Project 3

Due Saturday by 11:59pm **Points** 100 **Available** after Oct 25 at 12:01am

Overview

In this project, you will get experience working with AJAJ and JSON by creating a website to host and manage multiple chat rooms.

Required Packages

- Google Chrome - All grading will be done in Google Chrome, so make sure your chat app works in Google Chrome.
- Python 3.x
- Flask
- Jinja2 Templating engine (included with Flask)
- This project is to be done using vanilla JavaScript with the standard library. This means that you cannot use external libraries like jQuery.
- You can represent all server-side data as Python data structures, you **DO NOT** need to use SQLAlchemy. This means your project's data does not need to survive restarting the app.

GitHub Classroom Link

Please click on the following link to get your project repository.
https://classroom.github.com/a/4aCYceJP (https://classroom.github.com/a/4aCYceJP)

Specification

Project 1 and Project 2 focused on the server-side and client-side respectively. Because this, I broke the project specifications into the various parts of what makes up a server-side and client-side application. Project 3 is an application that will have a dynamic server-side and client-side and will contain parts of the server-side, controller and views, the client-side, UI elements and Events, with AJAJ and polling bridging the two.

Below is a list of requirements. Break these requirements up into the server-side and client-side parts as you see fit in order to satisfy the requirements.

1. When visiting the page for the first time, users should be given the chance to create an account or login.
    - For this project, you do not need to hash the password before storing it in your Python data structure. However, you will need to verify no duplicate usernames. Any failed login or account

creation should present an error message to the user.
2. Once successfully logged in, the user should be given a list of possible chat rooms to join, or a message stating that none currently exist.
    - The user should also have the option to create a new chat room.
3. Once in a chat room, the user should be shown the history of messages for that chat room, as well as be kept up to date as messages are sent to the chat room.
    - The user should also have the option to post a new message to the chat room.
    - The user should further be given a way to leave the chat room.
    - Users can only be in one chat room at a time. This includes preventing the user from using a different tabs to enter additional chat rooms.
    - You must use AJAJ and JSON to update the list of messages posted to the chat room, and to post new messages to the chat room.
    - All AJAJ chat updates should send only *new* messages to the user. The user should not receive the full list of chat messages with every AJAJ update as this could grow quite large over time.
    - You must be sure that your application does not display "phantom" messages to the user.
        - I.e., All users in the same chat room should see the same messages in the same order and new messages should always appear at the end of the chat log, never in the middle of the chat log.
    - You should take a polling approach to ensure that new messages are always available to the user. Your application should have 1 second between polls.
4. Once a user leaves the chat room, they should again be shown a list of potential chat rooms to join (or a message if none exist).
    - The user should also have the option to delete any chat rooms that they created.
        - Any users still in a room when it is deleted should be shown a message informing them that the room was deleted and be again presented with the list of available chat rooms (or a message if none exist).
5. The user should always (on every page) be presented with away to log out while they are logged in.
6. Design and CSS is left up to you. The only design requirement is that the page is presented in a clear and readable manner.

Notes & Hints

- You may find the use of profiles (https://support.google.com/chrome/answer/2364824) helpful for testing multiple users logging into the chat site at the same time.
- You can use any of the examples from class for inspiration and you may use snippets from them in your code. However, **DO NOT** copy them in their entirety and edit them for this project

Submission Guidelines

- Place any comments you have for the grader to help them grade your assignment in the FOR_GRADER file
- Name the main (controller) file chat.py

- You must be able to run your application by setting the FLASK_APP environment variable to chat.py and running "flask run"
- Do not submit any IDE files, feel free to add them to the .gitignore file under "# Editor files"
- Be sure to remember to push the latest copy of your code back to your GitHub repository before the the assignment is due. GitHub Classroom will not stop you from pushing additional commits after the deadline but it will make the latest commit before the deadline the submission that the grader will grade. It will also notify us if previous work is edited after the deadline.