button.



refactor the constant fetchBooks to fetchBook. Because here we only deal with one single book. Right click the mouse, choose “rename symbol”, type new name ,it will change all the fetchBooks in the file.

fix url code. Use `` instead of “”, and add bookId in the end, delete the url line.

```
useEffect(() => {
  const fetchBook = async () => {
    const baseUrl: string = `http://localhost:8080/api/books/${bookId}`;
    const url: string = `${baseUrl}?page=0&size=9`;
    // fetching the url data
    const response = await fetch(url);
    //failure scenario
    if (!response.ok) {
      throw new Error("Something went wrong!");
    }
```

In the await fetch (), pass the baseUrl we just fixed :

```
useEffect(() => {
  const fetchBook = async () => {
    const baseUrl: string = `http://localhost:8080/api/books/${bookId}`;
    // fetching the url data
    const response = await fetch(baseUrl);
    //failure scenario
    if (!response.ok) {
      throw new Error("Something went wrong!");
    }
```

delete the constant responseData which we used to get the whole books objects. Because we just use bookId to get one book object, we don’t need to get the whole _embedded.books.

```
    // transfer the url data into json;
    const responseJson = await response.json();
    // get the data which is the object of embedded books
    const responseData = responseJson._embedded.books;
    // create a book array
    const loadedBooks: BookModel[] = [];
    // iterate the object in responseData, push them into the book array,and set the loading process finished.
    for (const key in responseData) {
      loadedBooks.push({
        id: responseData[key].id,
        title: responseData[key].title,
        author: responseData[key].author,
        description: responseData[key].description,
        copies: responseData[key].copies,
        copiesAvailable: responseData[key].copiesAvailable,
        category: responseData[key].category,
        img: responseData[key].img,
      });
```

For the same reason above, we need to change the constant “loadedBooks” (array) into “loadedBook” (single object) . set all the book’s properties into the constant . and delete the iteration code in the books array(responseData). Because here we just get one book object by its id. fix the “setBooks” error. Change the plural form books into Singular form, since here we only get one book object.

```
    // create a book array
    const loadedBook: BookModel = {
      id: responseJson.id,
      title: responseJson.title,
      author: responseJson.author,
      description: responseJson.description,
      copies: responseJson.copies,
      copiesAvailable: responseJson.copiesAvailable,
      category: responseJson.category,
      img: responseJson.img,
    };

    setBook(loadedBook);
    setIsLoading(false);
  };
  fetchBook().catch((error: any) => {
    setIsLoading(false);
    setHttpError(error.message);
  });
```

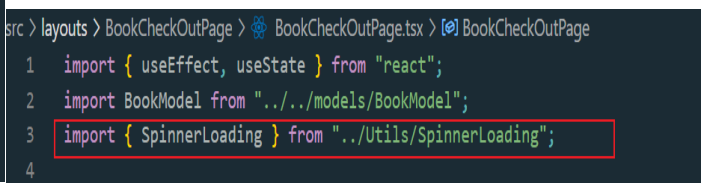
Go to Carousel.tsx file, copy the spinner Loading and http error part into our BookCheckoutPage.tsx.

```
52
53    if (isLoading) {      You, 2 days ago • Consume
54      return <SpinnerLoading />;
55    }
56
57    if (httpError) {
58      return (
59        <div className="container m-5">
60          <p>{httpError}</p>
61        </div>
62      );
63    }
```

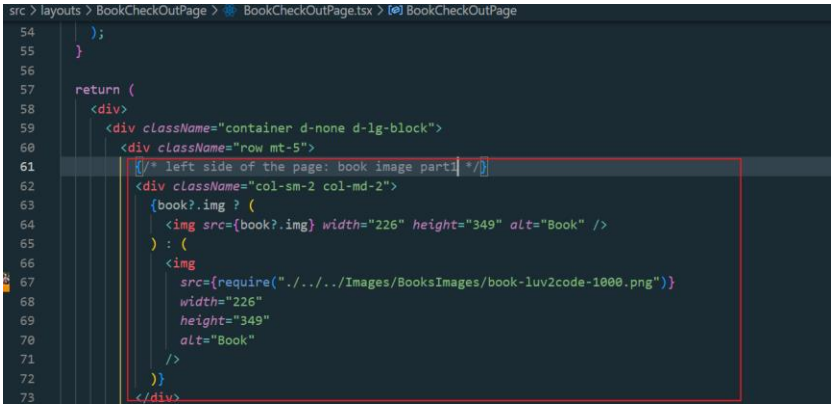
Just paste it after the fetchBook() catch error part.

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

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

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

  return (
    <div>
      <h3>Hi World!</h3>
    </div>
  );
};
```

fix the errors in the spinner Loading and http error part which we just pasted. Now, we finished all the useEffect part.



Go to return part, we first create an div to display our book image on the left half of the page.



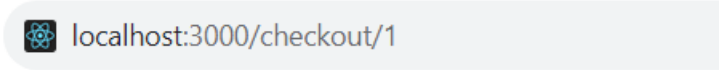
then we create another div to display our book info on the right half of the page.

```
74          {/* right side of the page: book info part */}
75          <div className="col-4 col-md-4 container">
76            <div className="ml-2">
77              <h2>{book?.title}</h2>
78              <h5 className="text-primary">{book?.author}</h5>
79              <p className="lead">{book?.description}</p>
80            </div>
81          </div>
82        </div>
83        <hr />
```

create book image and book info part for a mobile version page. Just change the wrapper div's className and copy the inside code from the web page version.

```
85        {/* mobile version */}
86        {/* image part */}
87        <div className="container d-lg-none mt-5">
88          <div className="d-flex justify-content-center align-items-center">
89            {book?.img ? (
90              <img src={book?.img} width="226" height="349" alt="Book" />
91            ) : (
92              <img
93                src={require("./../../Images/BooksImages/book-luv2code-1000.png")}
94                width="226"
95                height="349"
96                alt="Book"
97              />
98            )}
99          </div>
100         {/* book info part */}
101         <div className="mt-4">
102           <div className="ml-2">
103             <h2>{book?.title}</h2>
104             <h5 className="text-primary">{book?.author}</h5>
105             <p className="lead">{book?.description}</p>
106           </div>
107         </div>
108         <hr />
109       </div>
110     </div>
```

Let's run the application.Input the url: http://localhost:3000/checkout/1. We can see the page.

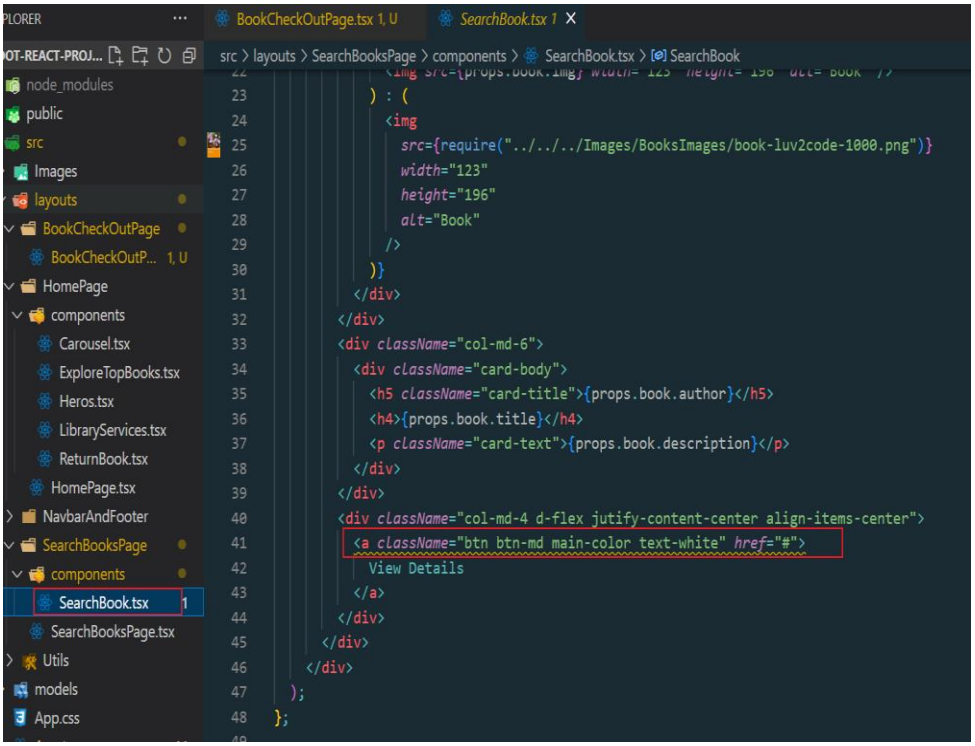


Mobile version:

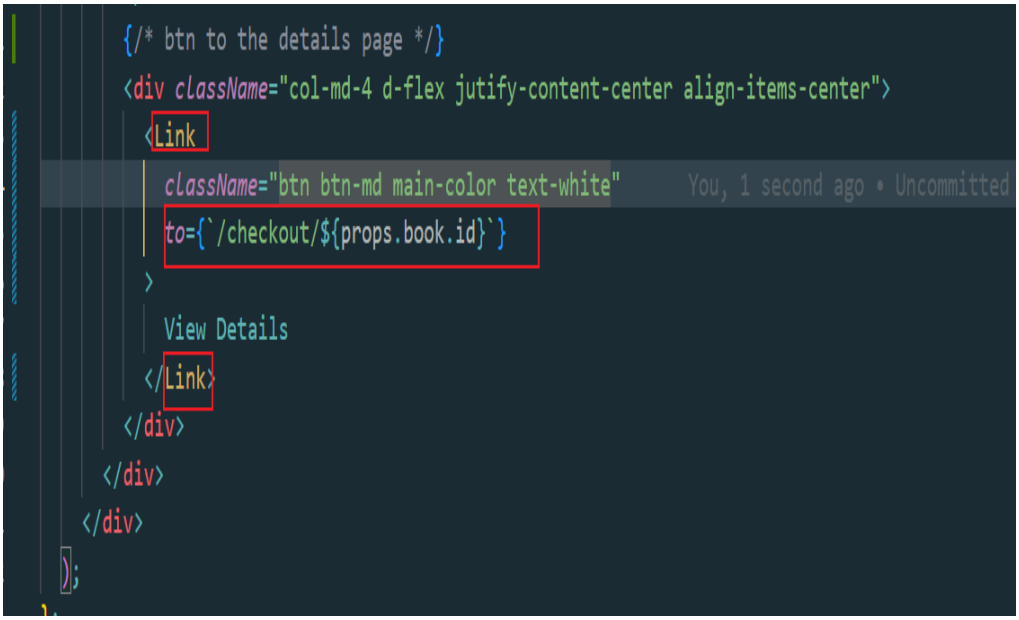
PART B. BOOK CHECKOUT LINK

We will fix all the links to our detailed book page(Checkout page).

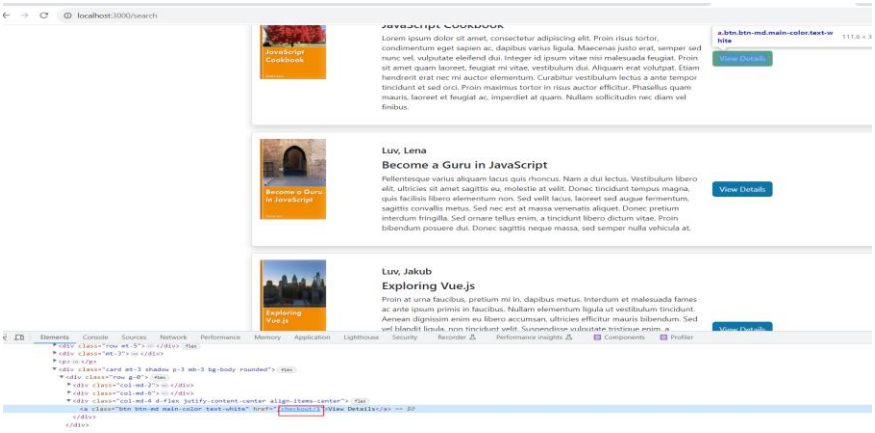
Step 1, Go to “SearchBook.tsx”, scroll down to the bottom, find the “view details” link.



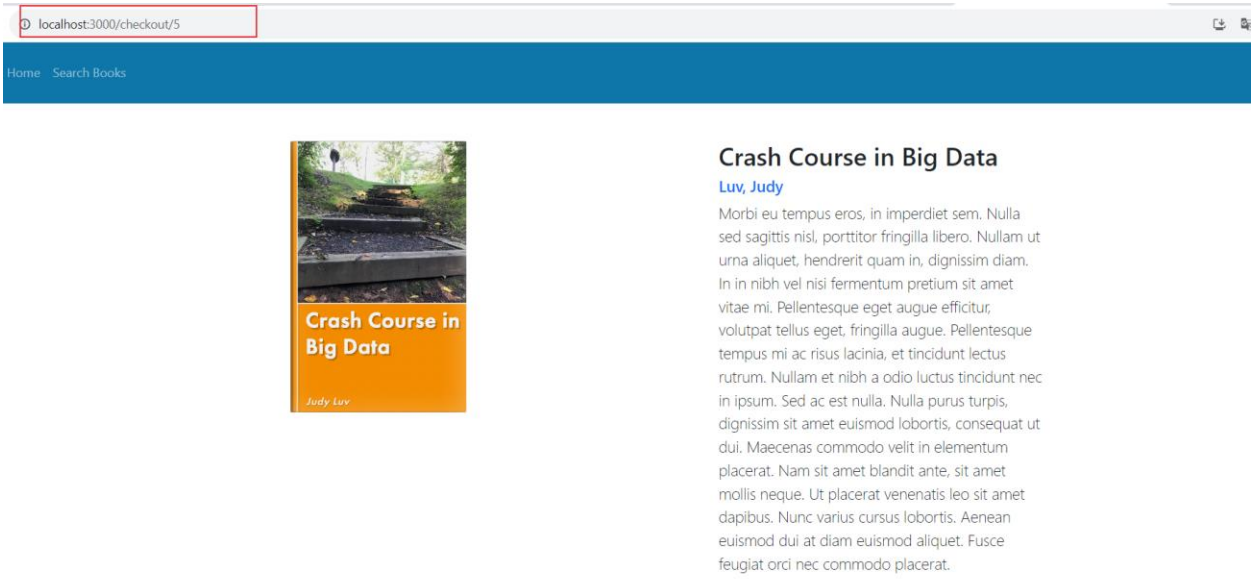
Change “a” to “Link” , “href” to “to”, vaule of href “#” to the value of {`/checkout/${props.book.id}`}



Now, run the application, open our search page. For the first book, check its “view details” button in the console. You will find that its href equals to “/checkout/1”. It means now the app gets its Id.

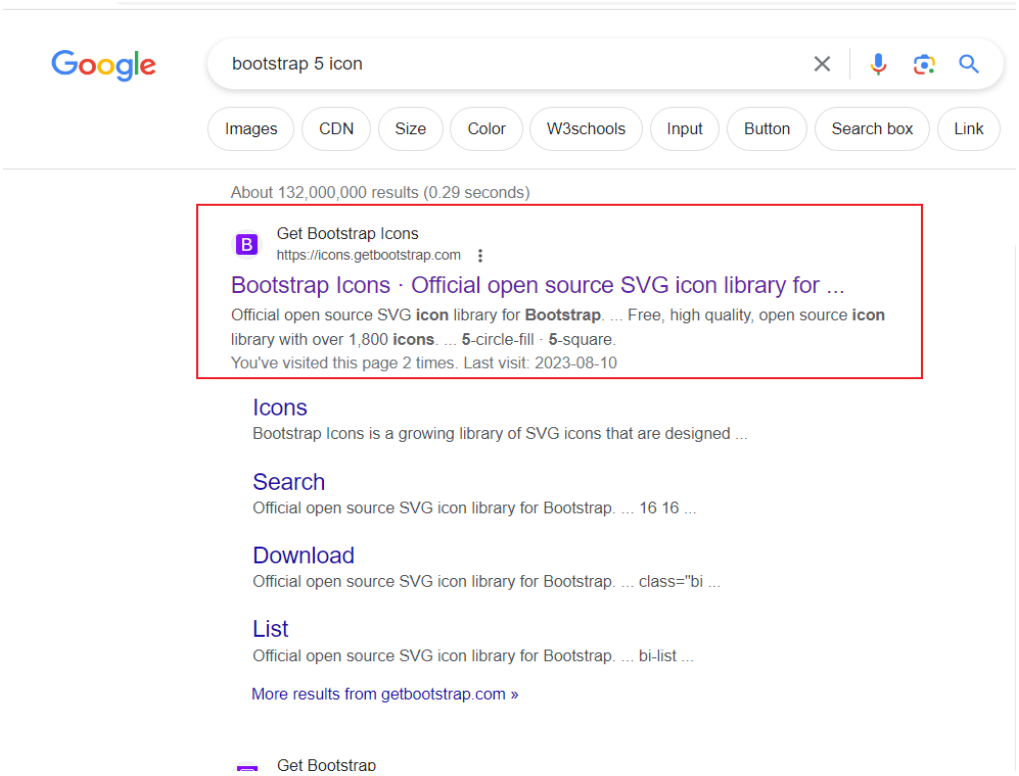


If we click on the “View Details” button, it redirects to value of the href we have just checked, shows the following page,

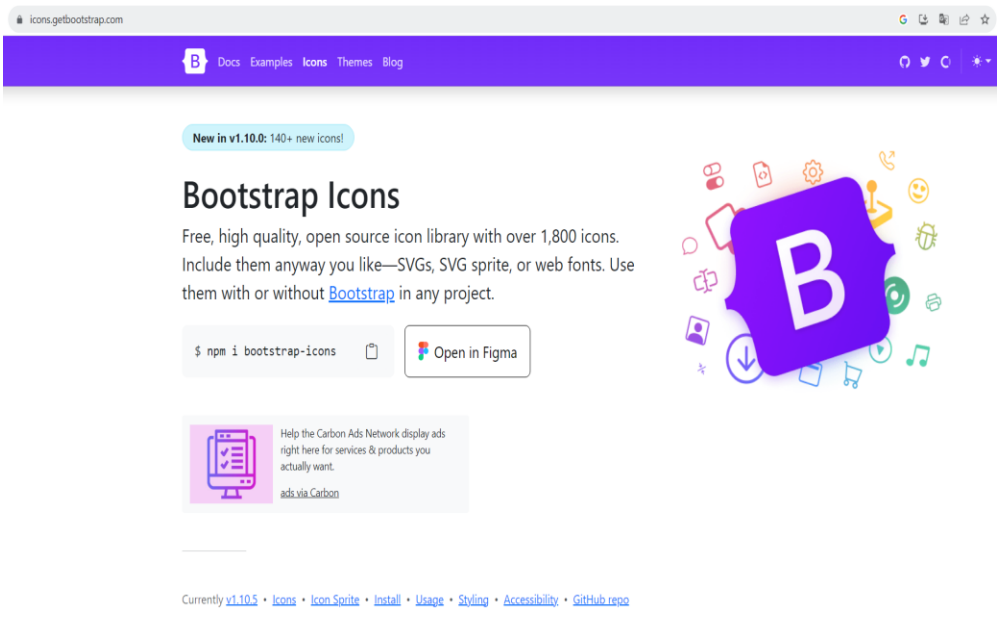


PART C. STAR REVIEW COMPONENT

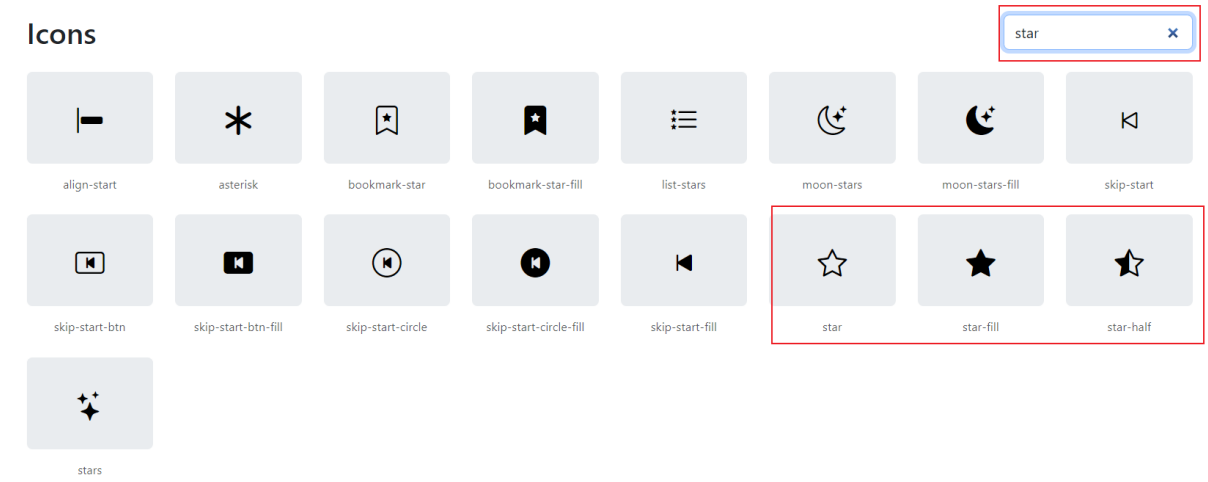
Google “bootstrap 5 icon.” Now, we will use bootstrap icon to display our star reviews. Since we use bootstrap version 5, google bootstrap 5 icon.



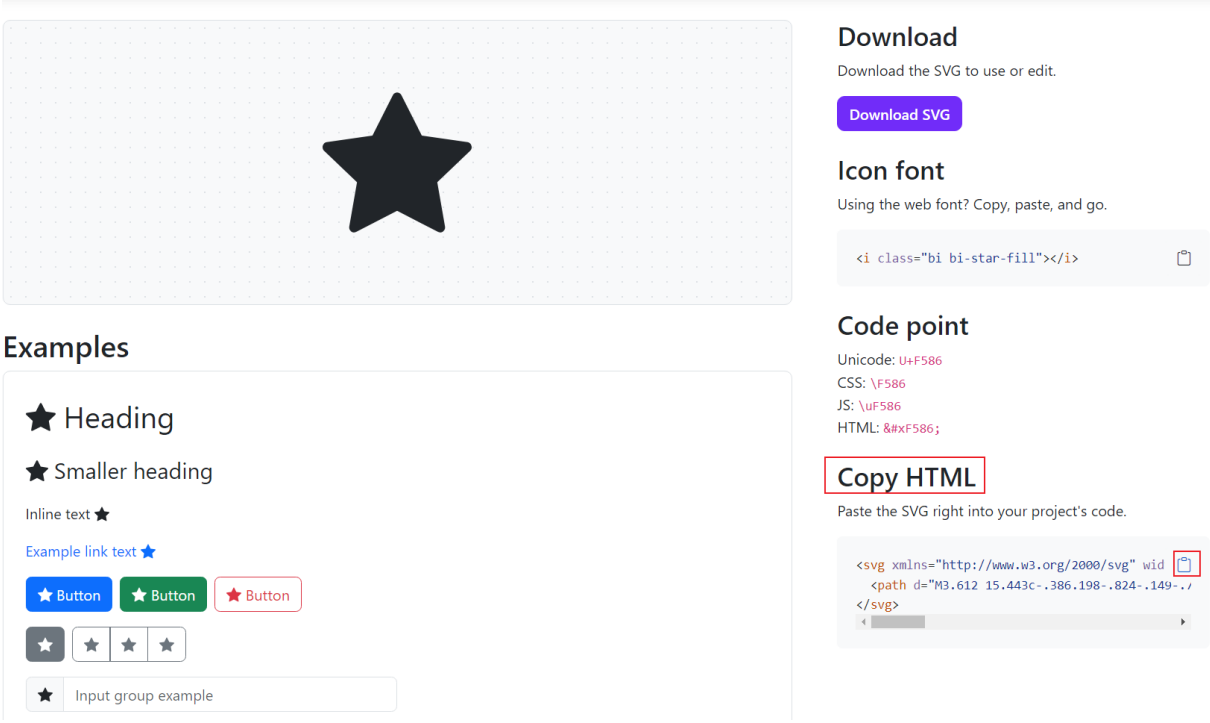
click the link : http://icons.getbootstrap.com



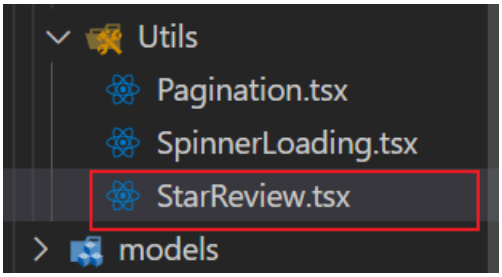
In the search box, input “star”. It will pop out many options. In our App, we use empty star, star-filled and half star.



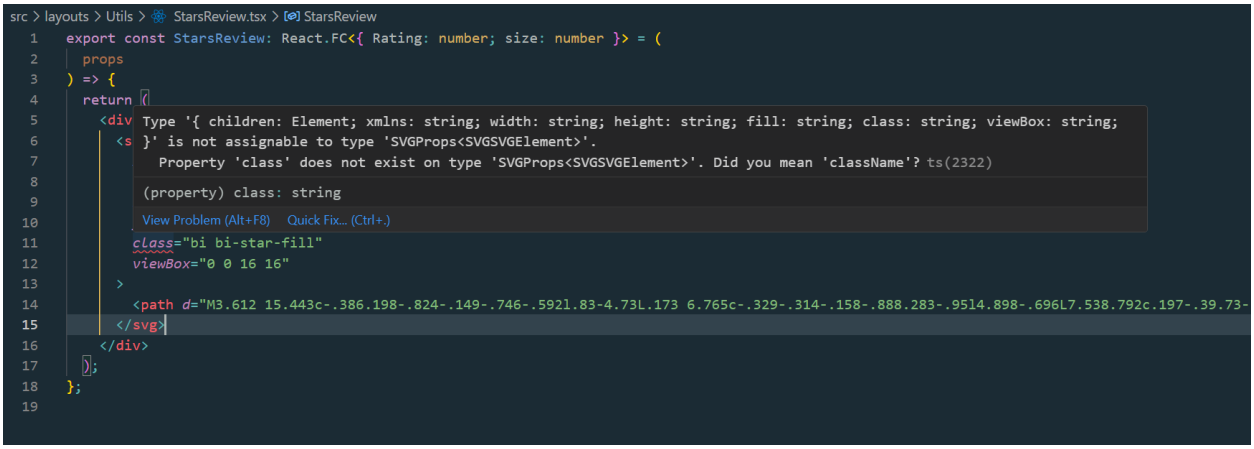
Click on the star-filled go to the main page of “star-fill”. Scroll down the page, click on the copy sign in the “Copy HTML” part.



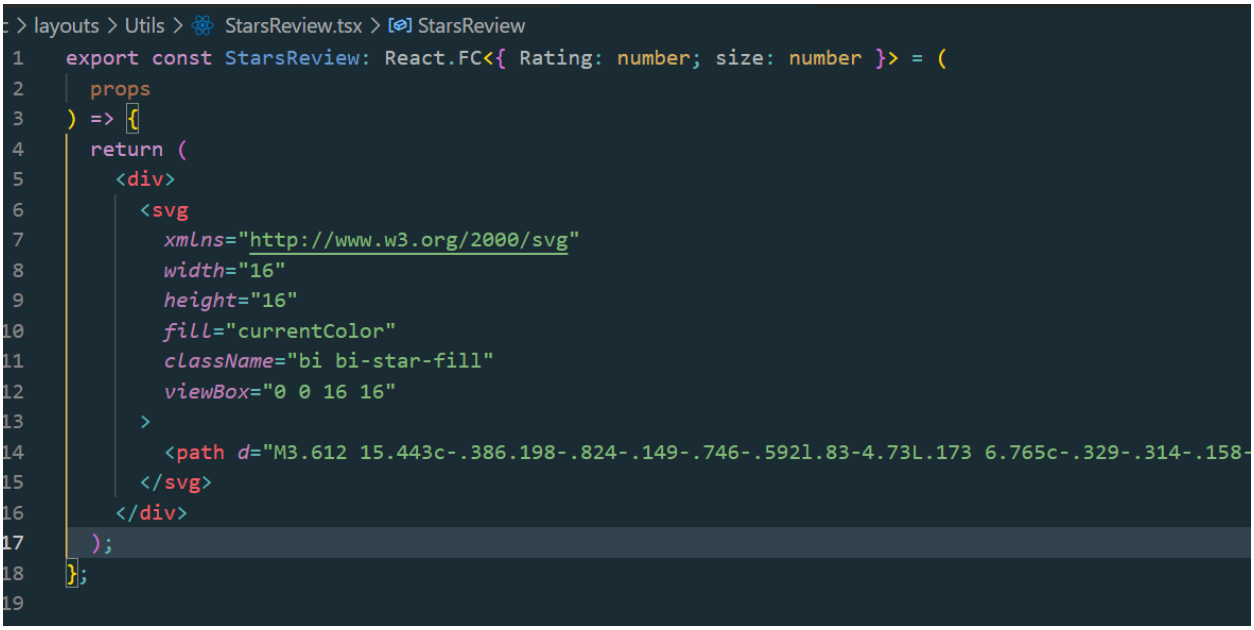
Back to our VS Code. Create a new file under folder “Utils”, name it “StarsReview.tsx”.



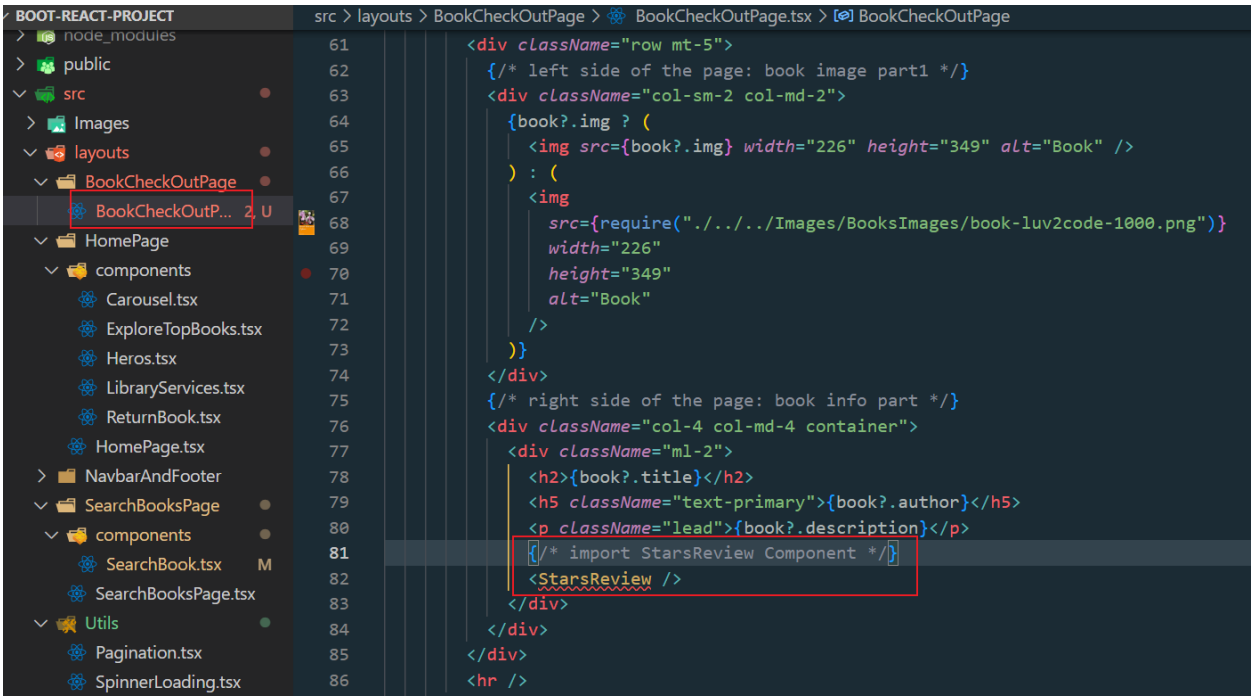
In the StarsReview.tsx, Write the code, paste the code we copied from the bootstrap icon website in the parentheses after “return”. We will get red error. Put the cursor on the red tilde, we can see the error description.



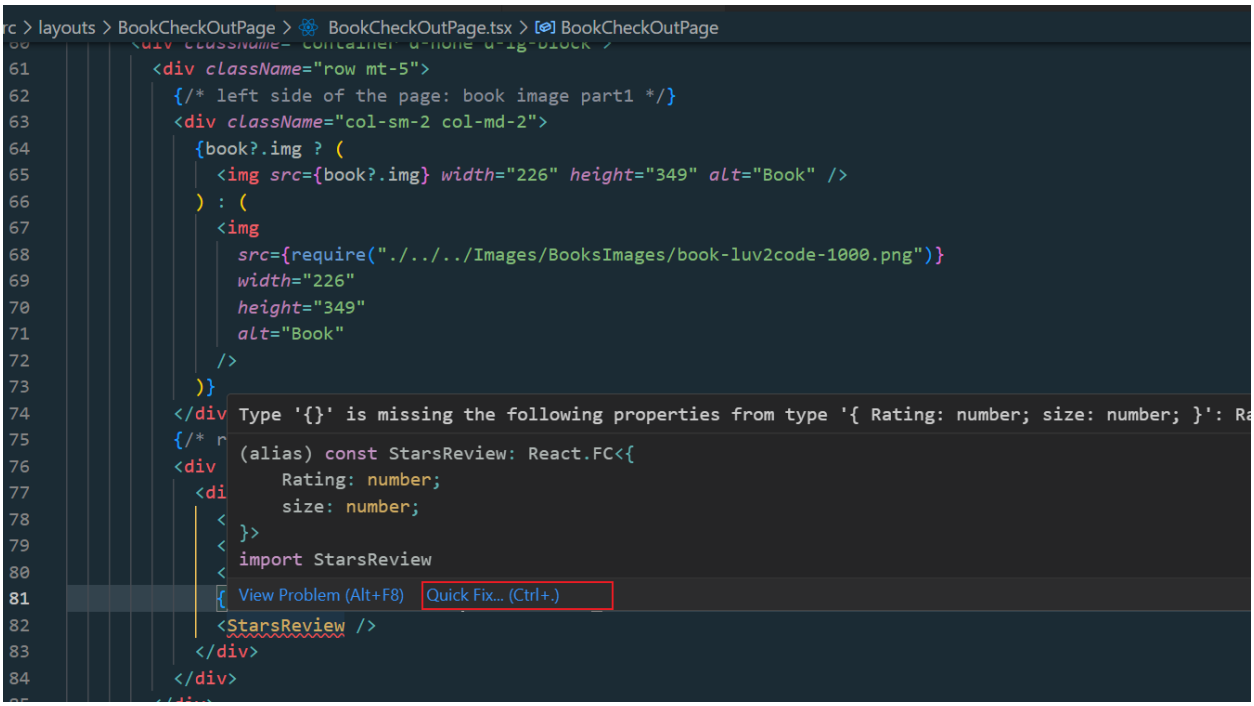
Fix the issue by click on the “Quick Fix”.



Go to BookCheckoutPage.tsx, import the StarsReview component we created just after the book description part. Fix the issue by import StartsReview Component.



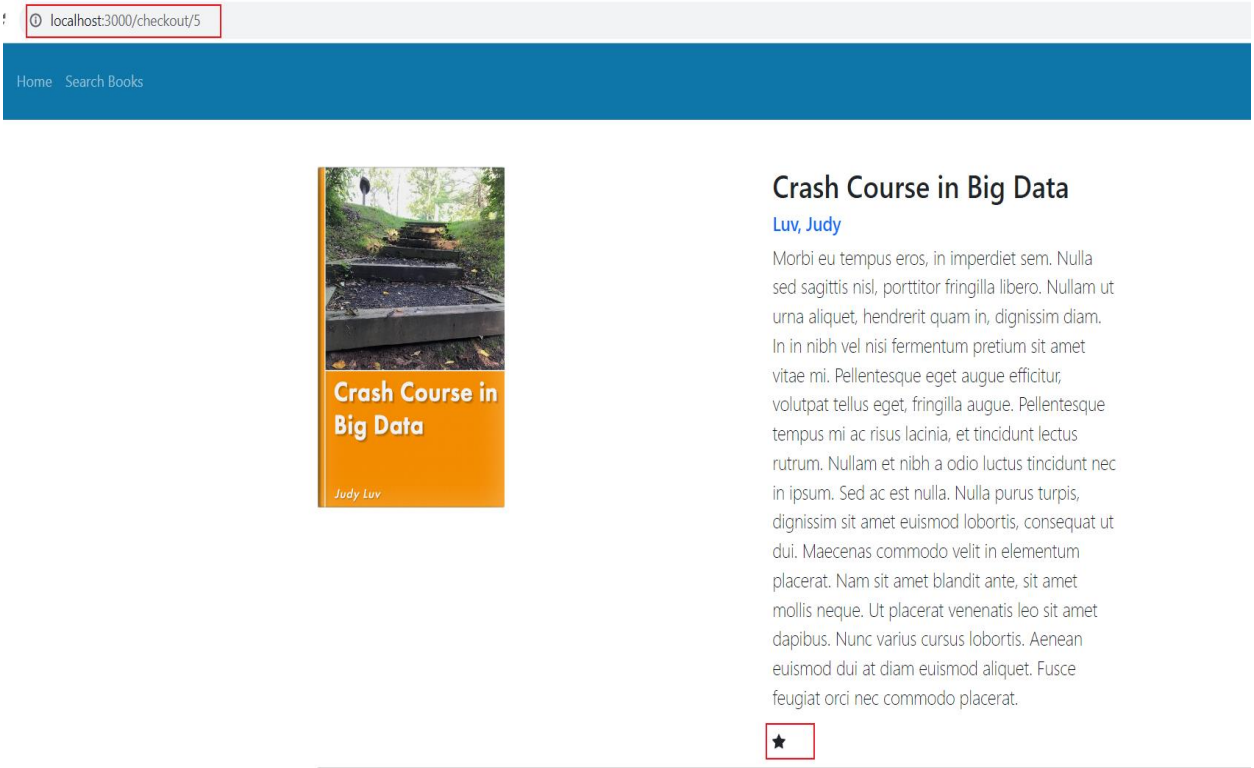
Fix the error. Our StarsReview component show an error. Right click the mouse, get the error description. We can see it's because our component is missing properties it's supposed to have. Click on the "Quick Fix" or add the rating and size properties directly.



We try to add “rating={4}, size ={32}”

```
75          {/* right side of the page: book info part */}
76          <div className="col-4 col-md-4 container">
77            <div className="ml-2">
78              <h2>{book?.title}</h2>
79              <h5 className="text-primary">{book?.author}</h5>
80              <p className="lead">{book?.description}</p>
81              {/* import StarsReview Component */}
82              <StarsReview rating={4} size={32} />
83            </div>
84          </div>
```

Run the application, go to the browser, input http://localhost:3%20000/checkout/5 Go to the book whose id is 5. We get one full star review.

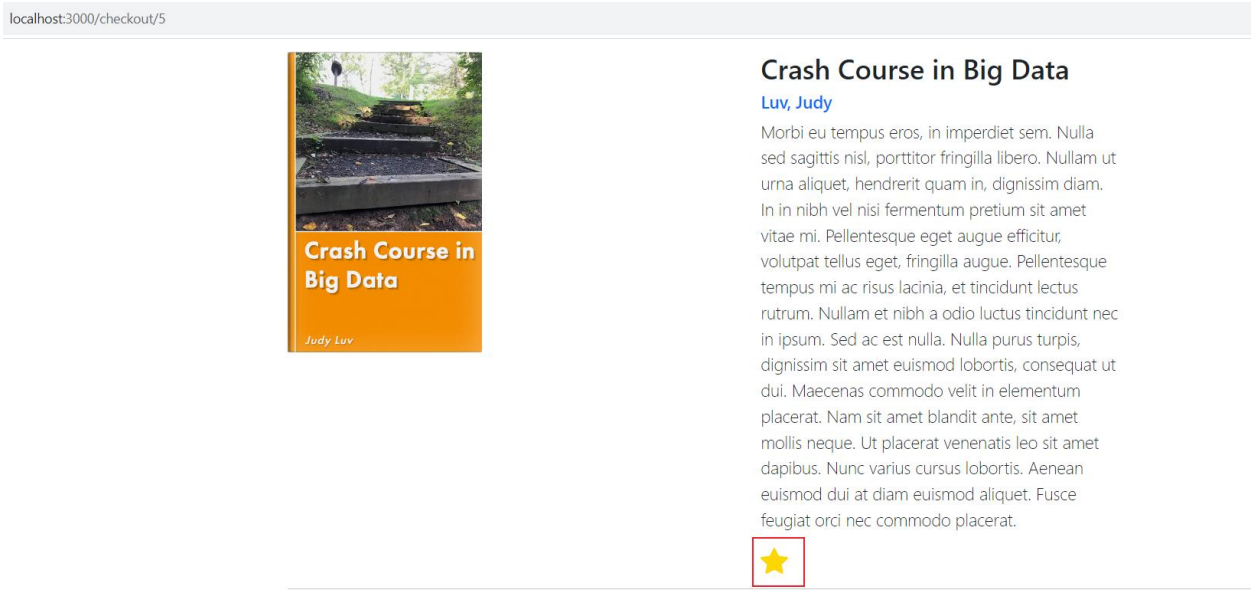


Go to StarsReview.tsx, change the value of width and height in the “svg” tag. And after className part, add style = {{color:“gold”}}. We get a one-star review under the book description. But there are still two problems to solve. One is parameter we passed does not work. We want rating ={4} and size={ 32} and it does not work. The other is we want to have a gold star in our App. Here we solve the size and the color problem. Values are changed as below:

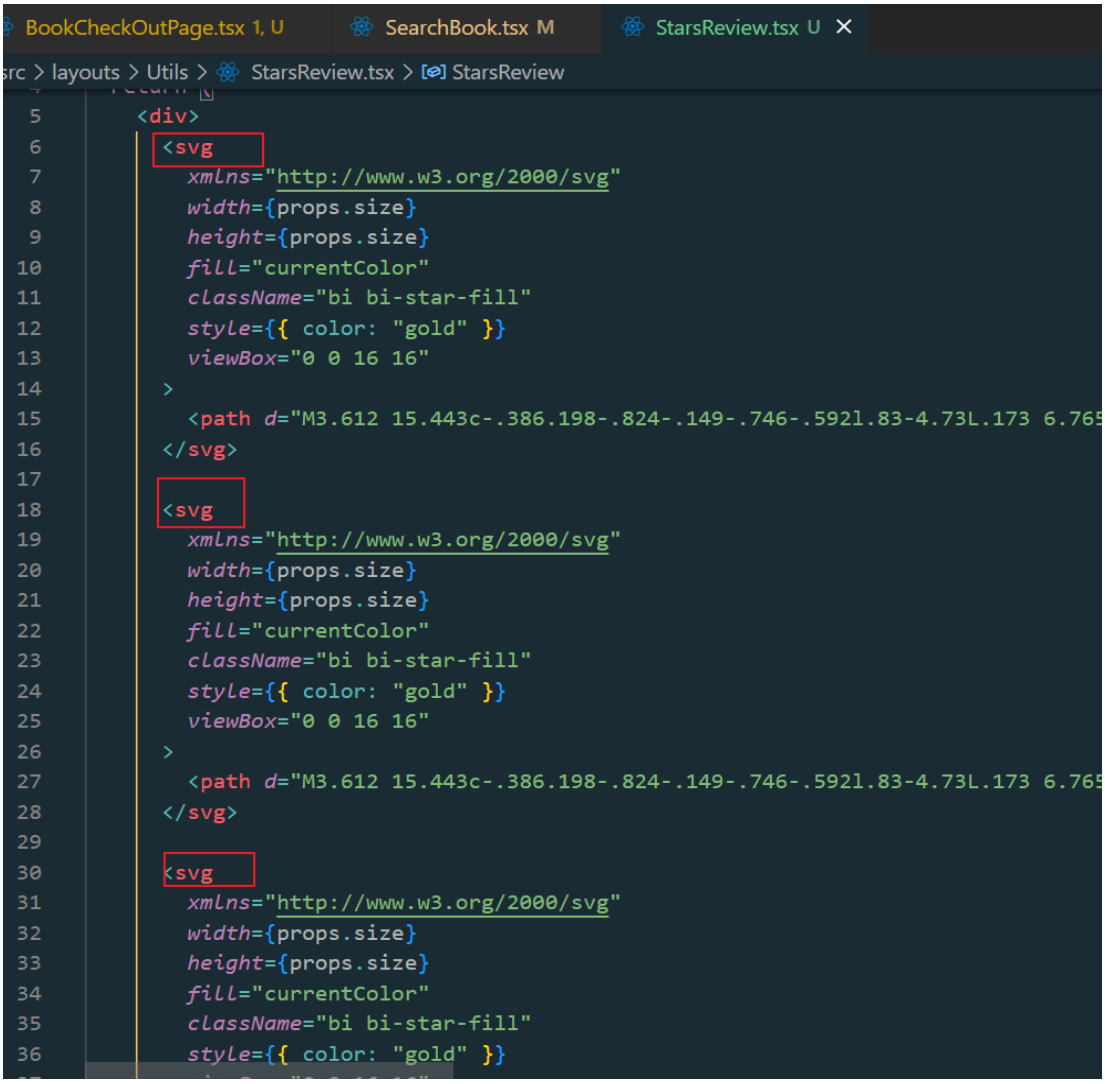
src > layouts > Utils > StarsReview.tsx > StarsReview

```tsx
export const StarsReview: React.FC<{ rating: number; size: number }> = (
  props
) => {
  return (
    <div>
      <svg
        xmlns="http://www.w3.org/2000/svg"
        width={props.size}
        height={props.size}
        fill="currentColor"
        className="bi bi-star-fill"
        style={{ color: "gold" }}
        viewBox="0 0 16 16"
      >
        <path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.
      </svg>
    </div>
  );
};

```

Let’s go back to our browser. Now we have a golden star review.



Now, if we copy “star fill” svg tag three times ,



Then copy “half star ” svg tag one time,

Then copy “star ” svg tag one time,

```
<svg
  xmlns="http://www.w3.org/2000/svg"
  width={props.size}
  height={props.size}
  fill="currentColor"
  className="bi bi-star"
  style={{ color: "gold" }}
  viewBox="0 0 16 16"
>
  <path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.8
</svg>
```

Refresh the page http://localhost:3 000/checkout/2, We will see a 3.5 golden star review, which equal to the value of each SVG tag’s occurrences.

Now, we create an array for each kind of star, and control the number of each kind of star by the length of the array. Let the length of the star fill be 3. Copy the original svg tag into our array function.

```
return (
  <div>
    {Array.from({ length: 3 }, (_, i) => (
      <svg
        key={i}
        xmlns="http://www.w3.org/2000/svg"
        width={props.size}
        height={props.size}
        fill="currentColor"
        className="bi bi-star-fill"
        style={{ color: "gold" }}
        viewBox="0 0 16 16"
      >
        <path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-
      </svg>
    ))}
```

The same process for the half star and empty star, make the length of these two arrays be 1. So now, we should also have a 3.5 stars review.

```
22      {Array.from({ length: 1 }, (_, i) => (
23        // half star
24        <svg
25          xmlns="http://www.w3.org/2000/svg"
26          width={props.size}
27          height={props.size}
28          fill="currentColor"
29          className="bi bi-star-half"
30          style={{ color: "gold" }}
31          viewBox="0 0 16 16"
32        >
33          <path d="M5.354 5.119 7.538.792A.516.516 0 0 1 8 .5c.183 0 .366.097.465.29212.184 4.327 4.898.6
34        </svg>
35      ))}
36
37      {Array.from({ length: 1 }, (_, i) => (
38        // star
39        <svg
40          xmlns="http://www.w3.org/2000/svg"
41          width={props.size}
42          height={props.size}
43          fill="currentColor"
44          className="bi bi-star"
45          style={{ color: "gold" }}
46          viewBox="0 0 16 16"
47        >
48          <path d="M2.866 14.85c-.078.444.36.791.746.59314.39-2.256 4.389 2.256c.386.198.824-.149.746-.59
49        </svg>
50      ))}
51    </div>
52  );
```

Open our browser, we still see a 3.5 golden star review, nothing changes.

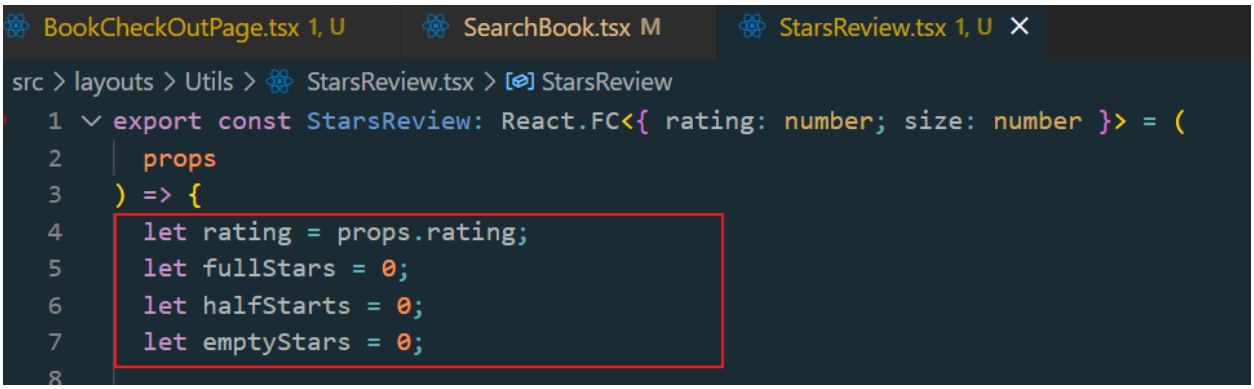
Crash Course in Big Data

Luv, Judy

Morbi eu tempus eros, in imperdiet sem. Nulla sed sagittis nisl, porttitor fringilla libero. Nullam ut urna aliquet, hendrerit quam in, dignissim diam. In in nibh vel nisi fermentum pretium sit amet vitae mi. Pellentesque eget augue efficitur, volutpat tellus eget, fringilla augue. Pellentesque tempus mi ac risus lacinia, et tincidunt lectus rutrum. Nullam et nibh a odio luctus tincidunt nec in ipsum. Sed ac est nulla. Nulla purus turpis, dignissim sit amet euismod lobortis, consequat ut dui. Maecenas commodo velit in elementum placerat. Nam sit amet blandit ante, sit amet mollis neque. Ut placerat venenatis leo sit amet dapibus. Nunc varius cursus lobortis. Aenean euismod dui at diam euismod aliquet. Fusce feugiat orci nec commodo placerat.

Now, our stars reviews has nothing related with our rating parameter. We will implement the code to make our star reviews change dynamically with the rating value.

In the Utils folder, StarsReviews.tsx, we initialize four parameter instances: rating, fullStars, halfStars, emptyStars.



Implement the logic relationship between rating and number of each kind of star. Calculate how many stars we have for each kind of star.

```
if (rating !== undefined && rating > 0 && rating <= 5) {
  for (let i = 0; i <= 4; i++) {
    if (rating - 1 >= 0) {
      // calculate how many full stars we can have
      fullStars = fullStars + 1;
      rating = rating - 1;
    } else if (rating === 0.5) {
      // calculate how many half stars we can have
      halfStarts = halfStarts + 1;
      rating = rating - 0.5;
    } else if (rating === 0) {
      // calculate how many empty stars we can have
      emptyStars = emptyStars + 1;
    } else {
      break;
    }
  }
} else {
  emptyStars = 5;
}
```

change the static value of array length of each kind of star to a dynamical value

```
{Array.from({ length: fullStars }, (_, i) => (
  // star fill
  <svg
    key={i}
    xmlns="http://www.w3.org/2000/svg"
    width={props.size}
    height={props.size}
    fill="currentColor"
    className="bi bi-star-fill"
    style={{ color: "gold" }}
    viewBox="0 0 16 16"
  >
    <path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.7
  </svg>
))}

{Array.from({ length: halfStarts }, (_, i) => (
  // half star
  <svg
    key={i}
    xmlns="http://www.w3.org/2000/svg"
    width={props.size}
    height={props.size}
    fill="currentColor"
    className="bi bi-star-half"
    style={{ color: "gold" }}
    viewBox="0 0 16 16"
  >
    <path d="M5.354 5.119 7.538.792A.516.516 0 0 1 8 .5c.183 0 .366.097.46
  </svg>
))}

{Array.from({ length: emptyStars }, (_, i) => (
  // star
```

Let's check if our code works. Go back to BookCheckoutPage, in the StarsReviews component, change the value of the rating to 2.5.

src > layouts > BookCheckOutPage > BookCheckOutPage.tsx > BookCheckOutPage

```
<div className="row mt-5">
  {/* left side of the page: book image part1 */}
  <div className="col-sm-2 col-md-2">
    {book?.img ? (
      <img src={book?.img} width="226" height="349" alt="Book" />
    ) : (
      <img
        src={require("./../../Images/BooksImages/book-luv2code-1000.png")}
        width="226"
        height="349"
        alt="Book"
      />
    )}
  </div>
  {/* right side of the page: book info part */}
  <div className="col-4 col-md-4 container">
    <div className="ml-2">
      <h2>{book?.title}</h2>
      <h5 className="text-primary">{book?.author}</h5>
      <p className="lead">{book?.description}</p>
      {/* import StarsReview Component */}
      <StarsReview rating={2.5} size={32} />
    </div>
  </div>
</div>
<hr />
```

Run the application, we get a 1.5 golden star here. Which means our code works perfectly.

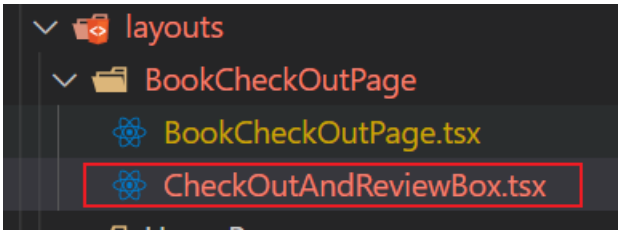
import our StarsReview component in our mobile version code

```
{/* mobile version */}
{/* image part */}
<div className="container d-lg-none mt-5">
  <div className="d-flex justify-content-center align-items-center">
    {book?.img ? (
      <img src={book?.img} width="226" height="349" alt="Book" />
    ) : (
      <img
        src={require("./../../Images/BooksImages/book-luv2code-1000.png")}
        width="226"
        height="349"
        alt="Book"
      />
    )}
  </div>
  {/* book info part */}
  <div className="mt-4">
    <div className="ml-2">
      <h2>{book?.title}</h2>
      <h5 className="text-primary">{book?.author}</h5>
      <p className="lead">{book?.description}</p>
      {/* import StarsReview Component */}
      <StarsReview rating={2.5} size={32} />
    </div>
```

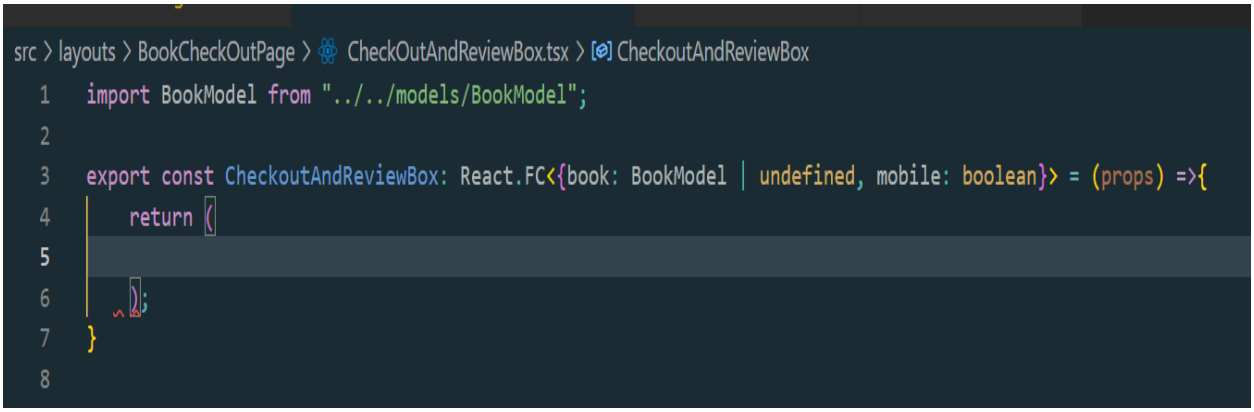
Go to our mobile version application, Under the description of the book, there is also a 2.5 golden star review.

PART D. CHECKOUT COMPONENT

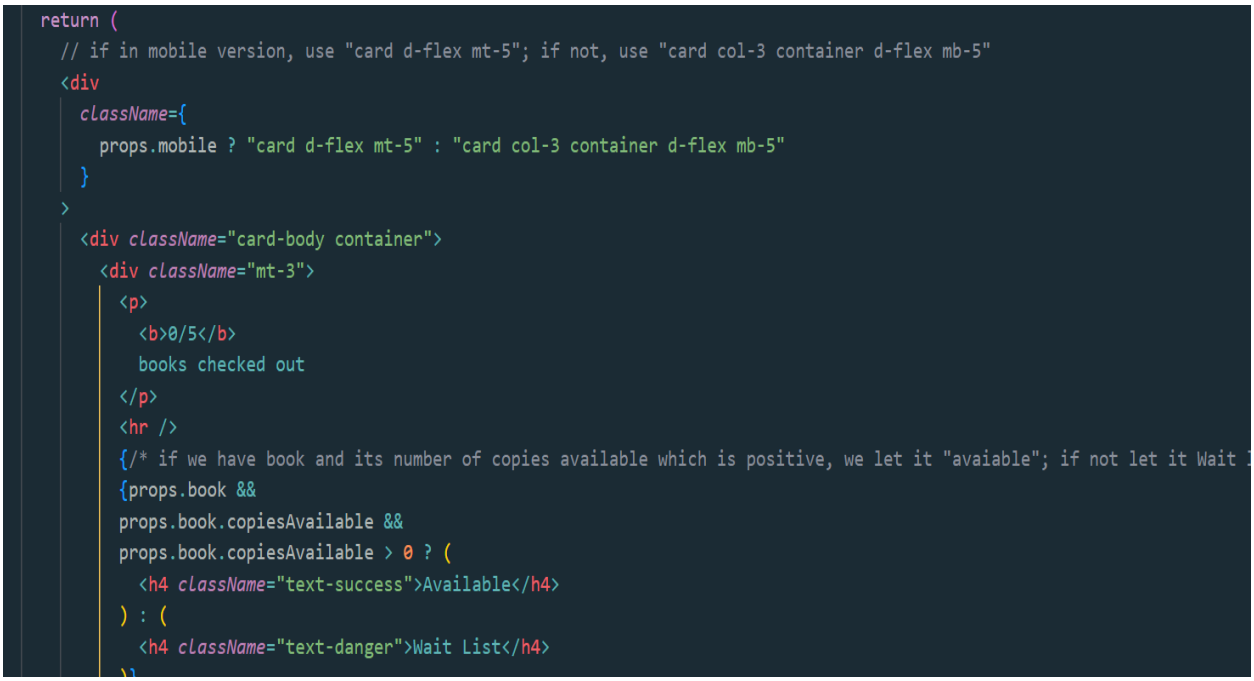
Go to folder BookCheckoutPage, right click and choose “New file” to create a new file.name it “CheckoutAndReviewBox.tsx”.



as before, we create export const structure. Pass the “book” which is BookModel object as props.



In “return” statement, create the layout for the page. As it’s a responsive page, we add another props “mobile” in the constant CheckoutAnsReviewBox.



In “return” statement, then create “copies” and “available” part.

```
{/* text for copies and copies available */}
<div className="row">
  <p className="col-6 lead">
    <b>{props.book?.copies}</b>
    copies
  </p>
  <p className="col-6 lead">
    <b>{props.book?.copiesAvailable}</b>
    available
  </p>
</div>
</div>
```

In “return” statement, create “sign in ” part.

```
{/* sign in button */}
<Link to="/#" className="btn btn-success btn-lg">
  Sign In
</Link>
<hr />
<p className="mt-3">
  This number can change until placing order has been complete.
</p>
<p>Sign in to be able to leave a review.</p>
</div>
</div>
```

Go to BookCheckoutPage.tsx, import “CheckoutAndReviewBox” component in the part of code for the desktop version, and set mobile={false} in its properties.

```
c > layouts > BookCheckOutPage > BookCheckOutPage.tsx > BookCheckOutPage
57    }
58
59    return (
60      <div>
61        <div className="container d-none d-lg-block">
62          <div className="row mt-5">
63            {/* left side of the page: book image part1 */}
64            <div className="col-sm-2 col-md-2">
65              {book?.img ? (
66                <img src={book?.img} width="226" height="349" alt="Book" />
67              ) : (
68                <img
69                  src={require("./../../Images/BooksImages/book-luv2code-1000.png")}
70                  width="226"
71                  height="349"
72                  alt="Book"
73                />
74              )}
75            </div>
76            {/* right side of the page: book info part */}
77            <div className="col-4 col-md-4 container">
78              <div className="ml-2">
79                <h2>{book?.title}</h2>
80                <h5 className="text-primary">{book?.author}</h5>
81                <p className="lead">{book?.description}</p>
82                {/* import StarsReview Component */}
83                <StarsReview rating={2.5} size={32} />
84              </div>
85            </div>
86            <CheckoutAndReviewBox book={book} mobile={false} />
87          </div>
```

Scroll down the page , Go to the part of code for the mobile, import “CheckoutAndReviewBox” component, and set mobile={true} in its properties.

```
105          {/* book info part */}
106          <div className="mt-4">
107            <div className="ml-2">
108              <h2>{book?.title}</h2>
109              <h5 className="text-primary">{book?.author}</h5>
110              <p className="lead">{book?.description}</p>
111              {/* import StarsReview Component */}
112              <StarsReview rating={2.5} size={32} />
113            </div>
114          </div>
115          <CheckoutAndReviewBox book={book} mobile={true} />
116          <hr />
117        </div>
118      </div>
119    );
120  };
```

Go to http://localhost:3000/checkout/2, we can see the golden star review and checkout and review box.

Mobile Version. We can see the checkout and review box still work.