CS 235 Assignment 2 Design Report

Lily Howan

Introduction

Spinebound

Spinebound is an online book catalog, which allows users to browse its database of books, view information about specific books and leave reviews with a message and a rating out of 5 stars.

New Feature

As a new feature on top of the assignment specifications, I chose to add a “My Books” feature. This acts as a virtual bookshelf where users can view their favourited books. The user’s favourites are stored as a list in each User object, as defined in the User class. I chose to add this feature as I feel it is genuinely useful and enables users to keep track of books they are interested in.

Book Page

To add a book to their favourites, the user must be logged in. While a book’s page can be viewed by any user, logged in or not, when the user clicks on the “favourite” button they will be redirected to the login page if not currently logged in (i.e. user_name is not present in session).

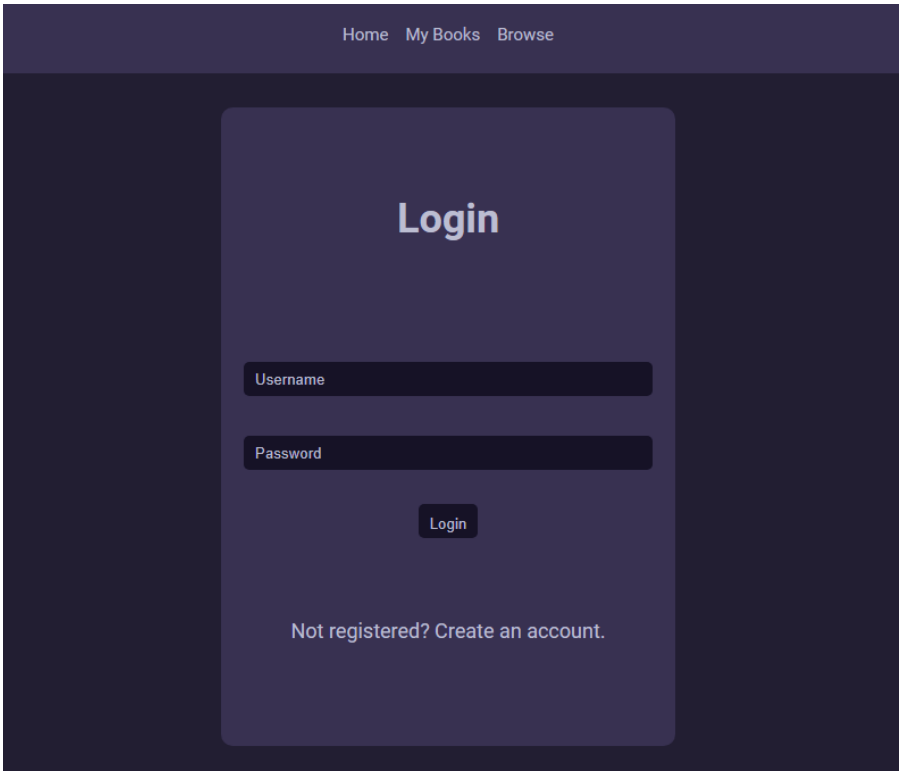

Figure 1: Spinebound login screen

Once logged in, the user can add the book to their favourites list by clicking on the red “favourite” button.

Figure 2: Book page display when book is not in user’s favourites

After doing this, the heart of the button will become a darker red, indicating that the book is in the user’s favourites.

Figure 3: Book page display when book is in user’s favourites

Clicking this button submits a form, which when validated in the Book blueprint will call the book/services.py method add_or_remove_book_from_favourites. This method gets the user and book from the repo and then checks if the book is in the user’s favourites. If the book is in the user’s favourites, it is removed from their favourites, otherwise it is added. This allows users to add and remove books from their favourites as desired. Once the user has added a book to their favourites, they can view it from the “My Books“ page.

My Books

In order to access the “My Books” page, the user must be logged in. Similar to commenting on an article in the Covid web app, I implemented this by redirecting the user to the login screen if they are not currently logged in (as seen above).

Once logged in, the user can access the “My Books” page via the link in the page header. This is functionally similar to the “Browse” page. My initial idea was to implement the “My Books” feature as a new blueprint. However, as “My Books” and “Browse” both exist for the purpose of displaying books and have many shared methods, I decided to implement “My Books” as a route inside of the Browse blueprint. I believe that this follows the Single Responsibility Principle, as it means that the Browse route has responsibility over a single part of the app’s function, i.e. displaying books.

Both the sort and search features from browse are present on the My Books page, but the title and message displayed when no books are available is different.

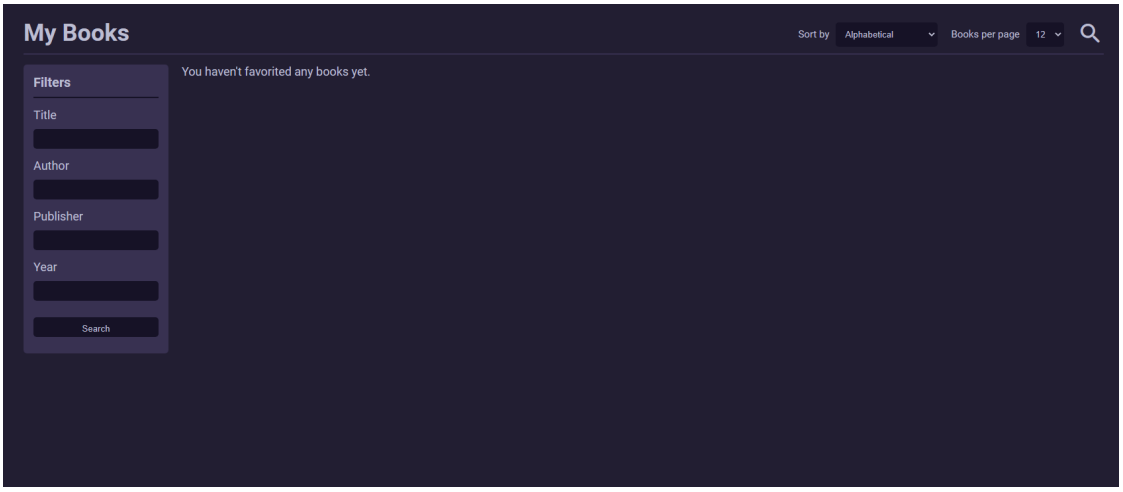

Figure 4: “My Books” page with no books present in user’s favourites

Once a user has added at least one book to their favourites, the page will start to display these books.

Figure 5: “My Books” page with one book in user’s favourites

The “My Books” page supports all features of the “Browse” page, i.e. sorting, selecting number of books to display per page, searching by title, author, publisher or year and the forward/back page buttons.

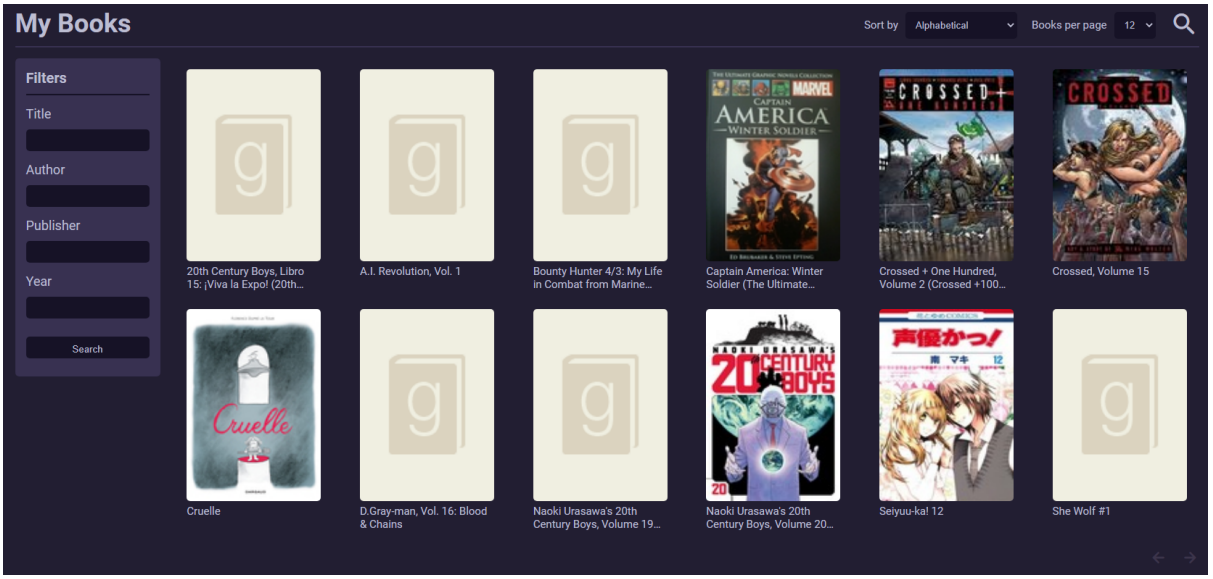

Figure 6: “My Books” page with 12 books in user’s favourites. As this is the default option for the number of books to display per page, the next page button is not active.

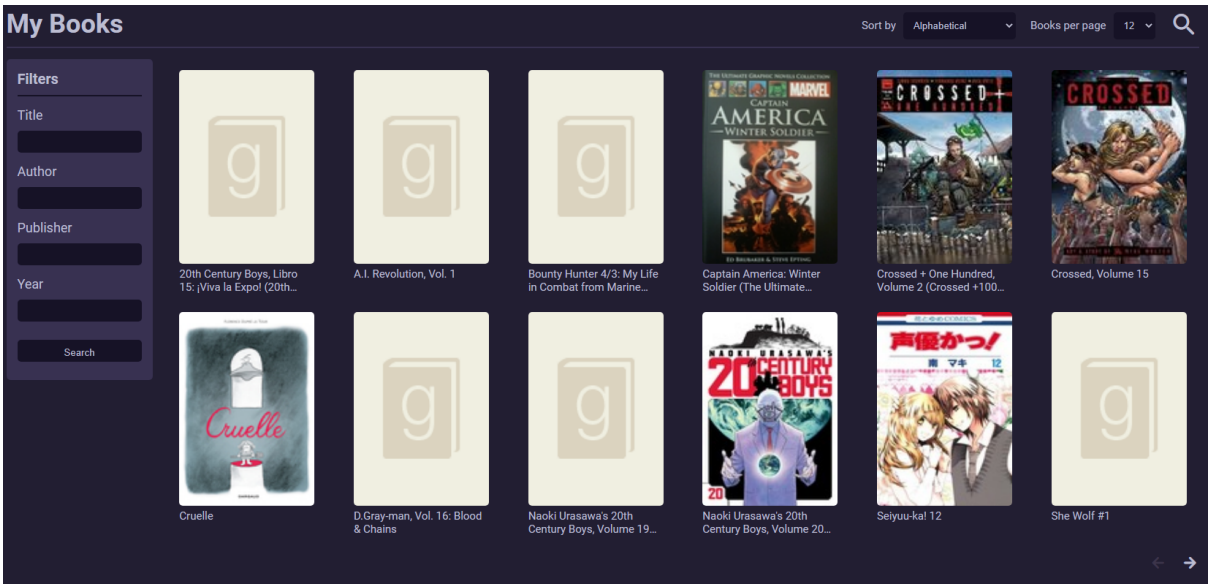

Figure 7: “My Books” page with >12 books in user’s favourites. There are more books to display than the number of books to display per page, so the next page button is active.

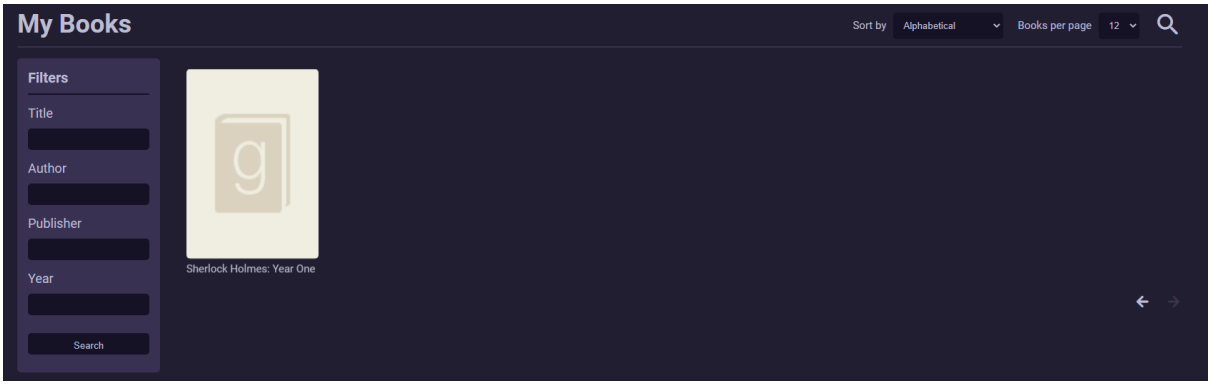

Figure 8: “My Books” page after clicking on the next page button. The previous page button is active.

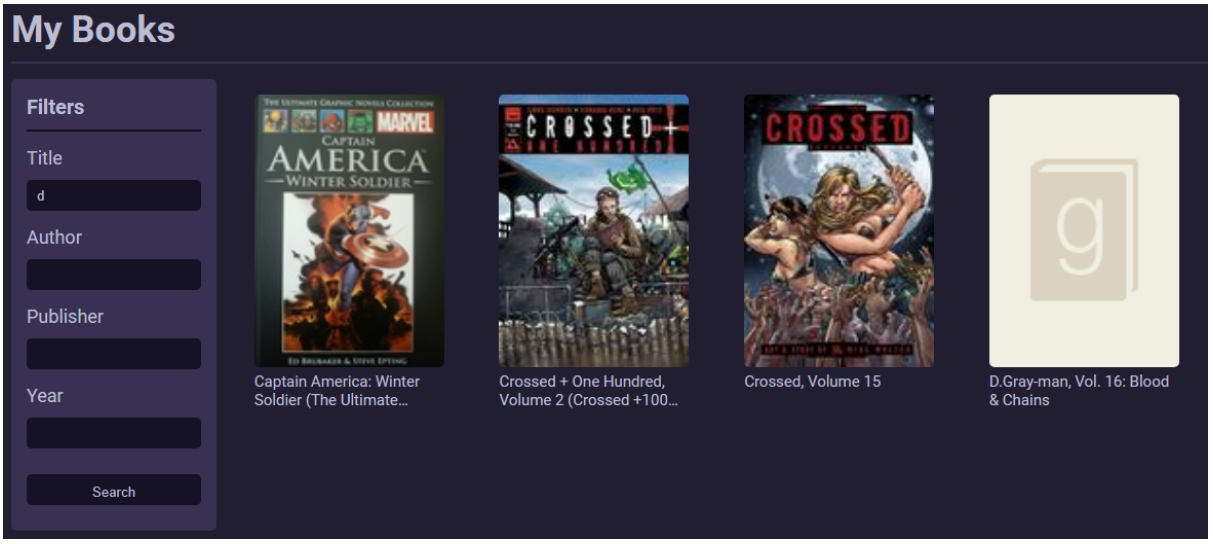

Figure 9: Searching by title includes all favourited books with titles that include the search string.

Next Steps

If I were to further extend this feature in the future, I would add the option to differentiate between a user’s favourite books, books that they have read and books they would like to read in the future. One method of implementing this could be through a book list dictionary, where each key is the name of a list. This would allow for predefined lists (e.g. “Books I’ve read”, “Books I want to read”) as well as the ability for users to create custom lists. While the ability to add to favourites on its own is useful, I believe that users would enjoy the option to customize their bookshelf as they desire.