BookWrym

Team 015-2

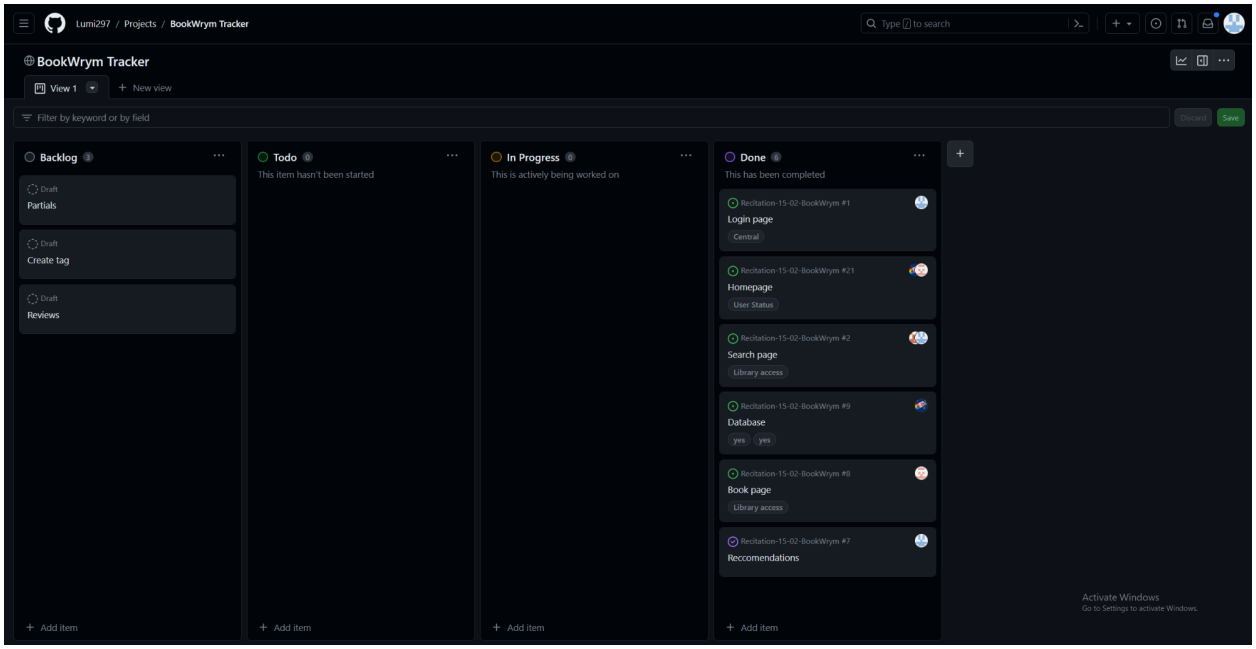
By: Nikko Gajowniczek, Brandon Lehman, Jeremy Schuberth, Bobby Vu and Ben Zabihi

Project Description:

Bookwyrm is a way of organizing and searching books from our extensive database powered by the Google Books API. This gives our users the ability to favorite books and view tags/genres attached to them and get recommendations. The basis of the project was to create a separate database of the Google Books API database in order to personalize one's own list of books. Our website begins with a login and register page. Here, the customer is able to create their private account which is stored into a table. Once they create an account or login, they will be redirected to their own homepage which shows lists of previously added books. The user can interact with this page as they are able to directly view the book's information by hovering and clicking on the card. The book information is stored onto a book page which will show the corresponding title, image, author(s), tags, genres, and description if provided within Google Books. The search function allows users to search for a variety of books based on their choice of title, author, or genre. All of these aspects combine to create a functioning website for all who enjoy reading.

Project tracker

Link: https://github.com/users/Lumi297/projects/3



VIDEO: Google Drive Video

VCS Link :

https://github.com/Lumi297/Recitation-15-02-BookWrym

Contributions:

Nikko Gajowniczek: I created a database module, which handled any functions requiring data retrieval. I integrated many existing functions that were in our routing page, and created many new functions to handle synchronizing results from our database and API to make the best use of both. I also helped maintain the main branch in github handling pull requests.

Brandon Lehman: Brandon was the one who thought up the idea for the group project. Brandon worked with Jeremy and Nikko mainly in the back end of the project. He dealt with the API routing and testing. He worked to keep the team organized through github management of the project board, and presented us wireframes and website framework. In particular, he provided a skeleton of the book page which allowed the front end to understand what was needed/wanted on it.

Jeremy Schuberth:

I worked on the search functionality. I worked with the api creating some basic error handling and used the google books api to make the baseline javascript search functionality that was later improved upon and I added the ability to search by author, title, or genre as well as the ability to get a certain number of search results. I also created the basic outline of our search ejs page which was improved upon by the front end team.

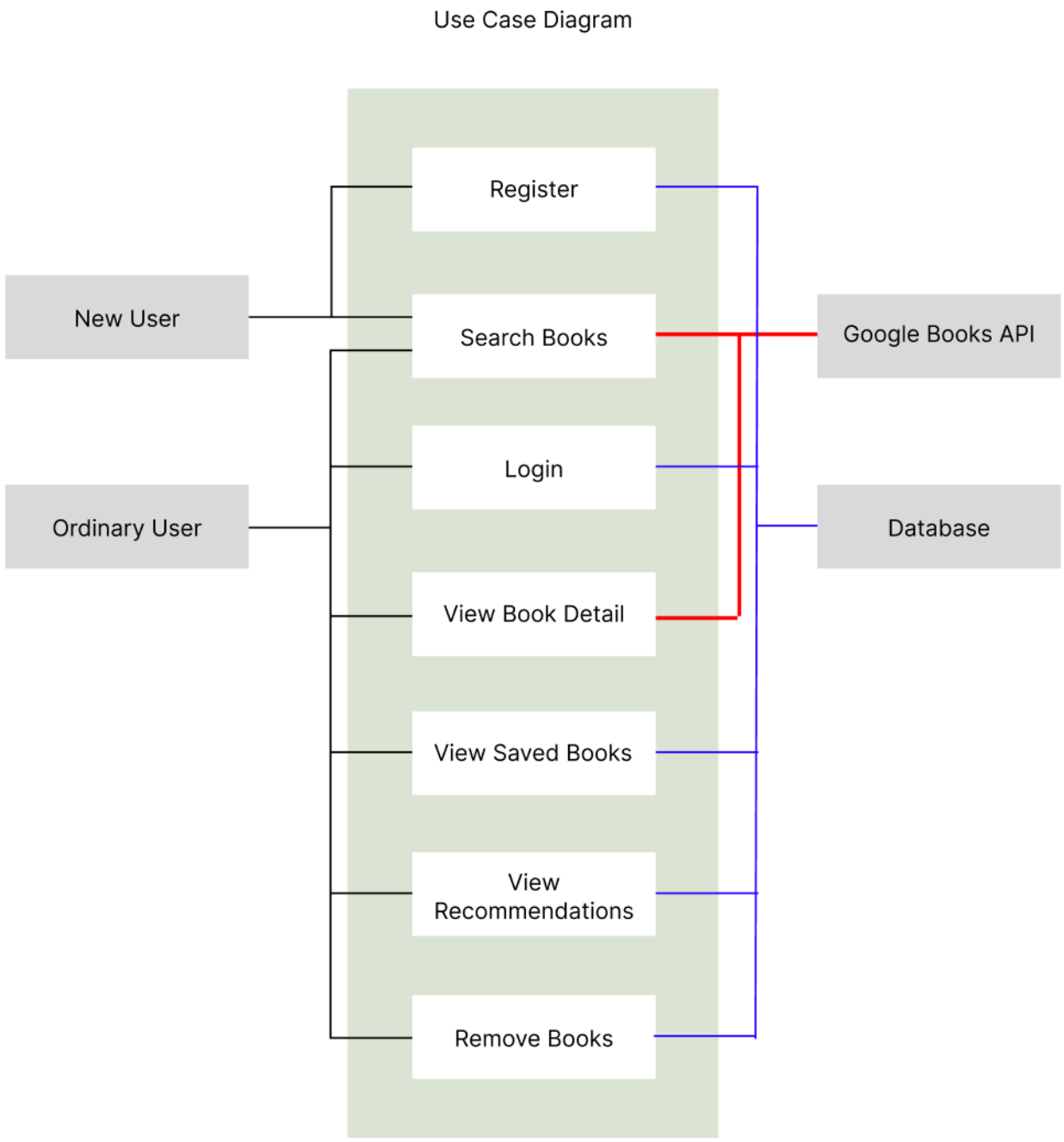
Bobby Vu :

I worked on a lot of front end issues. I created the login and register pages as well as the search and book page. I worked off of what Ben had created as well in order to integrate other parts. For example, he made the navbar and I added search functionality into it. I finalized the appearance of the website before presenting. I also worked on debugging some of the javascript files.

Ben Zabihi:

I worked mostly on the front end development of this project. I integrated the Navbar and footer for this project using bootstrap. I designed and coded the About Us and Account page as well. I also created the logout feature on the frontend and backend side as well as touched on front end appearance work in HTML and CSS for the search page and login/register pages.

Use Case Diagram:



Test Cases:

Feature 1. Login

Observation: The user is attempting to login. They are trying to login without having officially made an account. Their behavior is consistent with the use case. As a new user and not an ordinary user, they aren't allowed to login.

Result: As a result, they are unable to login to the website. An error is thrown because the username and password is unrecognized.

Feature 2. Register

Observation: The user is attempting to create an account. They would like to make an account to use for our website. This is a very simple case and options are limited to only signing in at this point.

Result: No error occurs as the username and password entered is taken as the correct format (string), and we have set up the html such that it is required to fill both entries.

Feature 3. Add Favorite

Observation: The user is adding a book to their favorites. This is because they want to save this book for later reference. This is normal behavior. They also attempted to add the same book again. Again, this is normal behavior as it could be due to misremembering or simply just because. Upon more observation, we decided to add in a delete button for removing that book for the user.

Result: Books were able to be added to the users book list as well as updating the user_to_books table within the database. Duplicate books were added to their page, so we used this result to make changes in application such that only unique books were added to their page.

Feature 4. Change Password

Observation: The user wants to change their password. They attempt by entering their previous password and entering a new one. They are possibly doing this to create a stronger password. We used this to make changes to the application. We made it so that once a password is changed, the user session is destroyed, and the user must login again with the new password.

Result: An error occurs when the previous password is added incorrectly. If the password is entered correctly, there is no error and the user can proceed to login again with their new password.

Deployment:

Our website is currently down as we've ran out of credits for hosting out website. As of now, our website can be accessed and deployed through Docker Compose and local host. Open our repository with any IDE, get to correct directory using the line:

```
cd Project/project_code
```

then run the line:

```
docker compose up
```

and go to localhost:3000/ in a browser(preferably FireFox).