**Design Report: 3313 Lab4 – Chatroom App**

Robert Bishop

David Kapron

Saket Bedi

Navid Khan

**How does our application work together?**

The server.cpp file runs all backend processes. It is responsible for opening a socket for the clients to connect to, transfer data, and gracefully terminate the client(s) and server.

**Server**

Server extends Thread to create 2 types of threads: SocketThread and ServerThread.

**SocketThread** is used to handle client socket connections. Each thread is added to a list of other client threads and is stored on the server in a vector. This allows the server to terminate client threads upon request. This thread is also responsible for sending data to each chatroom.

**ServerThread** is used to handle server operations. It creates a list of all clients connected to the server upon new connections. When the ServerThread receives data/a new connection, it CREATES a SocketThread. This Thread is also responsible for using semaphores to block clients upon any Read() calls. To issue this block, a semaphore that links all clients via port number is used. To ensure messages are transmitted only to the users in a certain chatroom, the server only allows the Read() call to be executed from clients who correspond to the given socket thread’s chat room number.

**Client**

The client runs on python and connects to the server via IP and port number. It connects through the socket created by the Server and is then able to communicate with the server through this socket. The client application synchronizes with the server which allows the client to see what room they’re currently in, what members leave the chat, and what user is sending messages into the chatroom (via username). When sending message from the client, the room number is appended to the message so the server can route the message to the correct chatroom.

**Graceful Termination**

In the server, each socket thread removes itself from the vector of referenced client threads by issuing a Wait() function on their own semaphore and freeing themselves from the server’s memory. When the server is stopped, its destructor loops through all SocketThreads (clients) and closes them. It then swaps the vector of the references with an empty vector.