Tyler Wright
Prof. Iyengar
CPE-490
18 December 2020

Simple Chat Server in Go

The chat server consists of a frontend written in HTML, CSS, and JavaScript and a backend written in Go. The backend is a simple Go routine which defines a Message object containing email, username, and the contents. There are two backend functions. The first is handleConnections, which converts the JSON of new messages into Message objects and forwards them to the broadcast channel where all messages are viewable. The second function, handleMessages, forwards all messages from the broadcast channel to each user such that everyone should be able to see the same messages. The backend was fairly straightforward to write, but the frontend turned out to be more difficult. The HTML and CSS were easy enough to write as it is just simple formatting and I am familiar with it, but the JS is doing a lot of heavy lifting as it is responsible for parsing message data and ensuring it appears correctly in-browser. Vue is used to make the interface dynamic, and it helps to convert message data into strings of HTML and to prevent any unexpected display errors. Finally, ngrok is used to make the local server publically accessible by simply forwarding data through a generated public URL.

The most difficult part of this project was definitely a debugging issue where the server was launched and accessible, but the messages would always appear as “invalid”. I had no idea where the problem was coming from, and I spent hours pouring over the code to make sure there wasn’t a missed typo or unimplemented feature it was expecting. After all this time, it turned out that I had mistaken a grave character for an apostrophe, where apostrophes are only used for the rune datatype in Go. The apostrophes were causing Go to skip over the definition of the Message object attributes, and once that was fixed, I was able to send messages smoothly.

If given the opportunity to continue this project, I have several ideas on how I would improve the final product. I had been interested in creating a chat server with multiple channels from the beginning and had done some research on the subject. It seems simple to implement in Go but would require an overhaul on the frontend, making it risky to attempt before the deadline. I also considered exploring a cybersecurity angle as I would be interested in making the chat end-to-end encrypted and seeing how easy a simple chat like this would be to exploit. Overall, this was a fun and challenging project where I learned a lot from the minor errors that I ran into, and I am excited to continue exploring these concepts in the future.