Progress Report

- Increment 3 -

Group #5

*The documents were written a few days before the due date, and we have changed some minor parts of our web app after that.

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 having 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 added new features to our web app and improved function of existing features.

Accomplishment During Itr3:

- Bug fixes: Fixed the displaying issue on the homepage, fixed booklist display issue, fixed buy order list display issue, etc. (Almost all the detail for bug fixes are under the issues under label itr3)
- Book detail page: We've added features where if the post belongs to a user, the user would be able to modify the post or even delete it.
- Login page: Added the feature to keep user login for 30 days as long as the back-end is still running during the 30 days session.
- Personal Profile Page: This is where the information of the currently logged in user will display. It allows the user to change password, view all posted books(sell orders), and view/delete the buy orders they posted.
- 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 recently posted books he/she posted.

- Post buy order page: This is where users can post the textbook's information that the user wants to buy or trade. By clicking the submit button it will display validation messages. Inadition, only the user who has been registered can post information.
- Buy order list page: This will display all users interested in buying textbook's information posted by the Post buy order page. Also, It is able to change the sort method from post time, alphabetically, and price; filtering by colleges, or status.
- Picture upload feature:
    - Allowing users to upload avatars during registration (Dennis), and implemented functionality for users to change their own avatar on their personal page(Zack).
    - Implemented functionalities to allow users to post books with a picture (Yuki).
- Fully migrated all features from back-end to front-end, clear up the files in the back-end (flask).
- Implemented About us, Q&A, and contact us page.

Accomplishment Overall:

- 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 or status. 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.
- 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

Please describe here in detail:
- anything that was challenging during this increment and how you dealt with the challenges
- any changes that occurred in the initial plan you had for the project or its scope. Describe the reasons for the changes.
- anything that went wrong during this increment

One major challenge that I faced during this iteration is to do conditional check to pages that display different things to different users (i.e. the book detail page). What I'm trying to accomplish is making the post editable if the post belongs to the currently logged in user, and make the text themselves editable for the user so the overall design will look good. After a couple days of struggling I was able to accomplish this through using angular conditions to compare currently logged in users with the owner of post at html directly. Then, to make the editing of posts look more smooth, I basically made the texts themselves editable, so there wouldn't be any extra prompts for input fields that make the page look bad. Additionally, to ensure when the edit is performed by the correct user within the session of the token, I've implemented a test for the token on the backend as the user submits the edit request, where the token is authenticated at the backend.
One unexpected problem that occurred during this iteration is the development of email components for our web app. I(Zack)'ve spent 2 days investigating and struggling with it's implementation, however nothing functional could be implemented. As we're short on time and some of our team failed to finish their work or struggle very hard on work that was assigned to them, me and Yuki have to move our attention to their work to ensure the functionality of our website, therefore the email feature is dropped.

One minor challenge I faced during this increment is finding a way to display both the booklist and the buy order list in the personal profile page. The part that I struggled on is trying to make the page not messy and user friendly. Firstly I tried to let the page display two sections, one for booklist and one for buy order list. I did not do this because it takes a lot of space, and I feel that it is not necessary to show everything on the page simultaneously. In the end, I decided to use a button to control the display of different sections. The page will display the booklist section by default and change back and forth with the booklist and buy order list as the users click on the button. Another challenge is finding a way to upload book pictures and profile pictures. Since this is pretty difficult, we all attempted to implement it. I found it very difficult to save the picture user uploaded to the assets/images directory and give every picture an unique name to save into the backend database. By the time this document is written, this issue is still not solved, and we might discard this feature completely.

One challenge that I faced during this increment was trying to upload a profile picture from the register page to the local computer. I found a lot of tutorials on how to store pictures in a database as a long binary, however none of them were useful since our database was set up to take a string to represent the location of a picture; if I were to change the structure of the database to store pictures as long binaries it would also mess up a lot of other areas of our code.

5) Team Member Contribution for this increment

Video Contribution:

a. Wesley White
b. Zhixi Lin (Zack)
c. Yuanyuan Bao (Script 320 words), Dennis (recording of this part)
d. Hanyan Zhang (Yuki)

Zack & Yuki is also responsible for reviewing all the member's scripts and providing necessary modifications individually.

6) Plans for the next increment

This is the final increment for our project.

7) Link to video

https://youtu.be/AJgfdW-BC3U

*The video was recorded on Thursday, Apr. 22, and the website demo is not our final version. We have changed minor things and added additional features Thursday night and Friday.