EasyReads

By: Mreedul Gupta, Nico Tsiridis, Jiahao Xu, Mitchell Rheins, and Quinn Nicodemus

Project Description:

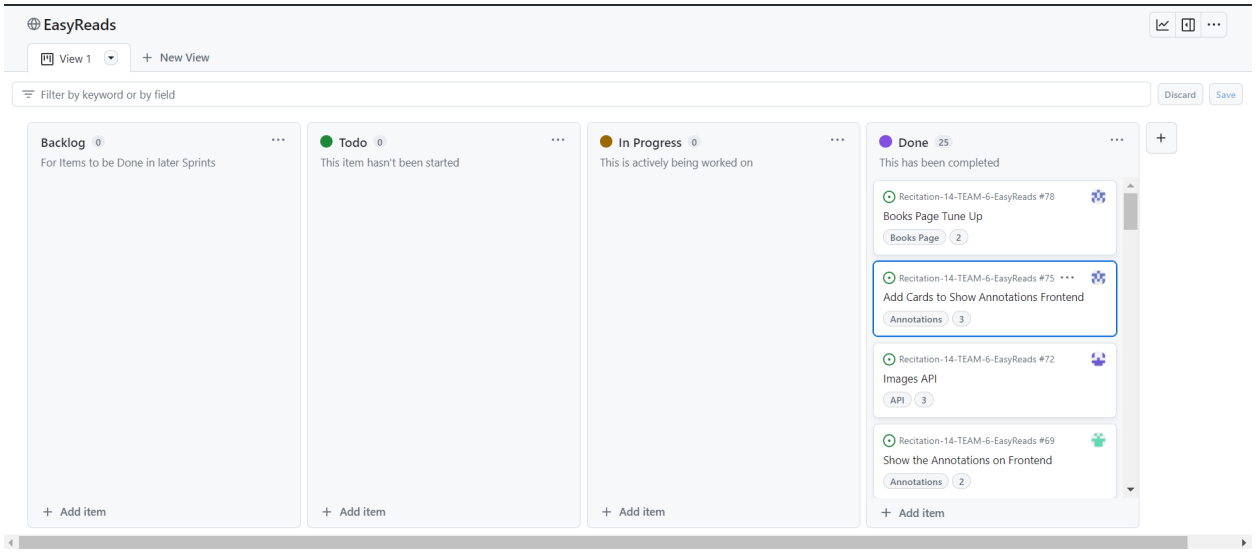
EasyReads will provide a catalog of literary works such as books of all genres and class notes where users can upload their own annotations for a specific book and their own notes for each class. In a world of continuous advancements in technology, EasyReads aims to consolidate all literary ideas in one place to facilitate continuous innovation. Currently as it is designed, EasyReads is very scalable and is capable of adding more texts such as articles, scholarly papers, and many more!

Upon registering on our website, users will be sent to a page that explains all of the different features of the app and how to fully maximize each! If they choose to explore our books page they will be met with the option to view our collection of over 70,000 books. In addition, EasyReads supports world-wide partnerships with books of many different languages ranging from Russian to French. Upon viewing any of these books, these users will have the option to create annotations, which will be their own notes for the specified texts. Furthermore, users will also have the option to publish their own class notes to the EasyReads community to provide academic help to whomever needs.

Project Tracker – Github Project Board:

Link: https://github.com/users/NanoNewt/projects/2/views/1

Pictures:



Video Demo of Project:

[EasyReads Demo Video]

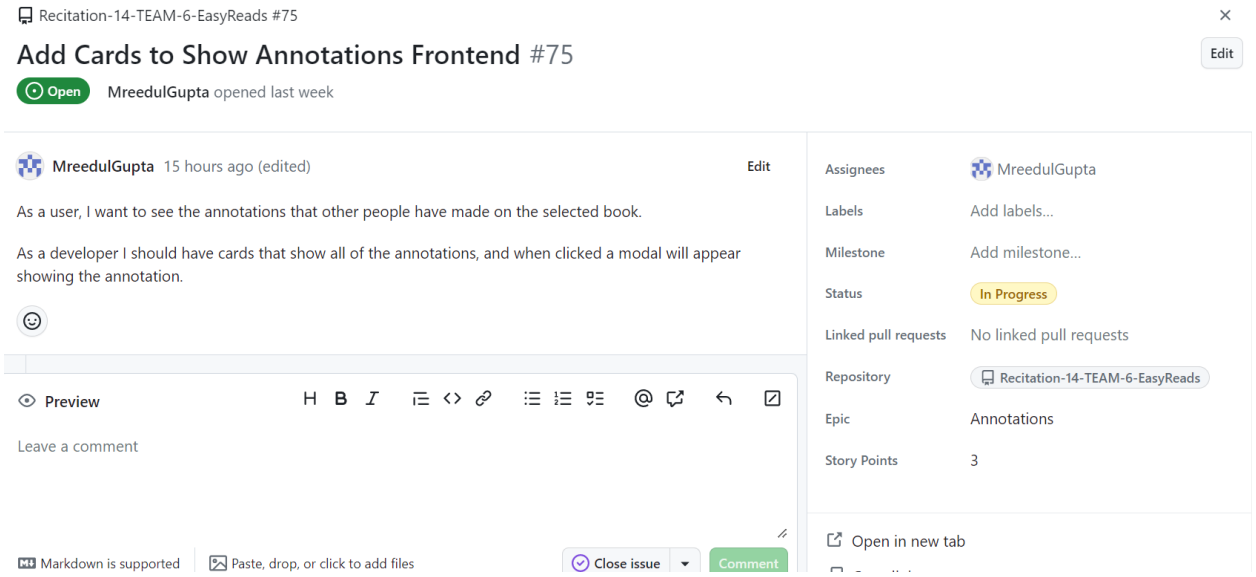
VCS:

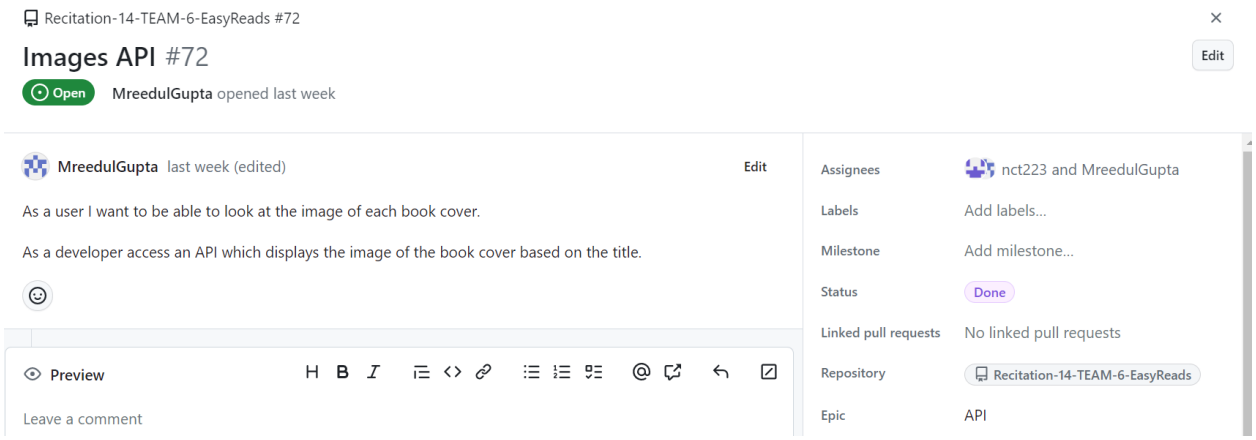
Link: https://github.com/NanoNewt/Recitation-14-TEAM-6-EasyReads

Contributions:

Mreedul:

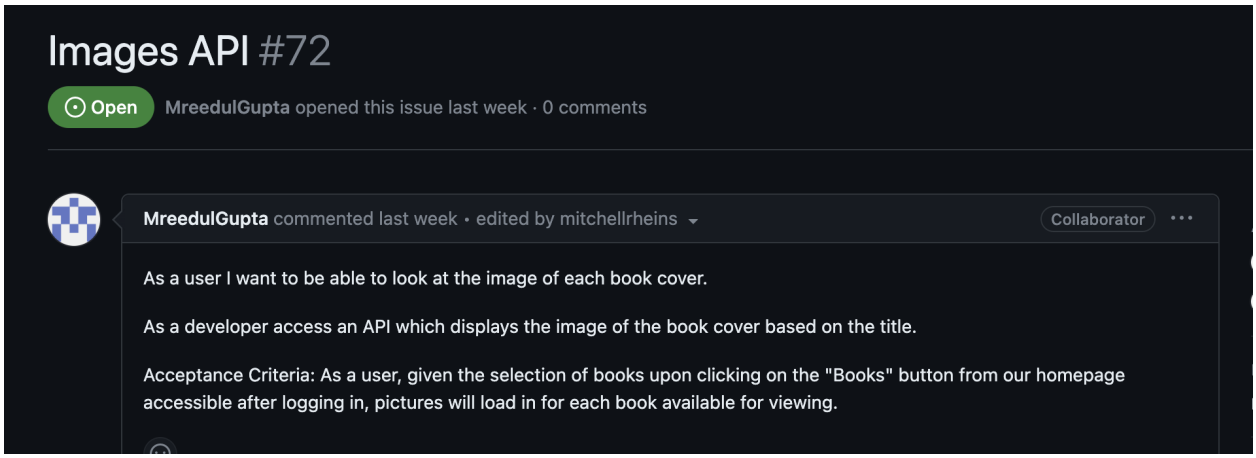
I worked on a little bit of everything with the main thing being frontend development. I developed both the home page and the books page. The home page tells everyone everything about EasyReads and the books page makes calls to the API and displays the information regarding each book. I also implemented both the search feature and the bookmarks feature which allows users to search for any book and allows them to bookmark their favorite books. Along with this, I also worked on the backend and SQL implementation required to make these features work. Below are screenshots of the project board of *a few* of the other tasks I worked on:

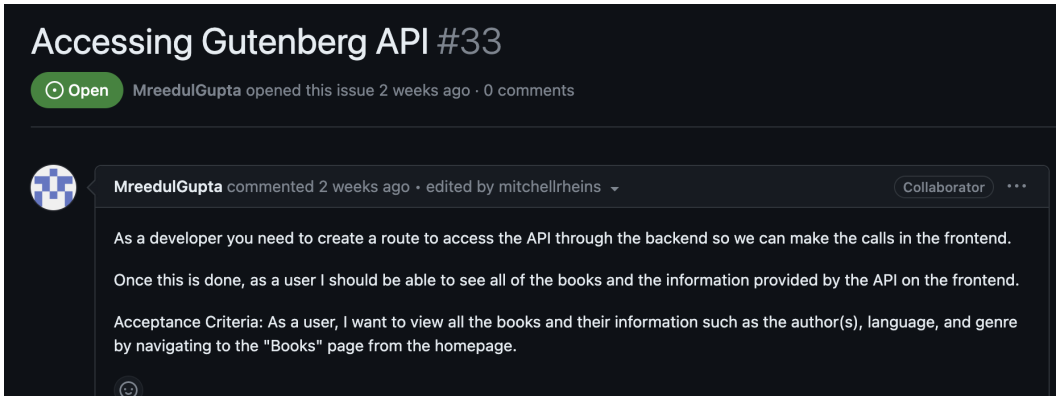




Nico:

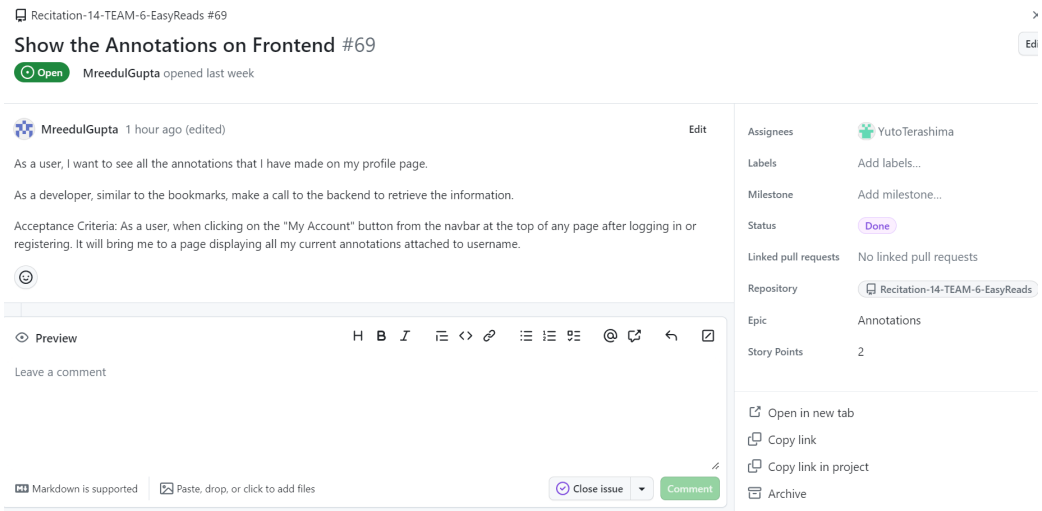
My main focus throughout this project has been implementing both API's. The Gutenberg API was used to fetch book data and display it on our website in the form of a list of books as well as the ability to click on a book and view its contents. I wrote the complete index.js routes associated with these API calls and implemented the respective embedded javascript in both the books (book search) and single book pages. I then implemented the routes associated with fetching book cover information. However, not all book covers were fetchable with the openlibrary.org API. There was however a last minute optimization made by Quinn that allowed for us not to use the openlibrary API and instead use an extension of the aforementioned Gutenberg API to fetch book covers. Finally, I worked on cleaning up the UI component. One commit was done for me by Mitchell as shown below.

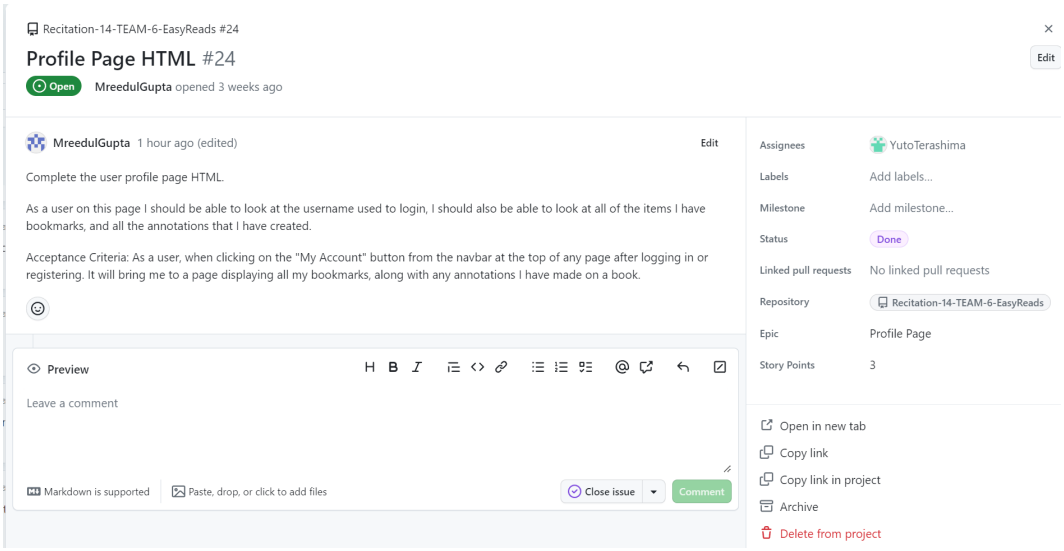




Jiahao:

My primary role was to construct the entire database, comprising tables for users, books, and other relevant data. Additionally, I developed the Profile page, complete with its corresponding API. This page offers two functions: the first involves a loop that retrieves and displays all of the user's favorite books and annotations on the page. The second function enables the user to delete any books or annotations in their collection. I also managed the book marks feature, utilizing the function and API to save the necessary data to the appropriate database. Finally, I made necessary adjustments to Quinn's API for storing comments and annotations so that they could be properly saved to the appropriate table. I have included several screenshots of the project board showcasing some of the other tasks that I worked on:





Mitchell:

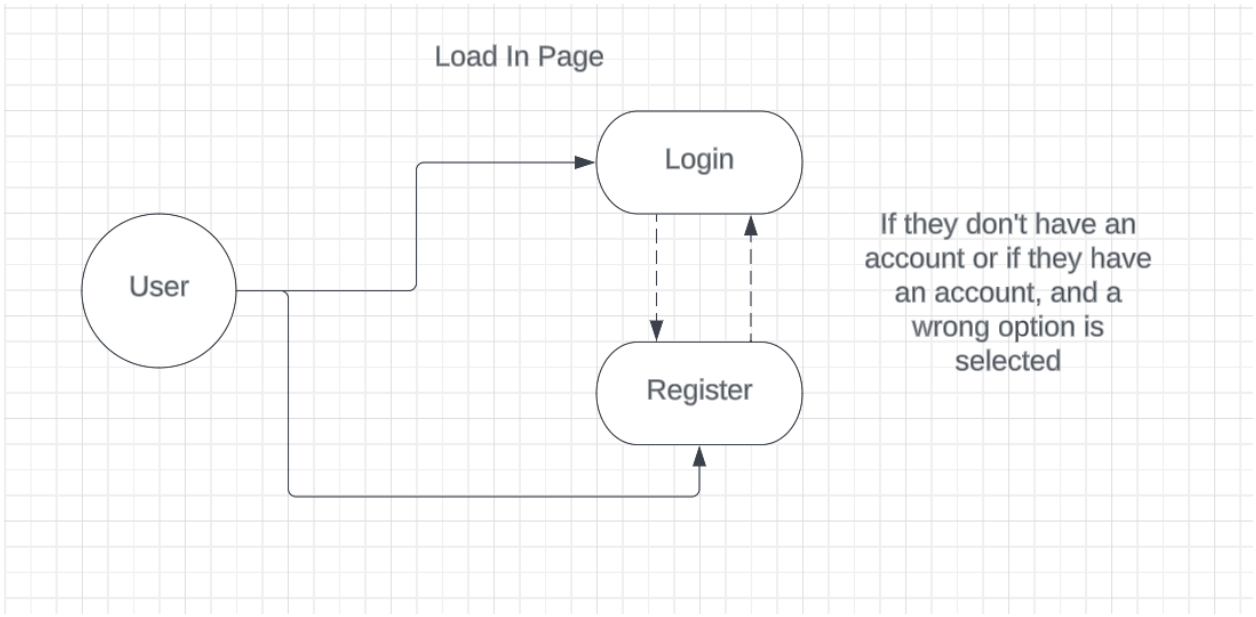
I played a pivotal role in initiating our project by designing an engaging and visually appealing splash page that effectively welcomes both new and returning users. In addition to this, I spearheaded the implementation of the login and registration features on the frontend, ensuring a seamless user experience.

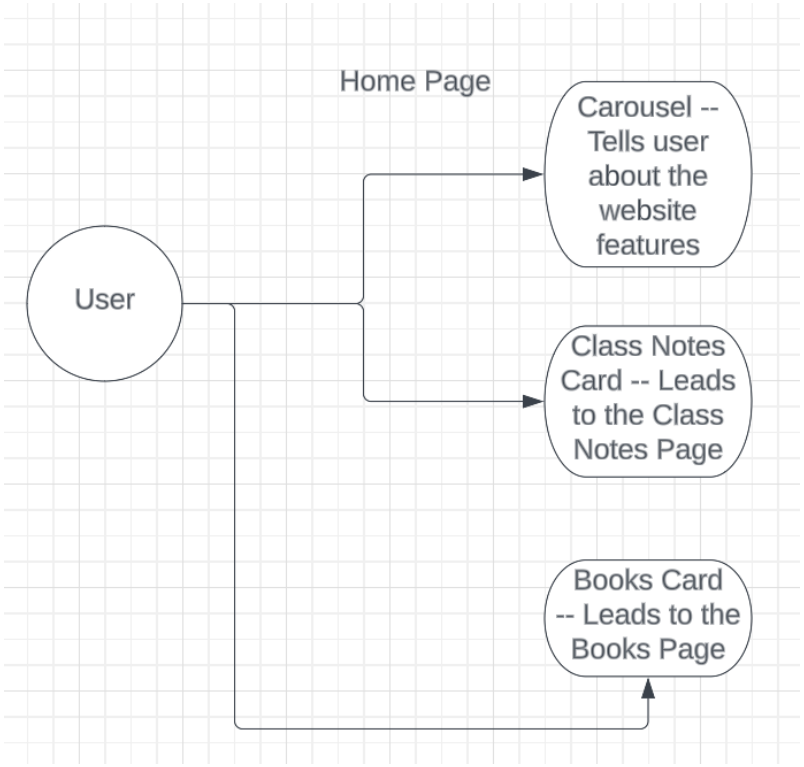
As one of our project's main features was the class notes platform, I was responsible for the end-to-end development of the class notes section. This entailed comprehensive involvement in frontend, backend, and database implementation. My primary objective was to create an intuitive and user-friendly system that allows users to effortlessly store, access, and manage their class notes. By prioritizing reliability and ease of use, I aimed to instill confidence in users that their notes would be consistently available whenever needed.

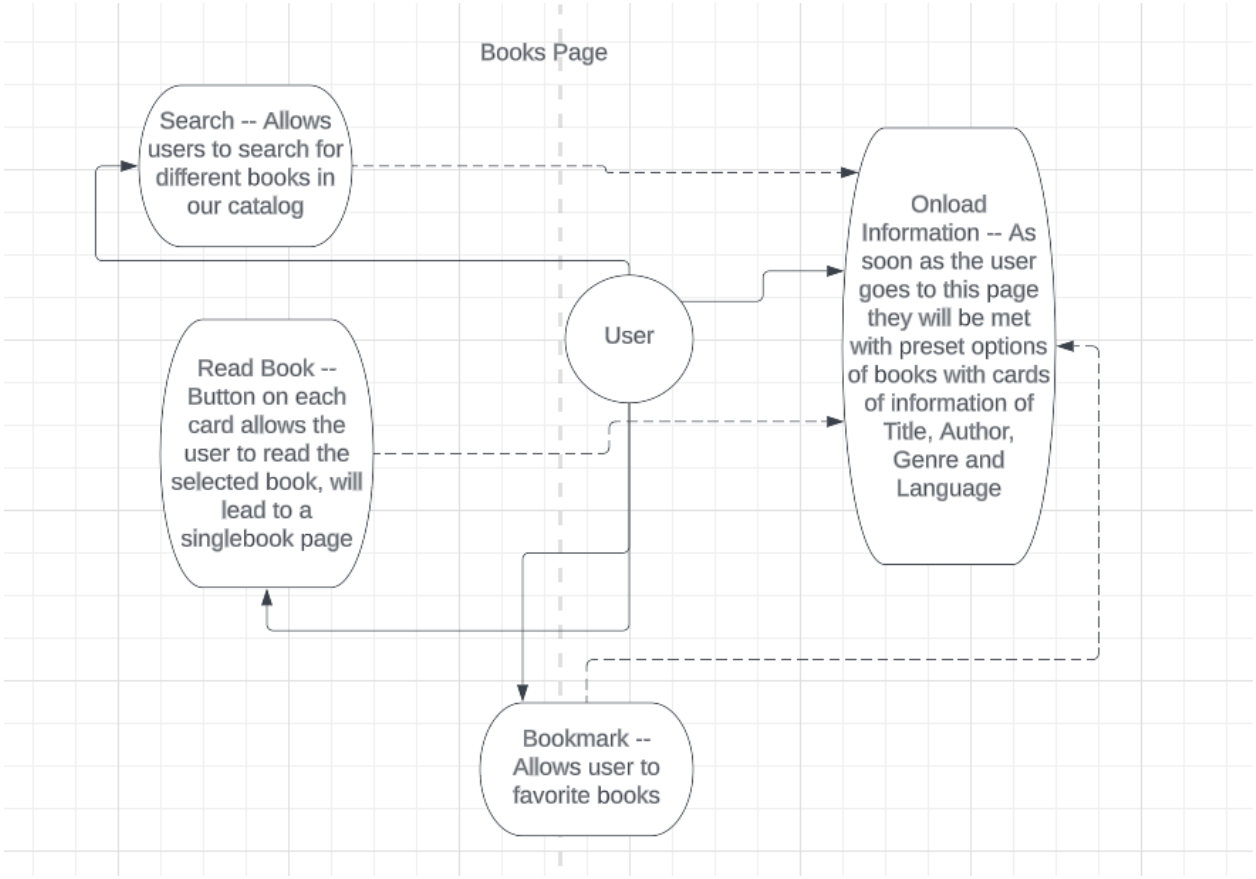
Quinn:

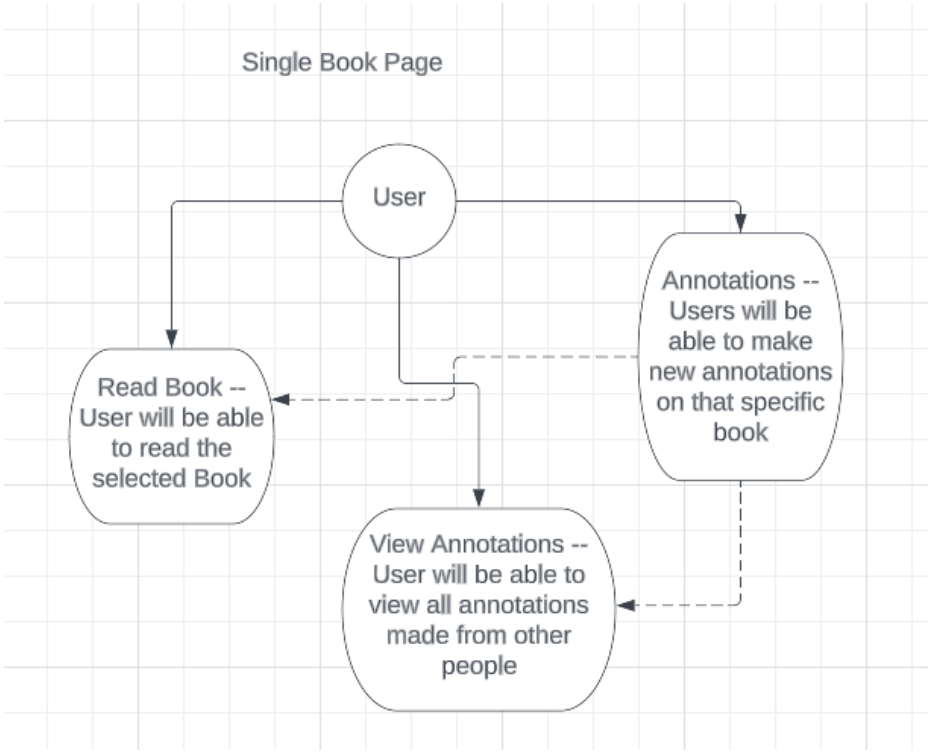
I added some stuff from the project template, and scribed one week's team meeting and a TA meeting, but most of my efforts were in the annotation page (which eventually turned into the singlebook page). I made significant modifications to the annotation page's html/ejs. In addition to, creating many front-end javascript functions and many API routes in order to facilitate the creation, storage, and retrieval of annotations and comments on annotations. I also merged the annotation page and single book page, and made it so that when we query Project Gutenberg for a new book it is stored in our database in order to reduce the number of external API calls we need to make.

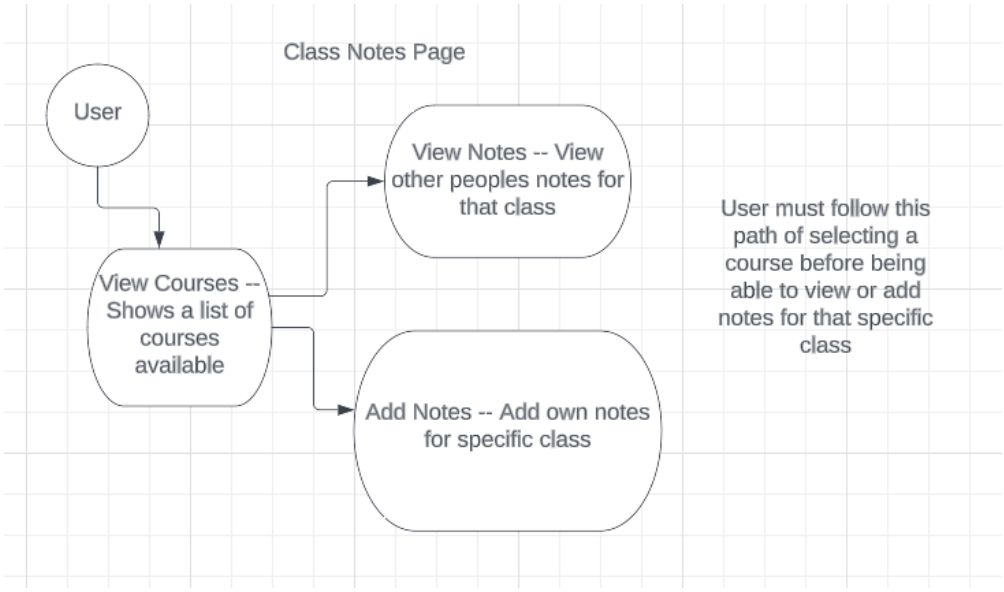
Use Case Diagram(s):

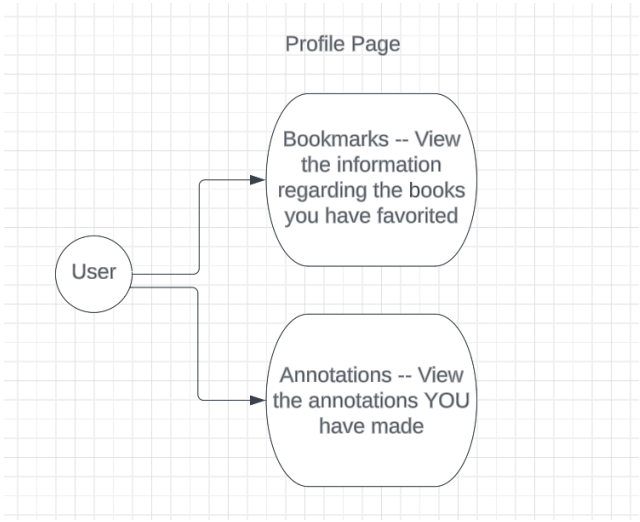

Load In Page
Login
User
Register
If they don't have an account or if they have an account, and a wrong option is selected


Home Page
Carousel -- Tells user about the website features
User
Class Notes Card -- Leads to the Class Notes Page
Books Card -- Leads to the Books Page


Books Page
Search -- Allows users to search for different books in our catalog
Onload Information -- As soon as the user goes to this page they will be met with preset options of books with cards of information of Title, Author, Genre and Language
User
Read Book -- Button on each card allows the user to read the selected book, will lead to a singlebook page
Bookmark -- Allows user to favorite books


Single Book Page
User
Annotations -- Users will be able to make new annotations on that specific book
Read Book -- User will be able to read the selected Book
View Annotations -- User will be able to view all annotations made from other people





Test Results with Adithya Bhaskara:

Use Case 1: Testing the books page functionality, if the user is fine with the load time of accessing the API and all the information that is retrieved. This information includes Book Cover image, title, author, genre, and language.

User Thoughts:

The first thing the user did was click on a book to read, they commented on the fast loading time for some of the smaller books such as Romeo and Juliet, but some of the longer books took a long time to load. The user liked looking at which author wrote each book, for example was surprised that Shakespeare wrote some books. Their behavior was consistent with the use case:

he looked at all the information on the cards before making a decision. One thing we changed is that for the images that don't load, we added a backup image.

Use Case 2: Testing the bookmarks and search functionality of the books page. This will include if the user can successfully find a book of their choice (assuming the API has the book) and will see if the user can bookmark the books of their choice. The user will also be required to remove the bookmark by going to the profile page.

User Thoughts:

The user first searched for Euclid's book, was surprised that the API had access to it. The user also bookmarked Euclid's book, and accidentally bookmarked another book and went to the profile page to remove it. The behavior was consistent with the use case as this was how the application was designed. The change we made was that when the user goes back to the books page the bookmark disappears, which was fixed in our application.

Use Case 3: Testing the entire functionality of the class notes page. Can the user use the search to find their appropriate course. Upon clicking the course see if the user can find the note they are looking for (if it exists). Also see if the user is able to add a class note to their desired course.

User Thoughts:

The user immediately opened the Algorithms course as they are taking the course right now, and they immediately added a note for that class. They also looked at the preset notes in the CSCI 2400 section. Also searched for many different courses, and liked how it was a dynamic search. The behavior was consistent with the use case, but instead of using alerts to insert a note we are now using modals.

Use Case 4: Testing the singlebooks/annotations page. Can the user read through the entire book and navigate through the different pages. Can the user add annotations to any specific part they wish, and are they able to view the annotations of others on the bottom of the page?

User Thoughts:

Commented on how cool it was to be able to read the entire book. Tested what would happen if he went to an out of bounds page, the error handling worked. Also added annotations to the book, and made sure that those annotations did not show up on another book's page. Also viewed the preset annotations made by other EasyReads users. Behavior was consistent with the use case.

Deployment:

Note this app is currently shut down due to Microsoft's free limit

Link: http://recitation-14-team-06.eastus.cloudapp.azure.com:3000/

Steps to host server if Repository is not clone:

1. First get the .pem file from microsoft azure and run the command: chmod 400 csci3308_key.pem
2. SSH into the VM made in azure and install all dependencies
3. Install docker in order to support the different functionalities on our website
4. Create an SSH key and link it to your github account
5. Clone our project repository and add the .env if it doesn't exist
6. Run sudo docker-compose up -d and navigate to our website with the link above.

Steps to host if Repository is already cloned:

1. SSH into the VM made in azure and update all dependencies if needed
2. Update docker if needed
3. Run sudo docker-compose up -d and navigate to our website with the link above.