Reflection

John Wang

Product

I believe our final product turned out spectacularly well. We were able to implement all of the required features while adding some of our own as well. Since I handled the client side, I will talk about the GUI and different features that were added to the client.

First, we were able to create a prototype GUI which worked reasonably well. This was easy to create and did not have any dynamic features (it consisted of a menu with different buttons for starting and joining conversations). However, we decided to make a login screen, a main screen, and a help screen to differentiate between when a user was logged in or not. This caused a couple of problems, because at first we were using an intermediate login screen. This intermediate screen displayed some simple information, and waited for the server to respond to a connection.

This intermediate screen ran into trouble, however, because initializing and constructing the intermediate screen usually took longer than the server’s response. This would break the program frequently. To solve this problem, we decided to use a CardLayout and initialize all the panels of the GUI at the beginning of the launch. Then, we would only make the appropriate panel visible depending on the user’s status.

It was unexpectedly hard to get the GUI to work correctly because many of the commands send from the server had race conditions. Thus, we had to synchronize parts of the GUI so that only it would only enter a certain state given a certain type of previous state. This happened quite frequently when error commands from the server occurred, because the GUI would be unsure as to whether to logout or stay on the main screen. Eventually, we created specific error messages so the GUI could differentiate these different cases.

If I were designing the chat system again, at least from the GUI side, I would spend a lot more time planning out what buttons and features I would add to the GUI, and how I would change frames on the GUI. Since Swing is not thread safe, I would also think a lot harder about how to receive commands from the server.

Team

I believe our team did an excellent job in sharing the workload. Akash Badshah implemented the server, Jake Bograd-Denton created the tests and the stubs, and I worked on the client. Our team met a couple of times during the beginning of the project to create a client-server grammar. This helped immensely because we could then work on the client and server side completely separately. In fact, Akash spent a week away at a conference, and we were still able to finish the product with time to spare because of how well we separated our tasks.

As a team, I believe we worked together extremely well. Each one of us had a specific role which was suited to our strengths. The different roles meant that there was almost no bickering on the team, and we were able to complete the project in an outstanding manner. If I could, I would work with this exact team again in a corporate environment.

Individual

I believe that I personally did a good job developing the client side. Although I probably should have spent more time designing the GUI beforehand, I ultimately created something that worked and was relatively pleasing to the eye. I created almost the entire GUI, which I believe was a substantial and rewarding task. Moreover, I went through a couple of iterations in the overall look and feel of the GUI.

One can notice that during our demonstration meeting, the GUI looked radically different than it does now. This was due to two or three large scale changes to the GUI. In addition, I added the extra emoticon feature which was very satisfying.