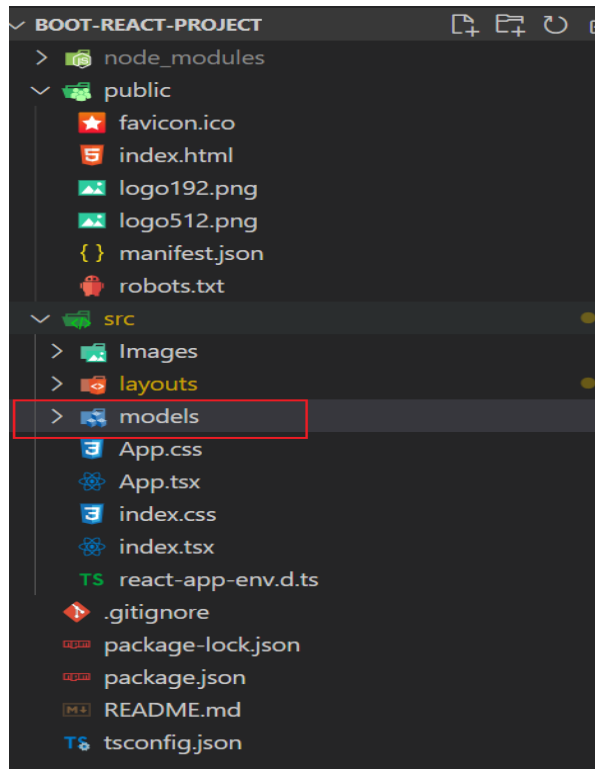


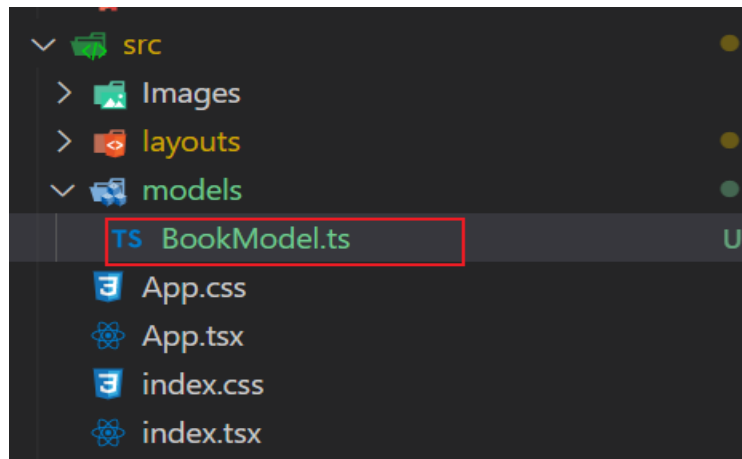
# S03 CONSUME CAROUSEL BOOK API

## 1. SETUP BOOK MODEL

Create a new folder “models” under the src directory



Create a new file “BookModel.ts” under the models folder.



In BookModels.ts, create the class with the same properties as the bookModel in our springboot app.

```
src > models > TS BookModel.ts > BookModel > constructor
1  class BookModel {
2      id: number;
3      title: string;
4      author?: string;
5      description?: string;
6      copies?: number;
7      copiesAvailable?: number;
8      category?: string;
9      img?: string;
10
11  constructor(
12      id: number,
13      title: string,
14      author: string,
15      description: string,
16      copies: number,
17      copiesAvailable: number,
18      category: string,
19      img: string
20  ) {
21      this.id = id;
22      this.title = title;
23      this.author = author;
24      this.description = description;
25      this.copies = copies;
26      this.copiesAvailable = copiesAvailable;
27      this.category = category;
28      this.img = img;
29  }
30  }
31
32  export default BookModel;
```

## 2. USESTATE AND USEEFFECT

import useSate , useEffect from react. And import our new created BookModel.

```
src > layouts > HomePage > components > Carousel.tsx > ...
You, 3 minutes ago | 1 author (You)
1 import { ReturnBook } from "../ReturnBook";
2 import { useEffect, useState } from "react";
3 import BookModel from "../../models/BookModel";
4
```

create useState for books, isLoading and httpError.

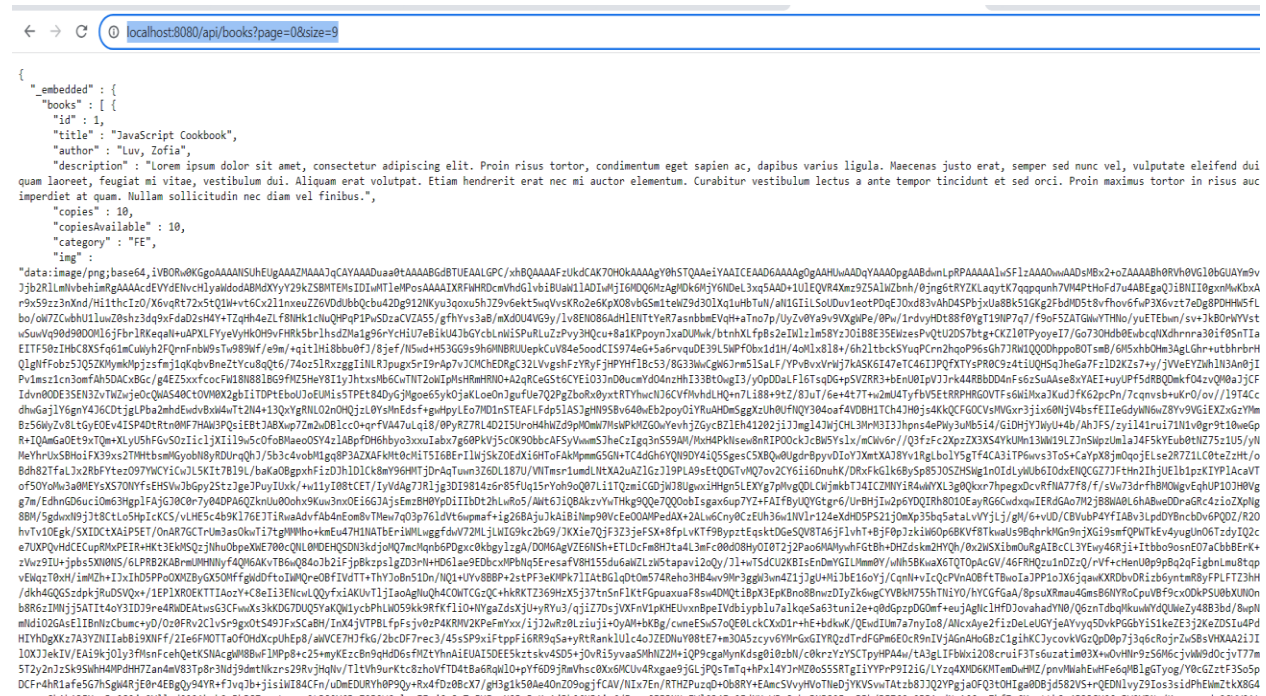
```
export const Carousel = () => {
  // Create a book array
  const [books, setBooks] = useState<BookModel[]>([]);
  // For judging whether in loading
  const [isLoading, setIsLoading] = useState(true);
  // API call failure scenario
  const [httpError, setHttpError] = useState(null);
}
```

create useEffect. useEffect is going to be our second hook where we're able to call some kind of function or API.

create an aysnc function fetchBooks() to call the backed Api.

```
useEffect(() => {
  const fetchBooks = async () => {
    const baseUrl: string = "http://localhost:8080/api/books";
    const url: string = `${baseUrl}?page=0&size=9`;
    const response = await fetch(url);
    if (!response.ok) {
      throw new Error("Something went wrong!");
    }
    const responseJson = await response.json();
    const responseData = responseJson._embedded.books;
    const loadedBooks: BookModel[] = [];
    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,
      });
    }
    setBooks(loadedBooks);
    setIsLoading(false);
  };
  fetchBooks().catch((error: any) => {
    setIsLoading(false);
    setHttpError(error.message);
  });
}, []);
```

when we visit <http://localhost:8080/api/books?page=0&size=9>, we can get all the books in our database.



Add loading process and httpError judgement.



Pass a book object as props. Then we set the image be this book's image, the title be this book's title, the author be this book's author.

```
export const ReturnBook: React.FC<{ book: BookModel }> = (props) => {
  return (
    <div className="col-xs-6 col-sm-6 col-md-4 col-lg-3 mb-3">
      <div className="text-center">
        {props.book.img ? (
          <img src={props.book.img} alt="book" width="151" height="233" />
        ) : (
          <img
            src={require("../Images/BooksImages/book-luv2code-1000.png")}
            alt="book"
            width="151"
            height="233"
          />
        )}
      <h6 className="mt-2">{props.book.title}</h6>
      <p>{props.book.author}</p>
      <a className="btn main-color text-white" href="#">
        Reserve
      </a>
    </div>
  </div>
);
};
```

In Carousel.tsx, call ReturnBook component part, passing the props we have add in the ReturnBook.tsx: a book object and its id as key. Add a slice() to specify the index of books.

```
<div className="carousel-inner">
  <div className="carousel-item active">
    <div className="row d-flex justify-content-center align-items-center">
      {books.slice(0, 3).map((book) => (
        <ReturnBook book={book} key={book.id} />
      ))}
    </div>
  </div>

  <div className="carousel-item">
    <div className="row d-flex justify-content-center align-items-center">
      {books.slice(3, 6).map((book) => (
        <ReturnBook book={book} key={book.id} />
      ))}
    </div>
  </div>

  <div className="carousel-item">
    <div className="row d-flex justify-content-center align-items-center">
      {books.slice(6, 9).map((book) => (
        <ReturnBook book={book} key={book.id} />
      ))}
    </div>
  </div>
</div>
```

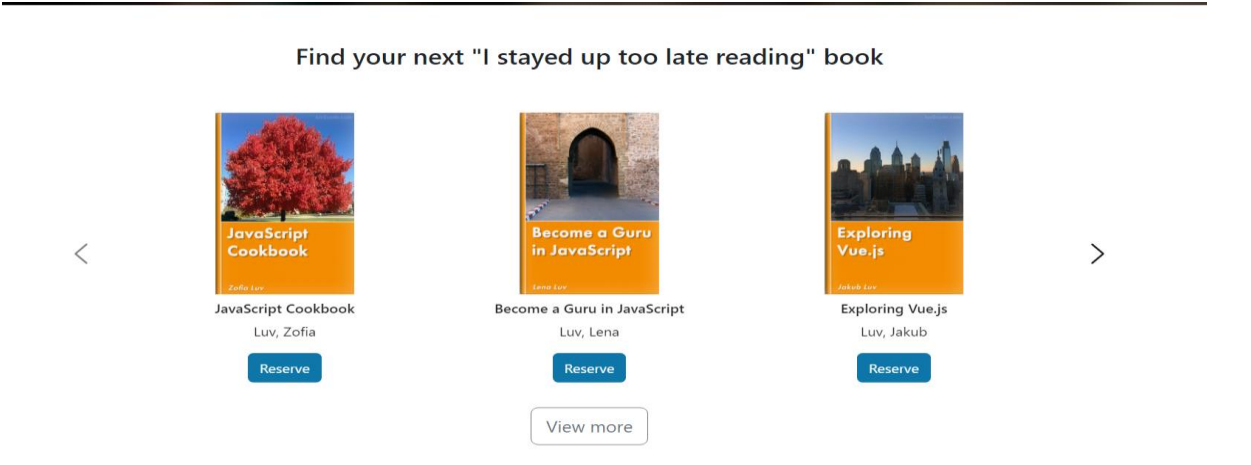
Fix the mobile version. In Mobile version, we just want to present one book in the Carousel component, so here we specify a book id to the key.

```

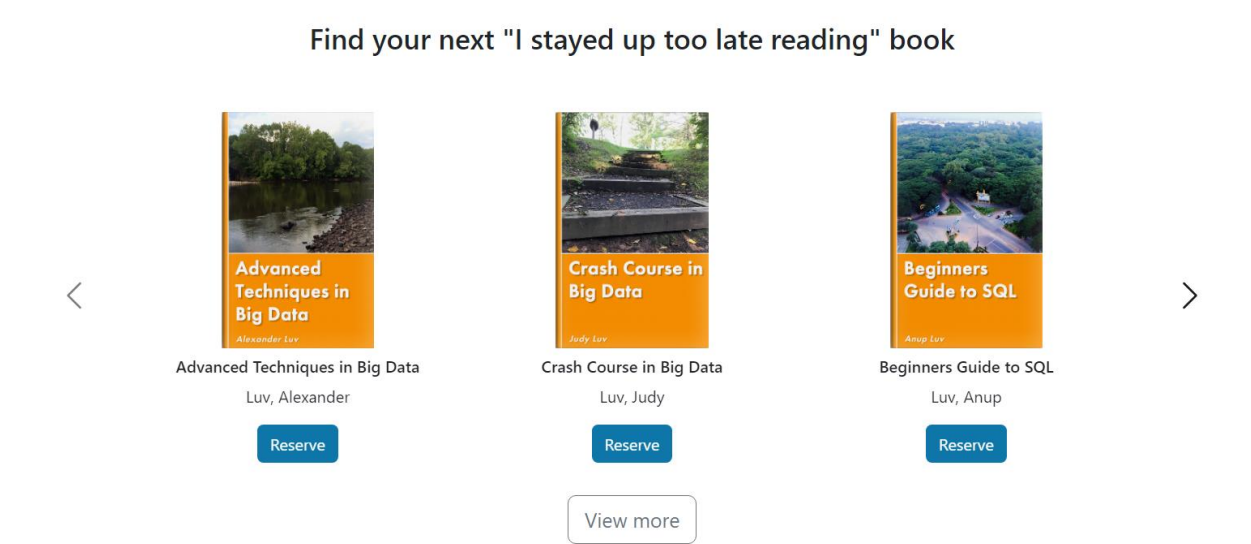
{ /* mobile */
  <div className="d-lg-none mt-3">
    <div className="row d-flex justify-content-center align-items-center">
      <ReturnBook book={books[7]} key={books[7].id} />
    </div>
  </div>
}

```

Run the application. We can get the page.

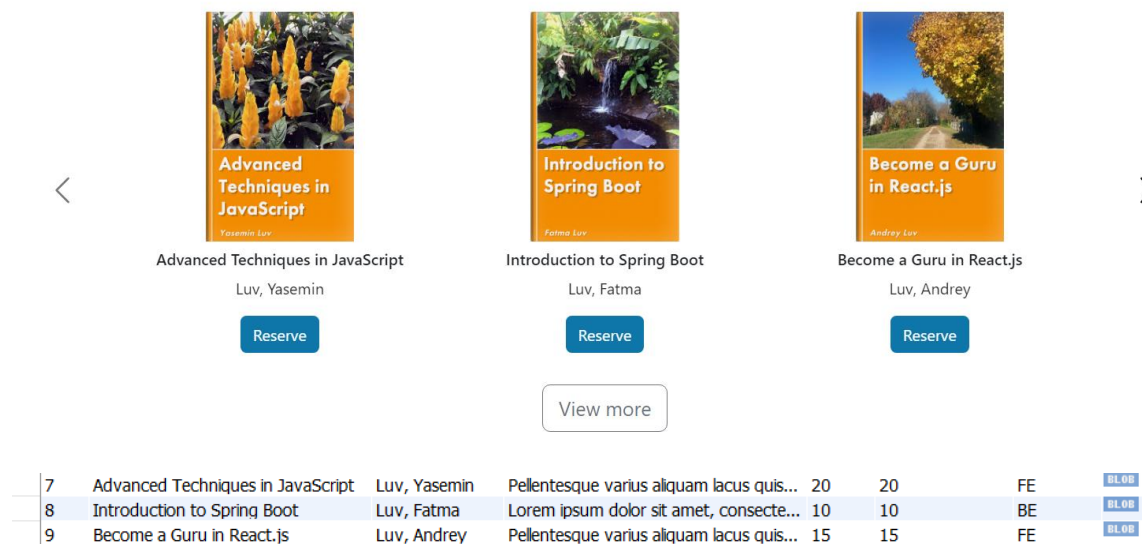


Click on the right arrow button, go to the second part of carousel items. The result page is as below. If we check our database, we can see they are the book 4 to book 6 in our database.



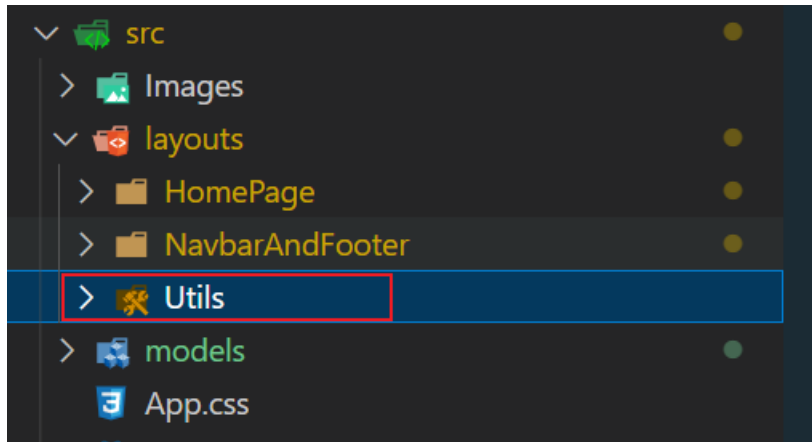
4	Advanced Techniques in Big Data	Luv, Alexander	Nunc eget lorem ac neque tincidunt ...	12	12	Data	BLOB
5	Crash Course in Big Data	Luv, Judy	Morbi eu tempus eros, in imperdiet se...	20	20	Data	BLOB
6	Beginners Guide to SQL	Luv, Anup	Lorem ipsum dolor sit amet, consecte...	20	20	Data	BLOB

Click on the right arrow button again, go to the second part of carousel items. The result page is as below. If we check our database, we can see they are the book 7 to book 9 in our database.

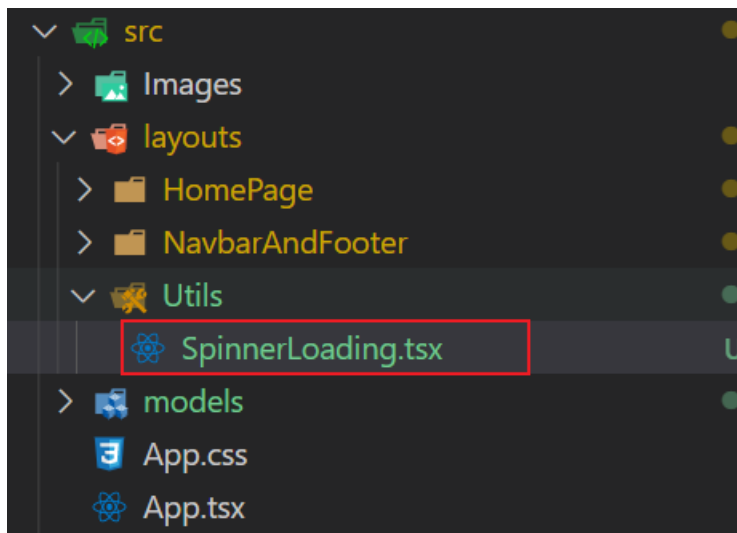


### 3. LOADING SPINNER

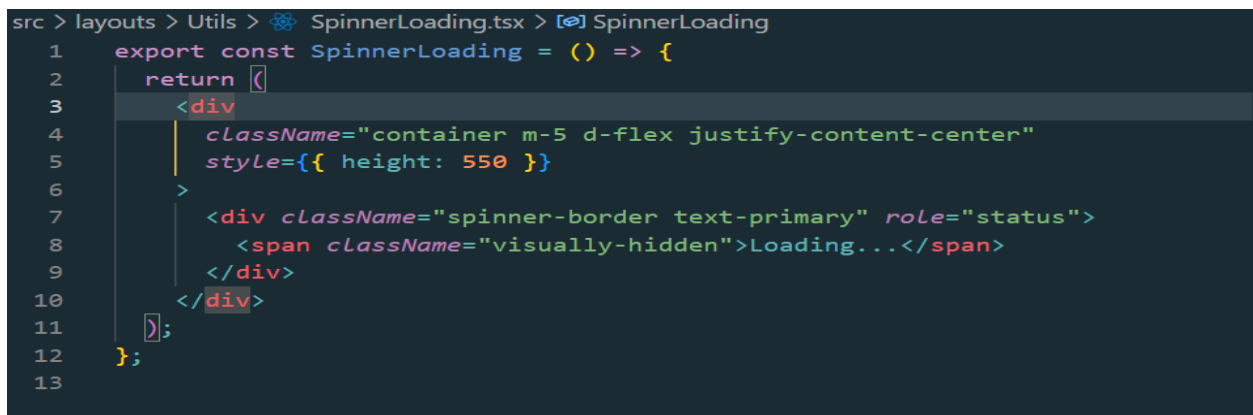
Create a new folder "Utils" under the "layouts" folder.



Under the Utils folder create a new file "SpinnerLoading.tsx"




Write the code in it.





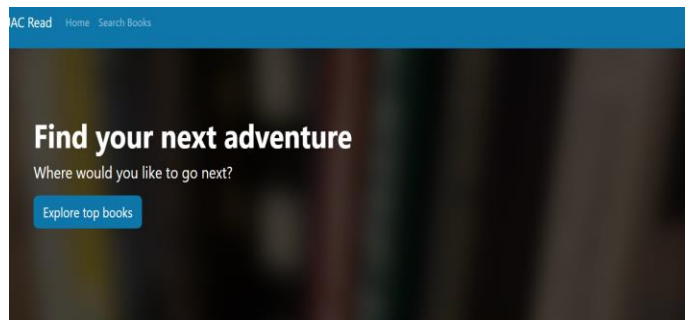
In Carousel.tsx, replace the static loading process code with our new created SpinnerLoading component.

```
if (isLoading) {  
  return (  
    <div className="container m-5">  
      <p>Loading...</p>  
    </div>  
  );  
}
```



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

Rerun the application. When the app is in loading process, a spinner will show:



In Mobile mode:

