Introduction

This document specifies a simple chat server which you're going to implement as the first step in the interview process with Expii. This is intended to showcase your technical competence with Python, HTML, CSS, and JavaScript.

We want a web application that allows users to chat with any other user currently connected to the server. The specification below can be seen as the minimum requirements. If you want to go above and beyond (in regard to features / design / etc.) feel free to do so as long as none of the basic specifications are compromised.

What we're looking for

We're looking to see an app that has been carefully architected and which has been built using best practices for modern web applications.

All code should be clean, concise, and readable. Extensive documentation is not required.

To better help us evaluate your work sample, please keep a rough estimate of how much time you spend working on the work sample, and also let us know what, if any, prior experience you had with the required technologies.

Specification

Requirements

Your chat server should roughly follow this design, however, feel free to make any UI/UX improvements above and beyond the design. Your application logic must follow these rules:

1. A user coming to the page for the first time should have just the logo and the bar at the top (no open chats). In order to register a username with the server, the user should input their name into the name box and press enter. It is fine to assume that all usernames will be unique.

2. User data *must not persist on the server*. The server should not have a database, and the server should not store any user data in memory (i.e. Don't do something like this `users= []`). However, you are free to use SocketIO rooms.

3. The client-side list of users should be continually updated as users join or leave the server. When a user selects another user in the dropdown, it opens a new conversation box with that user if one is not already open.

4. Each new chatbox should be placed adjacent to the previous chatbox or on a new row if it would not fit. The X button closes a chatbox.

5. A user can enter text in the input at the bottom of the chatbox and press enter to send it. A new message always appears at the bottom of the conversation (so if the box is full, then it appears at the bottom of the box).

6. All conversations should persist between sessions. For instance:
    - A user registers with the name “Bob”
    - Bob starts a conversation with Alice
    - Bob leaves the page
    - Upon coming back to the page, Bob re-registers as “Bob” and opens a conversation with Alice.
    - Both Alice and Bob should see all of their old messages with each other. (This data must also not be stored on the server)

Technologies

We require that you use the following technologies, as they mirror our own stack.

- React.js (for rendering the frontend, tutorial)
- Flask (for the server, tutorial)
- Flask-SocketIO (for client/server communication, tutorial)

If you are not familiar with any of these, feel free to read the accompanying tutorials.

If you’re new to Python, here’s a succinct guide to the language. You should use pip to install dependencies.

You can assume that your users are all using an up-to-date version of Chrome.

Handin

To turn in your project, email it to dev@expii.com with the subject “Candidate Work Sample: <your name>” and place all files inside a single file `worksample.tar/zip`/etc. Please also include two files: `requirements.txt` that contains pip dependencies (see: requirements.txt format) and `run.sh` which we will call to start your server.

If you have questions or clarifications, feel free to email Andrew or Rob at dev@expii.com.