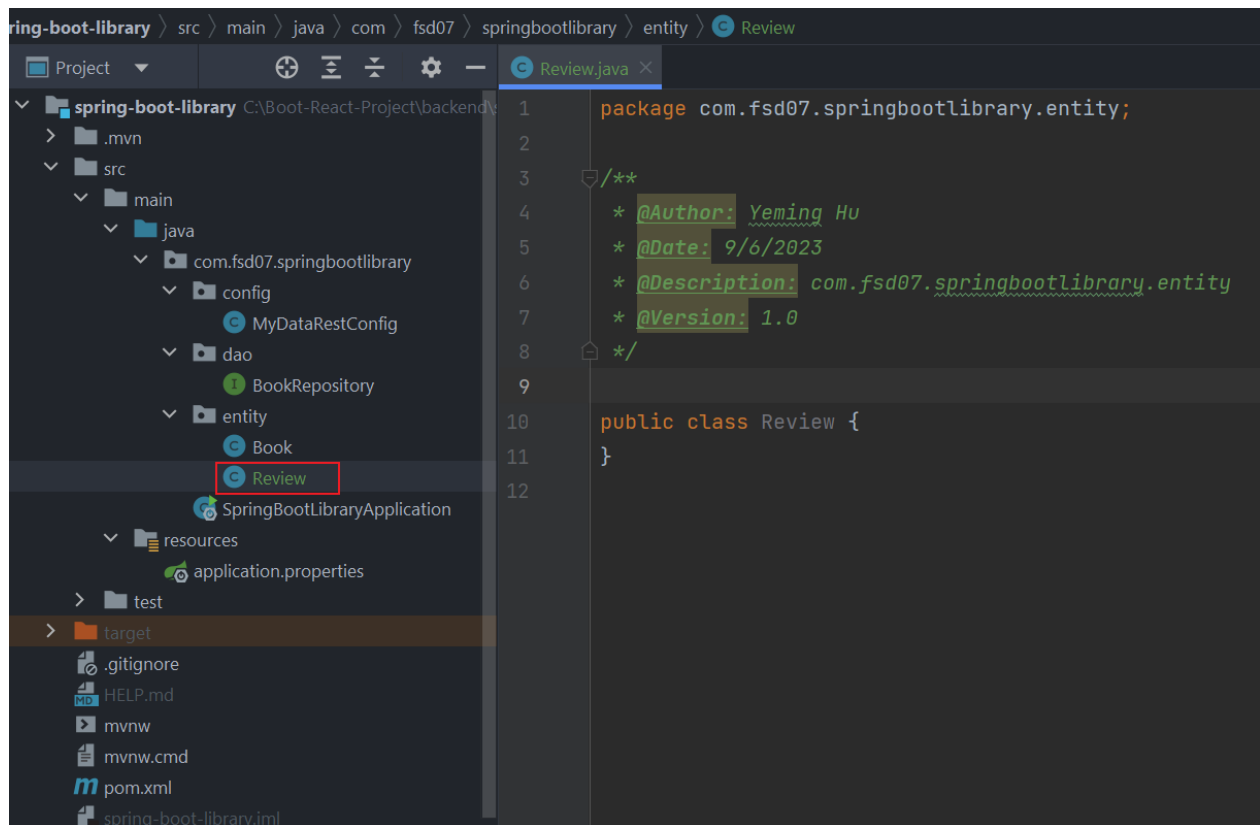


# S07 REVIEWS ON BOOK CHECKOUT PAGE

## 1. SPRINGBOOT-REVIEW ENTITY

Go to entity package, create a new class:Review.



Add @Entity, @Table and @Data annotations.

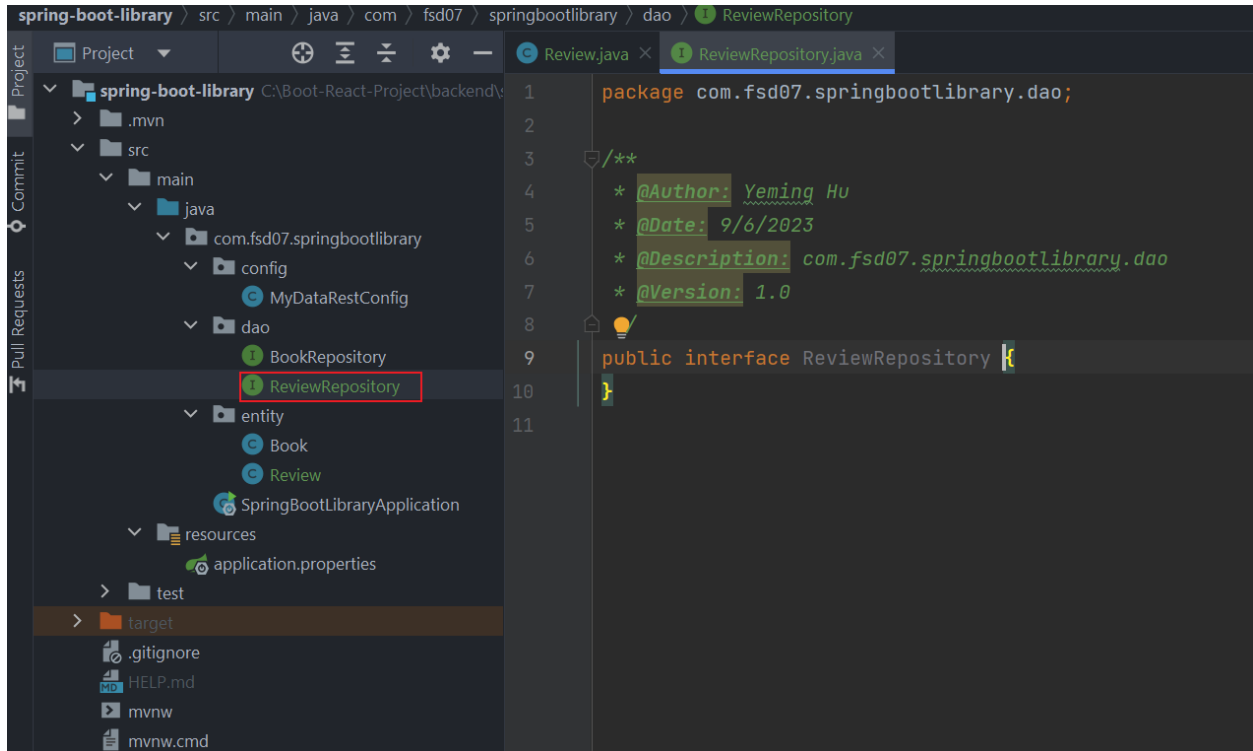
```
Review.java
1 import javax.persistence.*;
2
3
4
5 import javax.persistence.Entity;
6 import javax.persistence.Table;
7
8 /**
9  * @Author: Yeming Hu
10 * @Date: 9/6/2023
11 * @Description: com.fsd07.springbootlibra
12 * @Version: 1.0
13 */
14 @Entity
15 @Table(name = "review")
16 @Data
17 public class Review {
```

Add properties which are mapping with the columns of review table in our database.

```
15 @Entity
16 @Table(name = "review")
17 @Data
18 public class Review {
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     @Column(name = "id")
22     private Long id;
23
24     @Column(name = "user_email")
25     private String userEmail;
26
27     @Column(name = "date")
28     @CreationTimestamp
29     private Date date;
30
31     @Column(name = "rating")
32     private double rating;
33
34     @Column(name = "book_id")
35     private Long bookId;
36
37     @Column(name = "review_description")
38     private String reviewDescription;
39 }
```

## 2. SPRINGBOOT-REVIEW DAO

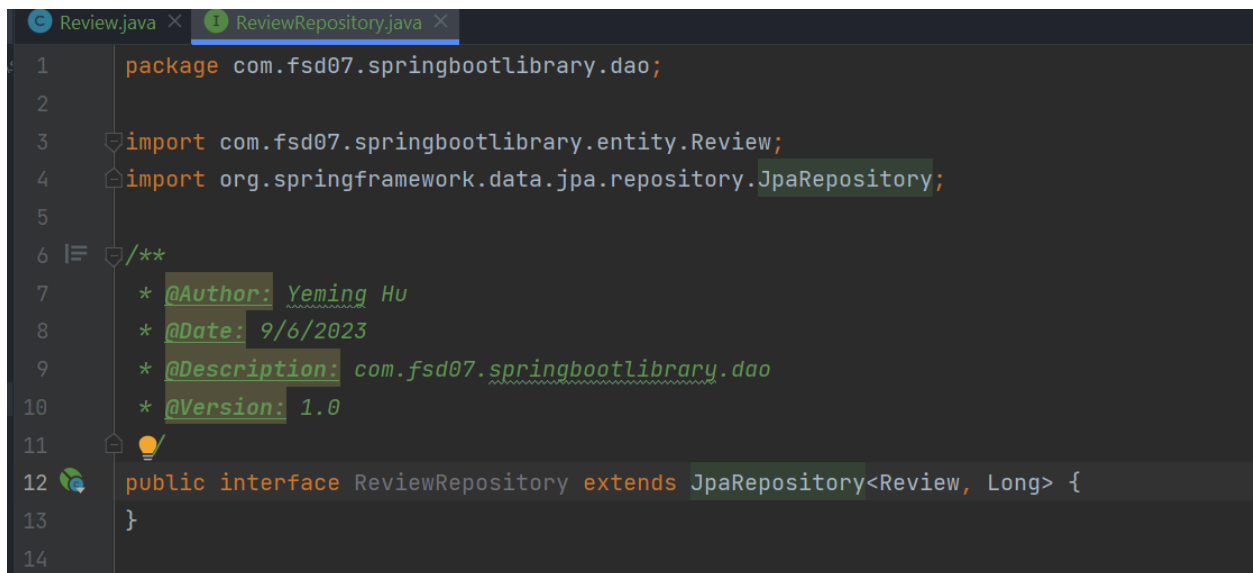
Create ReviewRepository in dao package, extends jpa repository.



The screenshot shows an IDE with a project named 'spring-boot-library'. The left sidebar displays the project structure, with the 'dao' package under 'com.fsd07.springbootlibrary' selected. The main editor shows the 'ReviewRepository.java' file. The code defines a package, a Javadoc comment with metadata, and a public interface 'ReviewRepository'.

```
1 package com.fsd07.springbootlibrary.dao;
2
3 /**
4  * @Author: Yeming Hu
5  * @Date: 9/6/2023
6  * @Description: com.fsd07.springbootlibrary.dao
7  * @Version: 1.0
8  */
9 public interface ReviewRepository {
10 }
11
```

Add “extends JpaRepository”



The screenshot shows the same IDE with the 'ReviewRepository.java' file updated. It now includes imports for 'Review' and 'JpaRepository', and the 'ReviewRepository' interface extends 'JpaRepository<Review, Long>'.

```
1 package com.fsd07.springbootlibrary.dao;
2
3 import com.fsd07.springbootlibrary.entity.Review;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 /**
7  * @Author: Yeming Hu
8  * @Date: 9/6/2023
9  * @Description: com.fsd07.springbootlibrary.dao
10  * @Version: 1.0
11 */
12 public interface ReviewRepository extends JpaRepository<Review, Long> {
13 }
14
```

Run the application.

open postman, Choose “GET”, Type <http://localhost:8080/api>, we can see that now we have a review api.

http://localhost:8080/api

GET http://localhost:8080/api

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results Status: 200 OK Time: 335 ms Size: 602 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_links": {
3     "reviews": {
4       "href": "http://localhost:8080/api/reviews?page,size,sort",
5       "templated": true
6     },
7     "books": {
8       "href": "http://localhost:8080/api/books?page,size,sort",
9       "templated": true
10    },
11    "profile": {
12      "href": "http://localhost:8080/api/profile"
13    }
14  }
15 }
```

Choose “GET”, Type <http://localhost:8080/api/reviews>, we can get all the reviews but no review's Id.

GET http://localhost:8080/api/reviews

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results Status: 200 OK Time: 377 ms Size: 1.42 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_embedded": {
3     "reviews": [
4       {
5         "userEmail": "example1user@email.com",
6         "date": "2023-09-03T23:20:14.000+00:00",
7         "rating": 4.0,
8         "bookId": 1,
9         "reviewDescription": "First book is pretty good book overall",
10        "_links": {
11          "self": {
12            "href": "http://localhost:8080/api/reviews/1"
13          },
14          "review": {
15            "href": "http://localhost:8080/api/reviews/1"
16          }
17        }
18      },
19      {
20        "userEmail": "example2user@email.com",
21        "date": "2023-09-03T23:20:14.000+00:00",
22        "rating": 4.5,
23        "bookId": 2,
24        "reviewDescription": "Second books is pretty good book overall",
25        ...
26      }
27    ]
28  }
29 }
```

In total, there are two reviews

```
27     "href": "http://localhost:8080/api/reviews/2"
28   },
29   "review": {
30     "href": "http://localhost:8080/api/reviews/2"
31   }
32 }
33 }
34 ]
35 },
36 "_links": {
37   "self": {
38     "href": "http://localhost:8080/api/reviews"
39   },
40   "profile": {
41     "href": "http://localhost:8080/api/profile/reviews"
42   }
43 },
44 "page": {
45   "size": 20,
46   "totalElements": 2,
47   "totalPages": 1,
48   "number": 0
49 }
50 }
```

Go to MyDataRestConfig class under config package, expose the reviews id and only allow get method.

```
@Override
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config,
                                                CorsRegistry cors) {

    HttpMethod[] theUnsupportedActions = {
        HttpMethod.POST,
        HttpMethod.PATCH,
        HttpMethod.DELETE,
        HttpMethod.PUT};

    config.exposeIdsFor(Book.class);
    config.exposeIdsFor(Review.class);

    disableHttpMethods(Book.class, config, theUnsupportedActions);
    disableHttpMethods(Review.class, config, theUnsupportedActions);
}
```

Restart the application.

Go to postman, Choose “GET”, Type <http://localhost:8080/api/reviews>, we can get all the reviews with review’s Id.

http://localhost:8080/api/reviews

GET http://localhost:8080/api/reviews

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2    "_embedded": {
3      "reviews": [
4        {
5          "id": 1,
6          "userEmail": "example1user@email.com",
7          "date": "2023-09-03T23:20:14.000+00:00",
8          "rating": 4.0,
9          "bookId": 1,
10         "reviewDescription": "First book is pretty good book overall",
11         "_links": {
12           "self": {
13             "href": "http://localhost:8080/api/reviews/1"
14           },
15           "review": {
16             "href": "http://localhost:8080/api/reviews/1"
17           }
18         }
19       },
20       {
21         "id": 2,
22         "userEmail": "example2user@email.com",
23         "date": "2023-09-03T23:20:14.000+00:00",
24         "rating": 4.5,
25         "bookId": 2,
26         "reviewDescription": "Second books is pretty good book overall",
27         "_links": {
28           "self": {
29             "href": "http://localhost:8080/api/reviews/2"
30           },
31         "review": {
```

### 3. SPRINGBOOT-FIND REVIEW BY ID

Go to ReviewRepository , add an interface function

```
15 public interface ReviewRepository extends JpaRepository<Review, Long> {  
16     Page<Review> findByBookId(@RequestParam("book_id") Long bookId,  
17                               Pageable pageable);  
18 }
```

Rerun the application. Go to postman, Choose "GET", Type <http://localhost:8080/api/reviews>, we can see it has a search endpoint.

```
37 {  
38     "_links": {  
39         "self": {  
40             "href": "http://localhost:8080/api/reviews"  
41         },  
42         "profile": {  
43             "href": "http://localhost:8080/api/profile/reviews"  
44         },  
45         "search": {  
46             "href": "http://localhost:8080/api/reviews/search"  
47         }  
48     },  
49     "page": {  
50         "size": 20,  
51         "totalElements": 2,  
52         "totalPages": 1,  
53         "number": 0  
54     }  
55 }
```

Choose “GET”, Type <http://localhost:8080/api/reviews/search>, we can see it has a findByBookId endpoint.

http://localhost:8080/api/reviews/search

GET <http://localhost:8080/api/reviews/search>

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON v ≡

```
1  {
2    "_links": {
3      "findByBookId": {
4        "href": "http://localhost:8080/api/reviews/search/findByBookId{?bookId,page,size,sort}",
5        "templated": true
6      },
7      "self": {
8        "href": "http://localhost:8080/api/reviews/search"
9      }
10   }
11 }
```

If we type: <http://localhost:8080/api/reviews/search/findByBookId?bookId=1>, we can get the result.

http://localhost:8080/api/reviews/search/findByBookId?bookId=1 Save v ≡

GET <http://localhost:8080/api/reviews/search/findByBookId?bookId=1> Send v

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> bookId	1			
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 20 ms Size: 999 B Save Response v

Pretty Raw Preview Visualize JSON v ≡

```
1  {
2    "_embedded": {
3      "reviews": [
4        {
5          "id": 1,
6          "userEmail": "example1user@email.com",
7          "date": "2023-09-03T23:20:14.000+00:00",
8          "rating": 4.0,
9          "bookId": 1,
10         "reviewDescription": "First book is pretty good overall",
11         "_links": {
12           "self": {
13             "href": "http://localhost:8080/api/reviews/1"
14           },
15           "review": {
16             "href": "http://localhost:8080/api/reviews/1"
17           }
18         }
19       }
20     ]
21   }
```



If we type: <http://localhost:8080/api/reviews/search/findByBookId?bookId=4>, we can find 0 element.

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/reviews/search/findByBookId?bookId=4`
- Method:** GET
- Query Params:** A table with one entry: 

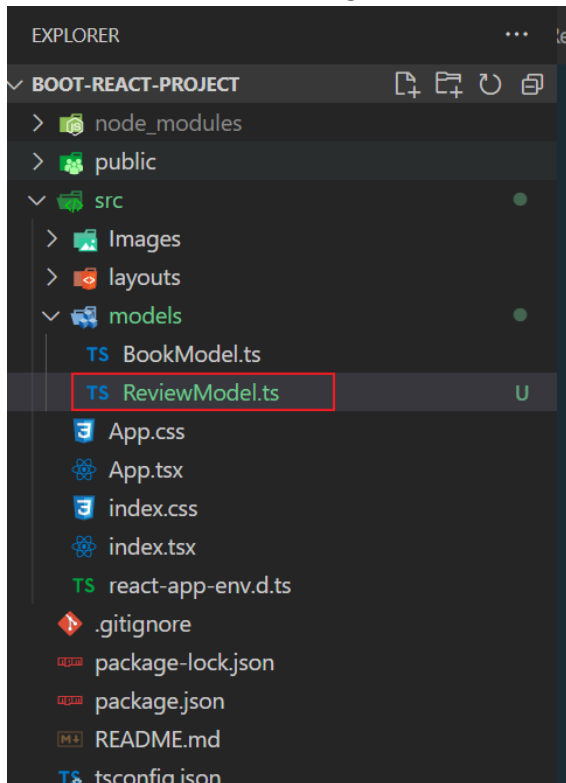
Key	Value	Description
bookId	4	
- Status:** 200 OK, Time: 16 ms, Size: 552 B
- Response Body (JSON):**

```
1 {
2   "_embedded": {
3     "reviews": []
4   },
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/api/reviews/search/findByBookId?bookId=4&page=0&size=20"
8     }
9   },
10  "page": {
11    "size": 20,
12    "totalElements": 0,
13    "totalPages": 0,
14    "number": 0
15  }
16 }
```

The `"page": { ... }` section is highlighted with a red box.

## 4. REACT- CREATE THE REVIEW MODEL

Go to our “models” folder, right click, choose “new file”. Name it ReviewModel.ts



In the ReviewModel.ts, create the properties which are the same as the review entity in our backend.

```
src > models > TS ReviewModel.ts > ReviewModel > constructor
1  class ReviewModel {
2      id: number;
3      userEmail: string;
4      date: string;
5      rating: number;
6      book_id: number;
7      reviewDescription?: string;
8
9      constructor(
10         id: number,
11         userEmail: string,
12         date: string,
13         rating: number,
14         book_id: number,
15         reviewDescription: string
16     ) {
17         this.id = id;
18         this.userEmail = userEmail;
19         this.date = date;
20         this.rating = rating;
21         this.book_id = book_id;
22         this.reviewDescription = reviewDescription;
23     }
24 }
25
```

## 5. REACT-REVIEW USE EFFECT

Go to BookCheckoutPage.tsx, create review useState and import ReviewModel

```
src > layouts > BookCheckOutPage > BookCheckOutPage.tsx > ...
You, 38 seconds ago | 1 author (You)
1 import { useEffect, useState } from "react";
2 import BookModel from "../../models/BookModel";
3 import ReviewModel from "../../models/ReviewModel";
4 import { SpinnerLoading } from "../Utils/SpinnerLoading";
5 import { StarsReview } from "../Utils/StarsReview";
6 import { CheckoutAndReviewBox } from "../CheckOutAndReviewBox";
7
8 export const BookCheckOutPage = () => {
9   // create useState
10  const [book, setBook] = useState<BookModel>();
11  const [isLoading, setIsLoading] = useState(true);
12  const [httpError, setHttpError] = useState(null);
13
14  //Review State
15  const [reviews, setReviews] = useState<ReviewModel[]>([]);
16  const [totalStars, setTotalStars] = useState(0);
17  const [isLoadingReview, setIsLoadingReview] = useState(true);
18
```

create review useEffect

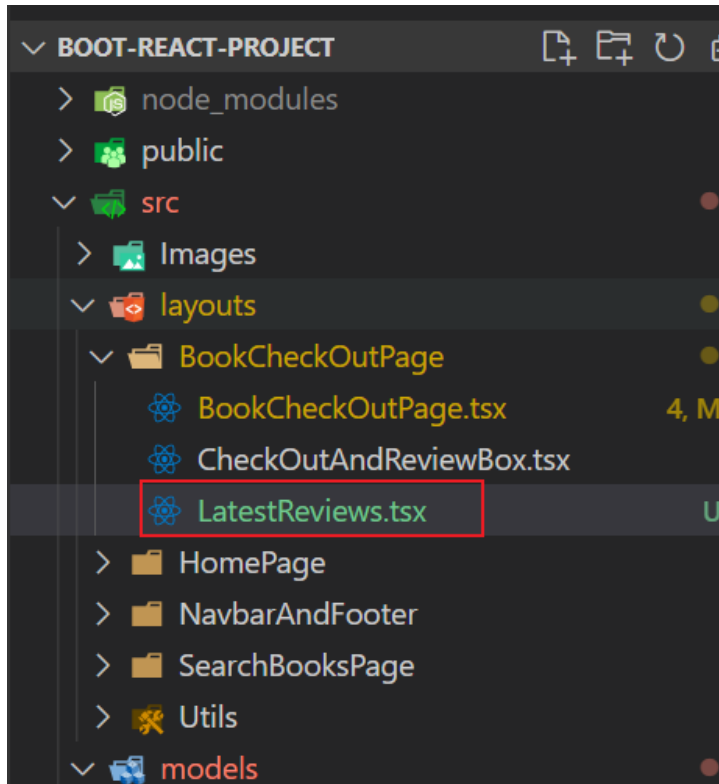
```
src > layouts > BookCheckOutPage > BookCheckOutPage.tsx > BookCheckOutPage > useEffect() callback > fetchBookReviews > book_id
53
54 //create review useEffect
55 useEffect(() => {
56   const fetchBookReviews = async () => {
57     const reviewUrl: string = `http://localhost:8080/api/reviews/search/findByBookId?bookId=${bookId}`;
58     const responseReviews = await fetch(reviewUrl);
59     if (!responseReviews.ok) {
60       throw new Error("Something went wrong!");
61     }
62     const responseJsonReviews = await responseReviews.json();
63     const responseData = responseJsonReviews._embedded.reviews;
64     const loadedReviews: ReviewModel[] = [];
65     let weightedStarReviews: number = 0;
66     for (const key in responseData) {
67       loadedReviews.push({
68         id: responseData[key].id,
69         userEmail: responseData[key].userEmail,
70         date: responseData[key].date,
71         rating: responseData[key].rating,
72         book_id: responseData[key].bookId,
73         reviewDescription: responseData[key].reviewDescription,
74       });
75       weightedStarReviews = weightedStarReviews + responseData[key].rating;
76     }
77
78     if (loadedReviews) {
79       const round = (
80         Math.round((weightedStarReviews / loadedReviews.length) * 2) / 2
81       ).toFixed(1);
82       setTotalStars(Number(round));
83     }
84
85     setReviews(loadedReviews);
86     setIsLoadingReview(false);
87   };
88   fetchBookReviews().catch((error: any) => {
89     setIsLoadingReview(false);
90     setHttpError(error.message);
91   });
92 }, []);
```

Add isLoadingReview in this code

```
93
94  if (isLoading || isLoadingReview) {
95      return <SpinnerLoading />;
96  }
97
98  if (httpError) {
99      return (
100      <div className="container m-5">
101          <p>{httpError}</p>
102      </div>
103      );
104  }
```

## 6. REACT-CREATE LATEST REVIEWS COMPONENT

Go to our “BookCheckoutPage” folder, create a new file “LatestReviews.tsx”



create export const LatestReviews() structure, adding a review object as its props.

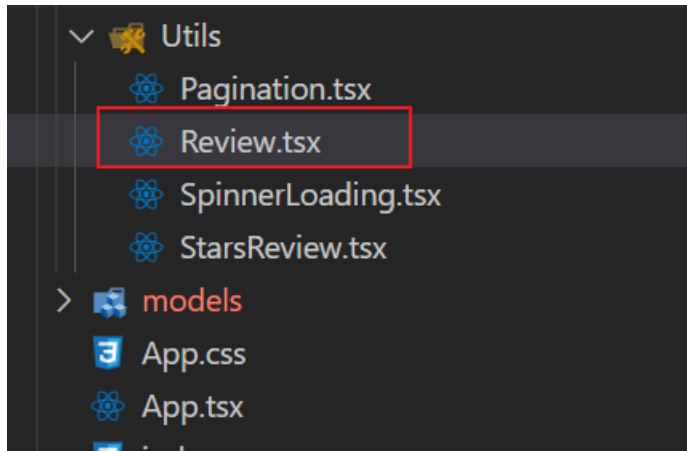
```
src > layouts > BookCheckOutPage > LatestReviews.tsx > LatestReviews
1  import ReviewModel from "../../models/ReviewModel";
2
3  export const LatestReviews: React.FC<{
4    reviews: ReviewModel[];
5    bookId: number | undefined;
6    mobile: boolean;
7  }> = (props) => {
8    return (
9
10   );
11  };
12
```

Fill in return part with HTML/CSS code

```
10 <div className={props.mobile ? "mt-3" : "row mt-5"}>
11   <div className={props.mobile ? "" : "col-sm-2 col-md-2"}>
12     <h2>Latest Reviews:</h2>
13   </div>
14   <div className="col-sm-10 col-md-10">
15     {props.reviews.length > 0 ? (
16       <>
17         {props.reviews.slice(0, 3).map((eachReview) => (
18           <Review review={eachReview} key={eachReview.id}></Review>
19         ))}
20         <div className="m-3">
21           <Link
22             type="button"
23             className="btn main-color btn-md text-white"
24             to="#"
25           >
26             Reach all reviews
27           </Link>
28         </div>
29       </>
30     ) : (
31       <div className="m-3">
32         <p className="lead">
33           Currently there are no reviews for this book.
34         </p>
35       </div>
36     )}
37   </div>
38 </div>
```

## 7. REACT-REVIEW COMPONENT

Go to our “Utils” folder, create a new file “Review.tsx”



create export const Review() structure, adding an object review as its props

```
src > layouts > Utils > Review.tsx > [🔗] Review
1  import ReviewModel from "../../models/ReviewModel";
2
3  export const Review: React.FC<{ review: ReviewModel }> = (props) => {
4    return <div></div>;
5  };
6
```

create useState.

```
src > layouts > Utils > Review.tsx > ...
1  import ReviewModel from "../../models/ReviewModel";
2
3  export const Review: React.FC<{ review: ReviewModel }> = (props) => {
4    // create a new date with the string that we're getting from the backend,
5    // adjusting some of the data so we can create a string and present
6    // the date how we want to present the date in the application.
7    const date = new Date(props.review.date);
8    const longMonth = date.toLocaleString("en-us", { month: "long" });
9    const dateDay = date.getDate();
10   const dateYear = date.getFullYear();
11   const dateRender = longMonth + " " + dateDay + ", " + dateYear;
12   return <div></div>;
13 };
14
```

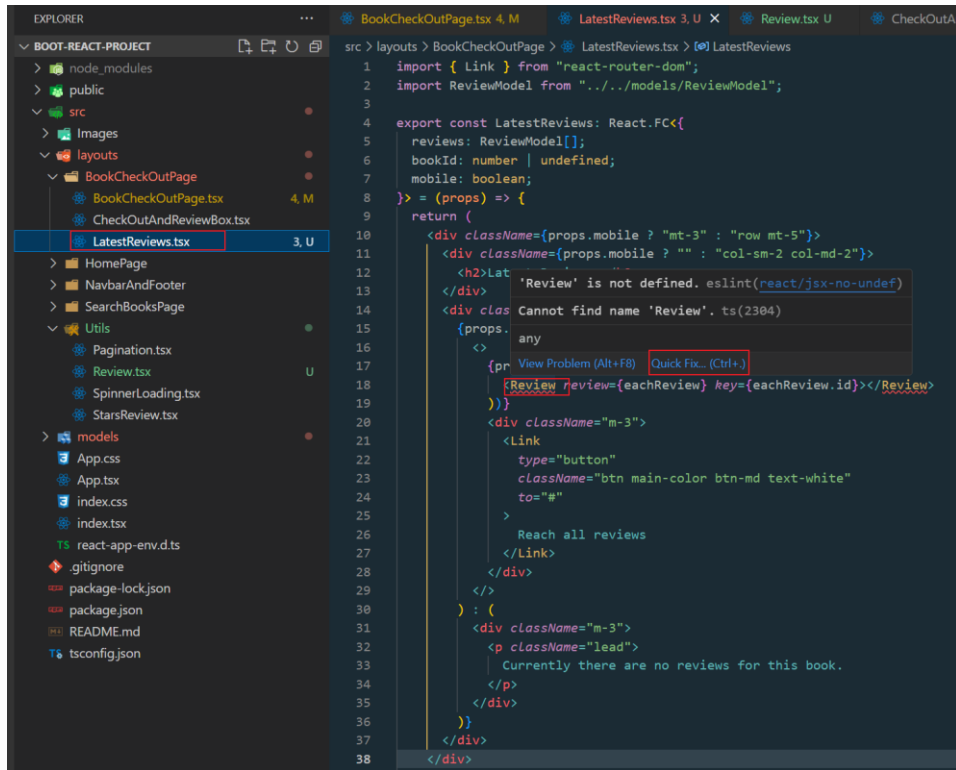
Fill in return part with HTML/CSS code.

```
13   return (  
14     <div>  
15       <div className="col-sm-8 col-md-8">  
16         <h5>{props.review.userEmail}</h5>  
17         <div className="row">  
18           <div className="col">{dateRender}</div>  
19           <div className="col">  
20             <StarsReview rating={props.review.rating} size={16} />  
21           </div>  
22         </div>  
23         <div className="mt-2">  
24           <p>{props.review.reviewDescription}</p>  
25         </div>  
26       </div>  
27     <hr />  
28   </div>  
29 );  
30 };
```



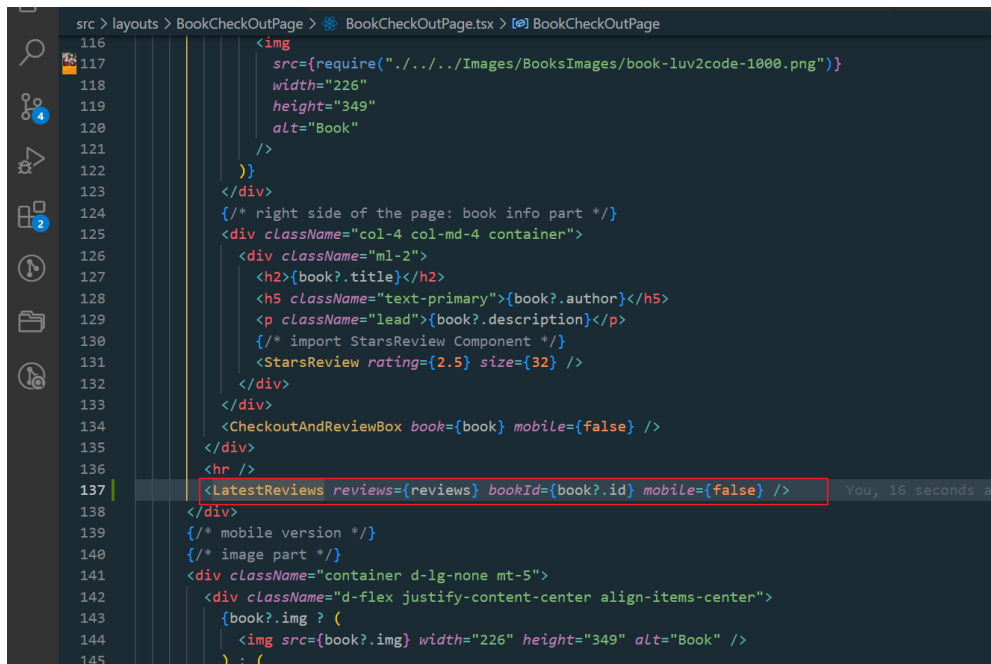
## 8. REACT-WRAP UP REVIEWS

Let's go to latestReview.tsx import review component, fix the error.



```
1 import { Link } from "react-router-dom";
2 import ReviewModel from "../../models/ReviewModel";
3
4 export const LatestReviews: React.FC<
5   reviews: ReviewModel[];
6   bookId: number | undefined;
7   mobile: boolean;
8 >> = (props) => {
9   return (
10     <div className={props.mobile ? "mt-3" : "row mt-5"}>
11       <div className={props.mobile ? "" : "col-sm-2 col-md-2"}>
12         <h2>Lat
13       </div>
14       <div clas
15         {props.
16         any
17         <pr
18         View Problem (Alt+F8) Quick Fix... (Ctrl+)
19         <Review review={eachReview} key={eachReview.id}></Review>
20       </div>
21       <div className="m-3">
22         <Link
23           type="button"
24           className="btn main-color btn-md text-white"
25           to="#"
26         >
27           Reach all reviews
28         </Link>
29       </div>
30     ) : (
31       <div className="m-3">
32         <p className="lead">
33           Currently there are no reviews for this book.
34         </p>
35       </div>
36     )
37   )
38 </div>
```

Go to BookCheckoutPage.tsx, add latestReview component under the CheckoutAndReviewBox component. In the desktop version part of the code, set the value of “mobile” as false.

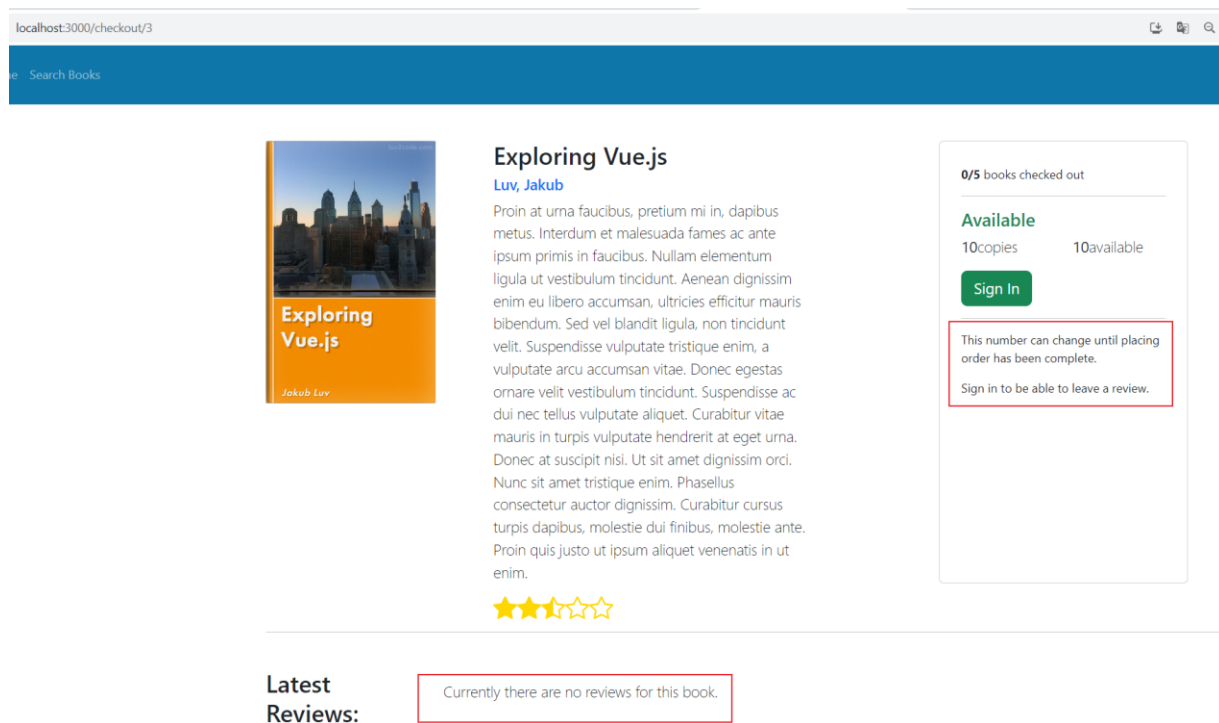


```
116 <img
117   src={require("../Images/BooksImages/book-luv2code-1000.png")}
118   width="226"
119   height="349"
120   alt="Book"
121 />
122 </div>
123 </div>
124 /* right side of the page: book info part */
125 <div className="col-4 col-md-4 container">
126   <div className="m1-2">
127     <h2>{book?.title}</h2>
128     <h5 className="text-primary">{book?.author}</h5>
129     <p className="lead">{book?.description}</p>
130     /* import StarsReview Component */
131     <StarsReview rating={2.5} size={32} />
132   </div>
133 </div>
134 <CheckoutAndReviewBox book={book} mobile={false} />
135 </div>
136 <hr />
137 <LatestReviews reviews={reviews} bookId={book?.id} mobile={false} />
138 </div>
139 /* mobile version */
140 /* image part */
141 <div className="container d-lg-none mt-5">
142   <div className="d-flex justify-content-center align-items-center">
143     {book?.img ? (
144       <img src={book?.img} width="226" height="349" alt="Book" />
145     ) : (
```

Add latestReview component under the CheckoutAndReviewBox component. In the mobile version part of the code too, but set the value of “mobile” as true.


```
154     { /* book info part */  
155     <div className="mt-4">  
156         <div className="ml-2">  
157             <h2>{book?.title}</h2>  
158             <h5 className="text-primary">{book?.author}</h5>  
159             <p className="lead">{book?.description}</p>  
160             { /* import StarsReview Component */  
161             <StarsReview rating={2.5} size={32} />  
162         </div>  
163     </div>  
164     <CheckoutAndReviewBox book={book} mobile={true} />  
165     <hr />  
166     <LatestReviews reviews={reviews} bookId={book?.id} mobile={true} />  
167 </div>  
168 </div>  
169 );
```

Then run the application, make sure your backend spring boot is running. If the book has no reviews, we will get this checkout page:



localhost:3000/checkout/3

Search Books



### Exploring Vue.js

Luv, Jakub

Proin at urna faucibus, pretium mi in, dapibus metus. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nullam elementum ligula ut vestibulum tincidunt. Aenean dignissim enim eu libero accumsan, ultricies efficitur mauris bibendum. Sed vel blandit ligula, non tincidunt velit. Suspendisse vulputate tristique enim, a vulputate arcu accumsan vitae. Donec egestas ornare velit vestibulum tincidunt. Suspendisse ac dui nec tellus vulputate aliquet. Curabitur vitae mauris in turpis vulputate hendrerit at eget urna. Donec at suscipit nisi. Ut sit amet dignissim orci. Nunc sit amet tristique enim. Phasellus consectetur auctor dignissim. Curabitur cursus turpis dapibus, molestie dui finibus, molestie ante. Proin quis justo ut ipsum aliquet venenatis in ut enim.

★★★★☆

0/5 books checked out

**Available**

10copies 10available

[Sign In](#)

This number can change until placing order has been complete.

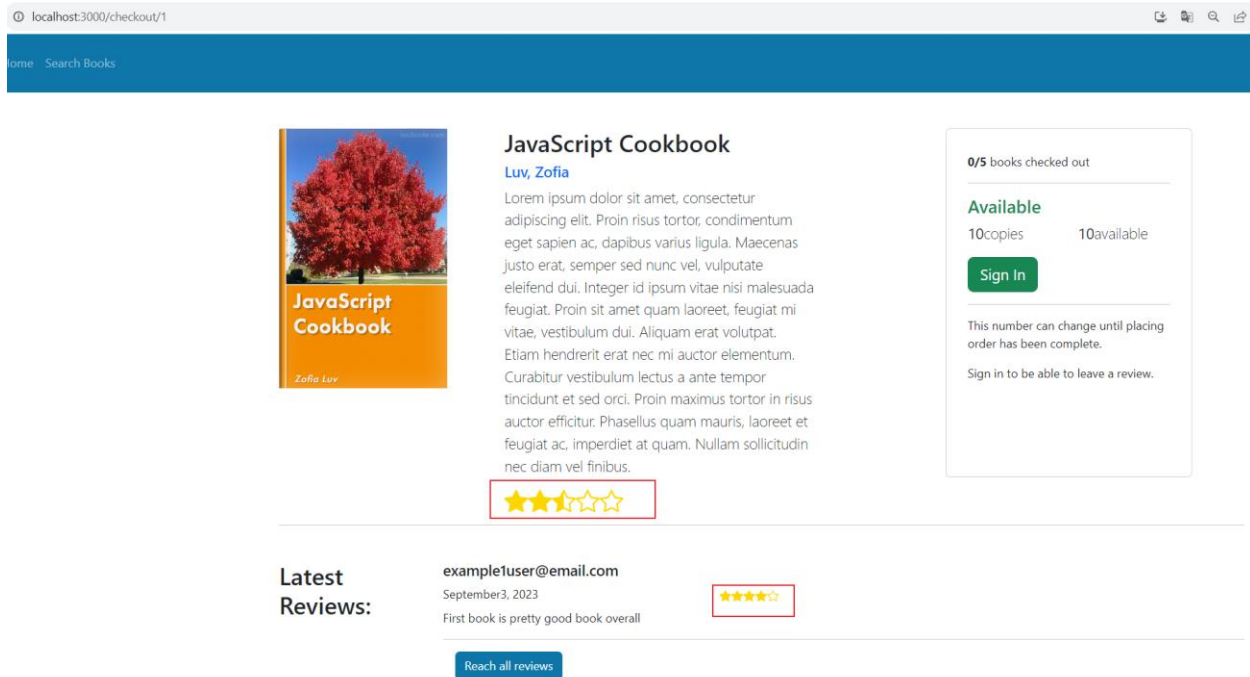
Sign in to be able to leave a review.

**Latest Reviews:**

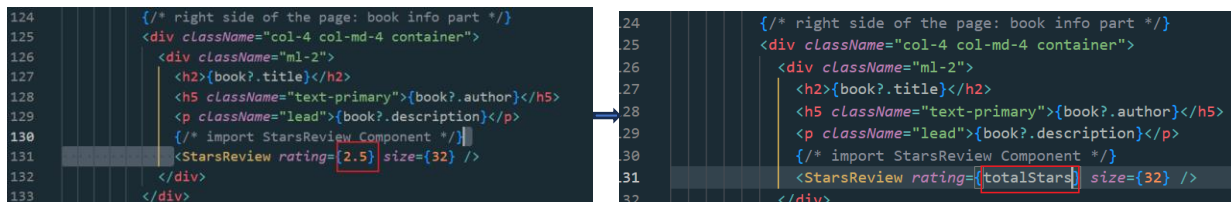
Currently there are no reviews for this book.

If the book has a review, we will get this checkout page.

Since we just have one review for this book right now, the star review value in this component should equal to the value in the book details above. But you can see, they do not equal. We will fix it.



Go to BookCheckoutPage.tsx, find StarsReview component, change the fixed rating value into dynamical value.



Rerun the application. Now, we can see these two star reviews' value become the same.

