Eliana Villafranca

Computer Networking

App Overview:

This is a basic messenger with a global chat, and a login page which requires a username, and checks to make sure that the chosen username is not currently taken. The login page also includes a purely aesthetic “forgot password?” link, which leads to a blank page containing a joke image. The server does not include a database, and therefore does not save messages. A user loses messages as soon as they log out, and cannot retrieve them even if they log in with the same username.

The login page, “forgot password” page, and messenger page are all styled with CSS, and both the messenger page and login page include a delicious gradient background. Flask is used as the HTTP server host framework. Socket.io is used to enable real-time bidirectional event-based communication because a messenger app should be have quick responses. The server-side of the application is written using Python, and the client-side is written in Javascript.

HTML:

There are four HTML documents included in the project. The first one, index.html, includes a template block named “body” which is included in every other HTML page and specifies the stylesheet (which uses bootstrap), and imports jQuery and Socket.io. The other three HTML pages were used to construct the login, messenger, and “forgot password” page. In order to provide feedback to the user, the login page includes the code:

which alerts the user if they attempt to log in with a username that is already taken. In the HTML for the messages page, the line

```
<script src="{{url_for('static', filename='messages.js') }}">
```

Imports the static file messages.js, so that the javascript needed to run the client-side will be used when the page is active.

Client/Server Side Overview:

The client-side javascript includes several console.log testing messages that were used to ensure that the code was working properly.

```
const socket = io();
```

is used to create a default connection through a socket over the global namespace. There are also two variables created

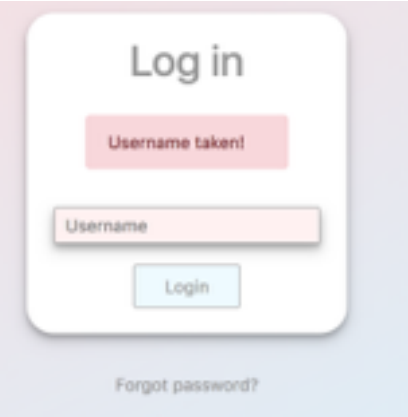
```
var active_users = [];

var user_name;
```

Which are used to hold the list of active users, and to hold the user name entered by the user. These are later used to ensure that each username is unique, and to display the usernames on the screen in the global chat.

There are three routes defined on the server-side. The route for messages and “forgot password” simply render the HTML for those pages, but the “login” route also includes instructions for checking whether or not the user’s username is already being used, by comparing it to the names inside of the list variable `users = []`. It also flashes a message if the username does match a currently used one.

Eliana Villafranca

Computer Networking



Login page displaying username error

Startup Instructions:

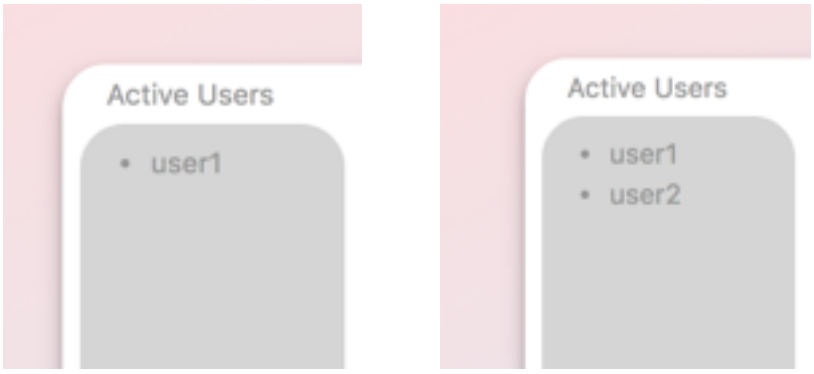
Assuming that the user has pip and python, the user must first create a virtual environment using `python -m venv env`. Once the environment is created, it must be activated with source `env/bin/activate`. Then, the user must navigate to the folder containing the application's app.py file. Then, the app can be run using `python app.py`, which will run the application with default settings, on localhost on port 5000.

App Usage

When a user first logs in, they type a username into the username bar on the login page. The server uses `emit` to send the username to the client. Then, the username is pushed into the list `active_users`, and the entire list of users is then re-added into the active users chat bar, to include the new user. It then appends the user's username to the list `users` and broadcasts the updated list to all of the clients using the server. Then, the list of users, including the new user, is updated for all of the other clients connected to the server.

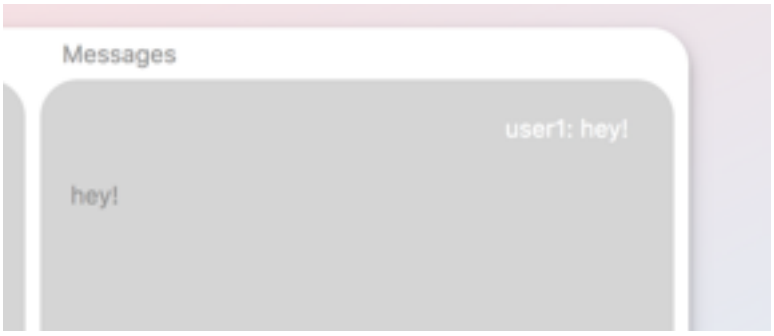
Eliana Villafranca

Computer Networking



Active user chat box before and after a second user has logged in

When a new message is sent, the server broadcasts it to all of the users in the global chat. If the message is sent by the user, the client ensures that it is placed on the left side of their screen and displayed in grey. If it has been sent by a different user, then the client displays it in white on the right side of the screen, and is preceded by the user's name and a colon. The message is appended to a list of messages, so that the messages can be scrolled through on the screen.



Message sent by a different user (top) compared to message sent by the user (bottom)

Eliana Villafranca

Computer Networking

The jQuery command `e.preventDefault();` keeps the message from being treated as a default HTML form. The default submission for an HTML form is to load a new page, so this command prevents the message page from changing in any way, aside from displaying the new message on the chat box. The method checks to see if the message is not empty, and then sends it to the server to handle it.

When a user logs out, the server emits a “disconnected” message to the client, which removes the user name from the `active_users` list, then re-adds the list to the active users chat box to exclude the user who has logged out. Then, the server removes the user’s username from the `users` list. Clicking the “Log Out” link returns the user to the login page.

Conclusion:

This project was very difficult for me because I had to learn how to use Javascript and jQuery, as well as HTML and CSS. As a result, I used both Flask and socket.io in an effort to make the coding for the project less complex, as a lot of the time spend on the project was spent understanding how to use the new programming languages. This project is very simple, and it is the first web application I have ever programmed, and also includes the first HTML pages I have ever coded. The experience of creating this messenger was very eye-opening and educational for me, and I believe I accomplished the task set before me, albeit very simply.