Overview:
The goal of the project is to create a website that five college students can use to buy, sell and give away items. It would aim to serve the Amherst community specifically, for example, selling books for classes that are specific to these colleges. Specifically, our application strives to provide a systematic ecommerce-like environment where students can easily login, search for items, flag items, view profiles, etc.

Team Members:
- Vedant Puri
- Shubham Mehta
- Sagar Thapar
- Zachary Kiihne
- Yifan Zhu

Github Repository: https://github.com/vedantpuri/ffs-web

Design Overview: Our login is based on the standard username and password verification, wherein a user must provide these credentials in order to become an authenticated user. Overall, our application provides the following: a homepage, search functionality, product upload functionality, flagged items view page, profile view page and edit profile functionality. We allow any user, either logged in or not logged in, to access the homepage and search for items. However, the other pages and functionalities are only accessible to logged in users. Consequently, we have adjusted the views such that users who have not yet logged in only see a 'Login' button and likewise, logged in users only see a 'Logout' button. Lastly, after logging out, the user is redirected back to the login page.

Problems/Successes:
Problems:
- A few dependencies remained among tasks assigned to different individuals. In the future, we wish to reduce this so that tasks can run in complete parallelism
- Difficulty in dynamically refreshing the view flagged items page (with updated flag counts) as soon as a user flags an item.

Successes:
- Redirection of logout directly to login
- Application restriction for unauthenticated users
- Profile edit and product upload functionality

Team Choice:
We have decided that each team member will implement a team choice component, hence there will be 5 team choice features and we will select the one which is most adequately implemented and suits our purpose:
- User registration: This will be a feature so that new users can register on our service on their own. This will be a form with several fields corresponding to the details we need for a user profile.
- View Seller: We will implement the view seller button so that a user can view the seller from the product description.
- Password reset: This is a feature so that users can reset their password if they forget it.
- Flag/UnFlag Item: This is a feature through which a user will be able to add a product to his flagged list by clicking on the flag item button and remove it using the unflag item button based on his preference.
- Search: This is a feature so that the user can search for certain products and obtain results based on keyword matches.
- Star rating for users: This is a feature so that users can rate a product based on satisfaction level, so that other users are able to make a more informed decision before a buying a product from a particular seller.

Instructions to Run Application:

- Please note that we experienced difficulties due to browser caching, which is why we suggest that our application be run in private browsing mode.
- Be sure to run the following command before running our application(in your django environment):

  pip3 install django-widget-tweaks
- Navigate to ffs-web/CS326TeamH and run **python3 manage.py runserver**
- We have created a user for you to preview. The name of this user is **Samuel Jackson** and the credentials are mentioned below:
  - ***Username:*** CS326TeamH
  - ***Password:*** ffsweb12345

Individual Writeups

Format:
Member name: Contribution %
Contribution explained in detail [bullets]

Vedant Puri: 22.5%

- Create the back-end mechanism for the Upload Product form.**[forms.py, views.py]**
- Create the back-end mechanism for the Edit Profile form.**[forms.py, views.py]**
- Collaborated to stylize the above two forms.
- Solve form resubmission bug and product image upload bug.
- Enable permissions for each page, preventing unauthorized users access to certain pages.
- Make login/logout buttons dynamic according to authentication.
- Create Logout mechanism and redirection back to login page.
- Improved style of the login page.
- Collaborated to map custom user to django user, and access custom user after authentication.
- Fix a few styling issues from the previous submission.**[Re-styling the login page]**
- Monitor the team's progress, help solve bugs, remove any redundancy.

Yifan Zhu: 22.5%

- Collaborated to implement Upload and Edit Profile form
- Collaborated to stylize the above two forms
- Extend base to landing page
- Add Edit Profile button and redirect it to edit profile page
- Implement search function of landing page and navigation bar
- Implement page redirection and success alert after uploading an item
- Implement pop up confirmation of uploading page
- Fix styling issues of footer

Shubham Mehta: 22.5%

- Setup User Groups and Custom Users
- Worked on mapping from Custom User to Django User
- Added login functionality to our application and its URL mapping
- Added logout functionality to our application and linked it to the logout button
- Enabled redirection of logout directly to login page
- Fixed minor styling errors
- Assisted teammate with debugging and model explanation.
- Studied extra materials on authentication in an effort to add to our team's knowledge-base
- Completed the 1-page write-up

Sagar Thapar: 22.5%

- Created add flagged view which allows items to be flagged.**[branch: improve-flagged]**
- Detected issues in the flagged items model.
- Created a form on the profile view to add flag item functionality.
- Worked on fixing issues on a separate branch named “improve-flagged”.
- Added forms to allow items to be flagged from the search results.
- Added urls so that the user gets redirected to the flagged items page after flagging an item.
- These changes are not included in this submission because it requires more testing.

Zachary Kiihne: 10%

- Changed user model to include django’s user model
- Used django’s user model to create a Custom User model with additional details while maintaining the functionality of the inherent user model
- Eliminate redundancy from user model
- Changed database to incorporate new user model
- Created separate branch to experiment with user model before changing the master
- Helped team members to understand how to use the user model
- Created write up