# 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 CONNECTIONS**

***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. Monitor listening socket (via select) (ERROR CHECK)
3. Check for new client connections (using FD\_ISSET) (ERROR CHECK)
4. If new connection is made
   1. Go to Update Client List state
5. If a socket is ready to receive data
   1. Go to READ SOCKETS*[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

**READ SOCKETS**

1. Go through each client
   1. If the client is not connected
      1. Skip this client
   2. If the socket has indicated it is ready to send
      1. Make a buffer to hold the message
      2. Read the socket until all data has been sent
      3. Go to EXTRACT ID[STATE]

**EXTRACT ID**

1. For each entry in the client array which has an actual file descriptor (is greater than 0)
   1. If corresponding file descriptor is set
      1. Read from client
      2. Go to Determine client Recipients with message

**DETERMINE CLIENT RECEPIENTS**

This state determines if any clients should not receive this message.... I'm not sure this needs to exist actually.

1. Access client relevant ignored by list
2. Pass ignored by list to Echo to Clients state alongside message

**ECHO TO CLIENTS**

This state echoes a given message to all viable clients.

1. Create a variable that records if this client should not get message (ignore)
2. For each currentClient in the client list
   1. Set ignore to false
   2. For each ignoredClient in the ignored By list
      1. If currentClient is equal to IgnoredClient
         1. set ignore to true
         2. exit inner for loop
   3. If ignore is true
      1. return to top of loop
   4. Write 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