**Design**

The **backend** will be a **Python Script** which will handle the Websocket connections.

(For handling multiple clients together without adding on to the latency, I can try on the multithreading module in Python which will enable me to look up for messages quicky, as well as check for any errors which may occur and then act accordingly).

The **Webclient** will be in **Javascript** which will interact with the UI (based on HTML/CSS) and connect with the Python server using sockets. It will take input from the form fields and buttons on the Appication page, and work accordingly). There is a built-in Websockets library available for JS, and I intend to use that.

**Client Side Modules/Functions:** (they’ll work through the primary socket which will established at the very beginning of the connection)

1. *openConnection*: this will be responsile for the creation of a duplex connection with the server, and if there is an error, it will show an error message.

This fnction needs to be called when the button for “New User” is clicked on the front-end/or a person logs in with a previously stored username.

1a. *loadChats*: This function will either load the original chat from the existing .db file or will create a new database (or encrypted text files) depending on the event created in the above function. The chat file will be stored on the client side (??)

2. *recieveClientMessages*: this function will be an indefinitie loop which will run constantly, and look whether the uffer has any new data, and if yes it will add the data to the open chat UI (using a new function addToChat, and add it to the specific user/room which has een asked for)

3. *closeConnection*: This function will e responsile for cleanly closing the conenction, as well as storing the chat data into the chat files.

4. createNewRoom: This will send a event to the server that there is a need for creation for a new room.

Authentication will be added in the later phase. Initially, only a username and then a login based system will be added.

**Server Side Modules/Functions:** (they’ll mostly consist of functions with indefinite loops as they need to constantly look for certain messages/flags from the clients)

The server will manage the rooms and P2P by making use of lists, so that it knows which message needs to be forwarded where.

1. *onConnection*: this will store the name of the client in a list (the names of the active clients), and see who all are online

2. *listenClients*: this will recieve messages from the client and forward it to other socket connections (which are the other users/rooms in this case)

3. *removeClient*: This will remove the specfiic client form the list of online clients,

close his socket connection, (and alert the users in chatrooms that the person has left)

All the above main and important modules will serve as the modules for my unit test cases.

Mainly: testing the server script for incoming connections, database handling, client for interfacing with HTML, retrieving chat history, appending new messages, closing/opening connection, proper authentication, chat room creation etc.

These are the basic essential functions which are required for the working of this project. More functions will be added depending on the needs.