

S04 SEARCH BOOKS COMPONENT

Search Books Component

- Display all book information

Number of results and how many are being displayed

Book image from database

Book title from database

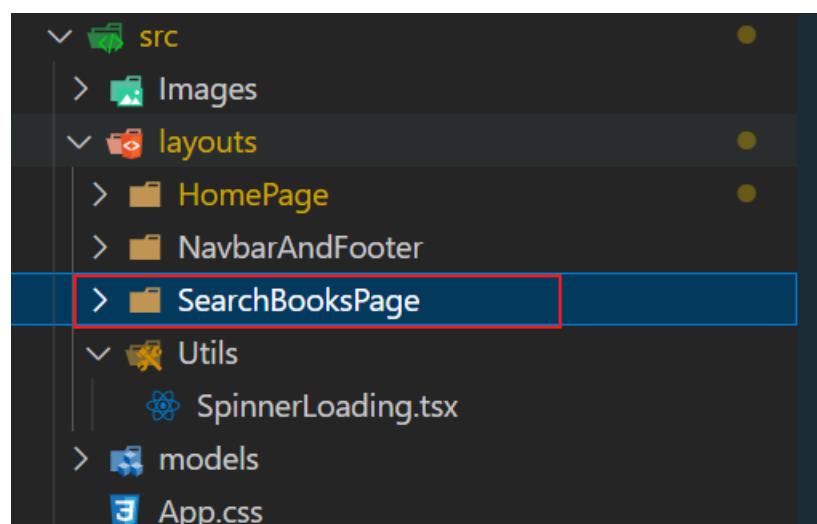
Book author from database

Book description from database

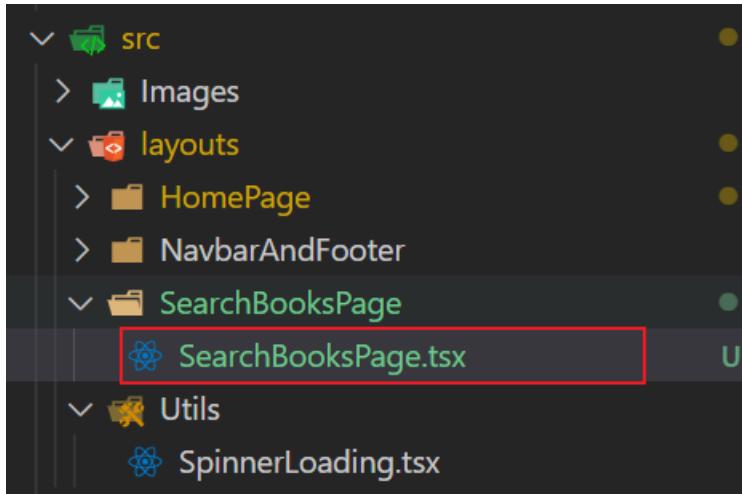
View Details Button

1. REACT-SEARCH BOOKS PAGE

Create a new folder “SearchBooksPage” under the layouts folder.



Create a new file “SearchBooksPage.tsx”



create export const SearchBookPage () structure. Create useState for books, isLoading and httpError. Go to Carousel.tsx file, copy these three useState.

```
/outs > SearchBooksPage > ⚙️ SearchBooksPage.tsx > ⚙️ SearchBooksPage
  export const SearchBooksPage = () => [
    // 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);
  ];
```

Fix the issues by importing some packages.

```
src > layouts > SearchBooksPage > ⚙️ SearchBooksPage.tsx > ⚙️ SearchBooksPage
1  import { useState } from "react";
2  import BookModel from "../../models/BookModel";
3
4  export const SearchBooksPage = () => {
5    // Create a book array
6    const [books, setBooks] = useState<BookModel[]>([[]]);
7    // For judging whether in loading
8    const [isLoading, setIsLoading] = useState(true);
9    // API call failure scenario
10   const [httpError, setHttpError] = useState(null);
11 };
12
```

Create useEffect in SearchBooksPage.tsx. Similar to the useState, we go to Carousel.tsx, copy the useEffect code. Paste them into SearchBookPage.tsx file.

```
> layouts > SearchBooksPage > SearchBooksPage.tsx > SearchBooksPage
1  useEffect(() => {
2    const fetchBooks = async () => {
3      const baseUrl: string = "http://localhost:8080/api/books";
4      const url: string = `${baseUrl}?page=0&size=9`;
5      // fetching the url data
6      const response = await fetch(url);
7      //failure scenario
8      if (!response.ok) {
9        throw new Error("Something went wrong!");
10       }
11      // transfer the url data into json;
12      const responseJson = await response.json();
13      // get the data which is the object of embedded books
14      const responseData = responseJson._embedded.books;
15      // create a book array
16      const loadedBooks: BookModel[] = [];
17      // iterate the object in responseData, push them into the book array, and set the loadedBooks
18      for (const key in responseData) {
19        loadedBooks.push({
20          id: responseData[key].id,
21          title: responseData[key].title,
22          author: responseData[key].author,
23          description: responseData[key].description,
24          copies: responseData[key].copies,
25          copiesAvailable: responseData[key].copiesAvailable,
26          category: responseData[key].category,
27          img: responseData[key].img,
28        });
29      }
30      setBooks(loadedBooks);
31      setIsLoading(false);
32    };
33    fetchBooks().catch((error: any) => {
34      setIsLoading(false);
35      setHttpError(error.message);
36    });
37  }, []);
38
```

Fix the issue by importing the useEffect.

```
1  import { useEffect, useState } from "react";
2  import BookModel from "../../models/BookModel";
```

fix the url. Change the “size=9” to “size=5”. we just want to each page show 5 books

```
useEffect(() => {
  const fetchBooks = async () => [
    const baseUrl: string = "http://localhost:8080/api/books";
    const url: string = `${baseUrl}?page=0&size=5`;
```

Continue to copy and paste this part

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

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

Fix the issue by importing the SpinnerLoding component.

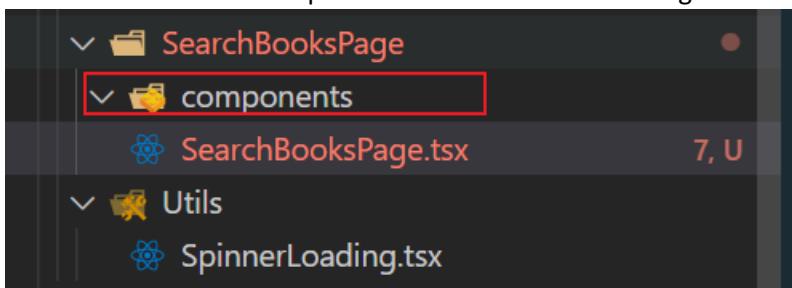
```
import { SpinnerLoading } from "../Utils/SpinnerLoading";
```

2. REACT-SEARCH BOOKS PAGE

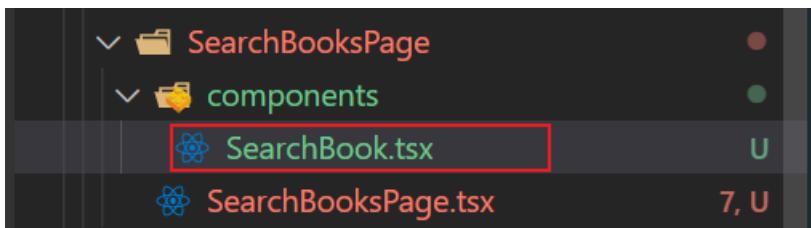
Write the code in the SearchBooksPage.tsx

```
return (
  <div>
    <div className="container">
      <div>
        <div className="row mt-5">
          <div className="col-6">
            <div className="d-flex">
              /* Search input box */
              <input
                className="form-control me-2"
                type="search"
                placeholder="Search"
                aria-label="Search"
              />
              <button className="btn btn-outline-success">Search</button>
            </div>
          </div>
          <div className="col-4">
            <div className="dropdown">
              <button
                className="btn btn-secondary dropdown-toggle"
                type="button"
                id="dropdownMenuButton1"
                data-bs-toggle="dropdown"
                aria-expanded="false"
              >
                Category
              </button>
              <ul
                className="dropdown-menu"
                aria-labelledby="dropdownMenuButton1"
              >
                <li>
                  <a className="dropdown-item" href="#">
                    All
                  </a>
                </li>
                <li>
                  <a className="dropdown-item" href="#">
                    Front End
                  </a>
                </li>
                <li>
                  <a className="dropdown-item" href="#">
                    Back End
                  </a>
                </li>
                <li>
                  <a className="dropdown-item" href="#">
                    Data
                  </a>
                </li>
                <li>
                  <a className="dropdown-item" href="#">
                    DevOps
                  </a>
                </li>
              </ul>
            </div>
          </div>
        <div className="mt-3">
          <h5>Number of results: (22)</h5>
        </div>
        <p>1 to 5 of 22 items:</p>
        {books.map((book) => (
          <SearchBook book={book} key={book.id} />
        ))}
      </div>
    </div>
  </div>
);
```

Create a new folder “components” in the SearchBooksPage folder.



Then create a new file in it. “SearchBook.tsx”



Write the code of SearchBook component.

```
src > layouts > SearchBooksPage > components > [?] SearchBook.tsx > [?] SearchBook
1  import BookModel from "../../models/BookModel";
2
3  export const SearchBook: React.FC<{ book: BookModel }> = (props) => {
4    return (
5      <div className="card mt-3 shadow p-3 mb-3 bg-body rounded">
6        <div className="row g-0">
7          <div className="col-md-2">
8            <div className="d-none d-lg-block">
9              {props.book.img ? (
10                <img src={props.book.img} width="123" height="196" alt="Book" />
11              ) : (
12                <img
13                  src={require("../../../../Images/BooksImages/book-luv2code-1000.png")}
14                  width="123"
15                  height="196"
16                  alt="Book"
17                />
18              )}
19            </div>
20          </div>
21          <div className="col-md-6">
22            <div className="card-body">
23              <h5 className="card-title">{props.book.author}</h5>
24              <h4>{props.book.title}</h4>
25              <p className="card-text">{props.book.description}</p>
26            </div>
27          </div>
28          <div className="col-md-4 d-flex justify-content-center align-items-center">
29            <a className="btn btn-md main-color text-white" href="#">
30              View Details
31            </a>
32          </div>
33        </div>
34      </div>
35    );
36  };

```

Go back to the SearchBooksPage.tsx, fix the issue by importing the SearchBook component.

```
1  ✓ import { useEffect, useState } from "react";
2  import BookModel from "../../models/BookModel";
3  import { SpinnerLoading } from "../Utils/SpinnerLoading";
4  import { SearchBook } from "./components/SearchBook";
```

Open the App.tsx, comment out the HomePage component

```
src > [?] App.tsx > [?] App
...
1  import React from "react";
2  import "./App.css";
3  import { Navbar } from "./layouts/NavbarAndFooter/Navbar";
4  import { Footer } from "./layouts/NavbarAndFooter/Footer";
5  import { HomePage } from "./layouts/HomePage/HomePage";
6
7  export const App = () => {
8    // create a bootstrap navbar
9    return (
10      <div>
11        <Navbar />
12        /* <HomePage /> */ []
13        <Footer />
14      </div>
15    );
16  };
17
```

Add the SearchBooksPage component.

```
export const App = () => {
  // create a bootstrap navbar
  return (
    <div>
      <Navbar />
      {/* <HomePage /> */}
      <SearchBooksPage /> You, 1
      <Footer />
    </div>
  );
};
```

Let's run the application.

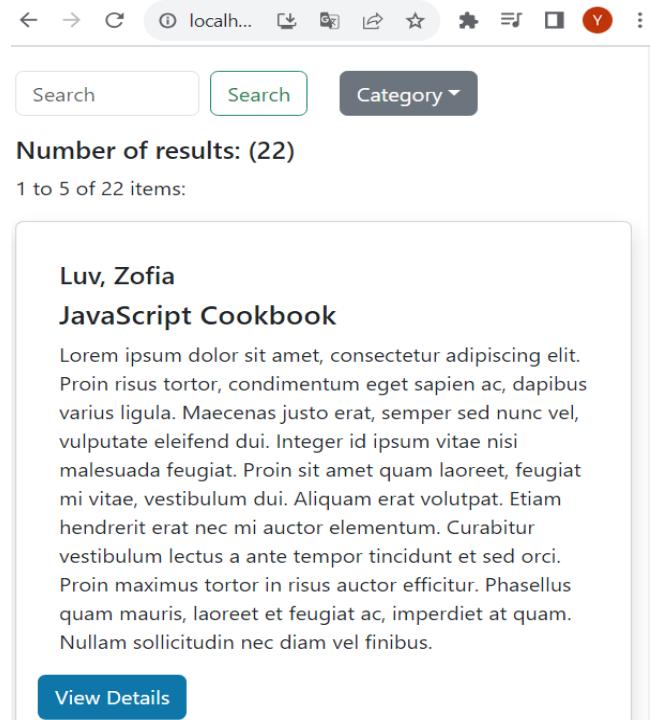
JAC Read Home Search Books

Search Category

Number of results: (22)
1 to 5 of 22 items:

| | |
|---|---|
|  | Luv, Zofia JavaScript Cookbook Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus. <input type="button" value="View Details"/> |
|  | Luv, Lena Become a Guru in JavaScript Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis eu, molestie at velit. Donec tincidunt tempus magna, quis facilisis libero elementum non. Sed velit lacus, laoreet sed augue fermentum, sagittis convallis metus. Sed nec est at massa venenatis aliquet. Donec pretium interdum fringilla. Sed ornare tellus enim, a tincidunt libero dictum vitae. Proin bibendum posuere dui. Donec sagittis neque massa, sed semper nulla vehicula at. |
|  | Luv, Jakub Exploring Vue.js Proin at urna faucibus, pretium mi in, dapibus metus. Interdum et malesuada fames |

But for the mobile mode, we noticed that the images can't be displayed.



The screenshot shows a web browser interface with a search bar, a 'Search' button, and a 'Category' dropdown. Below the header, it displays 'Number of results: (22)' and '1 to 5 of 22 items:'. Two book cards are visible:

- Luv, Zofia**
JavaScript Cookbook
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.
[View Details](#)
- Luv, Lena**
Become a Guru in JavaScript

Let me back to the SearchBook.tsx, we need to create a div for mobile mode.

```
export const SearchBook: React.FC<{ book: BookModel }> = (props) => {
  return (
    <div className="card mt-3 shadow p-3 mb-3 bg-body rounded">
      <div className="row g-0">
        <div className="col-md-2">
          <div className="d-none d-lg-block">
            {props.book.img ? (
              <img src={props.book.img} width="123" height="196" alt="Book" />
            ) : (
              <img
                src={require("../../../../Images/BooksImages/book-luv2code-1000.png")}
                width="123"
                height="196"
                alt="Book"
              />
            )}
          </div>
        </div>
        <div className="d-lg-none d-flex justify-content-center align-items-center">
          {props.book.img ? (
            <img src={props.book.img} width="123" height="196" alt="Book" />
          ) : (
            <img
              src={require("../../../../Images/BooksImages/book-luv2code-1000.png")}
              width="123"
              height="196"
              alt="Book"
            />
          )}
        </div>
      </div>
      <div className="col-md-6">
        <div className="card-body">
          <h5 className="card-title">{props.book.author}</h5>
          <h4>{props.book.title}</h4>
          <p className="card-text">{props.book.description}</p>
        </div>
      </div>
    </div>
  );
}
```

Let's rerun the application. Now we can see the images on the mobile mode.

A screenshot of a mobile browser interface. At the top, there is a navigation bar with icons for back, forward, refresh, and other browser functions. Below the navigation bar is a search bar with the placeholder "Search" and a green "Search" button. To the right of the search bar is a "Category" dropdown menu. The main content area displays the text "Number of results: (22)" and "1 to 5 of 22 items:". Below this, there is a card for a book titled "JavaScript Cookbook" by "Zofia Luv". The card features a thumbnail image of a red tree, the book title, and the author's name. The entire card has a light gray background.

Luv, Zofia

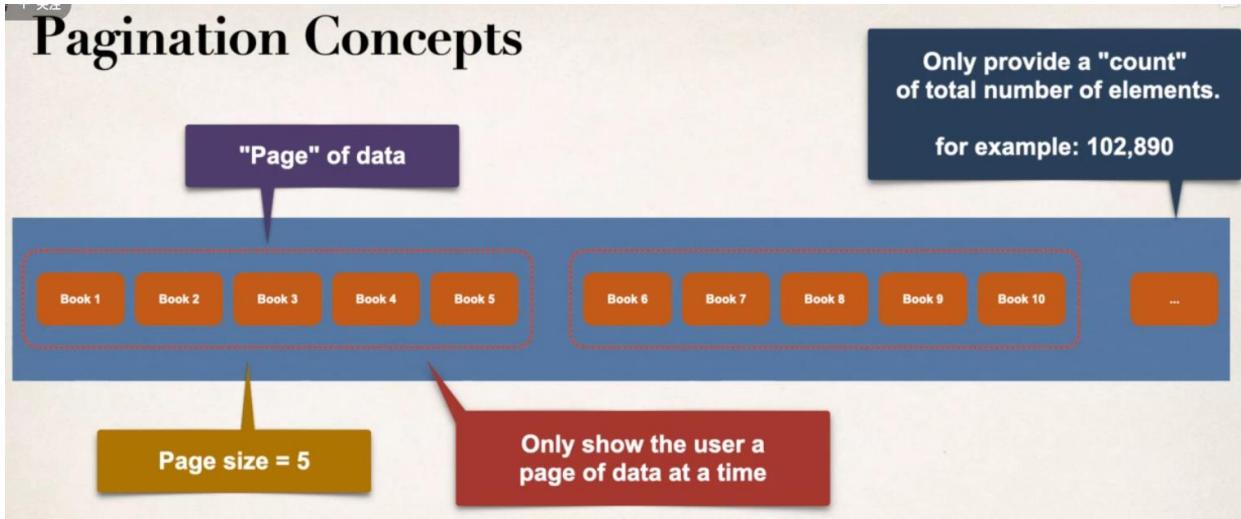
JavaScript Cookbook

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.

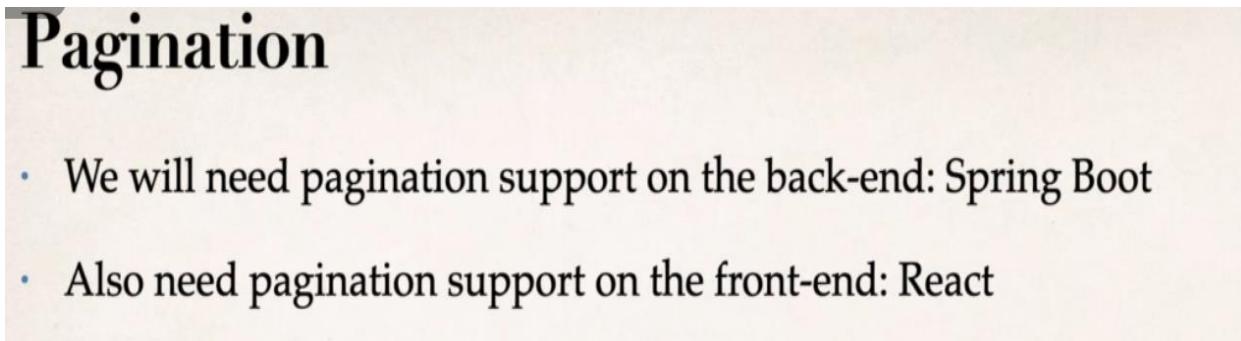
3. REACT-PAGINATION

Pagination

- Pagination is useful for handling large amounts of data
- Show the users a small subset of data: "page" of data
- The user can click links to view other pages



We need pagination support both on the front-end and back-end.



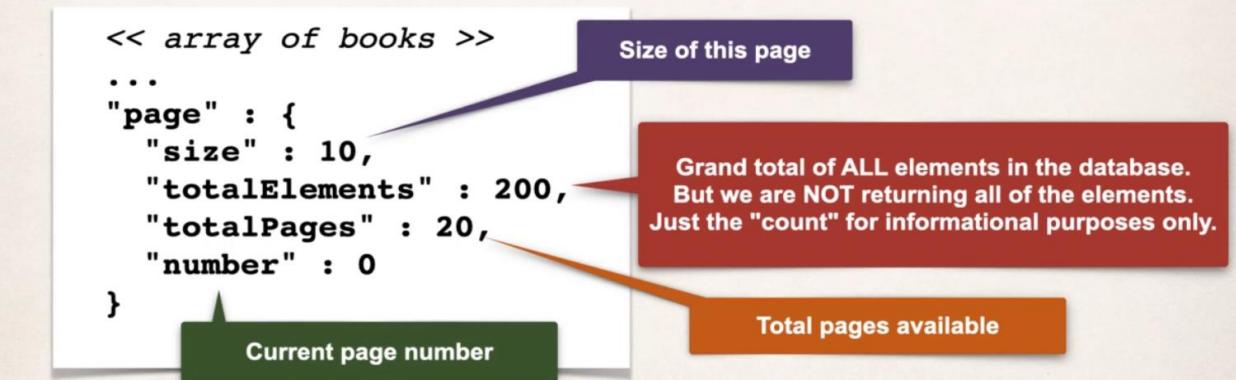
Spring Data REST - Parameters

- By default, Spring Data REST returns: 20 elements
- We can customize this by passing in parameters

| Parameter | Purpose |
|-------------------|---|
| <code>page</code> | The page number to access. 0-based ... defaults to 0. |
| <code>size</code> | The size of the page to return (items per page). Defaults to 20 |

Spring Data REST - Response Meta Data

- The response meta data has valuable information



Pagination with React:

Pagination with React

- Basic example of pagination component
- Create a new Pagination component that we can reuse within our application

Pagination.tsx

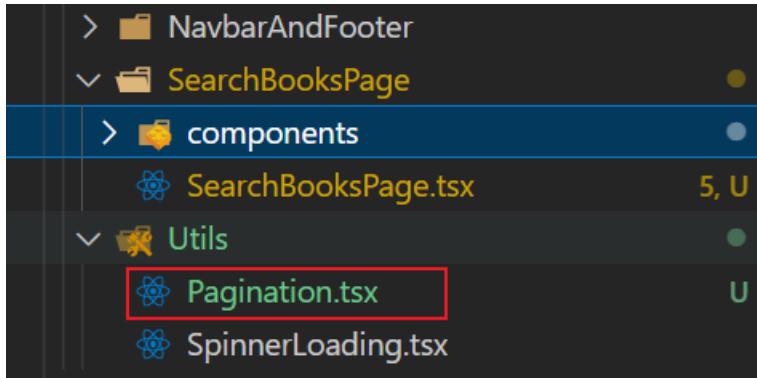
```
export const Pagination: React.FC<{}> = (props) => {
  < **Pagination Logic** />

  return (
    <nav>
      ..... * More Code * .....
    </nav>
  );
}
```

SearchBooksPage.tsx

```
<Pagination
  currentPage={currentPage}
  totalPages={totalPages}
  paginate={paginate}
/>
```

Create a new file “Pagination.tsx” in the Utils folder.



create export const Pagination() structure, passing currentPage, totalPages and paginate as its props. and add page number judgement.

```
src > layouts > Utils > ⚙️ Pagination.tsx > ⚙️ Pagination
1 < export const Pagination: React.FC<{
2   currentPage: number;
3   totalPages: number;
4   paginate: any;
5 > }> = (props) => {
6   const pageNumbers = [];
7   if (props.currentPage === 1) {
8     pageNumbers.push(props.currentPage);
9   }
10  if (props.totalPages >= props.currentPage + 1) {
11    pageNumbers.push(props.currentPage + 1);
12  }
13  if (props.totalPages >= props.currentPage + 2) {
14    pageNumbers.push(props.currentPage + 2);
15  }
16  else if (props.currentPage > 1) {
17    if (props.currentPage >= 3) {
18      pageNumbers.push(props.currentPage - 2);
19      pageNumbers.push(props.currentPage - 1);
20    }
21  }
22  pageNumbers.push(props.currentPage);
23  if (props.totalPages >= props.currentPage + 1) {
24    pageNumbers.push(props.currentPage + 1);
25  }
26  if (props.totalPages >= props.currentPage + 2) {
27    pageNumbers.push(props.currentPage + 2);
28  }
29}
```

Write the code in return part.

```
return (
  <nav aria-label="...">
    <ul className="pagination">
      <li className="page-item" onClick={() => props.paginate(1)}>
        <button className="page-link">First Page</button>
      </li>
      {pageNumbers.map((number) => (
        <li
          key={number}
          onClick={() => props.paginate(number)}
          className={
            "page-item" + (props.currentPage === number ? "active" : "")
          }
        >
          <button className="page-link">{number}</button>
        </li>
      ))}
      <li
        className="page-item"
        onClick={() => props.paginate(props.totalPages)}
      >
        <button className="page-link">Last Page</button>
      </li>
    </ul>
  </nav>
);
```

4. REACT-ADD PAGINATION TO SEARCH BOOKS PAGE COMPONENT

Go to SearchBooksPage.tsx, add useState of pagination.

```
6  export const SearchBooksPage = () => [
7    // Create a book array
8    const [books, setBooks] = useState<BookModel[]>([]);
9    // For judging whether in loading
10   const [isLoading, setIsLoading] = useState(true);
11   // API call failure scenario
12   const [httpError, setHttpError] = useState(null);
13
14   // pagination array
15   const [currentPage, setCurrentPage] = useState(1);
16   // each page show 5 books
17   const [booksPerPage] = useState(5);
18   // total amounts of books
19   const [totalAmountOfBooks, setTotalAmountOfBooks] = useState(0);
20   // total pages
21   const [totalPages, setTotalPages] = useState(0);
22
```

In the url, change the page number and books per page number to the dynamic value.

```
const fetchBooks = async () => {
  const baseUrl: string = "http://localhost:8080/api/books";
  const url: string = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;
```

And add these two statements.

```
useEffect(() => {
  const fetchBooks = async () => {
    const baseUrl: string = "http://localhost:8080/api/books";
    const url: string = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;
    // fetching the url data
    const response = await fetch(url);
    //failure scenario
    if (!response.ok) {
      throw new Error("Something went wrong!");
    }
    // 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;

    setTotalAmountOfBooks(responseJson.page.totalElements);
    setTotalPages(responseJson.page.totalPages);
```

And add these statement.

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

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

const indexOfLastBook: number = currentPage * booksPerPage;
const indexOfFirstBook: number = indexOfLastBook - booksPerPage;
let lastItem =
    booksPerPage * currentPage <= totalAmountOfBooks
    ? booksPerPage * currentPage
    : totalAmountOfBooks;

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

```
</div>
<div className="mt-3">
    <h5>Number of results: (22)</h5>
</div>
<p>1 to 5 of 22 items:</p>
{books.map((book) => (
    <SearchBook book={book} key={book.id} />
))
}
{totalPages > 1 && (
    <Pagination
        currentPage={currentPage}
        totalPages={totalPages}
        paginate={paginate}
    />
)
}
</div>
```

Let's run the application.

The screenshot shows a book search interface. At the top, there is a placeholder text area: "elit. Phasellus non viverra dolor. Pellentesque ligula mauris, congue quis neque quis, mollis scelerisque ligula. Pellentesque semper, erat commodo luctus mollis, nulla ipsum consectetur dolor, quis blandit massa sem fringilla libero. Maecenas eget mi nec est condimentum fermentum. Vivamus vehicula est sit amet ante gravida, eu finibus quam elementum. Proin egestas leo eu sagittis euismod." Below this, a book card is displayed for "Crash Course in Big Data" by Luv, Judy. The card includes a thumbnail image of the book cover, the author's name, the book title, a short description, and a "View Details" button. At the bottom of the card, there is a navigation bar with buttons for "First Page", "1", "2", "3", "4", and "Last Page".

We can see the pagination, but when we click the different page number, the book won't change.

We go back to the SearchBooksPage.tsx, add currentPage useState in the end of our array of state changes that are recall the use effect.

The screenshot shows a code editor with a dark theme. The code is part of a component that handles book fetching. It includes a function that sets books, marks loading as false, and then fetches books. It also handles errors and includes a lightbulb icon with a tooltip. A red box highlights the addition of the currentPage state variable to the useState array at the end of the code block.

```
    setBooks(loadedBooks);
    setIsLoading(false);
};

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

Then we refresh our page, if we click the different pages, we can find the books changed.

The screenshot shows the book search application after refreshing. It displays two separate book cards. The first card is for "Become a Guru in React.js" by Luv, Andrey, and the second card is for "Beginners Guide to Data Science" by Luv, Ajay. Both cards follow the same layout as the previous screenshot, with a thumbnail, author, title, description, and "View Details" button. Navigation buttons for "First Page", "1", "2", "3", "4", and "Last Page" are at the bottom of each card.

```

    },
    fetchBooks().catch((error: any) => {
      setIsLoading(false);
      setHttpError(error.message);
    });
    window.scrollTo(0, 0);
  }, [currentPage]);

```

Add this line of code, so that when we change the pages, it will always go to the top. And then change the static number to dynamic.

```

<div className="mt-3">
  <h5>Number of results: {[totalAmountOfBooks]}</h5>
</div>
<p>
  {[indexOfFirstBook + 1]} to {[lastItem]} of {[totalAmountOfBooks]} items:
</p>
{books.map((book) => (
  <SearchBook book={book} key={book.id} />

```

Then we refresh the page, when we change the page number, it will show the information dynamically.

localhost:3000

Number of results: (22)
11 to 15 of 22 items:

Luv, Anna
Advanced Techniques in Java
Nunc eget lorem ac neque tincidunt mollis. Fusce finibus laoreet nunc nec hendrerit. Curabitur eu placerat urna, sit amet pellentesque enim. Donec velit ligula, congue eu lobortis vel, interdum nec tellus. Nulla nisl ipsum, porta non egestas sed, vulputate quis nisi. Etiam pellentesque in velit non convallis. Nullam id risus quis augue posuere sodales vel maximus justo. Maecenas nec leo a nibh aliquet placerat nec sed massa. Duis sit amet nisi libero. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus non viverra dolor. Pellentesque ligula mauris, congue quis neque quis, mollis scelerisque ligula. Pellentesque semper, erat commodo luctus mollis, nulla ipsum consectetur dolor, quis blandit massa sem fringilla libero. Maecenas eget mi nec est condimentum fermentum. Vivamus vehicula est sit amet ante gravida, eu finibus quam elementum. Proin egestas leo eu sagittis euismod.

View Details

5. SPRINGBOOT-FIND BY TITLE

Let's go to BookRepository class, create findByTitleContaining interface function. Then import book class.

```
import com.fsd07.springbootlibrary.entity.Book;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.RequestParam;

/**
 * @Author: Yeming Hu
 * @Date: 9/3/2023
 * @Description: com.fsd07.springbootlibrary.dao
 * @Version: 1.0
 */
public interface BookRepository extends JpaRepository<Book, Long> {
    Page<Book> findByTitleContaining(@RequestParam("title") String title, Pageable pageable);
}
```

Rerun the application, then we open the postman.

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/books`. The response body is a JSON object representing a paginated list of books:

```
1,
  "self": {
    "ref": "http://localhost:8080/api/books"
  },
  "next": {
    "ref": "http://localhost:8080/api/books?page=1&size=20"
  },
  "last": {
    "ref": "http://localhost:8080/api/books?page=1&size=20"
  },
  "profile": {
    "ref": "http://localhost:8080/api/profile/books"
  },
  "search": {
    "ref": "http://localhost:8080/api/books/search"
  }
},
{
  "page": {
    "size": 20,
    "totalElements": 22,
    "totalPages": 2,
    "number": 0
  }
}
```

If we run the <http://localhost:8080/api/books>, use get method, send the request, we can get the information of 22 books.

If we input <http://localhost:8080/api/books/search>, choose “GET”, we will get:

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON 

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

If we add “findByTitleContaining?title=guru” to our search url string, we will get a “200 ok response”. All the books whose title contain “guru” returned. Totally we have found 3 books. REACT- FIND BY TITLE

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 70 ms Size: 1.74 MB Save Response  

Pretty Raw Preview Visualize JSON 

```
49
50   [
51     {
52       "_links": {
53         "self": {
54           "href": "http://localhost:8080/api/books/21"
55         },
56         "book": {
57           "href": "http://localhost:8080/api/books/21"
58         }
59       }
60     ],
61   "_links": {
62     "self": {
63       "href": "http://localhost:8080/api/books/search/findByTitleContaining?title=guru&page=0&size=20"
64     }
65   },
66   "page": {
67     "size": 20,
68     "totalElements": 3,
69     "totalPages": 1,
70     "number": 0
71 }
```

6. REACT- FIND BY TITLE

Go to SearchBooksPage.tsx. Create useState for search.

```
src > layouts > SearchBooksPage > SearchBooksPage.tsx > [e] SearchBooksPage
1 import { useEffect, useState } from "react";
2 import BookModel from "../../models/BookModel";
3 import { SpinnerLoading } from "../Utils/SpinnerLoading";
4 import { SearchBook } from "./components/SearchBook";
5 import { Pagination } from "../Utils/Pagination";
6
7 export const SearchBooksPage = () => [
8   // Create a book array
9   const [books, setBooks] = useState<BookModel[]>([]);
10  // For judging whether in loading
11  const [isLoading, setIsLoading] = useState(true);
12  // API call failure scenario
13  const [httpError, setHttpError] = useState(null);
14
15  // pagination array
16  const [currentPage, setCurrentPage] = useState(1);
17  // each page show 5 books
18  const [booksPerPage] = useState(5);
19  // total amounts of books
20  const [totalAmountOfBooks, setTotalAmountOfBooks] = useState(0);
21  // total pages
22  const [totalPages, setTotalPages] = useState(0);
23
24  const [search, setSearch] = useState("");
25  const [searchUrl, setSearchUrl] = useState("");
```

Get the url. If there is a search url string , then the final url should be the combination of base url and the search url.

```
useEffect(() => {
  const fetchBooks = async () => {
    const baseUrl: string = "http://localhost:8080/api/books";
    const url: string = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;

    // get the url
    if (searchUrl === "") {
      url = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;
    } else {
      url = baseUrl + searchUrl;
    }
}
```

We noticed that we get an error

```
co 'url' is constant. eslint(no-const-assign)
} & Cannot assign to 'url' because it is a constant. ts(2588)
const url: any
// if View Problem (Alt+F8) Quick Fix... (Ctrl+.)
url = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;
```

We change the “const” into “let” , and assign a empty string as its value.

```
useEffect(() => {
  const fetchBooks = async () => [
    const baseUrl: string = "http://localhost:8080/api/books";
    let url: string = "";

    // get the url
    if (searchUrl === "") {
      url = `${baseUrl}?page=${currentPage - 1}&size=${booksPerPage}`;
    } else {
      url = baseUrl + searchUrl;
    }
  ]
});
```

add searchUrl useState in the end of our array of state changes that are recall the use effect.

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

window.scrollTo(0, 0);
}, [currentPage, searchUrl]);
```

create a function which handle the searchUrl

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

const searchHandleChange = () => {
  if (search === "") {
    setSearchUrl("");
  } else {
    setSearchUrl(
      `/search/findByTitleContaining?title=${search}&page=0&size=${booksPerPage}`
    );
  }
};
```

Add a onChange listener to the value of input, so that we can get the search value, and add a onClick listener to search button, so each time the button is clicked, it will execute the search url handle function.

```
    /* Search input box */
    <input
      className="form-control me-2"
      type="search"
      placeholder="Search"
      aria-labelledby="Search"
      onChange={(e) => setSearch(e.target.value)}
    />
    <button
      className="btn btn-outline-success"
      onClick={() => searchHandleChange()}
    >
      Search
    </button>
  </div>
</div>
```

Let's run the application. If we type guru in the search box, then we can get the search result.

The screenshot shows a search interface with a blue header bar. Below it is a search bar containing the text "guru". To the right of the search bar are two buttons: "Search" and "Category ▾". The main content area displays search results for "guru". It shows two items:

- Luv, Lena**
Become a Guru in JavaScript
A small thumbnail image of a building with an arched entrance. The description below reads:

Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis eu, molestie at velit. Donec tincidunt tempus magna, quis facilisis libero elementum non. Sed velit lacus, laoreet sed augue fermentum, sagittis convallis metus. Sed nec est at massa venenatis aliquet. Donec pretium interdum fringilla. Sed ornare tellus enim, a tincidunt libero dictum vitae. Proin bibendum posuere dui. Donec sagittis neque massa, sed semper nulla vehicula at.

[View Details](#)
- Luv, Andrey**
Become a Guru in React.js
A small thumbnail image of a tree with autumn leaves. The description below reads:

Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis eu, molestie at velit. Donec tincidunt tempus magna, quis facilisis libero elementum non. Sed velit lacus, laoreet sed augue fermentum, sagittis convallis metus. Sed nec est at massa venenatis aliquet. Donec pretium interdum fringilla. Sed ornare tellus enim, a tincidunt libero dictum vitae. Proin bibendum posuere dui. Donec sagittis neque massa, sed semper nulla vehicula at.

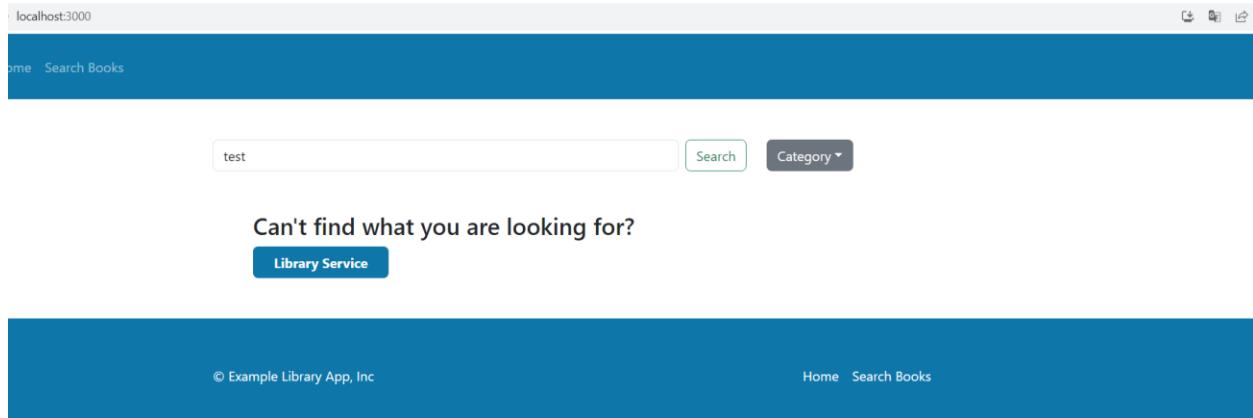
[View Details](#)

7. REACT-ALTERNATIVE TEXT FOR 0 BOOK

If the search result is 0 book found, it will show some message and a link to library service.

```
src > layouts > SearchBooksPage > SearchBooksPage.tsx > SearchBooksPage
176   [totalAmountOfBooks > 0 ? (
177     <>
178       <div className="mt-3">
179         <h5>Number of results: {totalAmountOfBooks}</h5>
180       </div>
181       <p>
182         {indexOfFirstBook + 1} to {lastItem} of {totalAmountOfBooks}{ " "
183           items:
184         }
185         {books.map((book) => (
186           <SearchBook book={book} key={book.id} />
187         )))
188       </>
189     ) : (
190       <div className="m-5">
191         <h3>Can't find what you are looking for?</h3>
192         <a
193           type="button"
194           className="btn main-color btn-md px-4 me-md-2 fw-bold text-white"
195           href="#"
196         >
197           Library Service
198         </a>
199       </div>
200     )
201   );
202   {totalPages > 1 && (
203     <Pagination
204       currentPage={currentPage}
205       totalPages={totalPages}
206       paginate={paginate}
207     >
208   )
209   </div>
210 </div>
211 );
212 };
```

We rerun the application, when we input “test” in the box, we will get 0 book returned, and get this page



8. SPRINGBOOT-FIND BY CATEGORY

Go to BookRepository class, create findByTitleCategory interface function.

```
5  public interface BookRepository extends JpaRepository<Book, Long> {
6      Page<Book> findByTitleContaining(@RequestParam("title") String title, Pageable pageable);
7
8      Page<Book> findByCategory(@RequestParam("category") String category, Pageable pageable);
9  }
```

Open Postman, input <http://localhost:8080/api/books/search>, choose “GET”, we will get:



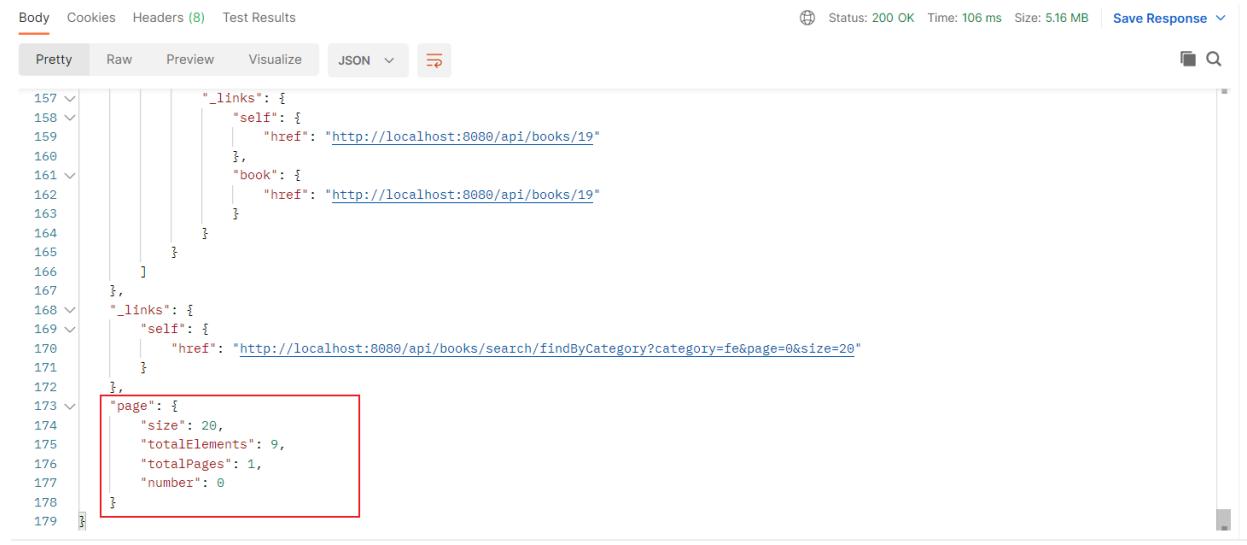
The screenshot shows the Postman interface with the following details:

- Body tab selected.
- Headers section shows 8 items.
- Test Results section is empty.
- Status bar: Status: 200 OK, Time: 20 ms, Size: 691 B, Save Response.
- JSON tab selected.
- Pretty printed JSON response:

```
1  "_links": {
2      "findByName": {
3          "href": "http://localhost:8080/api/books/search/findByName{?name,page,size,sort}",
4          "templated": true
5      },
6      "findByCategory": {
7          "href": "http://localhost:8080/api/books/search/findByCategory{?category,page,size,sort}",
8          "templated": true
9      }
10     "self": {
11         "href": "http://localhost:8080/api/books/search"
12     }
13 }
14 }
```

A red box highlights the "findByCategory" link in the JSON response.

If we add “findByCategory?category=fe” to our search url string, we will get a “200 ok response”. All the front end books returned. We can get 9 books.



The screenshot shows the Postman interface with the following details:

- Body tab selected.
- Headers section shows 8 items.
- Test Results section is empty.
- Status bar: Status: 200 OK, Time: 106 ms, Size: 5.16 MB, Save Response.
- JSON tab selected.
- Pretty printed JSON response (partial view):

```
157  [
158    {
159      "_links": {
160        "self": {
161            "href": "http://localhost:8080/api/books/19"
162        },
163        "book": {
164            "href": "http://localhost:8080/api/books/19"
165        }
166      }
167    },
168    {
169      "_links": {
170        "self": {
171            "href": "http://localhost:8080/api/books/search/findByCategory?category=fe&page=0&size=20"
172        }
173      },
174      "page": {
175        "size": 20,
176        "totalElements": 9,
177        " totalPages": 1,
178        "number": 0
179      }
180    }
181  ]
```

A red box highlights the "page" object in the JSON response.

9. REACT-FIND BY CATEGORY

Open SearchBooksPage.tsx, create useState for category selection.

```
src > layouts > SearchBooksPage > SearchBooksPage.tsx > SearchBooksPage
1  import { useEffect, useState } from "react";
2  import BookModel from "../../models/BookModel";
3  import { SpinnerLoading } from "../../Utils/SpinnerLoading";
4  import { SearchBook } from "./components/SearchBook";
5  import { Pagination } from "../../Utils/Pagination";
6
7  export const SearchBooksPage = () => {
8    // Create a book array
9    const [books, setBooks] = useState<BookModel[]>([]);
10   // For judging whether in loading
11   const [isLoading, setIsLoading] = useState(true);
12   // API call failure scenario
13   const [httpError, setHttpError] = useState(null);
14
15   // pagination array
16   const [currentPage, setCurrentPage] = useState(1);
17   // each page show 5 books
18   const [booksPerPage] = useState(5);
19   // total amounts of books
20   const [totalAmountOfBooks, setTotalAmountOfBooks] = useState(0);
21   // total pages
22   const [totalPages, setTotalPages] = useState(0);
23
24   // search by title, create two array for search
25   const [search, setSearch] = useState("");
26   const [searchUrl, setSearchUrl] = useState("");
27
28   // create useState for book category
29   const [categorySelection, setCategorySelection] = useState("Book Category");
```

create categoryField function to judge which category has been chose.

```
92
93  const searchHandleChange = () => {
94    if (search === "") {
95      setSearchUrl("");
96    } else {
97      setSearchUrl(
98        `/search/findByTitleContaining?title=${search}&page=0&size=${booksPerPage}`
99      );
10    }
11  };
12
13  const categoryField = (value: string) => {
14    if (
15      value.toLowerCase() === "fe" ||
16      value.toLowerCase() === "be" ||
17      value.toLowerCase() === "data" ||
18      value.toLowerCase() === "devops"
19    ) {
20      setCategorySelection(value);
21      setSearchUrl(
22        `/search/findByCategory?category=${value}&page=0&size=${booksPerPage}`
23      );
24    } else {
25      setCategorySelection("All");
26      setSearchUrl(`?page=0&size=${booksPerPage}`);
27    }
28  };
29
```

change the static word “category” into dynamical value.

```
152 |         <div className="col-4">
153 |             <div className="dropdown">
154 |                 <button
155 |                     className="btn btn-secondary dropdown-toggle"
156 |                     type="button"
157 |                     id="dropdownMenuButton1"
158 |                     data-bs-toggle="dropdown"
159 |                     aria-expanded="false"
160 |                 >
161 |                     {categorySelection}
162 |                 </button>
163 |                 <ul
164 |                     className="dropdown-menu"
165 |                     aria-labelledby="dropdownMenuButton1"
166 |                 >
```

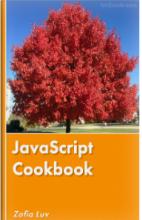
add onClick listener to each category dropdown item.

```
<li onClick={() => categoryField("All")}>
  <a className="dropdown-item" href="#">
    All
  </a>
</li>
<li onClick={() => categoryField("FE")}>
  <a className="dropdown-item" href="#">
    Front End
  </a>
</li>
<li onClick={() => categoryField("BE")}>
  <a className="dropdown-item" href="#">
    Back End
  </a>
</li>
<li onClick={() => categoryField("Data")}>
  <a className="dropdown-item" href="#">
    Data
  </a>
</li>
<li onClick={() => categoryField("Devops")}>
  <a className="dropdown-item" href="#">
    DevOps
  </a>
```

Let's run the application, If we select "fe" in dropdown menu, it will show us all 9 frontend books in our database.

Search FE ▾

Number of results: (9)
1 to 5 of 9 items:



Luv, Zofia
JavaScript Cookbook
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.

[View Details](#)

If we select "be" in dropdown menu, it will show us all 5 backend books in our database.

Search BE ▾

Number of results: (5)
1 to 5 of 5 items:



Luv, Fatma
Introduction to Spring Boot
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.

[View Details](#)

If we select "data" in dropdown menu, it will show us all 7 data related books in our database.

Search Data ▾

Number of results: (7)
1 to 5 of 7 items:



Luv, Alexander
Advanced Techniques in Big Data
Nunc eget lorem ac neque tincidunt mollis. Fusce finibus laoreet nunc nec hendrerit. Curabitur eu placerat urna, sit amet pellentesque enim. Donec velit ligula, congue eu lobortis vel, interdum nec tellus. Nulla nisl ipsum, porta non egestas sed, vulputate quis nisi. Etiam pellentesque in velit non convallis. Nullam id risus quis augue posuere sodales vel maximus justo. Maecenas nec leo a nibh aliquet placerat nec sed massa. Duis sit amet nisi libero. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus non viverra dolor. Pellentesque ligula mauris, congue quis neque quis, mollis scelerisque ligula. Pellentesque semper, erat commodo luctus mollis, nulla ipsum concomitantur dolor quis blandit massa cum fringilla libero. Maecenas eget mi.

[View Details](#)

If we select “devops” in dropdown menu, it will show us 1 related books in our database.

Search Devops ▾

Number of results: (1)

1 to 1 of 1 items:



Luv, Mariana
Exploring DevOps
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.

If we select “All” in dropdown menu, it will show us all 22 books in our database.

Search All ▾

Number of results: (22)

1 to 5 of 22 items:



Luv, Zofia
JavaScript Cookbook
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.



Luv, Lena