Flask Login Open Source Report

General Information and Licensing:

- Code repository
  - https://github.com/maxcountryman/flask-login.git
- License Type
  - MIT License
- License Description
  - fairly permissive license requiring preservation of copyright and license notices but otherwise allowing usage and distribution without any limitations
- License Restrictions
  - does not provide any warranty or liability
- Who Worked With This
  - Ahsan

LoginManager (class)

- Purpose
  - This library allows for behind the scenes session management of users by storing their session information through cookies (for authenticated users). It allows us to make the home page of our application private by requiring that users are logged in to access it. This library allows to make certain routes on the server only accessible if a user is currently logged in
  - *Please note that the below file and line numbers are NOT on the "main" branch yet, but instead on the "ahsan" branch*
    - flask_server.py – lines 13 to 16
      - this is where the LoginManager class is instantiated which will be used to keep track of session management
    - flask_server.py – line 26
      - this is where the library makes the "/" route only accessible if a user is authenticated
    - flask_server.py – lines 50 to 69
      - this is where more information about the current user is stored, with methods that LoginManager uses to verify if a certain user is

authenticated, if they are active, and their username given their session information

- flask_server.py – lines 72 to 73
  - this is a callback function that the library uses to return a User object which is used by LoginManager, with the username of that user attached

Magic

- The way that this library works when you look at the bigger picture is by using a User class and session cookies. When a user first registers through the HTML form on the /register route, the *load_user* function is called with their username, which creates a User object (as described in the LoginManager section above).
- Then, the *login_user* function is called with the aforementioned User object that was just created. This function is located in the *utils.py* file at lines 169 to 217.
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L169
  - the second parameter of this function is a Boolean called *remember*, which when set to True, will remember the user after their session expires
  - the third parameter of this function is a *datetime.delta* object called *duration*, which specifies the amount of time before the authenticated session cookie expires
  - the next step of this function is to create a *SessionMixin* object which is part of the *request* library
    - https://github.com/psf/requests/blob/main/requests/sessions.py
  - this object has a dictionary mapping called *session*, and if the *remember* Boolean was set to true, it sets the *"_remember"* and *"_remember_seconds"* keys to *'set'* and *duration* respectively (default is 365 calendar days from current date). The *"_id"* field is where the unique authenticated value for the cookie is actually set, with its value being the sha512 hash of the *User-Agent* HTTP header and the IP address of the user
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L403
  - the session cookie name and properties such as Secure or HTTP only for the *remember_token* cookie (authenticated session cookie) is set in the *config.py* file and is sent to the user in line 188 of *utils.py*

        - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/config.py#L4
        - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L216
- After the *remember me* cookie is created, the next step is for the any route that uses the *@flask_login.login_required* annotation to check the value of the incoming requests' *remember_token* cookie to match the value stored (which was the sha512 hash of the *User-Agent* HTTP header and the IP address of the user). Note that this is only done when the *User* object cannot be authenticated through the *is_authenticated* method. The method that does this is the *_load_user_from_remember_cookie* which is located here:
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/login_manager.py#L404
        - This in turn calls the *decode_cookie* method located in *utils.py*, which decodes the cookie using the app's secret key that is initially set when creating the *LoginManager* object and used to encode the *remember_token* cookie.
            - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/login_manager.py#L470
            - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L63
                - This returns the associated *User* object based on the *_user_id*
- If a user logs out (yet to be implemented in our current project) the *utils.py* file calls the *logout_user* function, which clears all of the cookie information that was set when the user initially authenticated and logged in using their password (the whole process described above)
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L220