Project 2: Flack

Objectives

- Learn to use JavaScript to run code server-side.
- Become more comfortable with building web user interfaces.
- Gain experience with Socket.IO to communicate between clients and servers.

Overview

In this project, you'll build an online messaging service using Flask, similar in spirit to Slack. Users will be able to sign into your site with a display name, create channels (i.e. chatrooms) to communicate in, as well as see and join existing channels. Once a channel is selected, users will be able to send and receive messages with one another in real time. Finally, you'll add a personal touch to your chat application of your choosing!

Milestones

We recommend that you try to meet the following milestones:

- Complete the Display Name, Channel Creation, and Channel List steps.
- Complete the Messages View and Sending Messages steps.
- Complete the Remembering the Channel and Personal Touch steps.

Getting Started

Python and Flask

As with Project 1, make sure that you have a copy of Python 3.6 or higher installed on your machine. You'll also need to install `pip`. If you downloaded Python from Python's website, you likely already have `pip` installed (you can check by running `pip` in a terminal window). If you don't have it installed, be sure to install it before moving on!

To run this Flask application:

1. Download the `project2` distribution code from https://cdn.cs50.net/web/2020/x/projects/2/project2.zip and unzip it.
2. In a terminal window, navigate into your `project2` directory.

3. Run `pip3 install -r requirements.txt` in your terminal window to make sure that all of the necessary Python packages (Flask and Flask-SocketIO, for instance) are installed.
4. Set the environment variable `FLASK_APP` to be `application.py`. On a Mac or on Linux, the command to do this is `export FLASK_APP=application.py`. On Windows, the command is instead `set FLASK_APP=application.py`.
5. Run `flask run` to start up your Flask application.
6. If you navigate to the URL provided by `flask`, you should see the text `"Project 2: TODO"`!

Requirements

Alright, it's time to actually build your web application! Here are the requirements:

- Display Name: When a user visits your web application for the first time, they should be prompted to type in a display name that will eventually be associated with every message the user sends. If a user closes the page and returns to your app later, the display name should still be remembered.
- Channel Creation: Any user should be able to create a new channel, so long as its name doesn't conflict with the name of an existing channel.
- Channel List: Users should be able to see a list of all current channels, and selecting one should allow the user to view the channel. We leave it to you to decide how to display such a list.
- Messages View: Once a channel is selected, the user should see any messages that have already been sent in that channel, up to a maximum of 100 messages. Your app should only store the 100 most recent messages per channel in server-side memory.
- Sending Messages: Once in a channel, users should be able to send text messages to others the channel. When a user sends a message, their display name and the timestamp of the message should be associated with the message. All users in the channel should then see the new message (with display name and timestamp) appear on their channel page. Sending and receiving messages should NOT require reloading the page.
- Remembering the Channel: If a user is on a channel page, closes the web browser window, and goes back to your web application, your application should remember what channel the user was on previously and take the user back to that channel.
- Personal Touch: Add at least one additional feature to your chat application of your choosing! Feel free to be creative, but if you're looking for ideas, possibilities include: supporting deleting one's own messages, supporting use attachments (file uploads) as messages, or supporting private messaging between two users.
- In `README.md`, include a short writeup describing your project, what's contained in each file, and (optionally) any other additional information the staff should know about your project. Also, include a description of your personal touch and what you chose to add to the project.
- If you've added any Python packages that need to be installed in order to run your web application, be sure to add them to `requirements.txt`!

Beyond these requirements, the design, look, and feel of the website are up to you! You're also welcome to add additional features to your website, so long as you meet the requirements laid out in the above specification!

Hints

- You shouldn't need to use a database for this assignment. However, you should feel free to store any data you need in memory in your Flask application, as via using one or more global variables defined in `application.py`.
- You will likely find that local storage will prove helpful for storing data client-side that will be saved across browser sessions.

How to Submit

1. If you haven't already done so, visit this link, log in with your GitHub account, and click Authorize cs50. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click Join course.
2. If you've installed `submit50`, execute

   ```
   submit50 web50/projects/2020/x/2
   ```

   Otherwise, using Git, push your work to `https://github.com/me50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `web50/projects/2020/x/2`.
3. Record a 1- to 5-minute screencast in which you demonstrate your app's functionality and/or walk viewers through your code. Upload that video to YouTube (as unlisted or public, but not private) or somewhere else.
4. Submit this form.

You can then go to https://cs50.me/cs50w to view your progress!