Socket Project Document for Milestone 1

Command format

For each command that a peer sends to the server, a specific format was made to be both simple and organized. A menu is presented to the client with a list of options they can choose, these being the commands. Once they select a command, they may be question further for input. An example would be when the “register” option is chosen, the user would then be asked to enter a username, an IP, and a port.

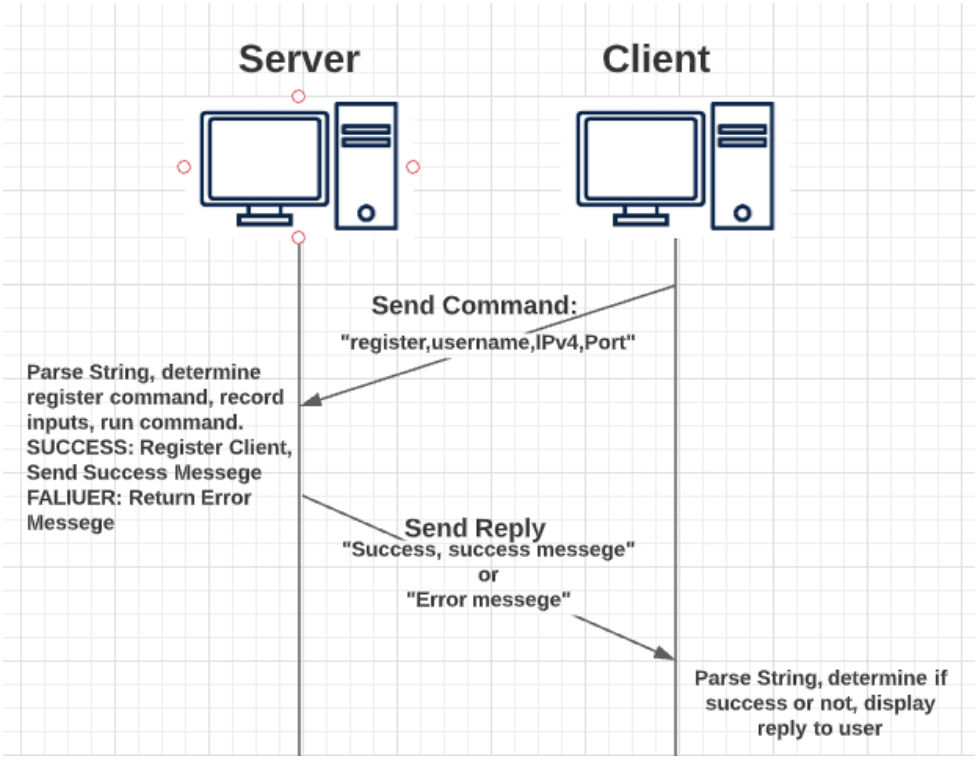
From there, once a command ahs been selected and additional input has been received, the peer service will then construct a string of the command to then send to the server.

EX: “register,Bob98,127.0.0.1,14002”

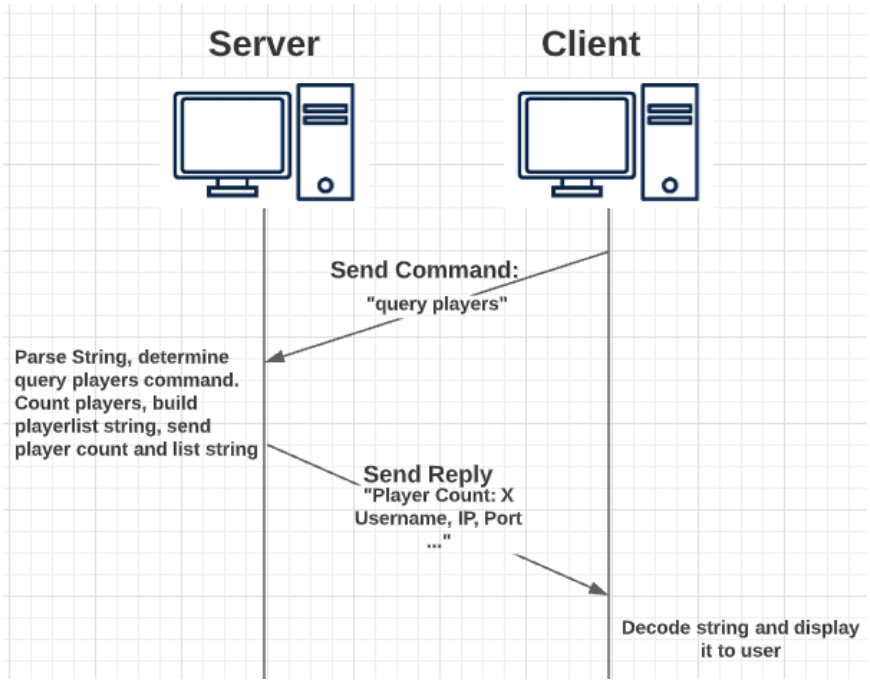
Once the server receives the message and decodes it, it then splits the string at all commas. In this, the server will be able to determine what command was sent to it and what additional info needed was given to it.

Time-Space Diagram

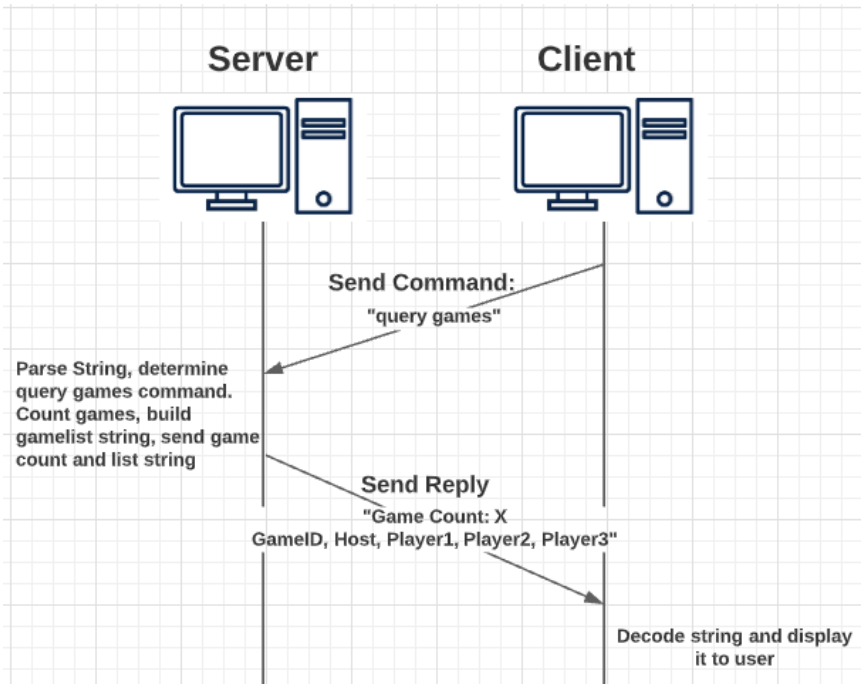
Register:



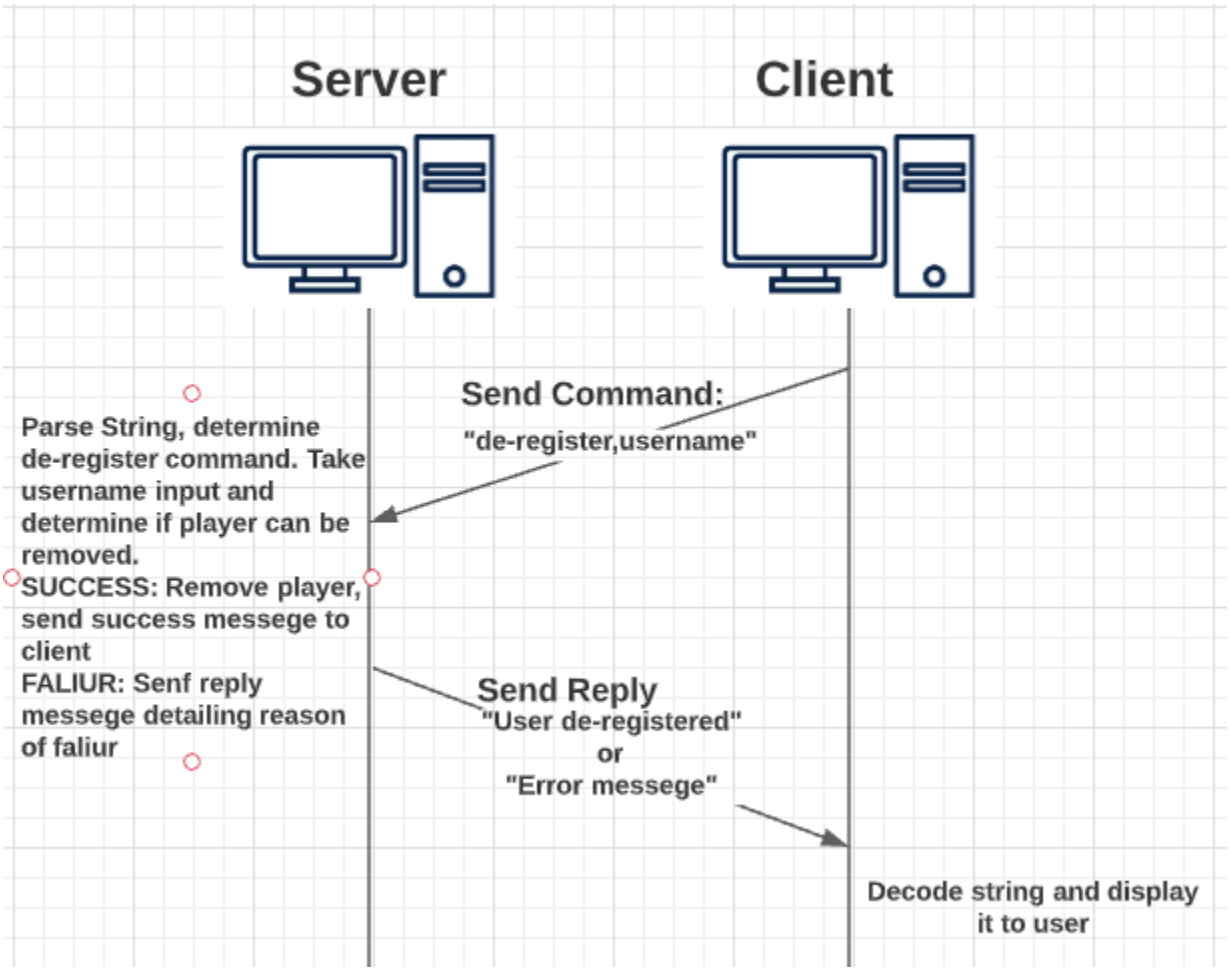
Query Players:



Query Games:



De-Register:



Data Structures and Algorithms

So far, for the most part things have been kept relatively simple. The peers and manager both user while statements to continue taking in commands and use if/else statements to determine steps to take.

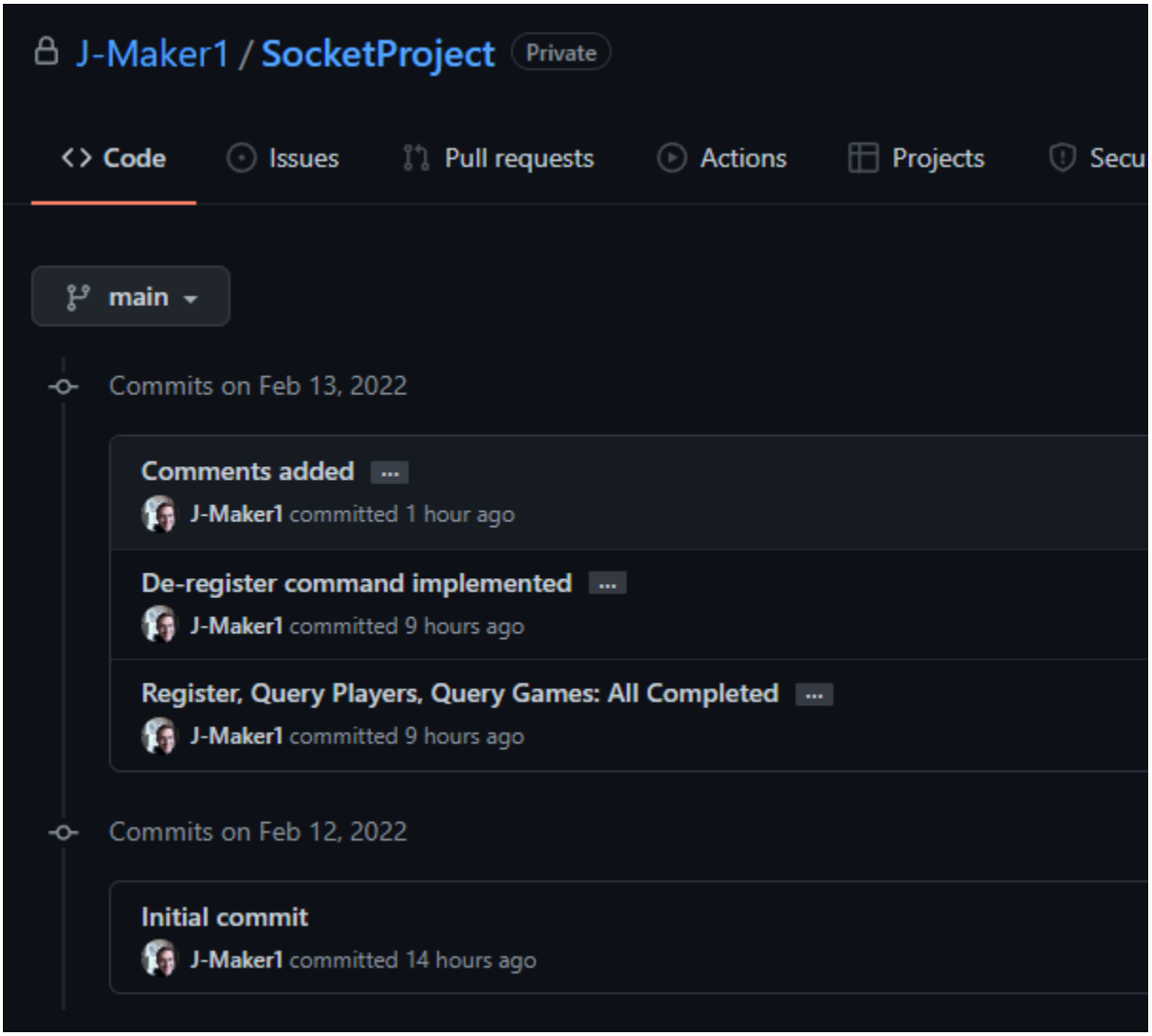
The peer does keep some data persistently, mainly a variable to track its registered status, and the socket it will use for playing games later on.

The server on the other hand has some more complicated data it holds, but still simple. For it's database, it keeps track of players by maintaining a 'list of tuples'. Each tuple is structured as "Username, IPv4 address, Port". It also has a 'list of tuples' for recording the list of games being played. The game tuple is structured as "GameID, HostName, Plyaer1, Player2, Player3"

There are no special sorting algorithms for the server, just simple loops with if statements for taking actions on a list.

Version Control System

For my version control system, I had made the decision to use githib. The github desktop app is easy to use, and the history of commits is helpful for backtracking if a change creates bugs too messy to fix.



Demonstration Video

https://youtu.be/97jbSkXe1AA

or

https://www.youtube.com/watch?v=97jbSkXe1AA&ab_channel=JustinMcGough