SNAKChat - Design and Implementation
CS 252 Project

Contributors:

Shaan Vaidya	150050004
Kanak Agrawal	150050016
Abhishek Kumar	150050020
Neeraj Dhake	150050022

Basic protocol:
Chat clients send all messages to the server, adding proper headers to the message for the server to decide after receiving it who send to and what action to take. Our chat app is a centralised app therefore.

Server side details:
Server side code is written in Python 2.7.

1. Important Libraries:
   - socket - sending and receiving data, socket programming
   - thread - for threading different connections
   - ldap - for ldap authentication

2. SERVER_IP and SERVER_PORT store the respective values for the server. The IP is edited in the code while the port number needs to be passed as an argument to the terminal command

3. Important system variables:

users_sockets	user to socket number mapping
socketid_name	socket number to user mapping
Ip_addr	name to ip addr mapping
users	user to timestamp (last seen or online) mapping
group_to_name	group to list of names in group mapping
name_to_group	name to 'groups in which the user is' mapping
chatDatabase	all chats stored as dictionary of dictionary of lists chatDatabase[user1][user2] returns list of chat messages between them as shown on user1's screen
groupChatDatabase	group chats, group to list of chat messages mapping

4. Threading
```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SERVER_IP, SERVER_PORT))
s.listen(1)
while True:
      conn, addr = s.accept()
      start_new_thread(new_client,(conn,))
```
This is the main code of the program. A socket is defined on which the server listens. Whenever a new connection is accepted, a new connection handler function is run in a new thread.

5. Receiving data
All messages end with a ‘$’
This is done because the messages get broken up into segments and therefore a complete message is received only after a ‘$’ is encountered.
```
temp = data[:data.find(":")]
```

Temp stores the header of the incoming message. Every message has a message type as the first word. Message format - “type”:<message>
Depending on the value of temp, the rest of the message is handled.

The following points are regarding the different types of messages and their handling on the server side.
6. login
validuser(<login message>) returns name of the user and if the credentials are valid.
In return, the server sends a login successful or unsuccessful message.
f the user is valid, corresponding changes are made in
users_sockets
socketid_name
ip_addr
users
... variables.
sendUserList() sends the list of all users and groups to all the online users which is called after the above is done.
If the new logged in user has some chat history, he is sent all his chat and group chat history from its values in chatDatabase and groupChatDatabase

7. msg
Messages of this type are normal one-one chat messages. It goes as:
msg:<towhom>:<message>
Server figures out who sent the message and sends the message msg:<fromwhom>:<message> and corresponding changes are made in the chat Databases

8. group
This message is a message sent to a group.
group:groupname:message
Server figures out who sent the message and sends the message group:<groupname>:<fromwhom><message> and sends it to all other members of the group and also updates the chat database as required.

9. NewGroup
This message is a message after the creation of a new group.
NewGroup:<groupname>:<list of users in the group>
The system variables are updated as required and the message that they have been added to a new group is sent to all the members of this new group by calling sendUserList()

10. logout
Logout message is of the type logout:<timestamp>
So the server stores the timestamp in the users dictionary and removes the corresponding user from user users_sockets, socketid_name

Client side details:

Libraries used on the client side:-

- Socket - For basic socket programming
- Thread - For implementing threads
- Libraries related to PyQt4 (QtCore and QtGUI) - Required for adding functionalities for PyQt widgets
- Ui_MainWinddow imported from output - Output.py file contains the GUI declarations of the window which is imported to change and add certain functionalities through the client side's code.

Functionalities of important global variables:-

- ➢ *temporaryGroupList* :- It's a list of members who appear in the checkboxes while creating groups.
- ➢ *username* :- stores the name of the current user logged in
- ➢ *users* :- It's a dictionary of all the users who have been logged in to the server at least once (Except the currently logged user himself) along with their time-stamps, if a user is online then the value corresponding to the dictionary shows online else it shows the time he/she was last online.
- ➢ groups:- Dictionary of groups and a list of members in a particular group.
- ➢ *currentUser*:- Shows the current user/group being shown in the main window of the GUI.
- ➢ *chatDatabase*:- Dictionary of dictionaries of every chat in which the current User is involved.
- ➢ *groupChatDatabase*:- Dictionary all the chats of groups in which the currently logged in user is present
- ➢ *authenticated*:- Checks whether user is authenticated.

The major functions and classes involved are:-

sendInfoToServer - To send group information to the server while making a new group
sendmessage - To send a message written in the text box to the server.
GUI_refresh - To refresh the GUI periodically.
Class GUI - It's the main class that determines the look and feel of the chat Application.
createGroup - This function is called to create a group with selected registered users.
submitGroup - This function is called when Submit button is pressed. It saves the group name and all other details in the respective arrays and calls sendInfoToServer
btnstate - this is a helper function for the createGroup function that specfies the users to be included in the group
logout - It logs out by sending a logout message along with the timestamp for last seen functionality
sendChat - This function basically updates the chatDatabase and the groupChatDatabase and calls the updateUi function.
updateUi - It's used to update the UI of the application(If at all new user comes online etc)

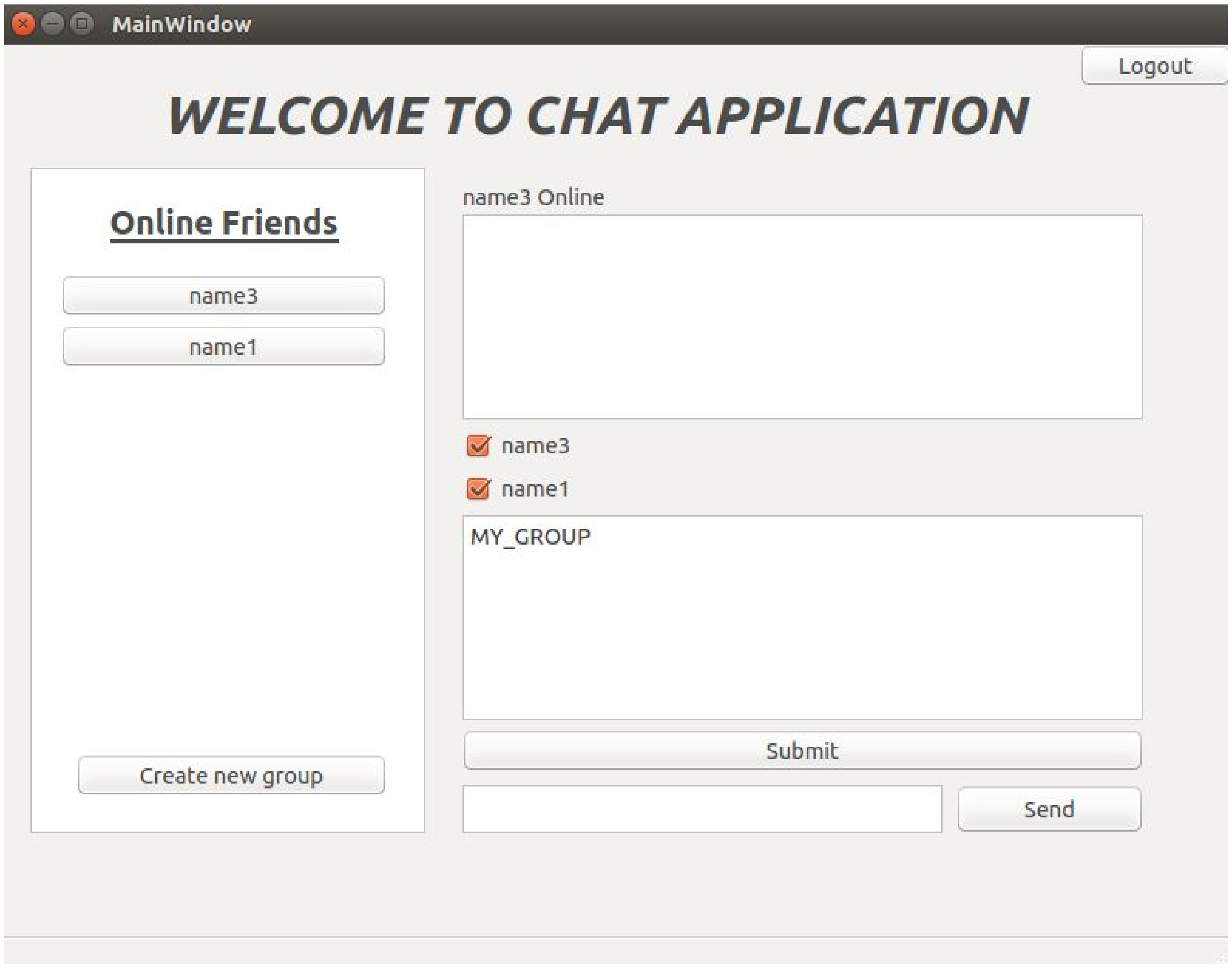
We have started two threads for the client corresponding to sending and receiving the data in main function using the thread class of python.

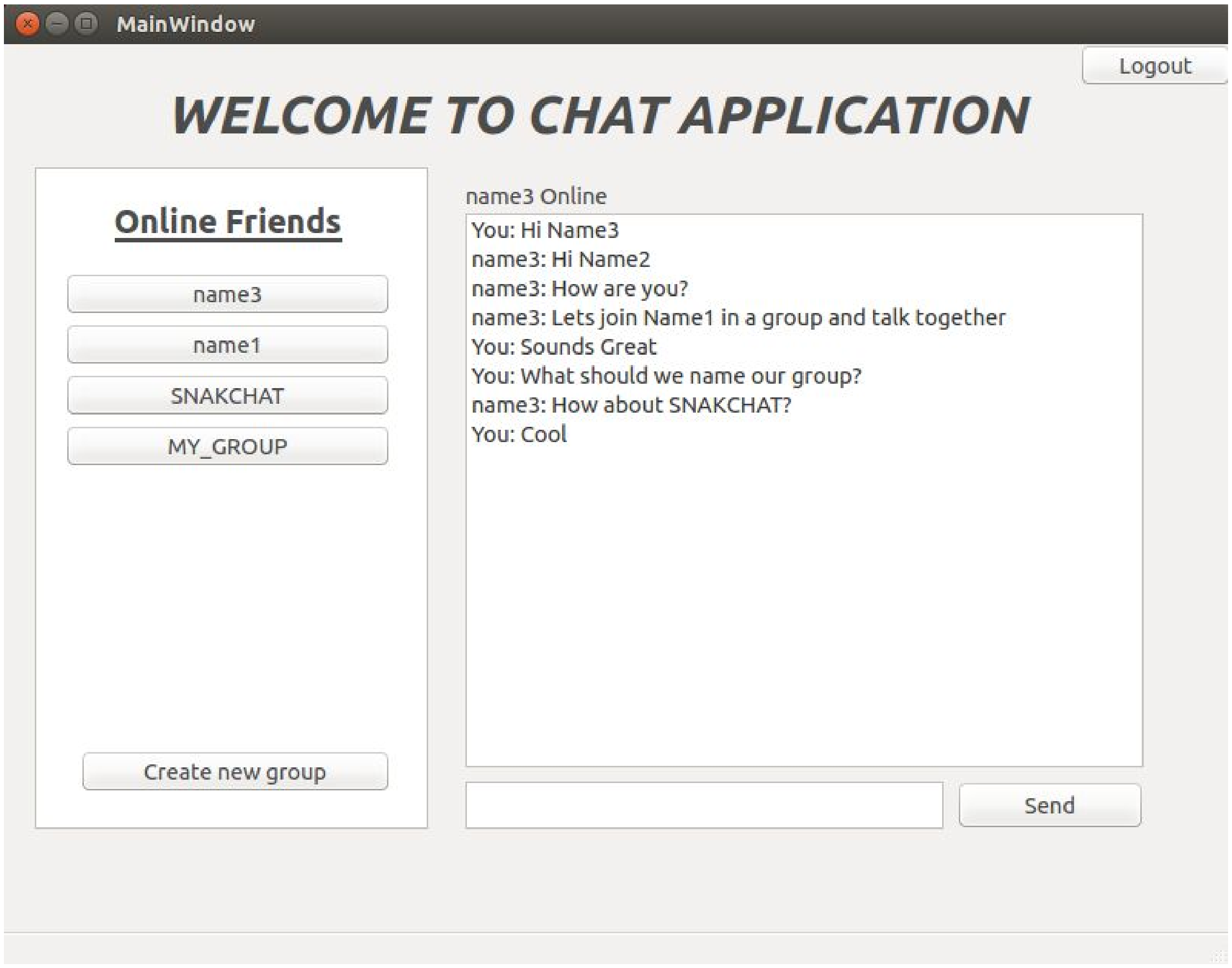
The function func basically checks for the header in the received data for a list of pre-defined headers like msg, group, list, grouplist, Database, GroupDatabse and Login and accordingly updates various dictionaries and lists at the client's side.

Login details:- The default server is assumed to be at cs252lab.iitb.ac.in. The client first sends his/her username and password to the server and waits for the authentication from it. The username and password are currently terminal based. I.e. one has to enter the details along with the python run script. For example:

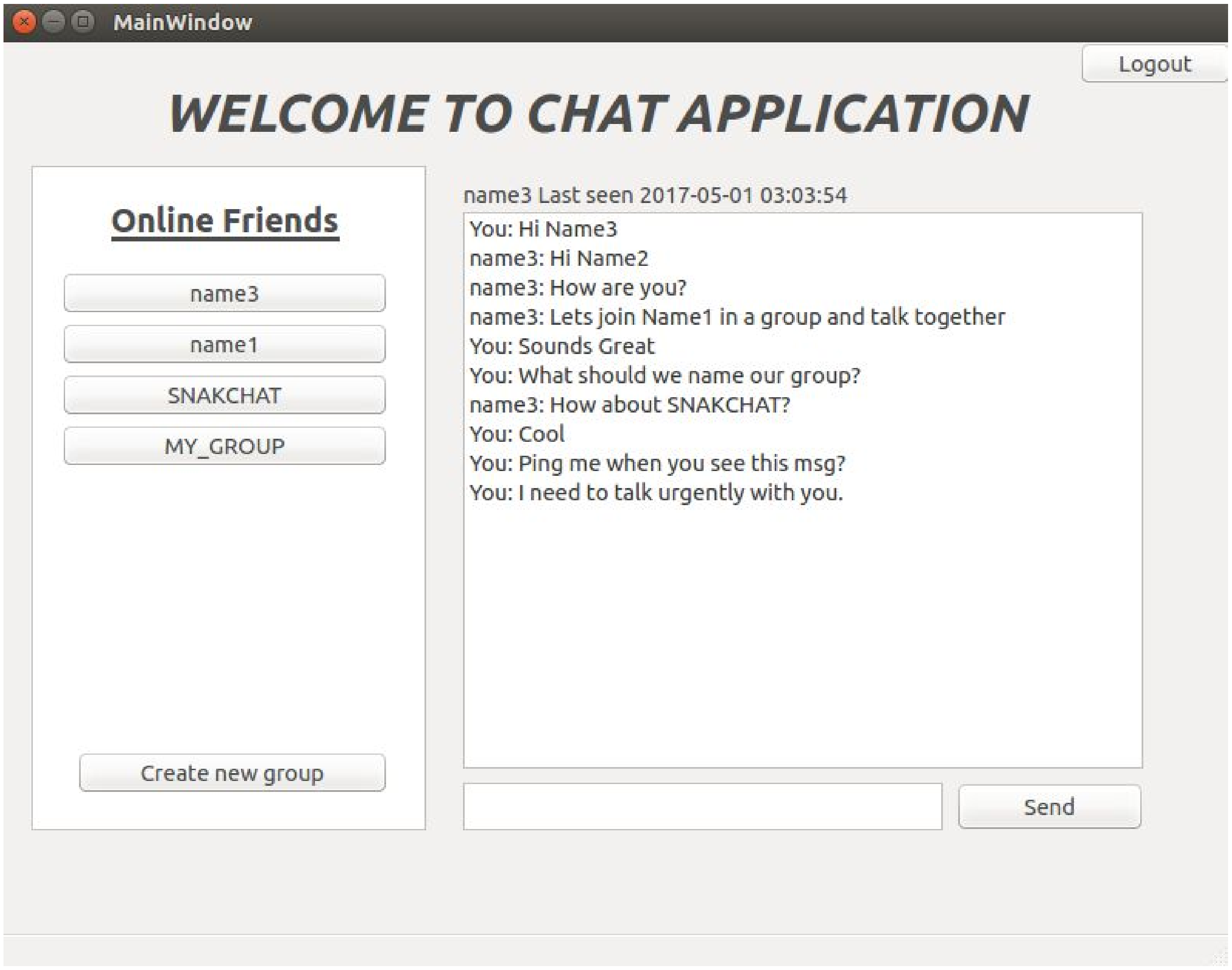
```
python client.py <server_ip_address> <port> <username> <password>
```

Screenshots of different views:





```
You: Hi Name3
name3: Hi Name2
name3: How are you?
name3: Lets join Name1 in a group and talk together
You: Sounds Great
You: What should we name our group?
name3: How about SNAKCHAT?
You: Cool
```



WELCOME TO CHAT APPLICATION

name3 Last seen 2017-05-01 03:03:54

```
You: Hi Name3
name3: Hi Name2
name3: How are you?
name3: Lets join Name1 in a group and talk together
You: Sounds Great
You: What should we name our group?
name3: How about SNAKCHAT?
You: Cool
You: Ping me when you see this msg?
You: I need to talk urgently with you.
```