**Progress Report**

**- Increment 1 -**

**Group #5**

# Team Members

Hanyan Zhang (Yuki): fsuid: hz19, GitHub ID: YukiKentyBonita, written in blue

Zhixi Lin (Zack): fsuid: zl19, GitHub ID: ZhixiL, written in red

Yuanyuan Bao: fsuid: yb19b, GitHub ID: yuan2021-cop, written in purple

Wesley White: fsuid: wlw19, GitHub ID: Wesley6801, written in green

Dennis Majano: fsuid: dvm17, GitHub ID: D3nnis-24, written in orange

1. **Project Title and Description**

FSU Book Trading Site

This web application is made for FSU Students to trade and sell textbooks with each other. Since textbooks are required for most courses and are generally considered expensive for students, FSU students experience problems such as spending hundreds of dollars on textbooks per semester and leaving them unused after only one semester. The FSU Book Trade web application is implemented to solve these problems for FSU Students. They are able to sell textbooks or trade their old textbooks for different ones needed for the current semester. Other students are able to buy the required textbooks at a lower price.

1. **Accomplishments and overall project status during this increment**

Overall, we have created a functional book trading web app containing the minimum amount of features an online trading site needs, with all data processed at the server-side (Will be separated into client-side & server side later). Below is the detail:

* Account system that keeps track of the user & connects with the sqlite3 database that’s managed by flask-sqlalchemy. The register page takes in the new user information, validates and processes at app.py, and sends it to the database under the Account model (table that stores account information in our database). Session attribute from Flask is also used to keep track of if a user has logged on.
* Index page with good-looking design (exceeding the initial planning), displaying the most recent 12 book listings posted. It also contains navigation to “post a new book page”, “all book posted page”, “sign in & sign up pages” for users that weren't signed in, and “sign out” for users that signed in.
* Search feature that if keyword matches with part of the “bookname” attribute of Post model (table that stores textbook post in our database), it will return those results. If no result is found, redirect to the home page. This feature is embedded into every .html that uses our layout template (e.g. homepage, listing page, post page).
* Login page that takes in the username & password from the user and validates with the “account” table in the database. If both match with any user, it will redirect to the homepage, else it will flash the message on why login was failed in the login page.
* Post page where users can post the information of their textbook to the website. The page will validate the user inputted data by making sure that the book title and author input field will not only contain numbers and the price input field will only contain numbers or the dot(.) for floating point numbers. The submitted data will be stored in the database for later use (display them in the booklist and homepage).
* Booklist page where all user posted textbooks are displayed by post time by default. The page has buttons for changing the sorting from post time to alphabetically and for filtering by FSU colleges.
* A page for registering an account. This page takes some basic information from the user such as username, password, email, name, and fsuid. If everything is valid then the information is entered into the database, if something is missing or isn’t correct then a flash message is displayed on the screen and the user is given the opportunity to finish filling in any missing or incorrect information.

1. **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

The biggest challenge of this project is to distinguish the difference between implementing the server side code and client side code. Since we were all new to web development, when we first started on this project, we did not know the advantages of having both server side and client side. At first, we planned to do everything on the server side using Python flask, which may result in some problems. Since we did not know exactly the concept behind it, we went to the Professor’s Zoom office hour as well as emailed the Professor to ask for some clarifications. We figured out that doing everything on the server side is possible but may be very tedious in terms of data management and transfer. By separating the implementation of the server side and client side, we can limit the number of round-trip server accesses and make our application more responsive. Due to the reason, we changed our plan of implementing everything on the server side and decided to render the HTML templates on clients machine and interact with the database with Python back-end.

Another problem I (Yuki) have encountered personally when implementing the html templates of our website is that the templates would not display correctly when we try to render the templates using Python flask. Based on Professor’s suggestions, we guessed that it might be because of flask’s render\_template method did not render the css file correctly, which can be fixed by having both server side and client side to use jsonify to pass information from the server to the client side and let the client side do the rendering. Later Zack figured out that part of this problem is caused by flask automatically loading the old css file, which will ignore the most recent css file. This can be fixed by changing the file name manually each time we change the css file or press Cmd+Opt+R to reload the page if we are using Safari on Mac. Due to this, we are able to continue using render\_template to test our application during the first increment and officially begin separating the server side and client side in the second increment.

I also had a hard time figuring out what to use to implement the client side code. The Professor suggested three most popular JS/HTML frameworks: React, Angular, AngularJS. At first, I was planning to use React, but I found out that the learning curve is steep, and I figured that I was not able to modify the code to work with React. As a result, we decided to use AngularJS instead because the Professor told us that AngularJS is very similar to what we are doing right row.

--- For me (Zack), the following are challenges that I’ve encountered during phase 1 ---

1. I wasn’t familiar with web development at all prior to this semester. Although my python class has taught me a small part of the database and web development, I struggled with many of these new things (e.g. SQLAlchemy, flask, the concept of database, concept of client-side & server-side, concept of API, etc.). However, Professor Mills has provided me great advice, pointed me in the right direction, and after reading several tutorials & watched several tutorial videos, my confusion is mostly resolved.
2. SQLAlchemy is quite difficult at the beginning, where I’ve struggled with its syntax and different combinations of attributes, which resulted in many bugs. However, after using the python terminal to test out the various way of using this ORM, I’ve discovered that this is extremely useful, and can be easily implemented.
3. Another difficulty that we’ve encountered with SQLAlchemy is getting it’s attributes. Since it returns a model object, it’s hard for me to pass it into the .html file to be displayed. However, after reading the references online for SQLAlchemy, I figured out that there’s a \_\_self\_\_ attribute that I can set up so when I call the object, it returns a list of tuple, which I can pass it directly into the .html, which made things a lot easier comparing to looping and store individual attributes one by one to form a tuple.

1. **Team Member Contribution for this increment**

For contributions on IT, RD, and progress reports, in each document we differentiated what we’ve contributed by using different color words.

General Contributions - For the planning & distribution of source code works, Zack & Yuki handled most of this part by creating issues with GitHub and talking to members in GroupMe.

Source Code:

* For the route controllers in app.py, everything has been grouped into comment sections with the contributors’ names on the top of each section.
  + Use filter (is:issue label:"Itr 1: Src Code" ) to see the issues of our source code.
* For other source code contributions, please refer to our Github issues with the label “Source Code”, where the contributors of each issue are its assignees.

Video Contribution (Based on 5 parts assigned):

1. Yuanyuan Bao
2. Zhixi Lin (Zack)
3. Hanyan Zhang (Yuki)
4. Dennis Majano
5. Wesley White
6. **Plans for the next increment**

For the next increment, we plan to keep using SQLite3 for our database software and flask for our server-side code, we are going to begin using AngularJS for our client-side code to resolve any rendering issues that we currently face using flask. We are also planning on using Axios to link both the client-side and server-side code, then finally we plan on implementing any other features and patching any issues that we currently have in our issues tab on Github to make sure our application is working correctly.

We will also implement new features for our web app. We will use Javascript to implement the sorting textbooks alphabetically and filtering by colleges available in fsu. We will formalize the use of foreign key and primary key as relation between the Account table and Post table. After this is implemented, we will be able to allow users to delete their previous posts once the transaction or trade is complete. Once the book is deleted, it will not show on our webpage.

In addition, we will also try to improve and integrate the design elements of all the web pages, making sure they all look with the same style by implementing more layout templates.

1. **Link to video**

[*https://www.youtube.com/watch?v=RidGyEy9m1o&feature=youtu.be*](https://www.youtube.com/watch?v=RidGyEy9m1o&feature=youtu.be)