Assignment Two Report

In this assignment, I implemented a sign up page, a login page, and logout functionality on the base of my first data analysis assignment. For the sign up page, if all the fields are valid, the system saves a new record to the database and starts a new session. For the login page, if all the fields are valid, the system checks if the username and password match the ones in the database, and if so, starts a new session. The logout functionality pops the session. This authentication system I built lets people register and identify them who use the site. While it works, it has two major flaws.

For the first one, if people log out of the app completely, but when he/she types '/home' into the address bar, that person can still access this page. The problem here is that this web app currently allowing anyone to access pages that require login. A safe application should only let users that are logged in to access this protected page. If a user is not logged in, he/she should be redirected to that login page and enter his credentials in that form.

This problem caused by the Python Flask routing principle. In my code, I used web templates to create a homepage, then in order to see the homepage in the browser. I need to create a template, indexed at html, and I placed it inside the templates folder. For seeing this page in the browser, I should to map a URL to it, and I do this in routes.py.  First as showing in the Figure 1, import the Flask class, and the function render\_template. Then create a new usable instance of the Flask class and save it into the variable app. Third, map the URL "/" to the Python function index. The Python function uses the Flask function render\_template to render index.html. So now when a user types in the URL "/", the function index will run and return the page index.html.

Figure 1. A Segment Code of Assignment Two

图片包含 屏幕截图



自动生成的说明

For the second flaw, if people log into this app successfully. He/she can get redirected to /home, as expected. The problem here is that now when that person go back to the sign up page, He/she can still see this form. The same is true for the login form. People can still see it. This web app is currently allowing signed in users to see the authentication forms. This doesn't make sense, either. The right way is only for users that are logged out to see these forms. If a user is logged in, he should be forwarded to the home page.

To fix the first flaw, I need to determine whether a user is logged in. For the next assignment, I will create a signup function in routes.py. I can set Session email equals newuser.email. This means, to log in a user, I created a new session. Similar to a Python dictionary, I associated the key email with the user's email. Therefore I can use this session object to determine if a user is logged in. If a user is logged in that key email should exist in this object. On the other hand, if a user is not logged in, the key email wouldn't have been set. It wouldn't exist in that object. So I can check whether a user is logged.

And the second flaw can be removed by the same way I did for the first flaw. If a user is logged in, the key email should exist in the session object. Therefore I can check this with an if statement. If email in session, return redirect(url\_for('home')). Same thing if a user goes to login. He/she get redirected to home. So now the app is protected.