# 4981 – Chat Room Design

## Client FSM:



## Client Pseudo:

**GET USERNAME**

***Get a valid username from the user (default username provided if nothing is entered [this default username is going to be the system name])***

1. Display system user name as a default
2. Prompt user for a new username
3. Check that the username is a valid one
   1. Length
   2. Special characters
4. Save username (recognize relationship between username and client IP)
5. Go to ***Setup TCP Socket*** state

**SETUP TCP SOCKET**

***Create TCP socket in order to: communicate with server (in order to send messages to other clients in group chat and, receive messages from another client in the group chat)***

1. Initialize Versioning (ERROR CHECK)
2. Create TCP socket (ERROR CHECK)
3. Go to ***Connect Host***  state

**CONNECT HOST**

***Connect to the server that relays the chat messages between all clients***

1. Get IP from IP text field
2. Do an IP lookup to check that it is a valid IP
3. Get Port from Port text field
4. INIT Addr struct
5. Connect to server thru TCP socket (ERROR CHECK)
6. Acknowledge a confirmation
7. Go to ***Extract Message from user*** state
8. Create a new thread for the ***Receive Message*** state

**RECEIVE MESSAGE**

1. Read TCP Socket
2. Parse Message
3. Extract Timestamp, Username, and Message content
4. Go to format Message state with Timestamp, Username and Message content

**SAVE CHAT SESSION**

This state allows the user to save the Message history to a file

1. Open a save file path browser
2. Create file at specified location
3. open file for writing.
4. Write each entry in the message history buffer to file
5. close file

**EXTRACT MESSAGE FROM USER**

1. Extract chat message from the GUI
2. Create a new buffer
3. Add the client IP and username to beginning of the buffer
4. Append chat message to the buffer
5. Go to FORMAT MESSAGE*[STATE]*

**FORMAT MESSAGE**

This message takes the distinct properties passed in and formats them correctly.

1. Create message buffer
2. Append Timestamp to message in format <Date - HH:MM>
3. Append Username and ':' character (colour code this)
4. Append message content
5. Append New line character
6. Store message at earliest available slot in message history buffer
7. Go to Update Message Window State

**DETERMINE IGNORE LIST**

1. Create a container of ignored recipients
2. Go through each connected client in the GUI
   1. If the user has checked an ignore box
      1. Add that client to the ignore list
3. Add this client to the ignore list
4. Go to SEND MESSAGE[STATE] with ignore list and message

**SEND MESSAGE**

1. Send the ignore list to the server
2. Send the chat message to the server

**UPDATE MESSAGE WINDOW**

This method updates the message window to display the most recent state of the conversation

1. Create Chat message
2. Clear message window
3. For each entry in the Message history buffer
   1. Append message to chat message
4. Write chat message to window
5. If message extends out of bounds, enable vertical scrollbar.

## Server FSM:



## Server Pseudo:

**CREATE TCP SOCKET**

***Create TCP socket as normal but set socket option to reuse address***

1. Initialize Versioning (ERROR CHECK)
2. Create listening TCP socket (ERROR CHECK)
3. Set the socket option to reuse the port (ERROR CHECK)
4. Initialize address information
5. Bind TCP listening socket (ERROR CHECK)
6. Go to ***Listen For Connection*** state

**MONITOR SOCKETS**

***Accept new client connections as they come and save them to an array of socket descriptors and add them to the set in which Select() checks thru***

1. Set listening socket to listen (ERROR CHECK)
2. In a forever Loop
   1. Monitor listening socket (via select) (ERROR CHECK)
   2. Check for new client connections (using FD\_ISSET) (ERROR CHECK)
   3. If a new connection request has been discovered
      1. Go to update client list state
   4. Go to Check Clients State

**UPDATE CLIENT LIST**

1. Go through each client
   1. If the socket has not been saved yet
      1. Save the accepted client to the current position in the client container
      2. Stop searching through the clients
2. Update the GUI with the latest client

**CHECK CLIENTS**

1. For every possible client
   1. If Client doesn’t exist
      1. Retrun to top of loop
   2. If client is signaled
      1. Read message
      2. Go to determine client recipients State with message
      3. If a close request was received
         1. Go to close Socket state

**DETERMINE RECEPIENTS**

This state determines if any clients should not receive this message.

1. For each Client
   1. If client doesn’t exist or is the current sender
      1. Return to top of loop
   2. send Message to client

**CLOSE SOCKETS**

1. GUI displays client has left the session
2. close the socket
3. clear the file descriptor
4. set the client container at the client's position to an invalid value