UNSW
SYDNEY

COMP9900 Project Report

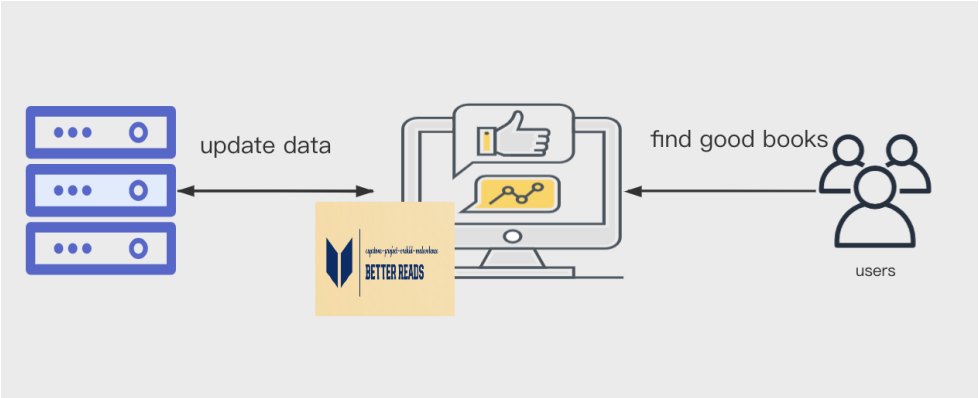
Team Orchid Malevolence

Team Members

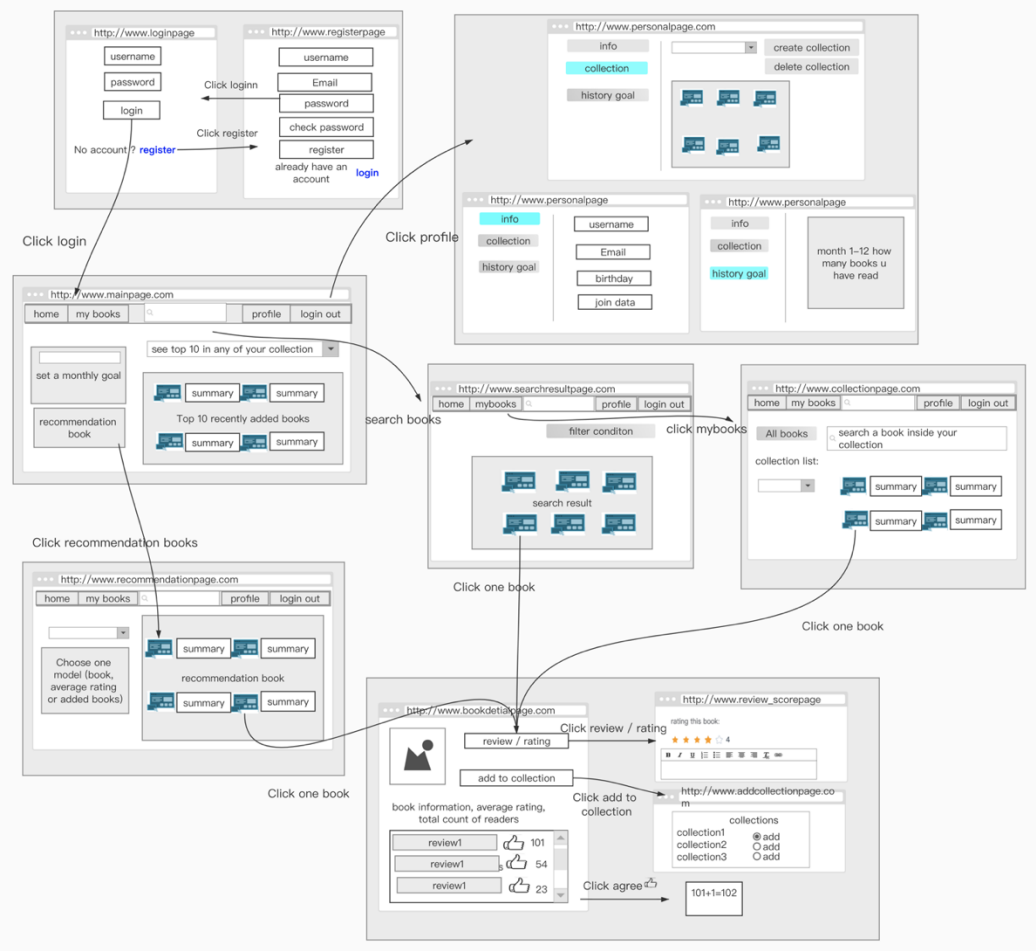
Yuwei Zhuge	z5141744	Scrum Master
Changlu Ma	z5140863	Developer
Yangyu Gao	z5223548	Developer
Yaodong Wang	z5184512	Developer
Yang Li	z5133109	Developer

Overview – Architecture

System flow



Front-end system flow



Technical stack

Frontend: Vue.js +JavaScript + Element-UI + Nginx

Backend: Python + Django + SQLIte+Redis

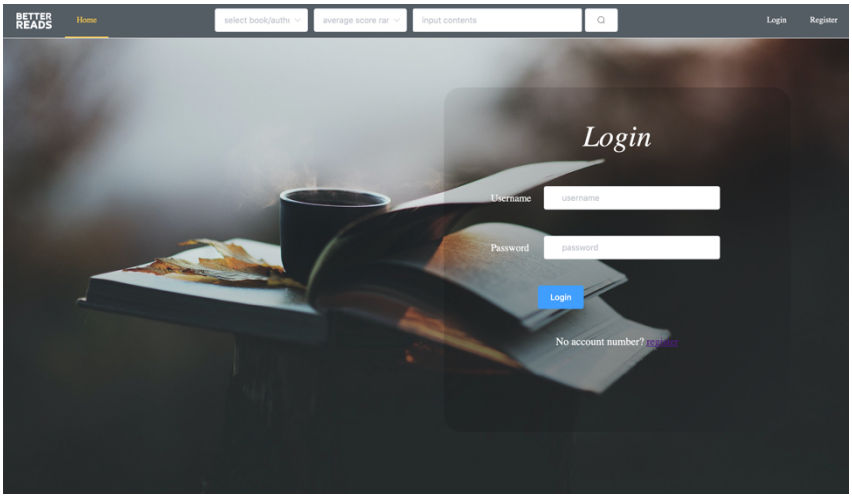
Functionalities

Frontend:

Login Page

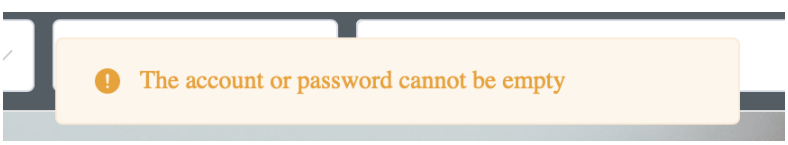
After entering the correct username and password, users can log into the Homepage. For the login interface, users do not need to re-enter the login page when they open the page for the second time. The system saves the user information until the user logs out.

Login Page is showing as follows:

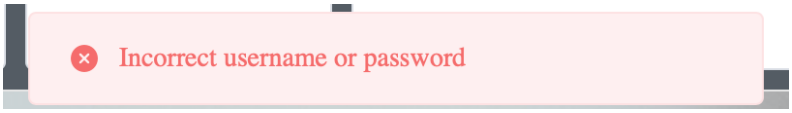


There are three states in which users log in

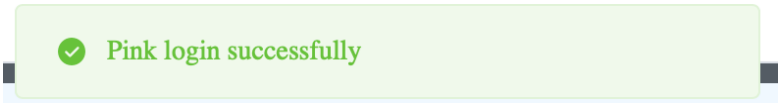
i. If the username or password is empty, prompt "The Account or password cannot be empty" information



ii. If the username or password is Incorrect, the message "username or password" is prompted



iii. If the username and password are all correct, the message "Username + Login Successfully" is prompted



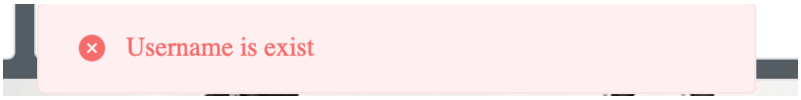
Register Page

Users can sign up for a new account after entering the correct username, email address, and password. Once the user has successfully registered, the page will go directly to the Homepage and there is no need to login again on the login page

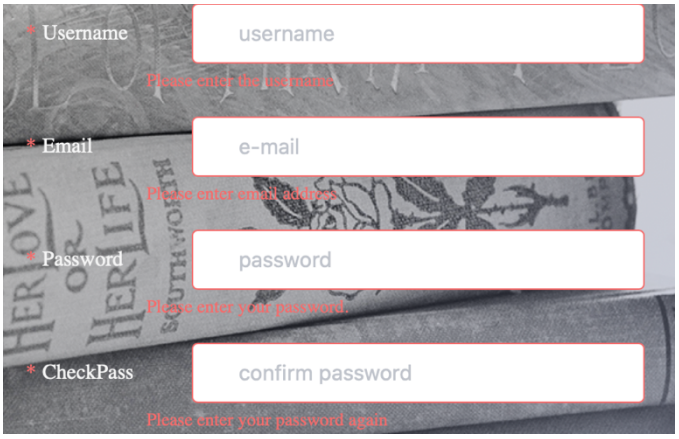
Register Page is showing as follows:

Statistically, there are two states when a user register

i. If the newly entered Username has been registered, the "Username is Exist" message will be prompted



ii. There will be a checksum mode for the length of the username, the format of the message, and the password are entered twice

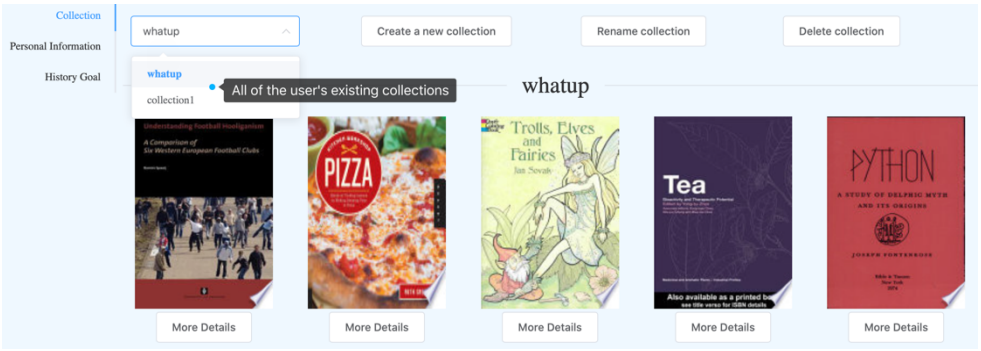


My profile Page

My Profile Page mainly contains three parts, namely Collection, Personal Information and History goal

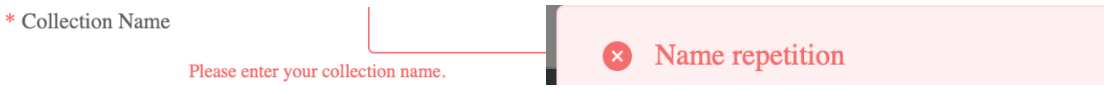
My Profile Page is showing as follows:

Component 1: Collection

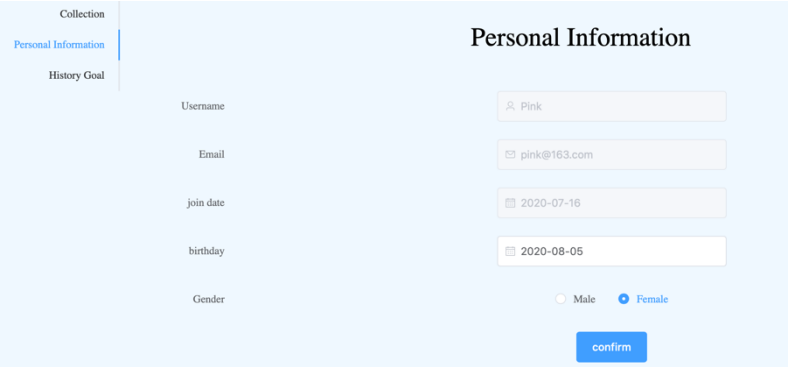


i. The Collection section mainly displays all the books collected by users. At the same time, users can create new collections, rename existing collections, or delete existing collections.

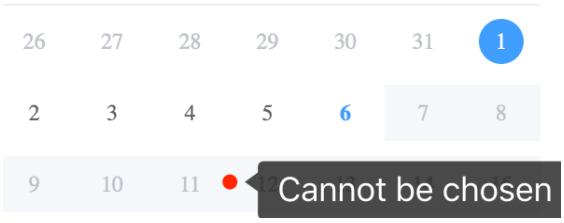
ii. When creating a new collection and renaming a collection, the system verifies that the entered collection name is null and that the name is repeated



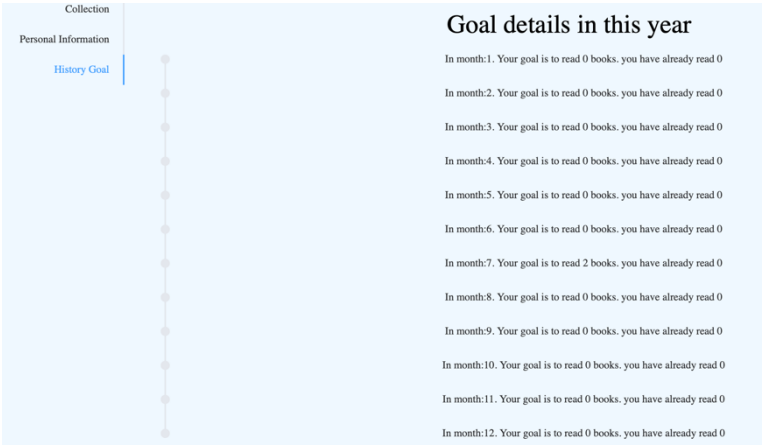
Component 2: Personal Information



i. The Personal Information section mainly displays the user's Personal information.
ii. Username, Email, and join Date are the information that was obtained at the time of registration and cannot be modified. Users can complete their birthday and gender
iii. For birthdays, the user can only select the date before the current date



Component 3: History Goal



i. History Goal shows the number of goals the user has set for this year and how well they have been achieved.
ii. For targets that are not set, the data shows the number of targets and reads is 0

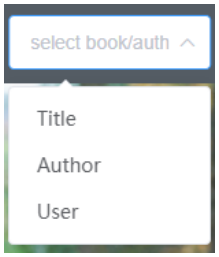
Homepage

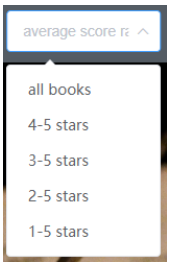
l. The website viewer could see two different types of homepage layout depend on whether he/she is login. Before the viewer login, he/she could see today's top 5 most popular books in this website. After login, the website viewer can see his/her top-10 added books in each collection. In addition, the homepage contains a goal setter to let user set and reset monthly goal. Moreover, through the homepage recommendation button, it can jump to the recommendation page.

Homepage before login is showing as follows:

Component 1: Header

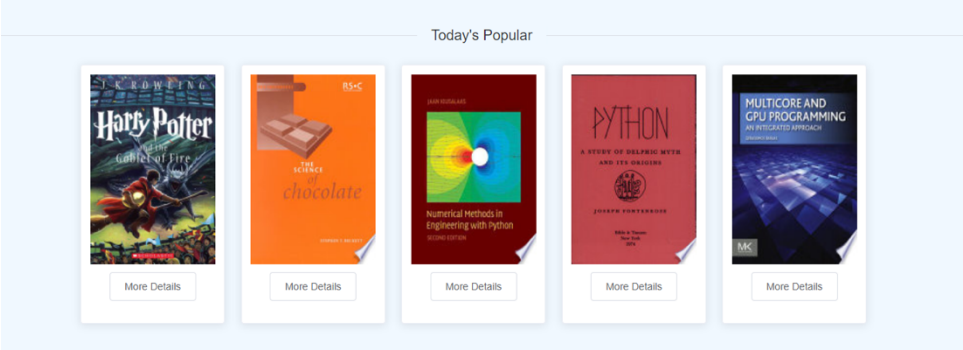
i. A website viewer can jump to homepage by clicking both the leftmost logo and the Home button.
ii. A website viewer can go to the login page to log into this website through account name and password by clicking Login button in the right-hand side.
iii. A website viewer can go to the register page if he/she does not have an account and want to register a new one by clicking the Register button in the rightmost of the header.



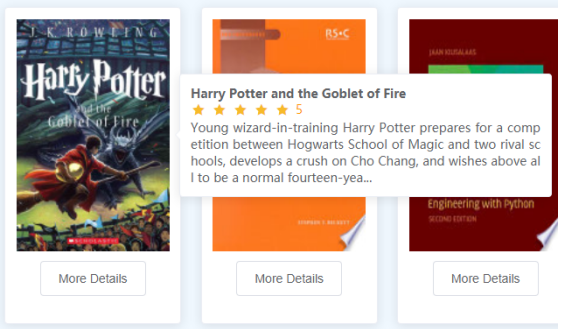


iv. A website viewer can search books depends on their book titles, author name, or see other user's collection by inserting their username. The search section contains two drop-down bar which representing select type and a filter based on average rating. After typing the input content, click search button and the website will jump to the search result page.

Component 2: Before login recommend

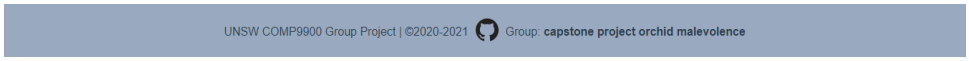


i. A website viewer could see 5 different books recommended by the backend recommendation system, which depends on average rating and the order of added number.



ii. A website viewer could see a brief introduction about this book when the mouse move on the book image. The introduction includes book title, average rating, and a brief book description.
iii. A website viewer can click the More Details button to jump to the specific book page to do more manipulations including rating and add reviews.

Component 3: Footer



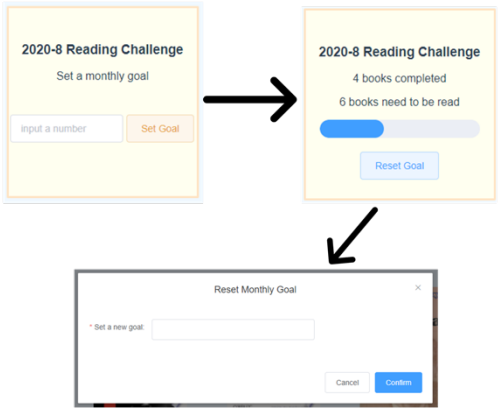
i. A website viewer could see a brief information about our team include team name and which project we have join in.
ii. A website viewer can see the source code of this project by clicking the GitHub icon which located in the middle of this footer.

Homepage after login is showing as follows:

Component 1: Header after login

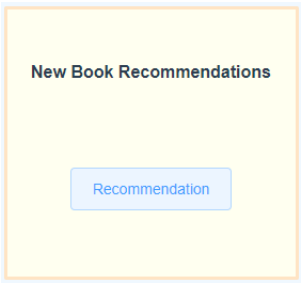
i. A website viewer can jump to see all their books by clicking My Books button, this button only appears after login.
ii. A website viewer can jump to his/her profile when he/she clicks the button next to Logout button. The button will show the username base on user's account.
iii. A website viewer can logout his/her account by clicking the Logout button. After logout, the layout of the homepage will go back to before login type.

Component 2: Monthly Goal Setter



i. A website viewer could set a monthly goal as his/her reading challenge if the user has not set this month's goal yet.
ii. A website viewer could change the goal by clicking the Reset Goal button, then an extra window will pop up and could let user to reset a goal.
iii. A website viewer could only set a goal in range [1, 99], due to it is a monthly goal.

Component 3: Recommendation



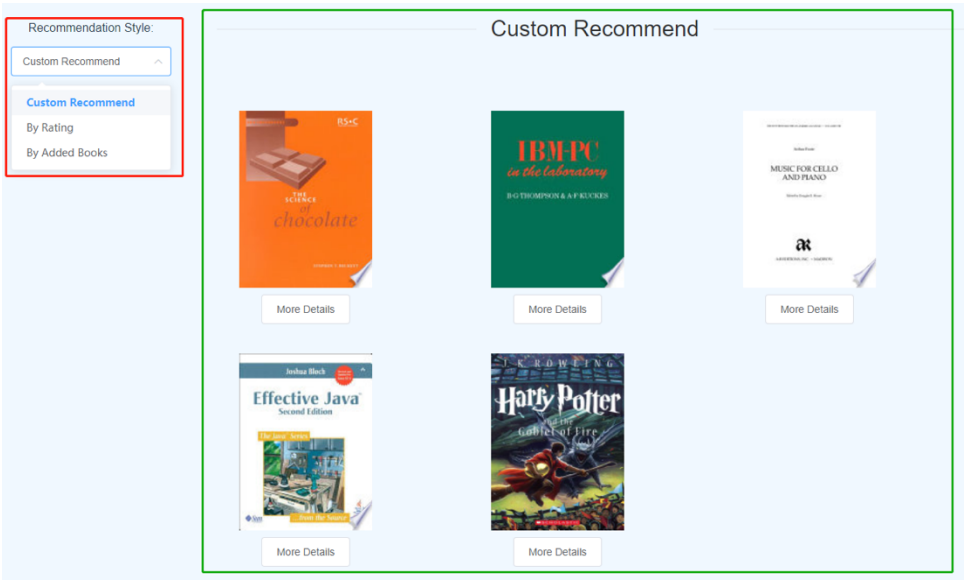
l. A website viewer could see the recommendation book page when he/she clicks the Recommendation button in homepage. The details of recommendation page will show in the introduction of recommendation page.

Component 4: Homepage Collection

i. A website viewer could take a brief look at all his/her collections. In this part, the result will only return the top 10 recent added books.
ii. A website viewer could see a divider above the books, which the title is high related the collection name.
iii. A website viewer could select to see which collection's books by clicking the selection bar.
iv. A website viewer could see a brief introduction which is completely same as the manipulation before login. In addition, user can view the details by clicking More Details button.

Recommendation Page

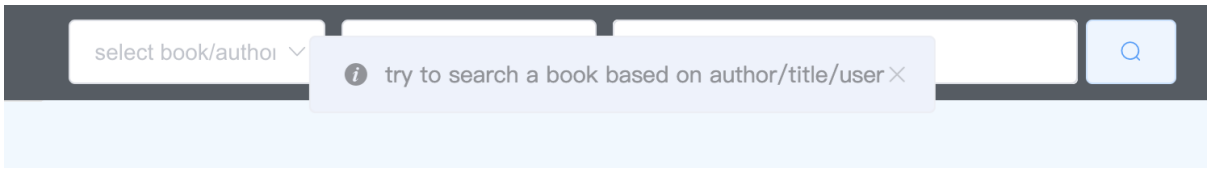
Component 1: Recommendation Style Selector



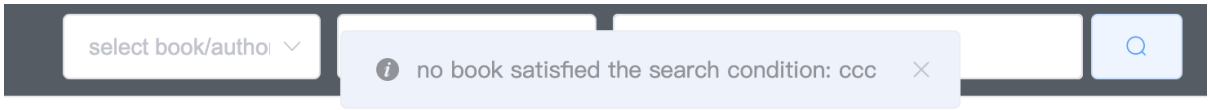
i. A website viewer could click the selection list to choose a recommendation style, each of the selection will return different results depends on different recommend algorithms.
ii. A website viewer chooses a recommendation style and the right-hand side will return at most 10 books that fit current user most.

Search result page

If user don't give a search condition, the website will return below message and re-route to homepage



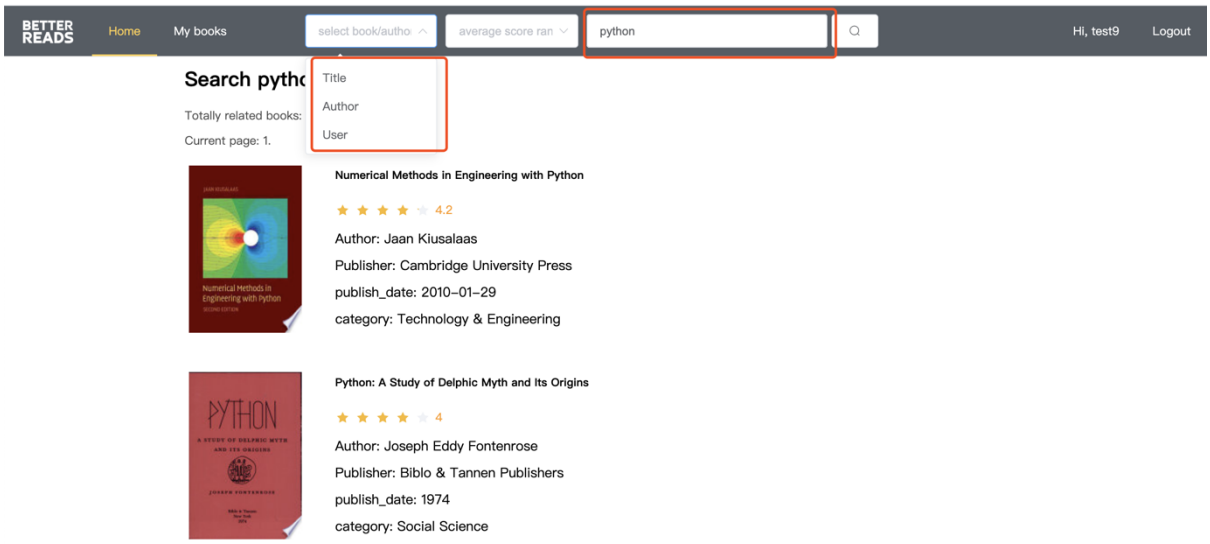
If user don’t give a search condition and type words that don’t exist in the database, it will return below message



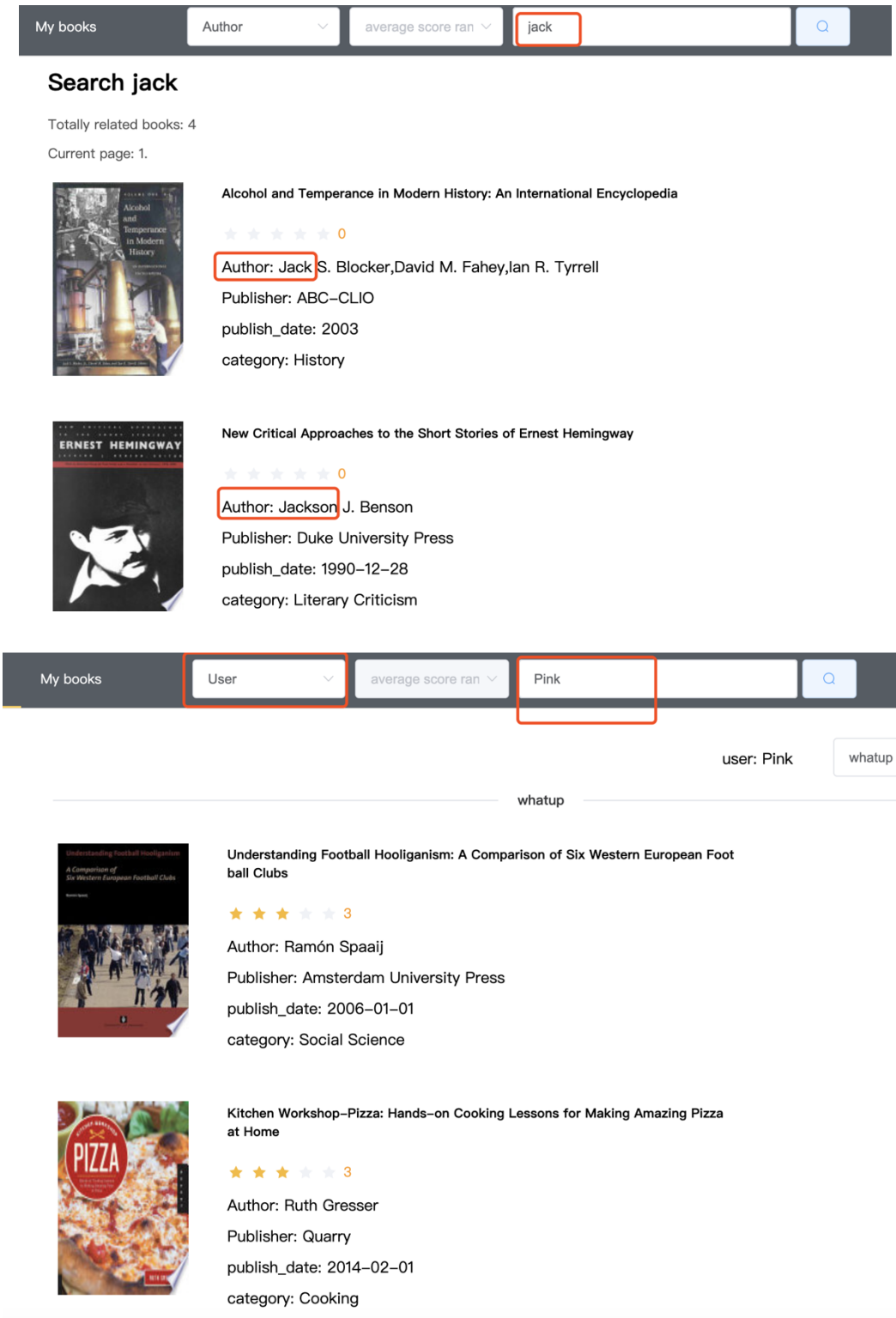
If user search based on username, the average score filter condition cannot choose

1. **To more easily locate books, readers must be able to easily search for books by name, author, or most recently added books to their collections.**

Users can search book based on title/author/user, and the result is sorting by average score

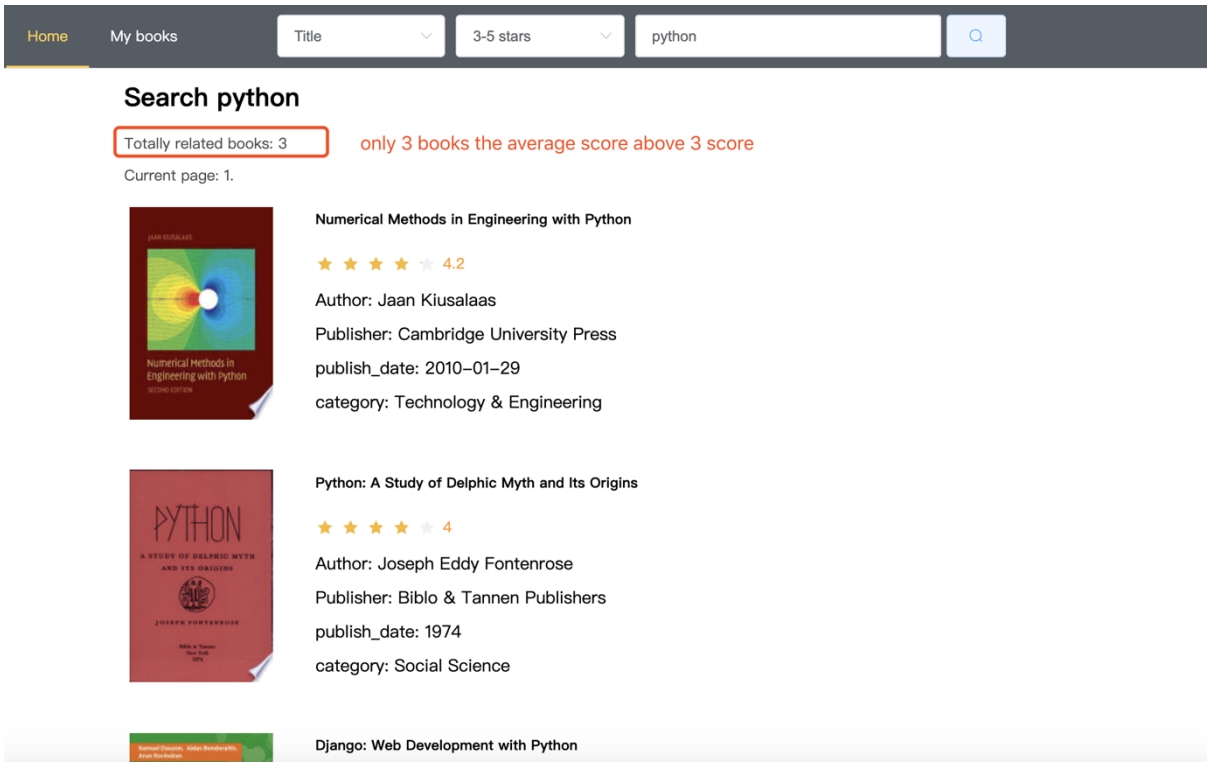


i. To make it easier to find books, readers must be able to search for books by name or author, which should bring up a results list that includes a summary of the name, author, and year of publication for each matching book.



ii. As a site visitor, I want to see a search frame in the main page so that I can search a specific book conveniently.

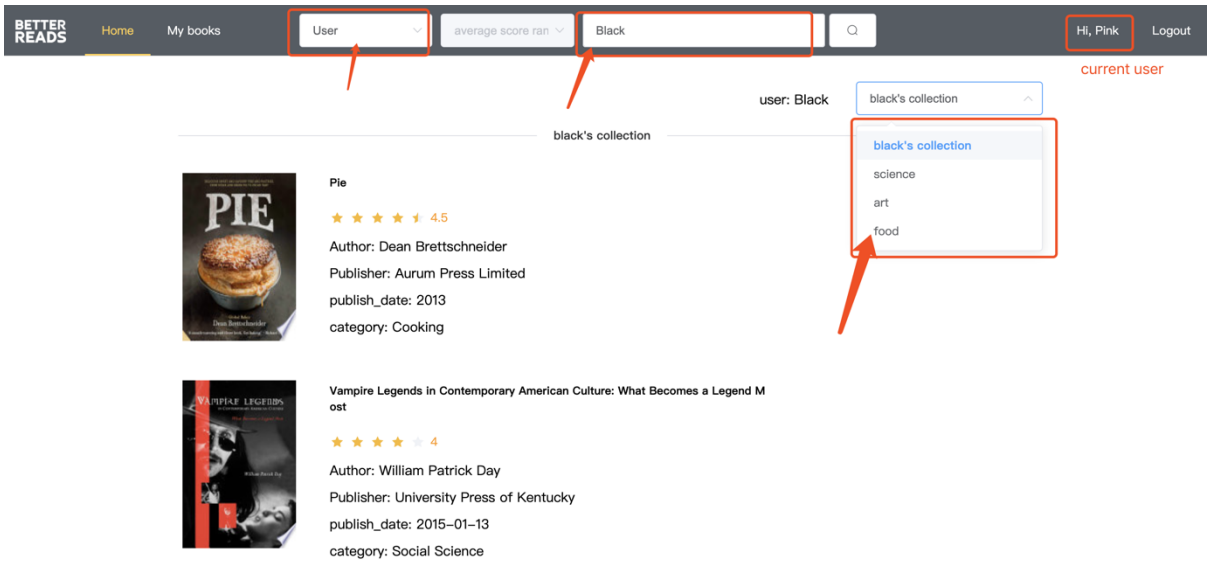
2. **They must be able to filter out books from search results that are under a given average rating.**
   User can search book based on a given average rating



i. Rating the book from 1 to 5, and the system will calculate the average rating score for every book

ii. Search book based on filter condition which under a given average rating.

3. User looking through other readers' collection, they can search any user by specific name and have a look at his/her collections:



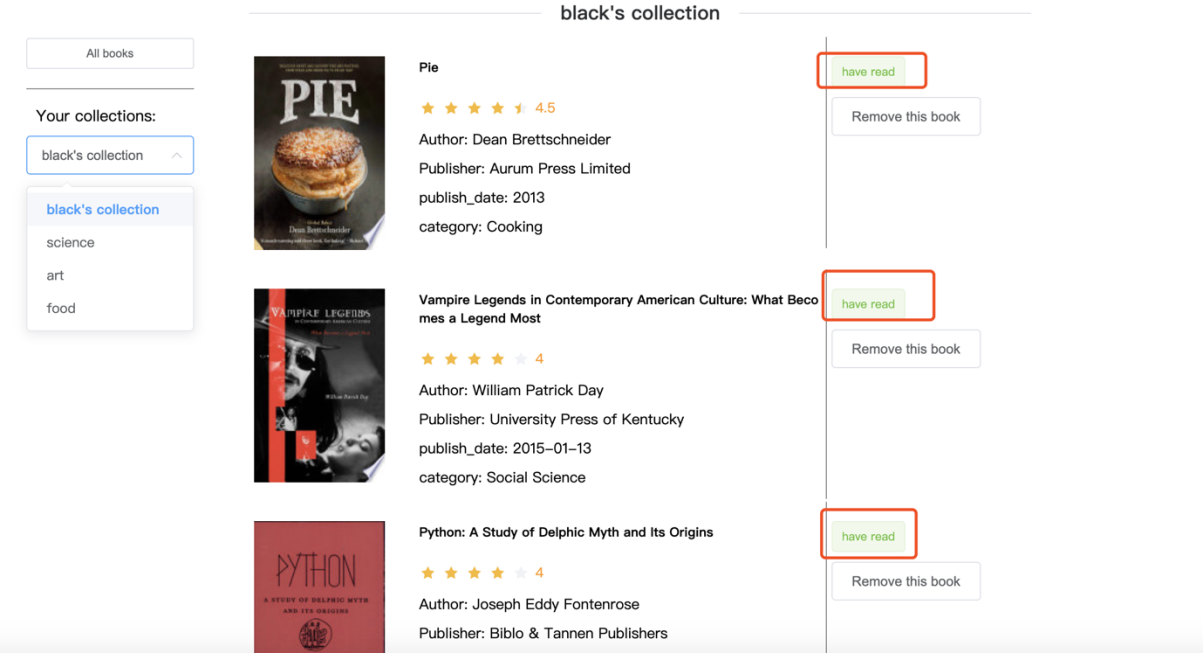
i. Search other users by inputting their username and see any book in any of the collections

ii. Readers must be able to view a list of their collections and click into any of these collections to view the books they contain.

My books page

1. **Readers must be able to keep track of books they have read in one of many easily identifiable book collections on their account, which contains the below user stories:**

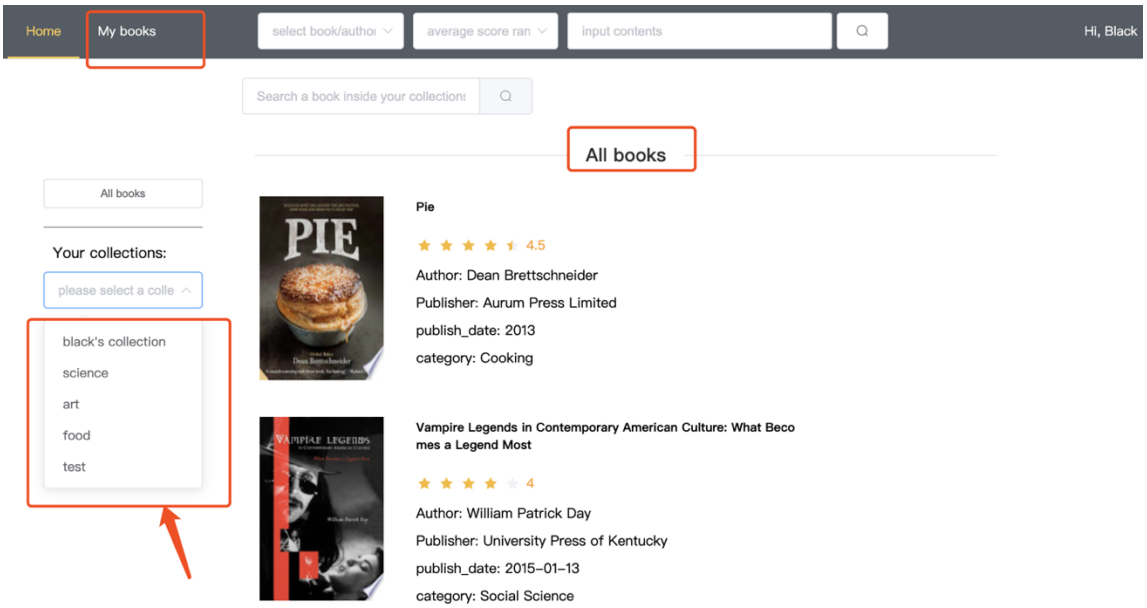
   A website user enables to do the below actions:



   i. See all books in any of the collection and check which book that the user has read
   ii. User must be able to remove books from any of their own collections.

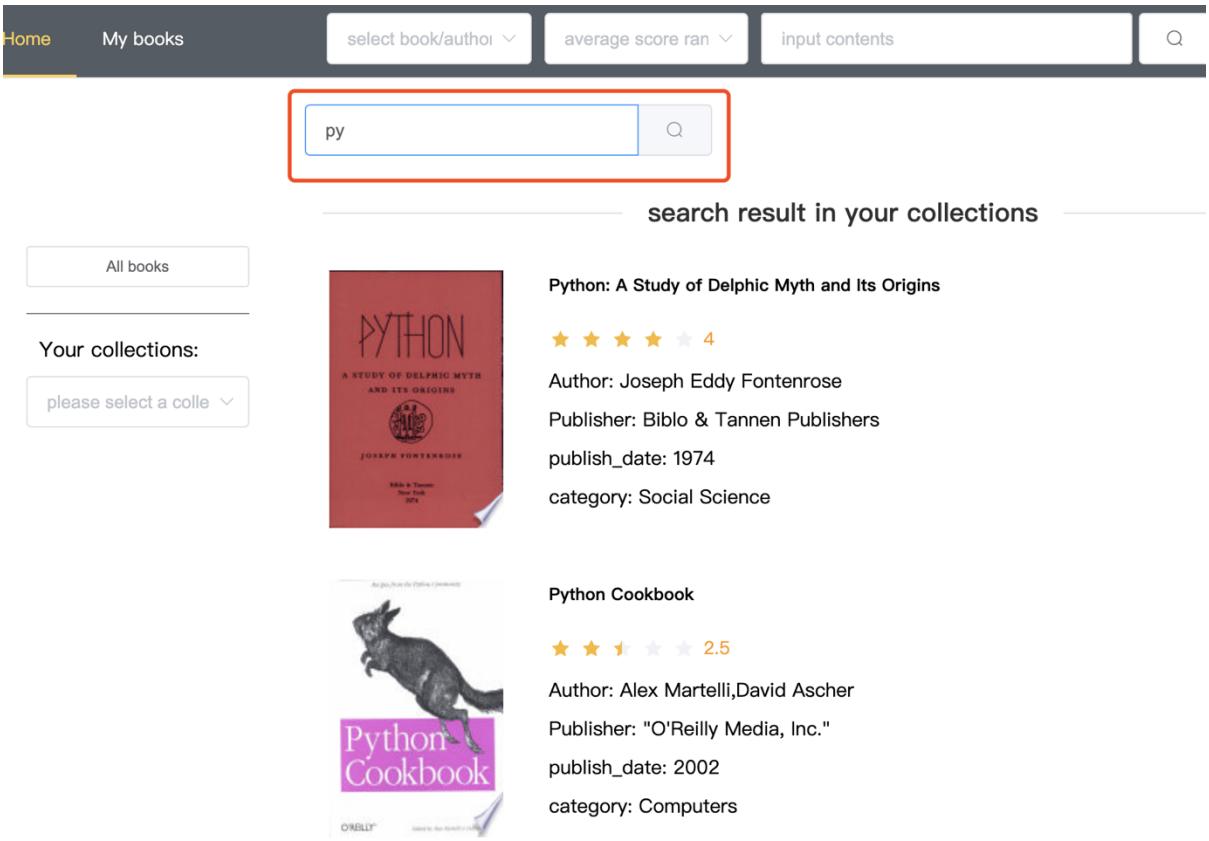
2. **User must be able to look through any such collection (their own and that of other readers (in search page can see other readers')).**

   user can click the button of my books in the header of the website and see his/her collections



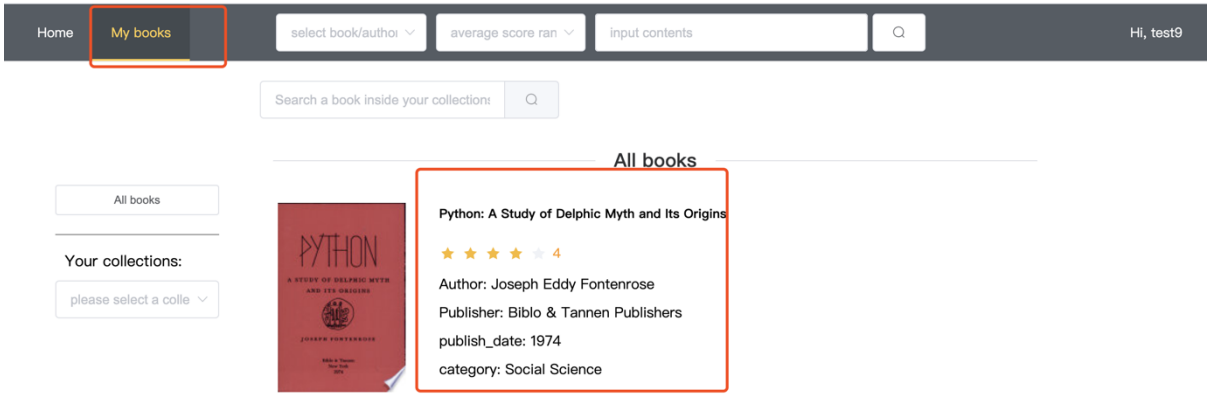
i. Readers must be able to view a list of their collections, and click into any of these collections to view the books they contain.

3. User can easily find a specific book inside his/her collection by search this book inside his/her collections



4. Readers must be able to view the full details of any book, (including all reviews), in any book collection.

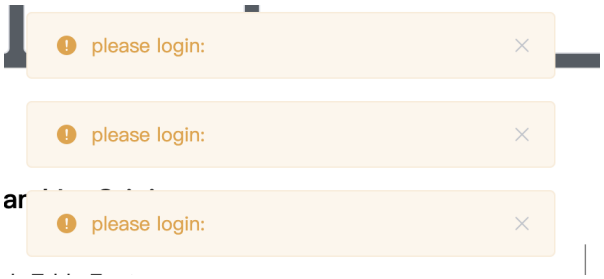
User can get the full details of any book by click it from any collections and get into a single book page, which contains title, author, publisher, publish data, category, ISBN and all reviews from readers



i. Write some reviews and rating a book from any book that has read.
ii. See all detail for a book from any collections

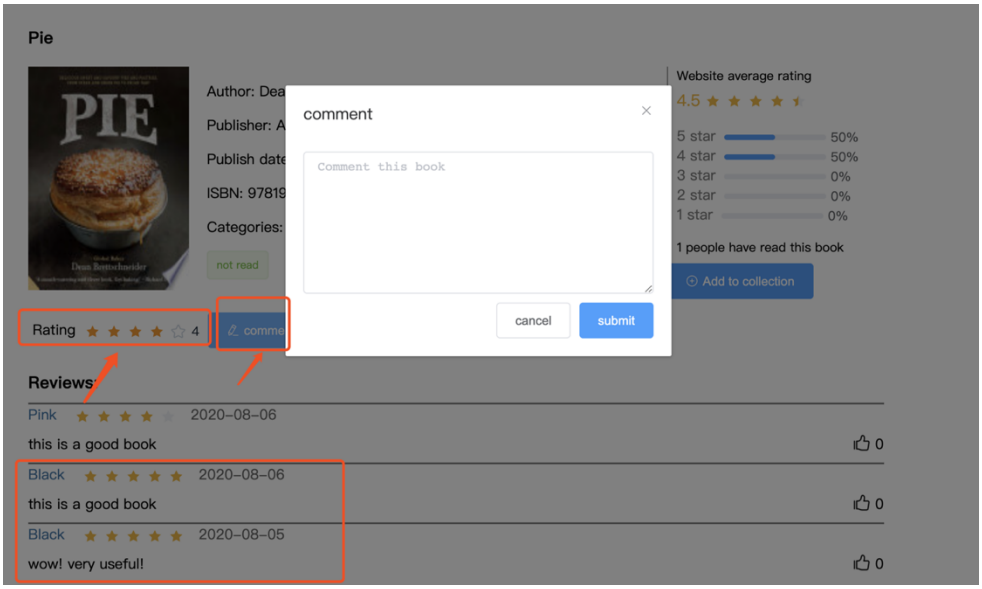
One book page

1. **If user is not login, then they cannot do any operation among rating, reviewing, adding a book to collection or like a review.**



2. **To help with selecting the right book, readers must be able to write reviews for a given book, including leaving a rating out of 5.**

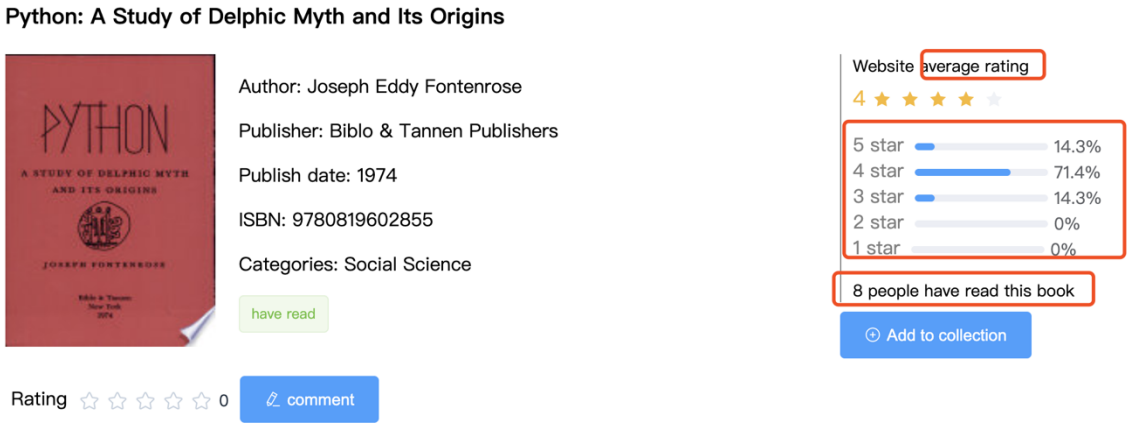
User can click a specific book and write many reviews, also give a rating out of 5 for this book. For every comment, user must write at least 3 words



i. readers must be able to add a review for any of the books that they've read, which should include a rating out of 5

3. **This full detail view must also display multiple aggregate statistics, including the book's average rating.**

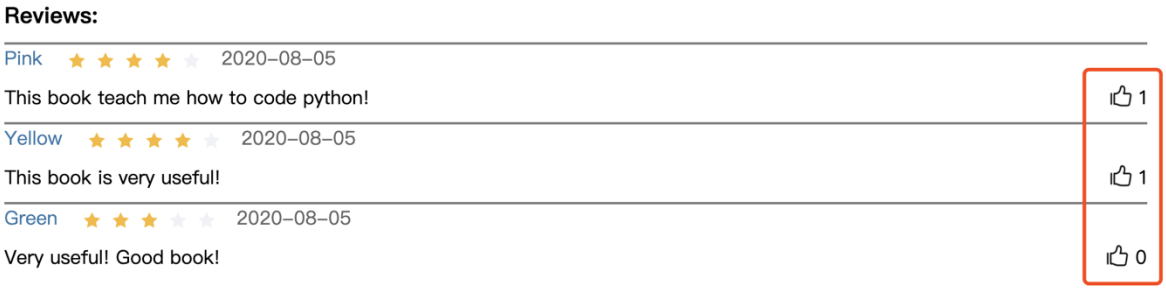
User can see may multiple aggregate statistics, including the books' average rating, how many people have read this book, and the percentage for rating from 1 to 5.



i. - Rating a book that has read from 1 to 5

ii. - Record the total count number on how many readers have read this book, and calculate the aver- age rating score about one book

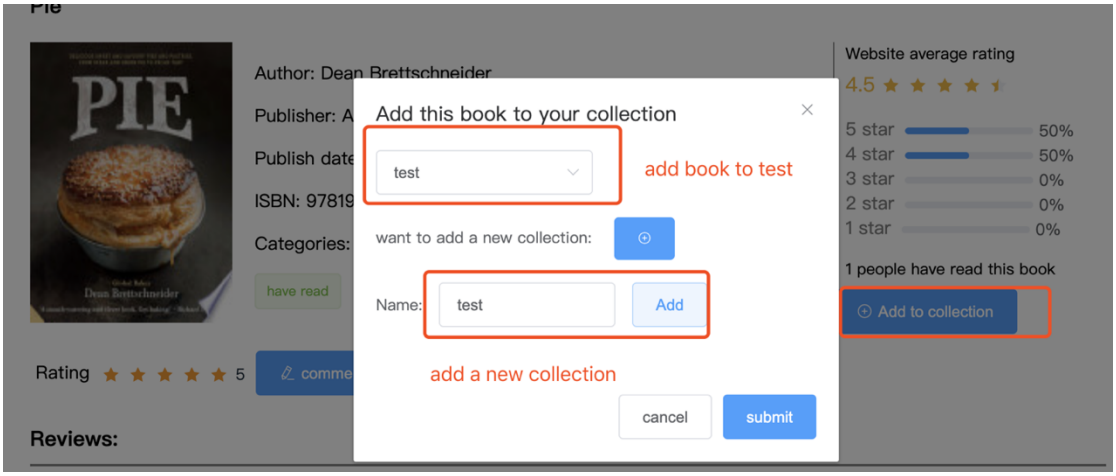
4. User can like any comment, and the reviews list will display based on like times, more people like this comment, then it will show on the top



i. - User can like comments from reviews.

ii. - Reviews list will display the reviews based on how many people like it.

5. Create different collections and add a specific book into their collections



i. - Readers must be able to add a book to their "main" collection, which should include details about the book title, author(s), publisher, publication date, and category.
ii. - User must also be able to add "named" collections to their account, which should include a name defined by the reader
iii. - User must be able to add books to any of their own collections.

Backend:

Api guideline , request detail and response detail, see here: More detail
Some main Apis:

1. User:
   Login and Register
   User can update some information.
2. Book:
   Admin can insert book into database.
   User only search book by title or author.
3. Collection:
   each user has default collection when user register success
   create new collection
   rename collection
   delete collection
4. Rating:
   create rating
   update rating
5. Review:
   write review
   update review
6. Like:
   create like
   update like
7. Search:

User can search book by title or by authors, and you can filter result with rating.

8. Recommend:
   If user rate book more than three times, system will provide recommend books for user. Else, user only can get top 10 rating books.
9. Bookdetail:
   User sends book id to acquire book detail. It is a complex json data.

Proper references and brief descriptions of ALL third-party

Library:

Frontend:

1. Axios
2. Vuex
3. Element-UI
4. Router

Vue is suitable for group work and we import several libraries to improve our work efficiency. Using Element-Ui enable us to fast build the front page and reduce amount of time to writing css. Also, it contains many features which can enable us to interactive with the dynamic data convenient. Router and Axios help us to fast exchange data from back-end and other pages.

Backend:

Dependencies are grouped by requirements.txt file.

1. Django: framework
2. django-cors-headers: fix cross connection issues
3. Rest_framework(rtf): Restful api, Authorization, serializer
4. Json: transform data type
5. Redis(library): control redis in python
6. Persarson Algorithm
7. Redis(server): as cache

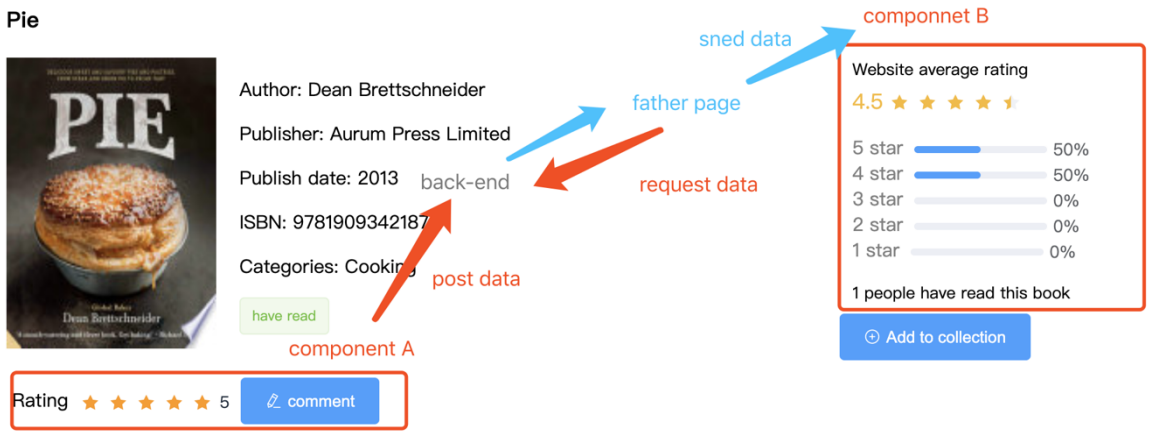
Implementation challenges:

Persist login

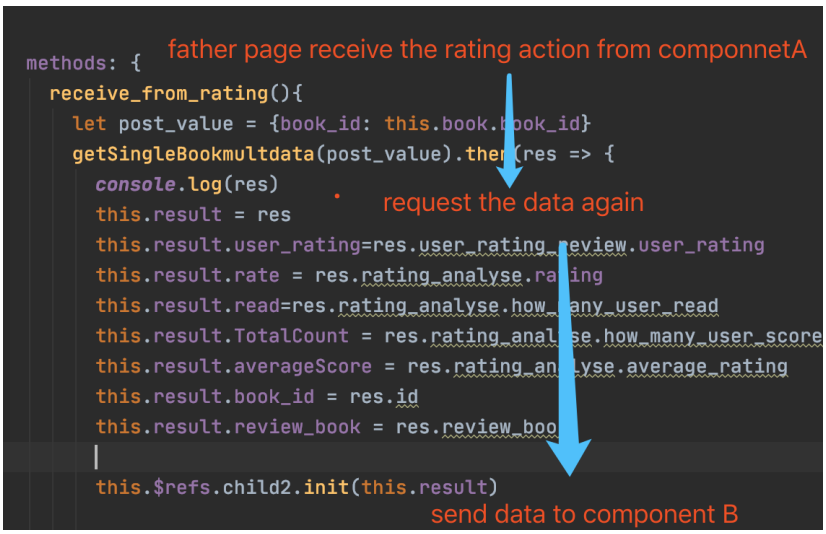
To achieve the functionality of persist login, the browser has to persist the token of last authenticated user working with the sessions in the backend theoritically. We save user information by using LocalStorage. LocalStorage enables the browser to retain data after the window has closed

Trick in sync request and asynchronous request

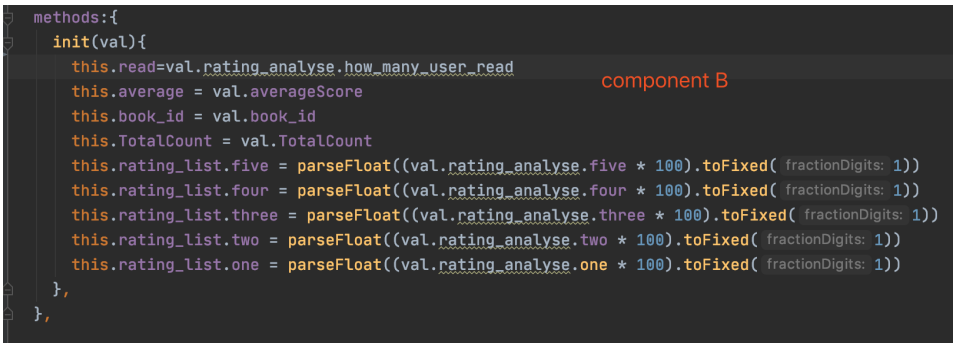
When I use axios to post the data from component A to the backend, I thought component B can display multiple aggregate statistics dynamically. However, I found axios is one kind of asynchronous request method. According to the Vue lifecycle, the request action from father will run before the post action of component A, and if using prop to transfer data from father page to component B, the statistics data will show without update. To solve it, I use ref to send updated date to component B after watching the rating action in component A and solve this problem.
The whole idea is like watch the action in component A, if father page received the rating action, then father page will request the data again and send the latest data to component B.



The coding like bleow：
Father page send updated data to component B



Component B receive the data



User-based Collaborative filtering Recommendation

$$Corr(x,y) = \frac{Cov(x,y)}{\sqrt{Var(x)Var(y)}}$$

$$\rho x,y = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

Above is the Pearson correlation coefficient between two variables, which is obtained by dividing the covariance by the standard deviation of the two variables. The greater the value of the covariance, the greater the degree of the same direction of the two variables.

```
# pearson similarity
#
def pearson_correlation(target, others):
    sum_xy = 0
    sum_x = 0
    sum_y = 0
    sum_xx = 0
    sum_yy = 0
    n = 0
    for key in target:
        if key in others:
            n += 1
            x = target[key]
            y = others[key]
            sum_x += x
            sum_y += y
            sum_xy += x * y
            sum_xx += pow(x, 2)
            sum_yy += pow(y, 2)
    if n == 0:
        return 0

    denominator = sqrt(sum_xx - pow(sum_x, 2) / n)  * sqrt(sum_yy - pow(sum_y, 2) / n)
    if denominator == 0:
        return 0
    else:
        numerator = sum_xy - (sum_x * sum_y) / n
        return numerator / denominator
```

The first step is to calculate the similarity between the user and others, and according to the level of similarity, we can find K neighbors that most similar to the user.

```
# calculate neightbour similarity
def neighbor_similarity(target,datas):
    distance_list=[]
    for key in datas:
        if(key!=target):
            dis=pearson_correlation(datas[target],datas[key])
            distance_list.append((key,dis))

    distance_list.sort(key=lambda i:i[1],reverse=True)
    return distance_list
```

Among the items that other neighbors like, we can calculate the recommendation degree of each item according to the distance between the user and other neighbors. Then, we can get a list of top 10 recommend books. Therefore, the website can recommend books to users according to the recommendation degree of each book.

However, the user-based nearest neighbor algorithm may spend a lot of time to calculate the list of distance. It will increase with the increase of the number of items and users, when

the amount of data is very large. For example, if we input 100,000,000 items, the time cost is too large.

```
def calculate_correlation(target,datas):
    recommend={}
    distance_list=neighbor_similarity(target,datas)
    total_cor=0
    target_book_list=datas[target]

    for i in distance_list:
        total_cor+=i[1]
    if total_cor==0:
        total_cor=1

    for i in range(len(distance_list)):
        weight=distance_list[i][1] / total_cor
        neighbor_id = distance_list[i][0]

        neighbor_books = datas[neighbor_id]
        for j in neighbor_books:
            if j not in target_book_list:
                if j not in recommend:
                    recommend[j]=neighbor_books[j] * weight
                else:
                    recommend[j]+=neighbor_books[j] * weight

    rec_list=list(recommend.items())
    rec_list.sort(key=lambda i:i[1],reverse=True)
    # get element which mark >= 0.
    res_list=[]
    for i in rec_list:
        if(i[1]>0):
            res_list.append(i)
    # return top 10.
    return res_list[:10]
```

Use Redis as cache

Compared to other key-value database, Redis is faster and supports data persistence, meaning that the data in the memory can be saved in the disk, and can be loaded again for use when restarting. The high performance makes it can read 110,000 times/s, and write 81,000 times/s. Also, Redis supports many data types, including Strings, Lists, Hashes, Sets and Ordered Sets data type operations in binary cases. Redis operations are atomic, and Redis also supports publish/subscribe, notification, key expiration and other features.

When user number is very large, user-base CF will cost too much time. So, I decide to user Redis to decrease the time cost. When user login, server already calculate the custom recommend result and store them into Redis. So, when client request recommend result, I can check Redis chche and get these data from Redis.

User documentation/manual:

Note: Make sure port 8000 and 8080 is not used!
You can use other high-performance server to run our code, but in here, to get the stable overview, we use default debugging Server to run our code.

Set-up:
Now, you have “project.zip”,

Unzip it!
You can get a file named project,

Click into project, there are five files:

1. First part-redis:
   i. click redis zip file:
   You can get redis-6.0.6 file in the same path

   ii. open your terminal:
   $~ cd redis-6.0.6/src
   $~ make

   Wait a few second
   $~ ./redis-server
   You can find that on your terminal:

Ok,redis server is OK. Keep this terminal!

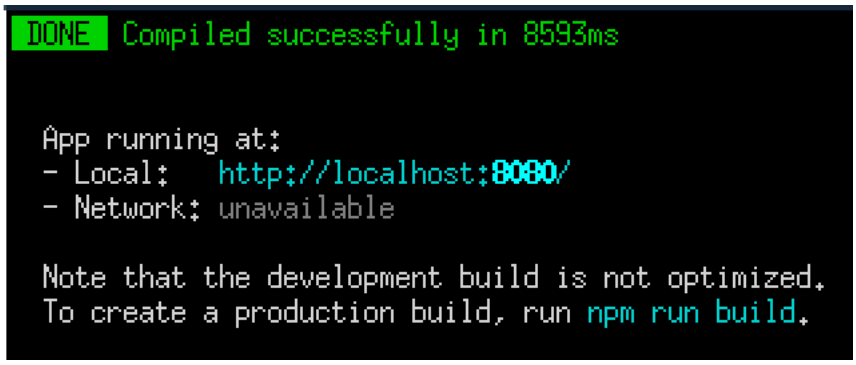
2. Second part Vue:
   i. Open a new terminal in project root path.

   $~ cd mainpage

   $~ npm install

   Wait few time~

   $~ npm run serve



   It means run Vue file on default debugging server successfully!

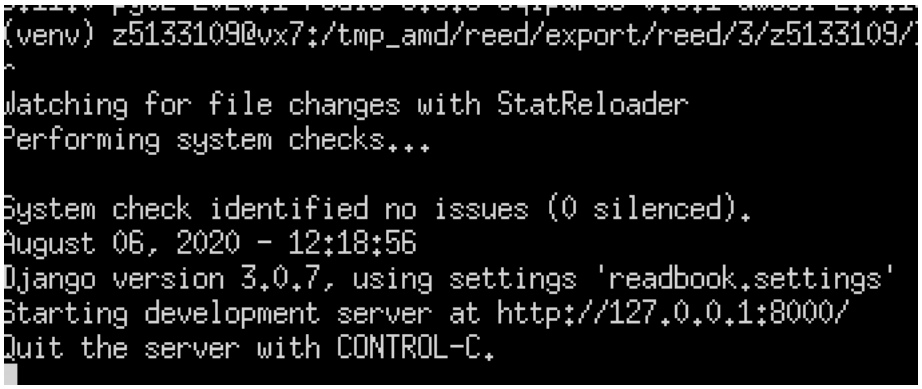
3. Third part Django
   i. Open a new terminal in project root path.

   In here, I choose to create virtualenv in project toot path, actually, you can create it in anywhere.

   $~ python3 -m venv venv

   ii. You can find a file named venv in your local path.

Run venv:
$~ source venv/bin/activate
$~ cd readbook
$~ pip3 install -r reuirements.txt
$~ python3 manage.py runserver



Django run successfully!

4. Enjoy your reading time!
   Now open a browser, recommend google chrome.
   Input localhost: 8080
   You can find our main page. You can search book, login or register!
   Ps: Some account info, username and password in account.txt!

Reference

M, Callum 2018, *Vue.js: Up and Running: Building Accessible and Performant Web Apps*, e-book, accessed 23 Feb 2018, <https://www.programmer-books.com/wp-content/uploads/2018/11/Vue-js-Up-and-Running.pdf>.

B, Ahmed 2019, A Beginner's Guide to Vue CLI, Medium, accessed 25 Jun 2019, < https://www.sitepoint.com/vue-cli-intro/>.