Bookshelf App
--code snippets

This assignment is called Bookshelf App. It is a really good exercise for a React JS beginner. React JS is a JavaScript library for building user interfaces. I finished it with the following steps.

1. Generate the main page and search page

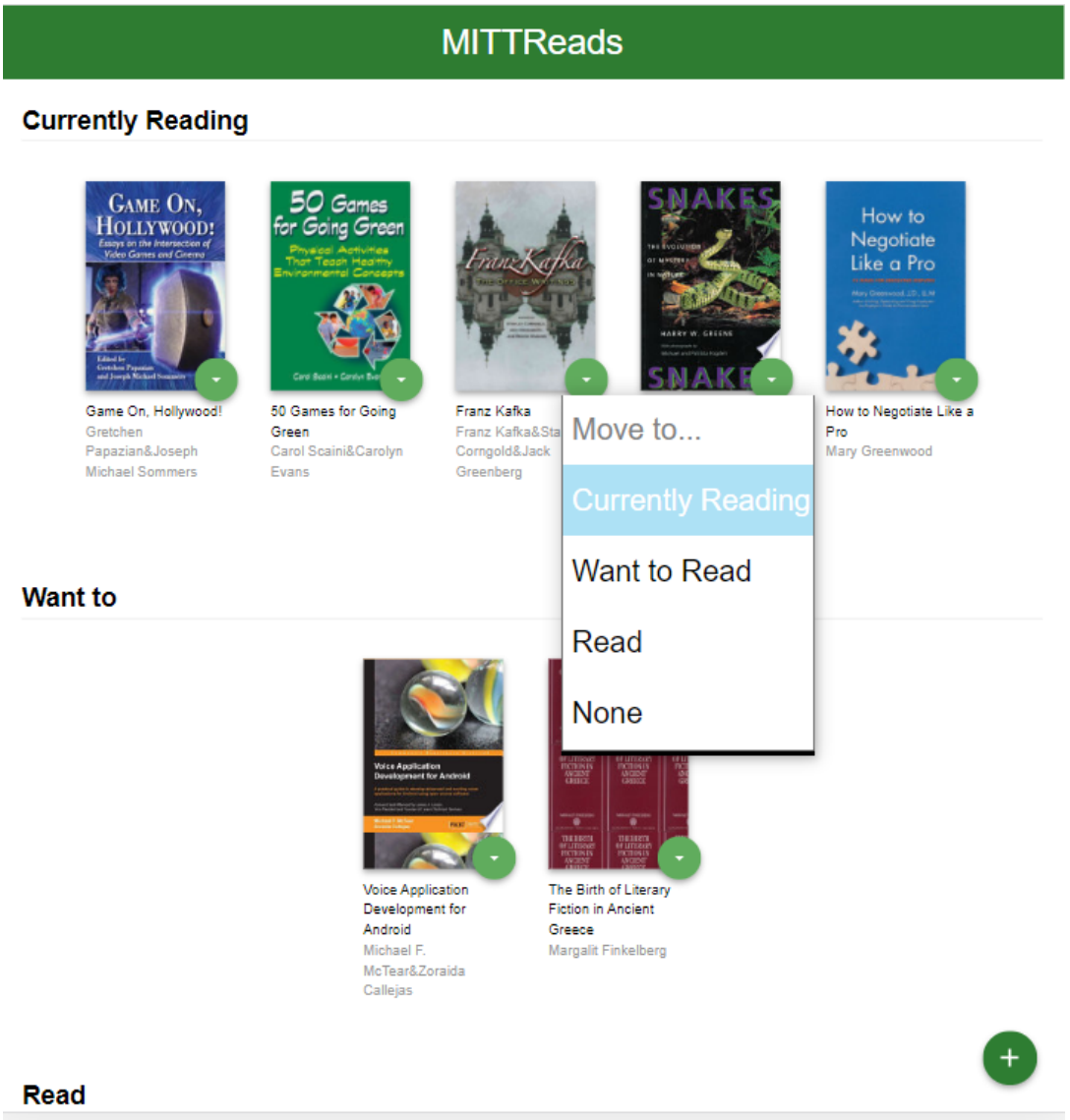

Figure 1. main page

The main page displays three bookshelves: *Currently, Reading*, *Want to* and *Read*. A button shows a dropdown menu at each book's right bottom corner to move this book to a different shelf.

If you want to jump to the search page, click the green “+” button at the bottom right of the page.

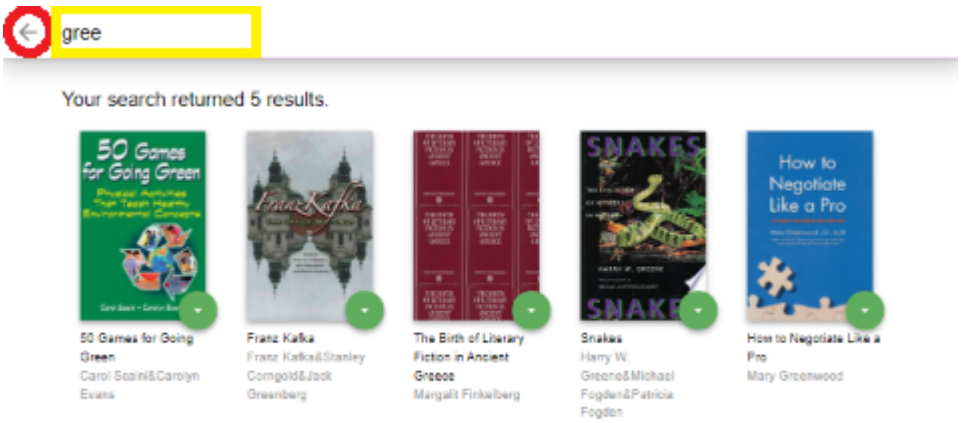

Figure 2. Search Page

On the search page, you can search for books by inputting the book’s name or author’s name in the yellow block.
By this time, each input letter can trigger an API request to the server, so the page will be refreshed very quickly.
If you want to return to the main page, click the left arrow button on the top left of this page.

./App.js:

```
function App() {
const [searchResults, setSearchResults] = useState([]);

useEffect(() => {
  getAllBooks()
    .then(data => {
      setSearchResults(data);
    });
}, [])

return (
    <Router>
      <div className="app">
        <Switch>
          <Route exact path="/">
            < BookShelfPage />
          </Route>

          <Route path="/search">
```

```
            <SearchPage />
          </Route>

        </Switch>
      </div>
    </Router>
  );
}
```

Figure 3. The main structure of the App function

The React App always starts from a single function. In this case, I chose hooks(`useState`, `useEffect`, Router, Switch, Route, and Link to establish this App.

`<Route exact path="/">` is setting the main page, and `<Route path="/search">` is setting the search page.

2. Compose two components for main page shelves and search page

In figure 3, two components were called: `BookShelfPage` and `SearchPage`. In this step, I established those components. BookShelfPage generates the main page, BookShelfPage calls BookOnShelf to put books onto different shelves where they belong.

SearchPage calls SearchBooks to require data from API and displays the results on this page.

3. Compose services API and utility function

./servers/BooksAPI.js

```
const baseUrl = 'http://localhost:5001/books';

export const getAllBooks = async () => {
  const response = await fetch(baseUrl);
  return await response.json();
};

export const changeShelf = async (book) => {
  book.completed = !book.completed;
  const response = await fetch(`${baseUrl}/${book.id}`, {
    method: 'PATCH',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
```

```
      shelf: 'none'
    }),
  });
  return response;
};
```

Figure 4. API for requiring data

In this case, I build a json-server for requiring data and update data.

./utility/Utility.js

```
export const joinAuthors = (book) => {
  if (book.authors) {
    return book.authors.join('&');
  }
  return '';
}

export const imageLink = (book) => {
  let imgLink = '';
  if (book.imageLinks) {
    if (book.imageLinks.smallThumbnail) {
      imgLink = book.imageLinks.smallThumbnail;
    } else if (book.imageLinks.thumbnail) {
      imgLink = book.imageLinks.thumbnail;
    }
  }
  return imgLink;
}
```

Figure 5. tools

Function `joinAuthors()` for joining several authors in a line joined by '&'; Function `imageLink()` check and chose an image link for the book's cover.

4. use debounce

```
onChange={debounce(handleSearch, 400)}
```

Figure 6. debounce

Then, when you input the searching key on the search page, it only triggers one API call if you type your key continuously.

In Summary, I learnt how to build a one-page app through React JS, and I had a lot of fun.

Assignment Instructions:
https://docs.google.com/document/d/1T8cvhfYDm7dGmm8AVJQEJrrywXvL6rzwB-PPU3dbuDA/edit?usp=sharing