Noah Coomer & Casey Satran
COMP 3825 - Networking
3/5/19

Project Design

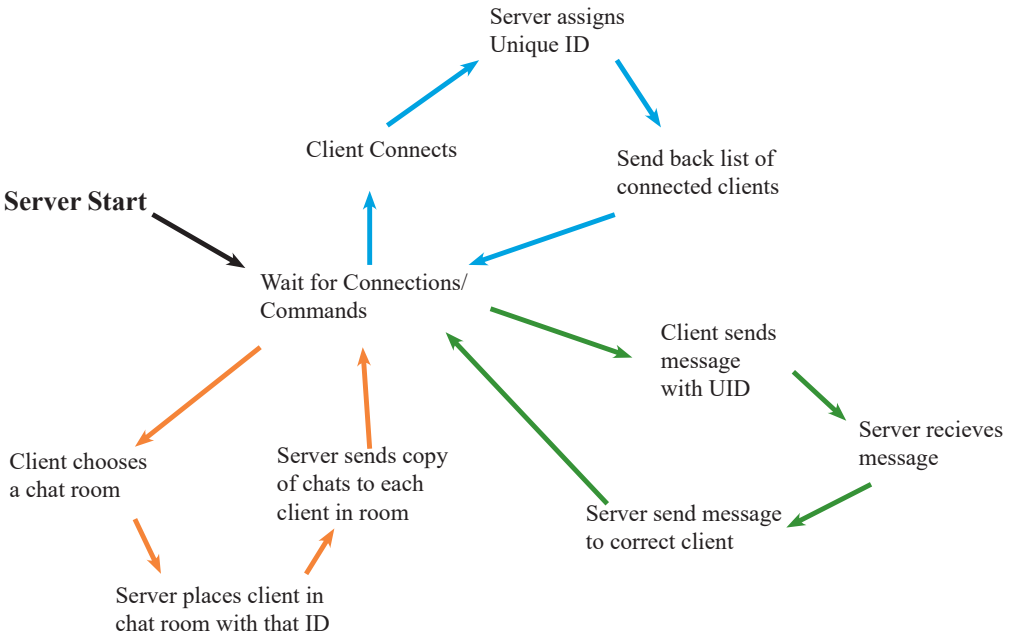
Project Overview:

Our project will is a reliable client-server chat room application written in Python 3. Once the server is started, it will wait for incoming client connections or commands from connected clients. When a client connects to the server, the server will create a new control thread for the client. The client will be assigned a unique identifier (UID) after entering a username. The client will then be echo'd back a list of all currently connected clients and chatrooms. The first client to connect to the server must wait until a second client starts a chat room with them. The client will then enter the UID of another client and a chat room will be initialized. If there already exists a chat room that the requested client is in, the new client will be added to the chat room. Clients can then enter messages which will be broadcasted to every client in the chat room. Any amount of clients can connect to the chat room.

Implementation:

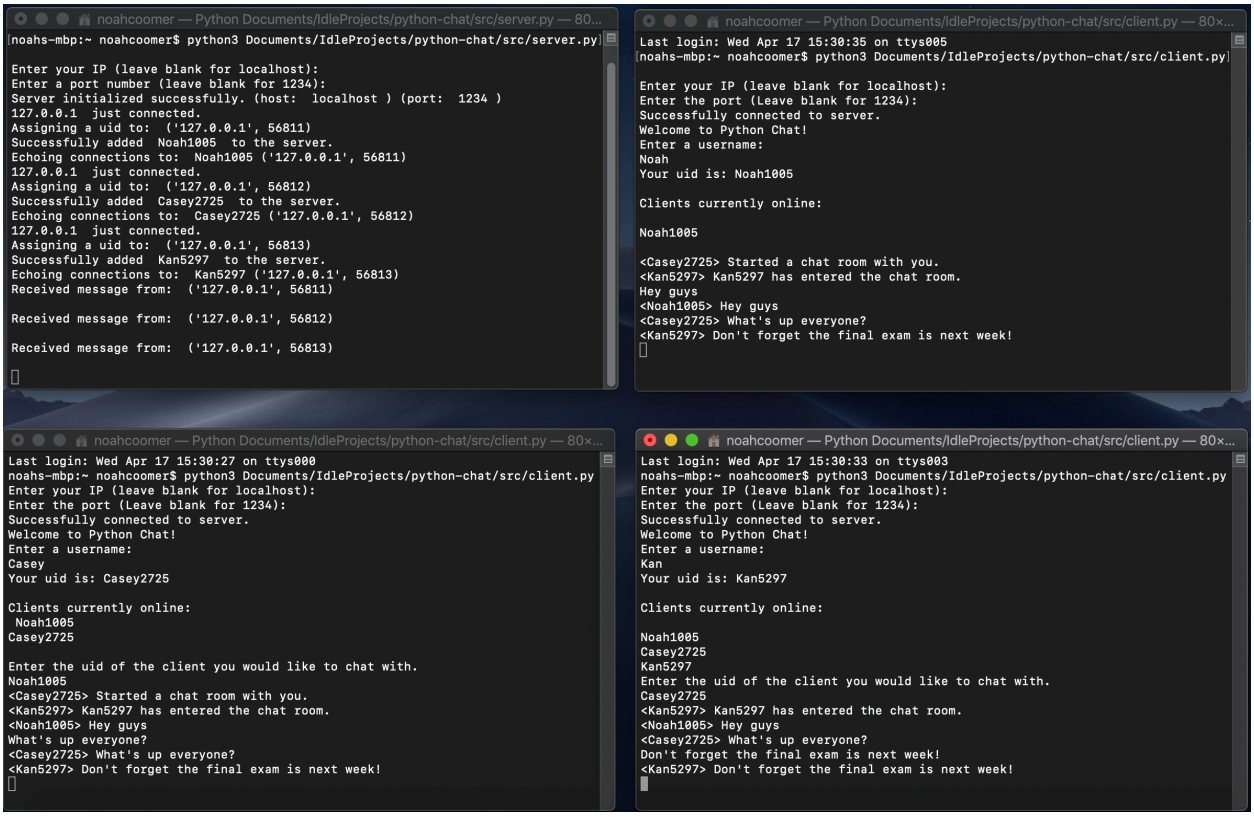
Our project will rely on several Python standard libraries: 'socket' and 'threading'. The 'socket' library will be used to initialize the connection objects and facilitate the transfer of information across the network. The 'threading' library is used extensively in this project. The server object starts a new control thread for each client, which handles UID and room assignment, as well as message broadcasting. The client is initialized with 2 threads, a send and receive thread. The send thread handles the sending of any data to the server while the receive handles data reception and output. No databases or external APIs are needed in order to echo messages between clients. The server object has several different data structures to keep track of clients, most of them being hash maps to support O(1) lookups. Client UIDs are mapped to which room they are in within the clients_to_rooms dictionary. Room numbers are mapped to a lists of participating clients in the rooms_to_clients dictionary. Finally, UIDs are mapped to socket objects in the clients dictionary. These data structures provide a very natural lookup process for the server, as messages sent to the server can be identified by the uid, which is stored in each client thread, then the room can be found by accessing room = clients_to_rooms[uid], all participating clients can be found by clis = rooms_to_clients[room], and then the clis list can be iterated through to get individual socket objects from clients to send messages to.

Program Flowchart:



Input and Output Screenshots:

An example usage of the program on localhost with 3 clients connected to a chatroom.



Challenges:

The biggest challenges each of us faced were multithreading. We both needed to do research and figure out how to make sure each thread was doing the correct job in both the client and the server as well as making sure all the data was synchronized.

Evaluations:

Casey worked on the client.py file which included the class declaration as well as the receive and send threading. Noah worked on the server.py file which handles the incoming connections and routes messages to different clients.