Project Design

List of classes:

1. Chat\_server: Waits for requests from the client and handles the connection to the clients. There will be 1 server to many clients. It will share the chat sessions among the clients. The server will contain a vector holding all of the chatrooms. Using the asio library, the server is constructed and will accept asynchronously (do\_accept(). Do\_accept() is a recursive function that runs indefinitely until there is an error. (SERVER)
2. Chat\_session: Contains a socket to connect to link the chat\_server to a chat\_room. It handles the reading and sending of user input messages among clients. (BOTH)
3. Chat\_participant: individualizes the participants and their messages. A pointer is also type-defined within the class so the client and server knows where it needs to direct messages to. (BOTH)
4. Chat\_room: deals with the mechanical processes of each chatroom. The join function deals with the process of each participant joining an individual room and being added to the participant container. The leave function deals with the process of a participant leaving a room and being erased from the participant container. The deliver function in the process of delivering any message in any chatroom. It will first store messages that are wanting to be sent by the participants. The class will also have a container of chat participant pointers and an object of a global deque container. (BOTH)
5. Chat\_client: The main client class that will be used by the user. It is how they will interact with others on Superchat. (CLIENT)
6. Chat\_message: Contains the message written by the user. Will be accessed and read by both client and server. The max body length for the message will be 150 characters. (BOTH)
7. User: inherited from chat\_participant, it has more information about individual users including the given userID, nickname, password, and super user privileges. The functions inside the class are get and set functions for these variables. (CLIENT)
8. Interface: Main setup for using ncurses as GUI. Multiple classes are inherited from it, including login, room, menu, and manager. The interface is set up using ncurses and its many GUI functions to create the windows that the user can see. (BOTH)
9. Login: Login GUI for users. Client-side, the user will enter a username and password to login as a specific user. The validate\_credentials() function will compare the input information to a file containing valid users and their passwords. If it matches the information, the client will then proceed to the next window. The user also has the option to create an account, using the input username and password. It will store the info into the file for use next time. (CLIENT)
10. Room: The interface for all chatrooms, will display current users, messages, and input box for the user. The variables inside of room include a pointer to a chat\_room, a room\_ID to differentiate the different rooms, and the name of the rooms. The functions for this class are list\_users(), add\_room(), get\_room(), display\_room\_list(), and delete room. The list\_users function will show the nicknames of all the of current users within the room. Add\_room and delete\_room or used respectively to add the room to the server and remove it from the server. Display room list will print a list all of the rooms available on the server. (CLIENT)
11. Menu: this GUI will open after a successful login and when leaving a room. The menu will have a list of the rooms available and an input text to enter a specific room. The create\_room() function is there to create a new chat\_room. Inside the constructor for the menu class, a default room will be created called ‘lobby’ (CLIENT)
12. Manager: a GUI to manage the server. Other functions necessary to see server-side issues and statistics will be put into this class. Super users will be able to manage the server, such as checking the uptime of the server (check\_uptime()), etc. Any other ideas we come up with similar to these will be added here.(BOTH)

A picture containing wall

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Identifier** | **Requirement** | **Functional/ Non Functional** | **Source** | **Client/Server/Both** | **Notes** | **Function** |
| 001 | Superchat should have all uses within one page/ window | NF | Details | Client | (user friendly) |  |
| 002 | Superchat needs to send messages within 1 second | NF | Details | Server | (responsive) |  |
| 003 | Create the login screen | F | Group | Client | \*ncurses | login() //constructor |
| 004 | Display the login screen | F | Group | Client | \*ncurses | create\_account() |
| 005 | Check if user info is valid | F | Group | Client |  | validate\_credentials(int id, str pw):bool |
| 006 | Create a new room | F | Details | Client | '/create ' ||\*asio (point to chat\_room) | room()||get\_command() |
| 007 | Add the room to the list of rooms | F | Details | Client | (in constructor) | add\_room() |
| 008 | Give the user an option of rooms to enter | F | Details | Client |  | get\_room\_list:vector<chat\_room>||display\_room\_list() |
| 009 | Create the menu screen with list of rooms | F | Group | Client |  | menu() |
| 010 | Display menu screen with list of rooms. | F | Group | Client |  | display\_room\_list() |
| 011 | Move into a new chatroom | F | Details | Client | \*asio | leave(cpp ptr)||join(cpp ptr)||change\_room(int id) |
| 012 | There will be a maximum of 10 chat rooms | NF | Group | Server | (if vector>10,do not create) in room constructor) |  |
| 013 | User will be able to create a nickname (nick”) | F | Details | Client |  | set\_nick() |
| 014 | There will be a default chat room called “lobby” that cannot be deleted | F | Details | Server |  | create\_lobby() |
| 015 | The server will keep track of all the previous messages | NF | Details | Server | (per chat\_room) |  |
| 016 | The server will support up to 50 users at once | NF | Details | Server | (break up into more req? chat\_room holds participants) |  |
| 017 | Users can be muted by other users | F | Details | Client | '/mute '|| (how to block messages on client) | mute(string nick)||get\_mute\_list():vector<string||get\_command() |
| 018 | File transfer between home directories will be allowed | F | Details | Client | '/transfer'||(figure out how to send files using asio) |  |
| 019 | There will be the ability to have your message coded | F | Details | Client |  |  |
| 020 | Superusers will be able to delete chatrooms | F | Details | Client | '/delete' | get\_command()||delete\_room(str n) |
| 021 | Users in a chatroom that is deleted will be moved to the lobby | NF | Group | Both |  |  |
| 022 | Superchat messages will be spell checked against a file provided | F | Details | Server |  |  |
| 023 | Superchat will support tab completion on messages | F | Details | Client | \*readline |  |
| 024 | Tab completion will be done from "common" replies | NF | Details | Client | \*readline |  |
| 025 | The file "~.SuperChat" will store persistant information across clients | F | Details | Client |  |  |
| 026 | Messages will be no more than 150 characters long | NF | Group | Client |  |  |
| 027 | Users can leave the superchat | F | Group | Client | '/exit' | get\_command()||leave() |
| 028 | There will be a file that keeps track of the common replies | F | Group | Server |  |  |
| 029 | Users that do not type something within 300 seconds will be kicked | F | Group | Client |  |  |
| 030 | There will be an option to broadcast a message to all chatrooms | F | Group | Client | '/all ' | get\_command()||broadcast() |
| 031 | The amount of time a user is in the server will be recorded | F | Group | Server |  |  |
| 032 | A user can input a command that will display their uptime | F | Group | Client |  |  |