Progress Report

- Increment 2 -

Group #5

*We basically just changed our web app from doing everything on the server side to separating client and server side in this increment, so some of the information in the three documents will be very similar to the last increment.

1) **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

2) **Project Title and Description**

   FSU Book Trading Site

   Textbooks are required for most courses and are generally considered expensive for students. Students at FSU need to spend hundreds of dollars per semester on these textbooks. But after only one semester many of them will be left on the bookshelf. If there was a way for students to trade their old textbooks for their old courses and get used textbooks for their new courses it would help students out. This web application is implemented to solve these problems for FSU students. 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. 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 using this web application at a lower price than elsewhere. This website will help students save money and reuse resources, which is also good for the environment.

3) **Accomplishments and overall project status during this increment**

   Overall, for this iteration we've successfully separated our client side (Angular) and server side (Flask). The program is functional with the user interface part done with Angular and data processing on the server side using Flask.

   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). Token-based authentication is used to keep track of whether a user is logged in and which user is logged in.

   Homepage: Flask is used for accessing the database and filtering out the 12 most recently posted books. After this, the data is converted to a json blob using jsonify, which is sent to Angular. Angular shows the design of the page and gets the jsonified data from flask and displays the textbooks in the recently posted books section.

   Post page: This is where users can post the information of their textbook to the website on the Angular side. The data will be sent to Flask, and Flask 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 (a Post system) for later use (display them in the booklist and homepage). A message will be used to track whether the data is successfully sent to the database. It will be set to different alert messages if something wrong happens and a success message otherwise. Then the variable will be jsonified and sent to Angular, where the message will be displayed for the users (post successfully or not). If successful, Angular will redirect the users to homepage. The users are not allowed to access this page if they are not logged in.

Booklist page: This is where all user posted textbooks are displayed by post time by default. The page is able to change the sorting method from post time to alphabetically or price and for filtering by FSU colleges. Just like the homepage, books are displayed in appropriate sections. A page navigation is also added where 16 books per page and allows page change whenever the users click on it.

Book Detail Page: Users will be able to click into the textbooks from the booklist, general profile, and index page. All information related to this book will be displayed in the book detail page.

General Profile Page: This will be displayed when the users click on the name of the person who posted the book in the book detail page. It shows that information about that person and the 8 most recent posted books he/she posted.

Login Page: This is where registered users can login to our website. After the users enter username and password on the Angular side, the inputs will be sent to the server side, and Flask will check if username exists and if username and password matches. Alert messages will be sent back to Angular if the wrong username and/or password is entered. If username and password are both correct, the user will be logged in and redirected to the homepage.

Register Page: This is where users are able to register to our website and they will be logged in automatically and redirected to the homepage if they successfully registers. The user API is done using Angular and validating the input fields and generating a token for login is done in the Flask backend. Appropriate alert messages or a success message will be sent to Angular after validation of the user input.

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).

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

The most significant change we did for the increment is that we decided to use Angular instead of AngularJS and axios for our client side. We made this change because we feel that Angular will be used more in our future career and it fits better with our project. Also, compared to AngularJS, Angular is more organized with different components, and we can do http requests internally (so no need to use axios).

One of the biggest challenges we face during this increment is to send book and user data from Flask to Angular. The two main problems of this are how to convert data from SqlAlchemy database to json blob and how to send this to the client side and display correct information on the page. Zack and I were stuck on the send data part for a few days, so we went to Professor Mills’ office hour twice to ask for help. Professor Mills sent us a sample code that is similar to our project for us to get some references. We tried to implement our code using the similar methods in the sample code, but it didn’t work at first. The sample code works when I try to run it. Due to this, I tried to add our project code

files one by one into the sample code project folder to figure out what went wrong, and it worked when I did this. Later Zack figured out that it was because we are using a newer version of Angular, which is not compatible with the methods in the sample code. After this issue is solved, we are able to send data from Flask to Angular.

Another minor challenge I personally faced is finding a way to database data to json blob so that I can send it to Angular. When I first tried to jsonify the data, Flask alerts me that the type of list is not json serializable. I searched online to find ways to solve this issue and found out that the marshmallow library can be used to fix this. I installed the package and attempted to use it, but python keeps saying that the package is not installed. I have to give up using marshmallow because of this. Later I found out that the dataclass from the dataclasses library can be used for this and is very simple. We just have to make sure the format of the blob is correct before we jsonify it.

Zhixi Lin (Zack)
One of the major, and the general challenge that I faced is to learn Angular. Angular is extremely messy in my opinion, as there are no back-compatibility for versions, which resulted in us being stuck for many days on minor issues due to the fact that we watched tutorials for Angular 2, 4, 7, etc. At least around 10 hours of our time are wasted watching/reading the incompatible tutorials/instructions. However, we eventually resolved it by setting our Angular version to 6, and only focused on that version's instruction (however often it's extremely hard to find the specific version or there isn't a version showing on the instruction).

Another challenge that me, Yuki, and Yuanyuan faced are trying to send data from flask to the server. At the beginning, when Yuki took over the Login Component, she was unable to establish a post connection with Flask server due to the CORS header problem. However, after hours of investigation, and a 3 hours long brainstorm between us 3, we figured out the format and syntax of the data weren't compatible, resulting in Flask reporting error, thus blocking the access.

The last challenge that I faced was converting from session(cookie)-based authentication to token-based authentication. There was little instruction online about this, and the time was short for me (4 days) to implement this as our assigned team members failed to complete this on time and there was little progress. It was very complicated to start with, however after approximately 15 hours of struggling, I was able to implement a complete token-based authentication for our web-app.

In general, it's very hard to find relevant instructions for what we're doing since there isn't a great official instruction that will perfectly suit our situation, however we managed to migrate everything over and done with front-end back-end separation.

One thing that went wrong during this increment is that when I pushed my code for the Login Component of the Login page, I accidentally merged the main branch with pieces of an outdated branch, overriding some of the previously deleted/overwritten files. Thankfully Zack and Yuki had a backup of a previous iteration of the branch to minimize any damage done to the overall project, and to be able to keep working without any more issues.

5) 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 back-end server app.py, everything has been grouped into comment sections with the contributors' names on the top of each section.
- Use filter (is:issue label:"Itr2") to see the issues that divided our works in this iteration.
- For the front-end Angular 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:
a. Zhixi Lin (Zack)
b. Yuanyuan Bao
c. Dennis Majano
d.Hanyan Zhang (Yuki)
e. Wesley White

6) Plans for the next increment

For the next increment, we are going to design minor pages (About us page and Commonly asked Q&A) and implement the components for these.

We are also going to implement the remaining function in the login page, including Forgot password and keep user login for 30 days.

We will implement a personal profile page which allows users to view their personal information and change their password. This personal profile page will only be available after the user is logged in and a user will not be able to see other users' personal profile page.

We will allow users to upload their personal profile pictures when they register to our website.

We will make the booklist page(also any other pages that display books) display an actual picture of the textbooks that the users uploaded instead of a temporary picture.

We will add filtering by status of the books in our booklist page.

We will try to fix any existing bugs.

We will improve our website design to make it look more professional.

We will clear up the app.py, making it only serve the purpose of data processing, delete all the existing html/css files.

Further integrate front server and back server, making them more efficient.

If Time Permits:

- We will try to implement a chat box feature or email notification feature.
- We'll start to use hashing to encrypt the password so the pain text will never be stored, for a higher security.
- Implement a component that allows users to post buy orders for books that they wish to purchase for a desired price.

7) Link to video

https://www.youtube.com/watch?v=ZGHQ_f405hw