**Software Requirements and Design Document**

**For**

**Group <5>**

Version 1.0

**Authors**:

Zhixi L

Hanyan Z

Yuanyuan B

Wesley W

Dennis M

# Overview (5 points)

This website will allow FSU students to easily trade used textbooks among themselves. Trading textbooks is much more economical for students than to have to buy new ones all the time. This website will help students save money and reuse resources, which is also good for the environment.

The website will have all the basic features that websites for trading used books should have, which include but not limited to user system, posting feature, and booklist page. The website will allow FSU students to trade used textbooks of previous semester for textbooks of the classes they are currently taking.

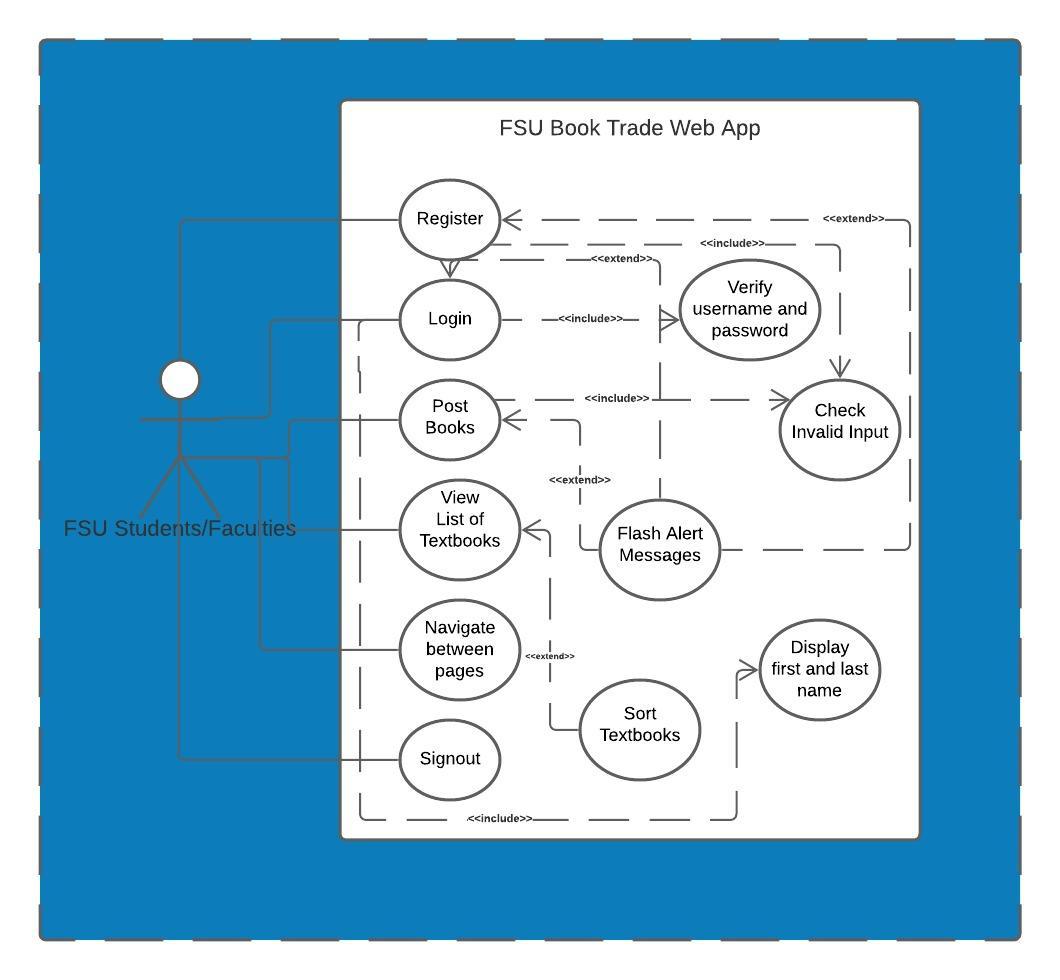
# Functional Requirements (10 points)

1. The login page shall authenticate the user with user-inputted username and password through Post request. If the user entered the wrong password or username, prompt the error message. If user input matches with a username and it’s password in the database, then redirect to homepage. (High)
2. If the user has already logged on, any access to the login page shall be redirected to the home page. (Low)
3. When a user searches for a textbook, the web app should compare the existing books’ names with the keyword. Display all matching books to the user. (Medium)
4. The header should display the full name of the user if already signed in, or “offline” if the user is not signed in. (Low)
5. In the booklist page, app.py should find all the posts in the post table, order by the time posted, then send it to be displayed on the booklist page. (High)
6. When the user accesses the home page, app.py should automatically fetch 12 most recent posts and display it in the home page. (Medium)
7. In the booklist page, the users should be able to change the ordering of the books by clicking on different buttons. The default ordering is by the time in which the books are posted. Users will be able to change it to order alphabetically. In addition to this, the booklist page should also support filtering by colleges. For example, if the users click on filtering by the College of Computer Science, the list should only display books in courses belonging to this college. (Medium)
8. In the homepage, the users should be able to click on About Us and Commonly Asked Q&A to get a general idea of how this web application works and commonly asked questions regarding the uses. (Low)
9. The web application shall be able to redirect the users to the homepage from any pages whenever they click on the website logo located in the header. This requirement is derived because not every page has a “back to homepage” link, which may result in difficulties for the users when they are stuck in a certain page and cannot return to the homepage. (Medium)
10. In the post page, users can upload basic information about the textbooks that need to be traded. Such as the name of the textbook, its degree of newness, price, physical photos, and the classification of the textbook, such as the textbook of which college it belongs to. All listed are required to the user, because those informations will be used for the search and order area. The Post page shall validate user inputted data and prompt alert messages it the input is not in the correct format. After all data are validated, the data will be send to the database after the user clicks Submit(High)
11. For the category of the textbook, which is a “drop down menu”. It listed all colleges of department. The user only needs to select one. This category also helps users to search textbooks.(Medium)
12. The description area for the user to add more details about the textbooks, but it is not a required.(Low)

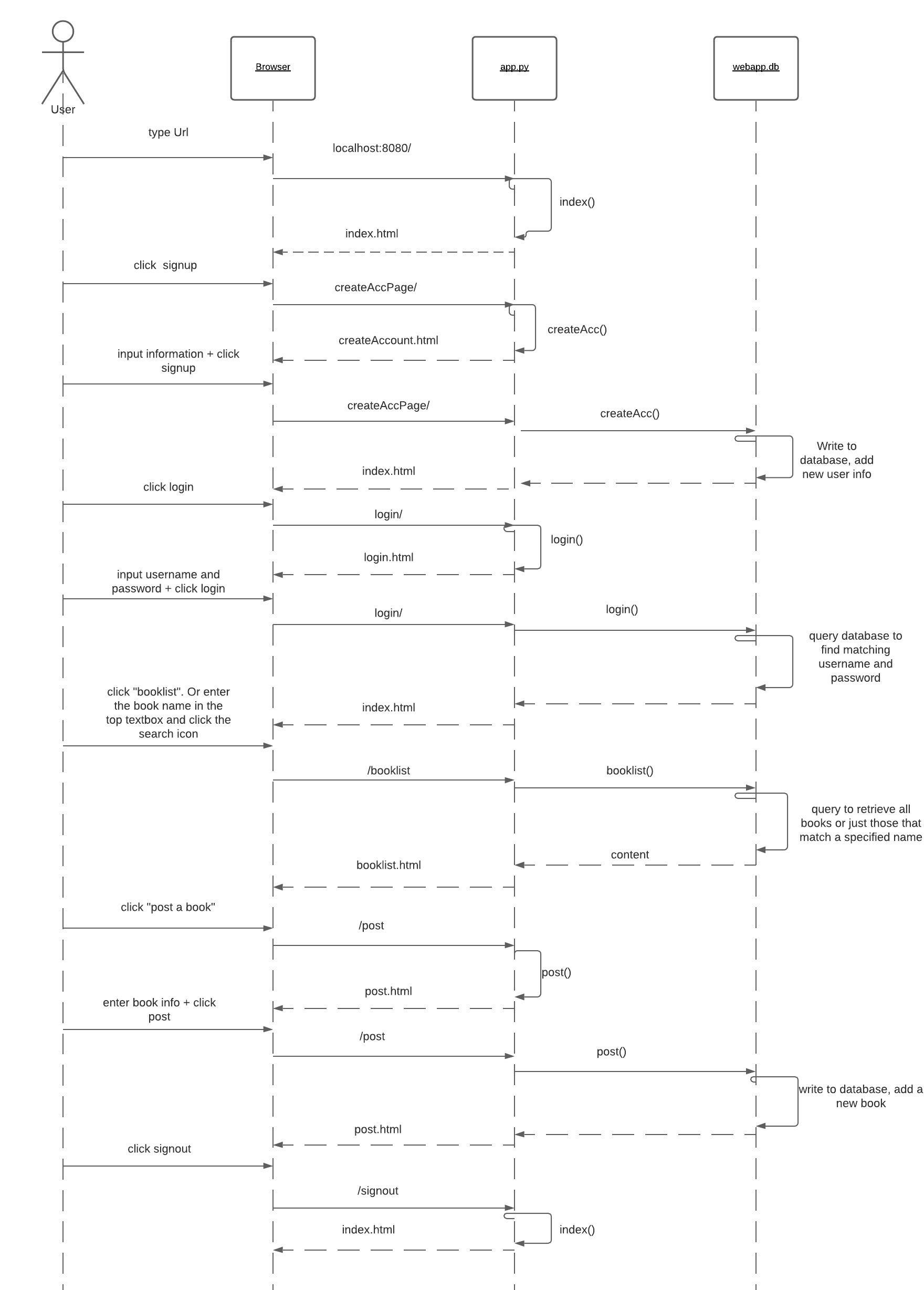
# Non-functional Requirements (10 points)

1. Use SQLAlchemy as the ORM to manage the database.
2. Use SQLite3 that came with python as the local database to store account & post tables.
3. Route Controller app.py should not send sensitive data to users such as the password of a user.
4. Search functionality for people to find the posts in 3 seconds.
5. System should keep track if the user has signed in or already signed out at all time.
6. The system shall keep user information secure all the time.
7. The front-end should be implemented using HTML/CSS/JS and AngularJS.
   1. Sorting and any other similar features should be implemented in the front-end using JavaScript and AngularJS. This requirement is derived because implementing everything on the server side can make the application less responsive due to the round trips the data has to go through.
8. The system shall not become unresponsive in any cases and shall be able to restart itself whenever an error occurs.
9. The system should store user information in the correct database/area.
10. The server-side should be implemented using Python and Flask.

# Use Case Diagram (10 points) \*Yuki did this part



# Class Diagram and/or Sequence Diagrams (15 points) \*Dennis did this part

**

# Operating Environment (5 points)

For server to operate:

The Windows 10 or MacOS 10.15.7 both with at least python 2.7.18

Installation of flask, SQLAlchemy is also required.

For Client to access web app:

This web application should be able to operate on any browsers including Safari and Chrome.

# Assumptions and Dependencies (5 points)

* Assuming the user is not accessing this webpage through the internet as the current web app we’ve developed can only be accessed from the local network, we haven’t implemented casting & buying url for our website yet.
* One of the assumed factors is that the webpage designs are tested using only Safari and Chrome, so we are not sure whether the web pages will look the same way on other browsers. If this will cause problems, the potential operating environment will change since it does not work on browsers other than Safari and Chrome.