#### 1. Introduction

The project aims to design software in order to manage Instant messaging services across devices that are located anywhere in the world. This would be achieved through the use of a variety of network systems located around the world to enroll real time communication across teams of many different sizes. Using offline storage systems to allow for communication even when no network is available or a user is not currently online. We would also allow for friend management systems which would allow for a user to add other users in order to message them directly apart from group messaging services.

#### 2. Requirement Analysis

2.1 Requirements system wide

* Both the server and clients will need to run the program form the source code.
  + No installation is needed.
  + User can start the program by opening the source code and running it.
* Communication is conducted through the wireless access points and wire network in the campus.
  + The campus network does not need to make any changes to accommodate the program
* Clients IP and port will be are embedded in the server side software.
* The client will send messages: Direct, Broadcast, Add Friend, Remove Friend, View Friends, Quit, -h or -help
  + The Direct message is used so that the clients who are registered friends with each other can have a direct message with each other.
  + The Broadcastmessage is used so that any client can be in the group message. All clients will be able to see all messages, regardless of friend status.
  + Add friend message will request the server to add a new line in the friend request file.
  + Remove friend will remove a friend on the servers friends list.
  + View friends will have will report back to the client, as to whom they can direct message with.
  + -h or -help command will be initially displayed when the user logs in and will show a list of commands that the user can use to communicate with the server.
  + The Quit command is to disconnect to the server

#### 2.2 Requirements on the Client

The client is responsible for taking input from the user and sending an appropriate command to the server, processing responses from the server, and displaying the received and processed messages to the user. The client will initiate a help menu displaying the commands that the server can accept. The client will obtain the IP of the server.

* Accepting input
  + Direct, Broadcast, Add Friend, Remove friend, View Friends, Quit
* Validating the response
  + The accepted inputs by the client will only be processed if there is a successful connection to the server.
* Input command Processing
  + Broadcast command
    - Will send a message to the server requesting the client to view and message the open group message / broadcast
  + Add Friend command
    - Command followed by username will prompt the server to send a message to the request text file and requested user asking to accept or decline friend request
  + Remove Friend command
    - Command followed by username, will prompt the server to remove a friend from the client users friend group.
  + View Friends command
    - Will request the server to return a report of all the users current friends
  + Direct command
    - Command followed by username will prompt the server to have the client have a direct message to the requested user
  + Quit command
    - Will disconnect from the server
  + -h / -help command
    - Will display all the servers acceptable commands
* Responses message processing
  + Add Friend
    - The client will receive a message based on the response to the requested friend, either they accepted or declined the request
  + View Friends
    - Will recieve a list of all friends that the client/user has accepted
  + Remove Friend
    - Will receive a message saying that the the requested user has been deleted from the users friend group
  + Broadcast
    - Will start to receive messages from all users on the current server who are chatting in the open group chat
  + Direct
    - The user will receive messages from the requested user to whom they have requested to directly chat with
  + Quit
    - Will receive a message stating that the user has been removed from the server.

#### 2.3 Requirements on the server

The server is responsible for obtaining the request from the client, processing that request and sending the correct response back to the client as soon as possible while handling multiple clients at a time. The server is also responsible for storing the request and appropriate response into the database for future use if needed by the same user over a magnitude of devices that may be available to act as a client.

##### 2.3.1 Obtaining the requests from the clients

* Check the request, and make sure that it contains correct information that is needed to be processed, otherwise return error to the client to handle what is need to happen.
* Build support for maintaining multiple clients at one time and displaying information in a fast manner with accuracy being very important.
* The maximum number of connections that would be supported for the beta program we will be developing would be at least 100 concurrent connections.
* We do not need to use multiple threading in any language besides python which will be the major language used when developing this application.

#### 3. General Design

##### 3.1 General System Wide Design

* A normal client-server interaction between the clients would follow a normal chat application that you would see on a daily basis but within the command-line allowing for any terminal to experience real time communication without installing external tools. The system would also follow easy to use methods of experiencing a broadcasted message system from other people around the world, Easy to use direct messaging services, as well as adding people from the broadcast as a friend for direct messages to be delivered. The Rule of direct message support is that both users would need to allow friendship in order for direct messages to be delivered to each of the users. Friendship is generally not needed in order for the general broadcast however

#### Message Names

The messages from the client include many possibilities which include:

* Direct, Broadcast, Add Friend, Remove Friend, View Friends,Quit

Depending on the connection request After logging in or registering, The server will send the user a couple of possibilities which follow:

* Registered
* Error

1. Username in use
2. Email in use
3. Passwords do not match
4. Please enter all required fields

* Success

1. You have been successfully registered

* Login
* Success

1. You have been logged in successfully

* Error

1. Username/Email is incorrect
2. Password is incorrect

The Server will also send broadcasted messages to the client, or direct messages to each individual user. Directed messages will only be seen by the two users who are sending each other messages and will not see broadcasted messages while in direct message mode. Viewing friends will also clear out messages from direct and broadcasted view in order to allow the user a simple screen in which orders can be given to remove friends if needed or directly message a certain person in their friend list.

The Server will communicate any errors that are issued to the client to allow the user to fix anything that they may have sent, in order to allow for seamless integration with a simple user experience (UX)

#### Message Format

Since the output of the message will be over the socket layer, a normal HTTP message will be broadcasted over the socket and allow for client to server to client communication which in return would allow for a real time communication between clients.

Messages between client and server will communicate in order to allow the server to know which user has been removed the from broadcast or direct message list and then allow for saving messages offline.

Direct messages are the only form in which a message is saved for the user when the user is offline. Broadcasted messages are not saved for offline because they are for people who are currently active in the global chat for communication. This allows for a message to be visible when the user comes back online to know that direct messages are available for them to view.

Friendships are available to be seen from any time that a user requests to see them by using the request of “View Friends”. At this time, Users will be able to see a list of people who they can direct message, or even remove as a friend. Friendships are mainly used in order to direct message a user. Friendships are the only source of allowing a user to message them directly instead of messaging them from the broadcasted range in which they may not see a message direct at them depending on the amount of traffic that may or may not be seen during the course of the allotted time.

#### 3.2 Database Implementation

In the current stage, we do not involve the real database. Instead we will use formatted text files to store the data that we need to save for offline or general records. Usernames will only allow for alphanumeric characters only without any special characters allowed. This will create a better and easier way of searching files for usernames without needing to use regular expressions or custom edge cases to achieve the desired result.

1. Append a record

When a message is received from a user who is authenticated, the record will be stored only if it is a direct message or a friend request or approved friendship from both of the authenticated users. This will allow only important messages that a user needs to actually see to be stored and shown to the user when they login and ask for the records.

1. Search for a record

Searching for a record would allow the text file to be read into a buffer or cache and allow for each record to have a comma separated message to check different parts of that instance of the message that matches a certain search parameter.

1. Modify a record

The modification of a record for this use would be to delete a record, or add a record friendships or direct messages and the feature of “seen” messages. If a record has been “seen” then we will not count that message as unseen when a user would log in to notify them of any new direct messages while they were offline. This would make it easy for users to know of any actions that are occurring on the system that involve them in anyway.