K-Chat-Bot

CS 425: Computer Networks
Mini Project Report

Kanupriya Agarwal - 13338
Kaushal Agrawal - 14313 Kritagya Dabi - 14326

Group No. - 32
November 2017

1 Brief Description of Project

K-Chat-Bot is a mini chatting application that will fulfil the need for communication amongst a small organization or a closed group. It is a client-server application implemented in Python, essentially equipped with basic features like authenticated log-in, broadcasting & personal messaging among few other supplementary ones.

2 Objective

Among the plethora of options available to fulfill the communication requirements within a team, running your own application offers a unique proposition.

- *Reliable*: One doesn't have to bother in case the server of service provider is down. The application can be made available as and when required
- *Trust*: One doesn't have to worry on being snooped while exchanging confidential information within the group, unlike in services by external providers
- *Secure*: If the group is localized to a small geographical area, the application can be run on a local network. It will save from venturing into the 'internet', thereby resolving a lot of security issues.
- *Sleek*: It is custom built to serve the minimal needs of a group. While other application are doing everything under the sun, a dedicated service with no frills simplifies usage.
- *Modify*: Further, the application can be modified and updated to the changing requirements of a dynamic organization, while still keeping it sleek.

Thus, the above advantages of having one's own service application, propelled us to build the first version of the K-Chat-Bot.

3 Assumptions

- Only Messages of size less than 1024 Bytes are allowed in Broadcast and Message Options.
- We have allowed usernames and password of size 3 - 15.

- At a given time, only one user can delete account, update password or create a new account, since each of these functions writes to the same credential file at server's database.
- Number of simultaneous online users is limited to 1000.
- In the current implementation, once a user is added to chat group, he/ she remains a member, until the group is deleted.

4 Architecture

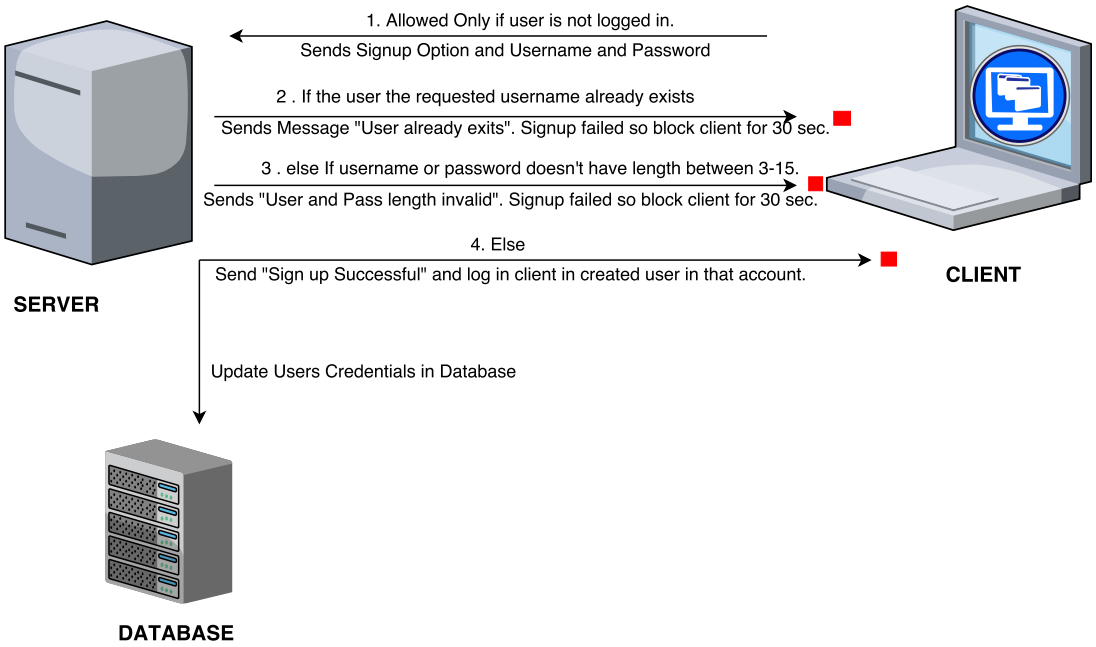
In our model all the relevant information is stored at server's database. Client creates a socket connection to the server and checks between socket input or stdin input stream. Multiple clients can connect to the server in parallel. Each new socket connection is run as a separate thread by the server.
User (client side) can choose an option through stdin which will be then sent to the server. Hereafter, the client & server will start executing module of that corresponding option. If there is any input at socket then client shows it to user through stdout.
To start the server we have to run following command-
`python server.py IP_server Port_server`
To start using ChatBot as a client we have to run following command-
`python client.py IP_server Port_server`
Note - In the above commands IP_server is the ip of the server and Port_server is the port on which we want to run the server.
Following are the option modules that is supported by our ChatBot :- (Model diagrams: In the following model diagrams, actions are depicted with sequence number and above the arrow is the condition when the action i.e. written below arrow will get executed. Also if we reach red mark function of that module ends.)

0 - Options

This will print all the Options that are supported for users.
We can choose this option by typing 0 or Options at user interface.

1 - Signup

Server Module - `serverUtilities/signup.py` Client Module - `clientUtilities/signup.py`

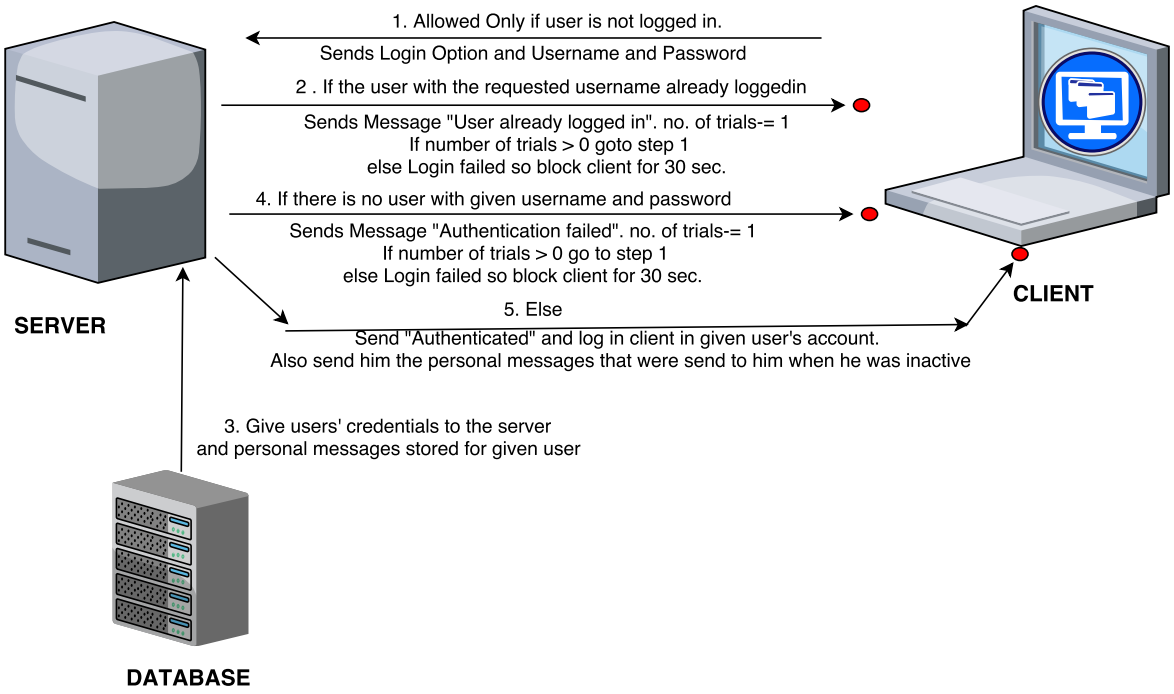


In this module if signup failed then the client is blocked for 30 seconds and not allowed to login or signup for that time. If the client again get failed at signup he will get blocked for 1 minute, in this way time period, for which client will get blocked, will keep doubling for consecutive failures at signup or login.
This option can be given by typing 1 or Signup at user interface.

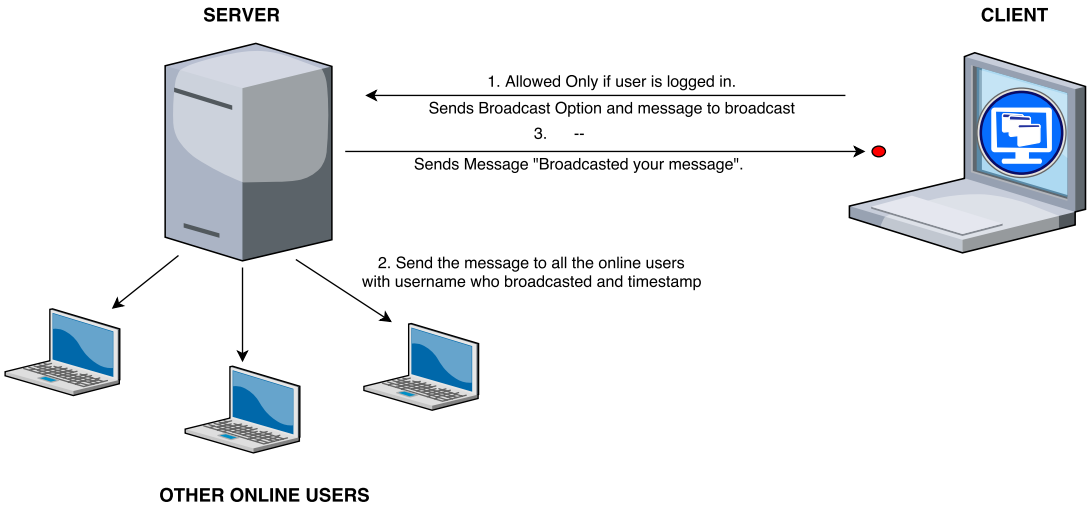
2 - Login

Server Module - `serverUtilities/authentication.py serverUtilities/asynchronous.py`
Client Module - `clientUtilities/authentication.py`

Login is allowed for 3 trials. Like Signup if Login fails for 3 trials then the client will get blocked for 30 seconds and consecutive failures will double the block time period.
This Option can be given by typing 2 or Login at user interface.



3 - Broadcast



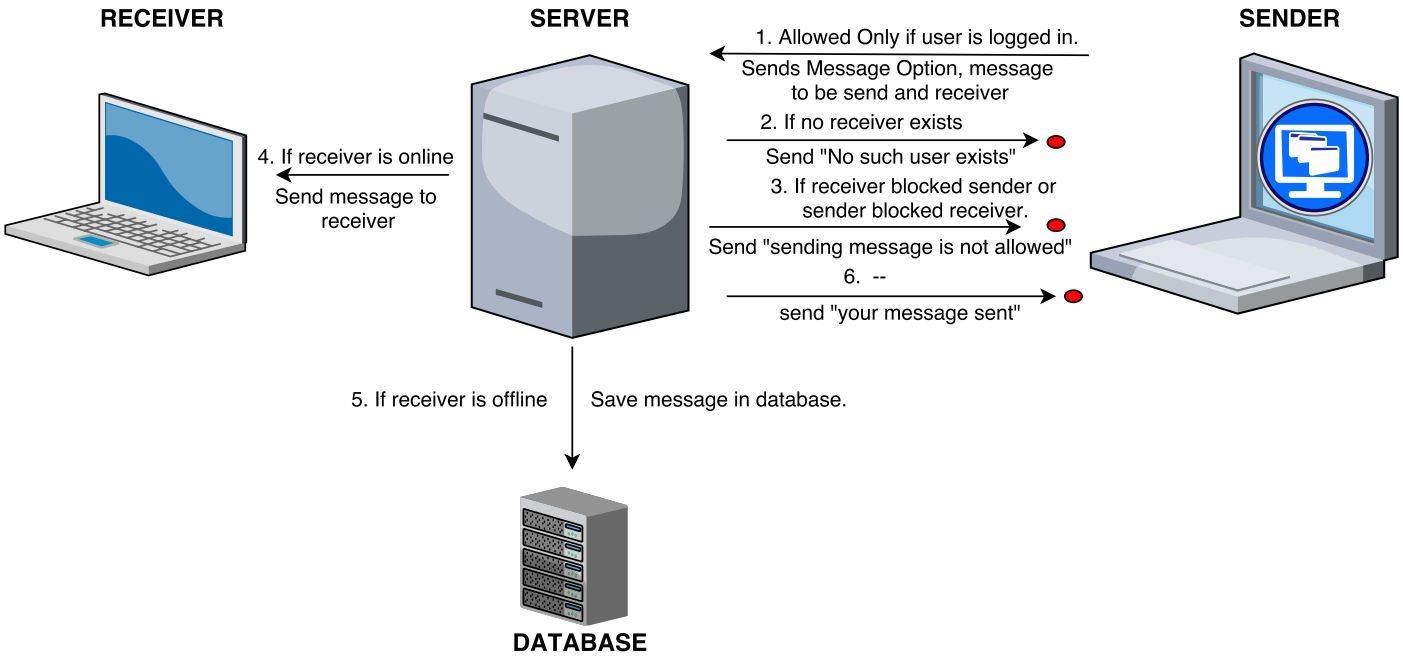
Server Module - `serverUtilities/broadcast.py`
Client Module - `clientUtilities/broadcast.py`

Broadcast will send your message to all the other online users with broadcast by user tag and time-stamp.
This Option can be given by typing 3 or Broadcast at user interface.

4 - Message

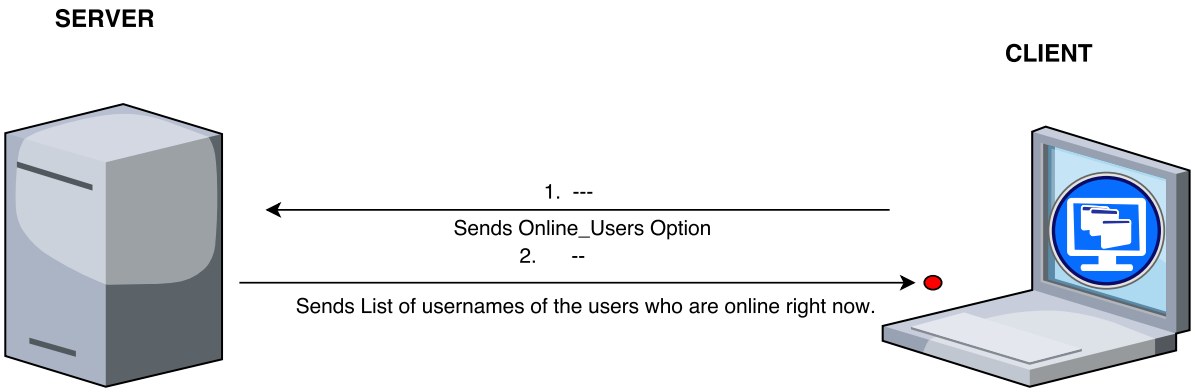
Server Module - `serverUtilities/message.py`
Client Module - `clientUtilities/message.py`

Message is to send personal message, it sends message with sender's username tag and time-stamp. If receiver is not online then save this message to be delivered later in database, this message will get to the receiver when he will login.
This Option can be given by typing 4 or Message at user interface.



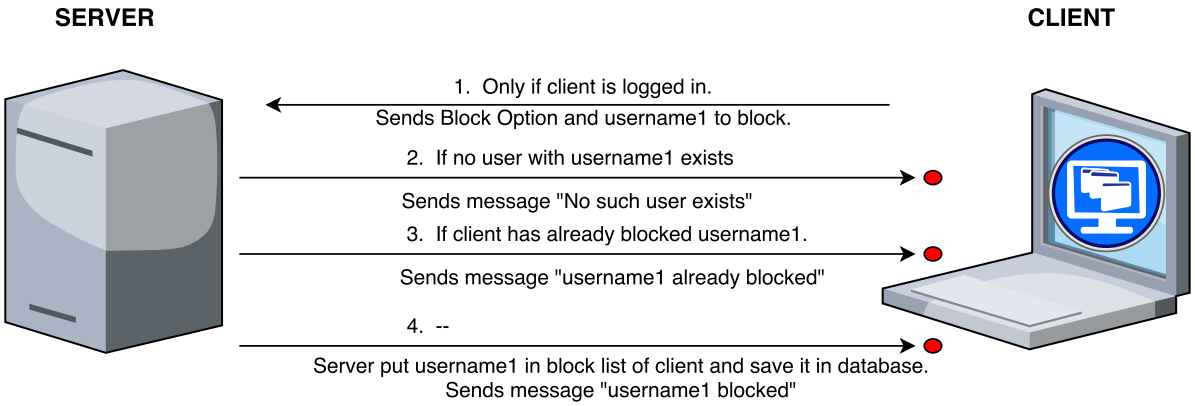
5 - Online_Users

Server Module - `server.py` Client Module - `client.py`
This is used to get list of all the users who are online right now.
This Option can be given by typing 5 or Online_Users at user interface.



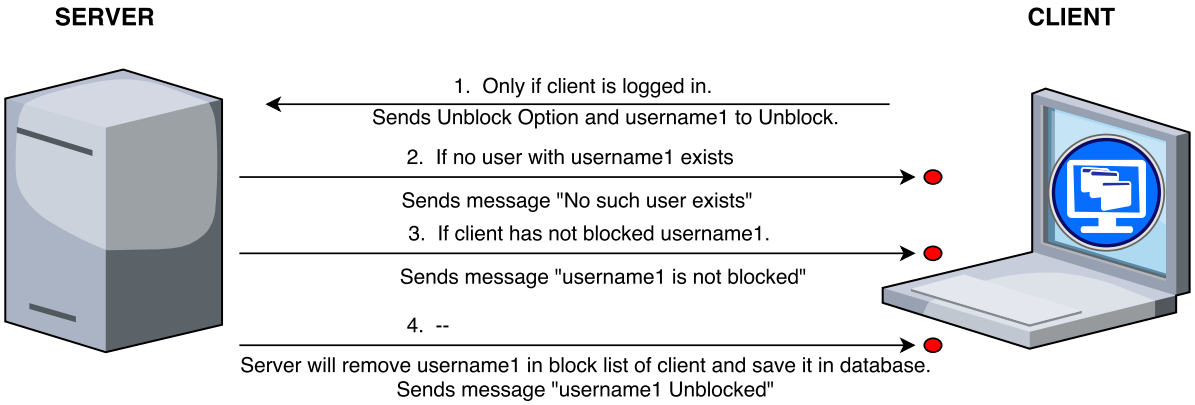
6 - Block

Server Module - `server.py` Client Module - `client.py`
If you block a user you will not be able to send him personal message and neither will he.
This Option can be given by typing 6 or Block at user interface.



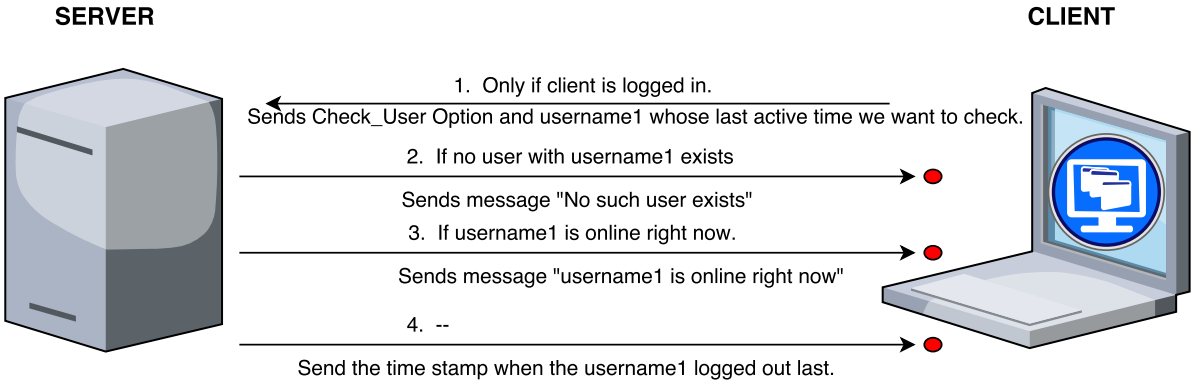
7 - Unblock

Server Module - `server.py` Client Module - `client.py`
Used to unblock already blocked user.
This Option can be given by typing 7 or Unblock at user interface.



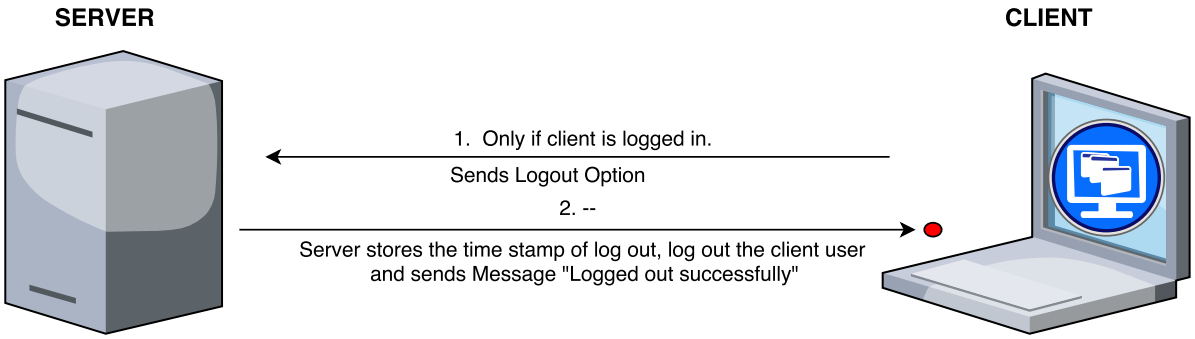
8 - Check_User

Server Module - `server.py` Client Module - `client.py`
Used to check the last time a user was active.
This Option can be given by typing 8 or Check_User at user interface.



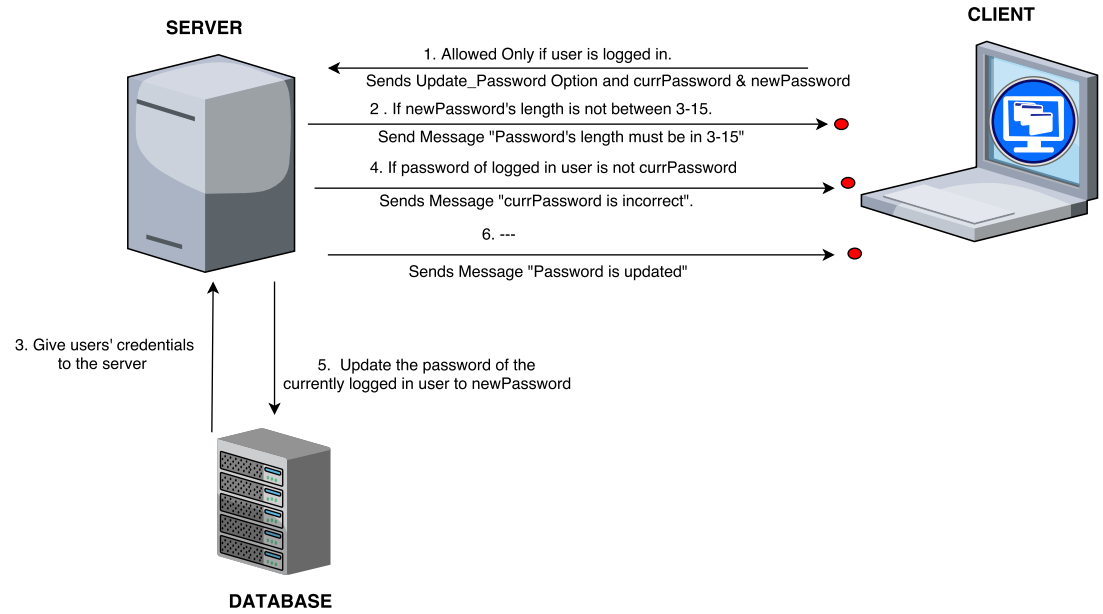
9 - Logout

Server Module - `server.py` Client Module - `client.py`
Server will log out the user and will save the time stamp of log out in Database.
This Option can be given by typing 9 or Logout at user interface.



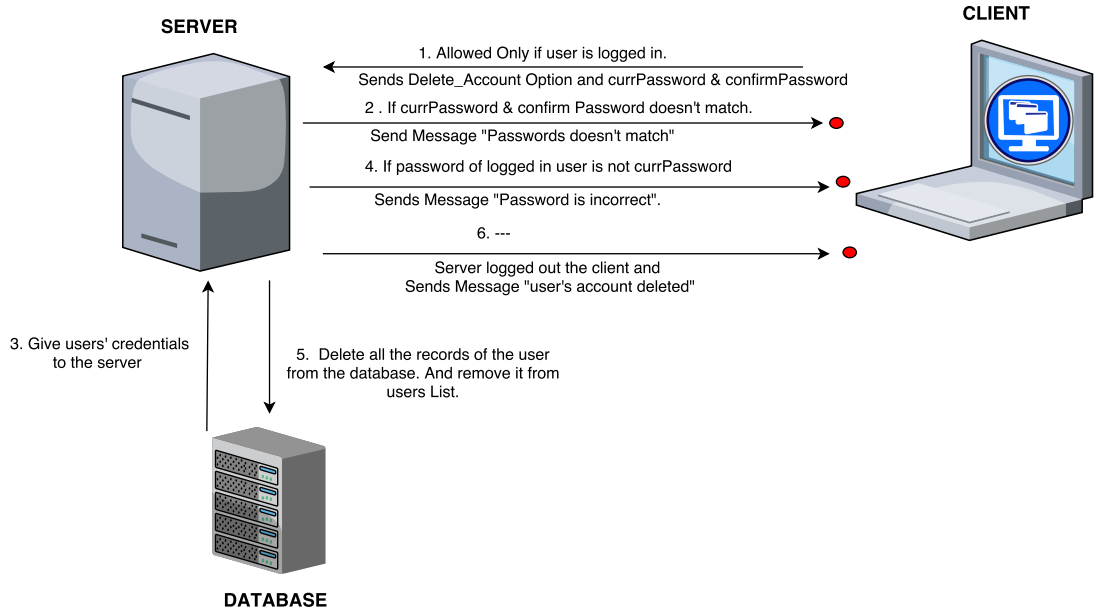
10 - Update_Password

Server Module - `serverUtilities/updatePassword.py`
Client Module - `clientUtilities/updatePassword.py`
Server will update the password of currently logged in user.
This Option can be given by typing 10 or Update_Password at user interface.



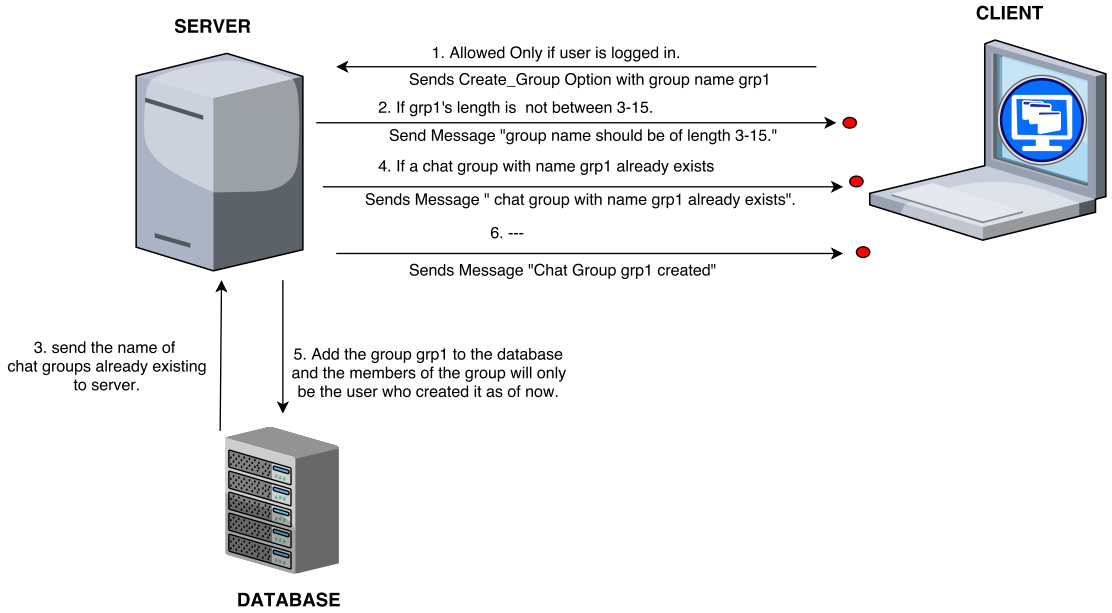
11 - Delete_Account

Server Module - `serverUtilities/deleteAccount.py`
Client Module - `clientUtilities/deleteAccount.py`
We take currPassword and confirmPassword as input to verify the user. If the confirm Password & currPassword matches the user's password then server will delete the currently logged in user's account.
This Option can be given by typing 11 or Delete_Account at user interface.



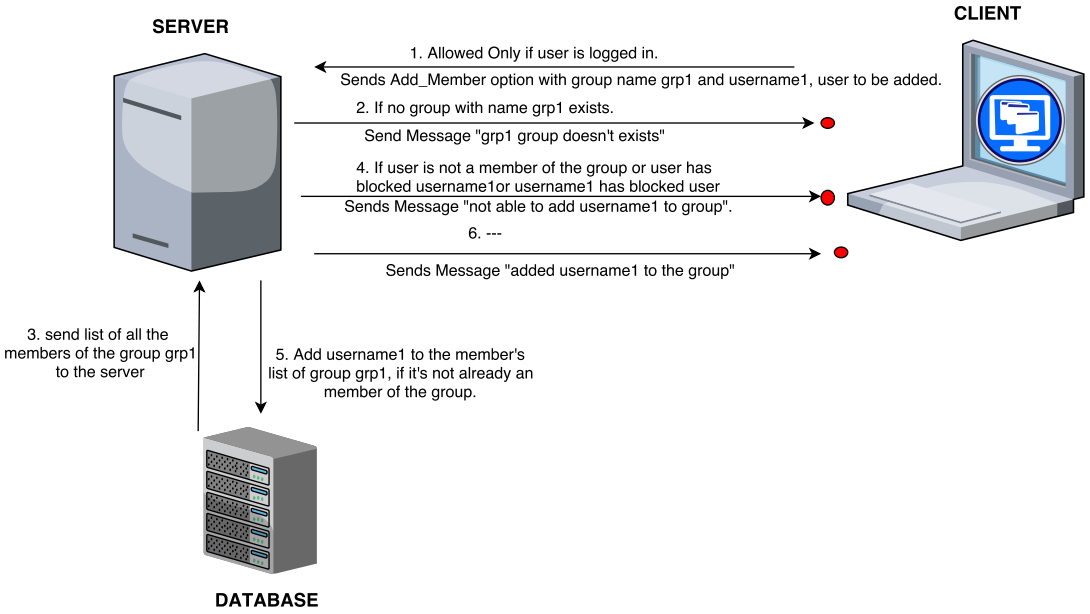
12 - Create_Group

Server Module - `serverUtilities/createGroup.py`
Client Module - `clientUtilities/createGroup.py`
This is used to create chat groups, for messaging selected users at a time. At the time of creation member of the group will only be the user who created the group. A member can further add other users to group using Add_Member option.
This Option can be given by typing 12 or Create_Group at user interface.



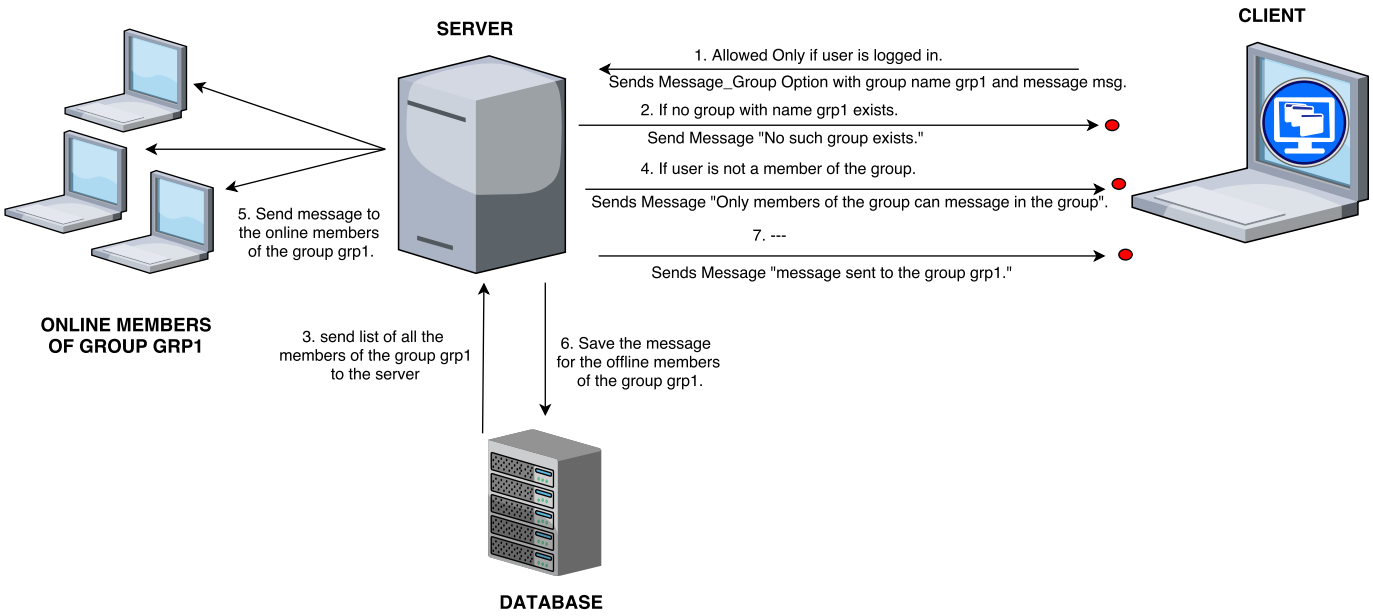
13 - Add_Member

Server Module - `serverUtilities/addMember.py`
Client Module - `clientUtilities/addMember.py`
Add Member function takes the name of the group and the username1 that needs to be added in that group. To add a member to any group, user must be member of that group. If username1 has blocked the user or user has blocked the username1, then also user can't add username1 to the group.
This Option can be given by typing 13 or Add_Member at user interface.



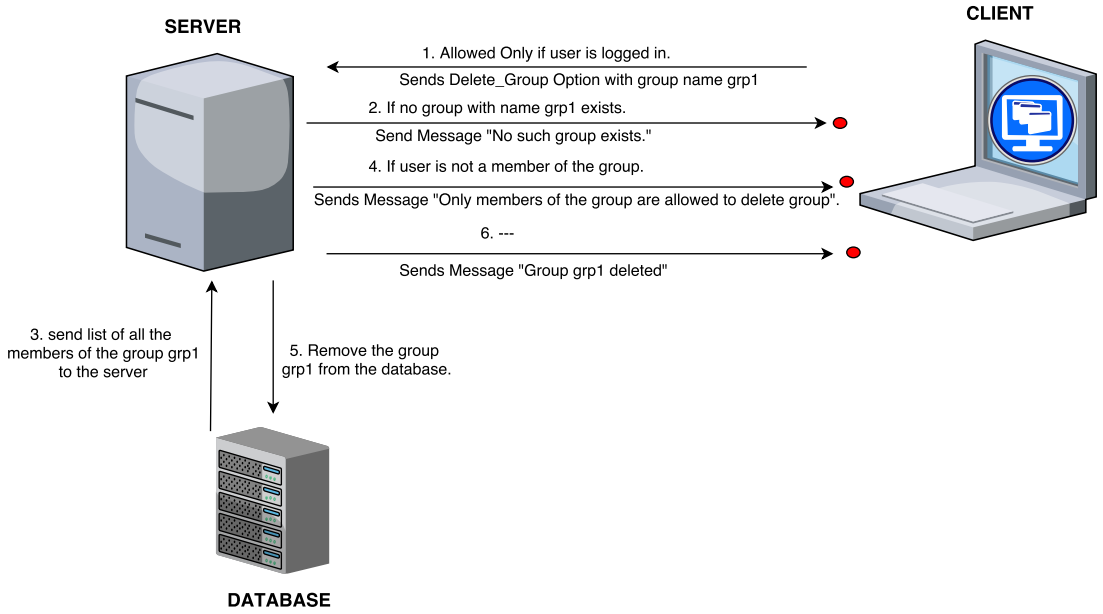
14 - Message_Group

Server Module - `serverUtilities/messageGroup.py`
Client Module - `clientUtilities/messageGroup.py`
Only a member of a group can message to the group. All the online members of the group will directly get the message while the members who are offline will get the message when they will come online.
This Option can be given by typing 14 or Message_Group at user interface.



15 - Delete_Group

Server Module - `serverUtilities/deleteGroup.py`
Client Module - `clientUtilities/deleteGroup.py`
Only a member of a group can delete the particular chat group.
This Option can be given by typing 15 or Delete_Group at user interface.



16 - Exit

Server Module - `server.py` Client Module - `client.py`
First this will call log out if client is logged in and then client will break connection and exit.
This Option can be given by typing 16 or Exit at user interface.

5 Implementation Environment

- **Language** : Python 2.7.12
- **Connection** : For network connections python socket APIs has been used.
- **Database** : For storing the credentials, groups information, messages and block lists, we have used text Files at server as our database.

6 Summary

This implementation of K-Chat-Bot serves as a chatting application with its feature highlights as below.

- Maintain separate user accounts
- Authenticated Log-in
- Update authentication details
- Broadcast message to all registered users
- Personal messaging
- Offline messaging
- Block and unblock users
- Create and maintain chat groups

However, current implementation has some limitations, some of which are evident from the assumptions stated earlier. Below are a few other features/modules that can be implemented in the next version, which will narrow down our assumptions and make it an all-round chat-bot!

- Synchronize writing to the 'credential file', so that simultaneous attempts to update account details do not fail
- Add an option to leave the chat-group
- Keep a history of messages, upper-bound by size. Give user an option to view history and search
- Develop a user-friendly client interface