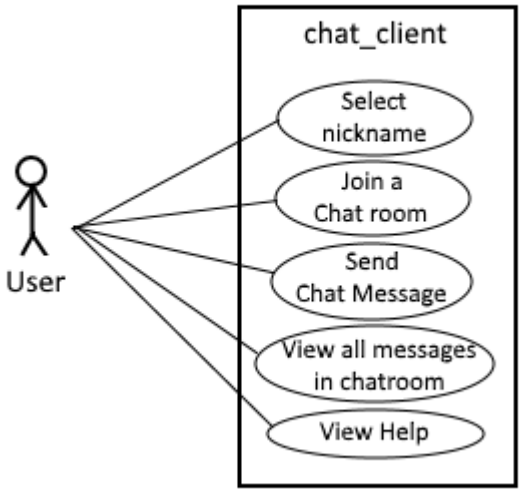
Introduction

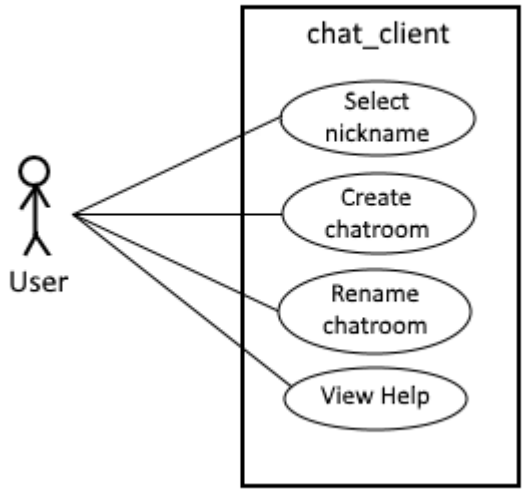
For the UberChat Software that we will be implementing and testing the stakeholders for the project are Professor Davis, teaching assistant Jianjin Deng, the members of this project (David Benepe, Jacob Cohn, Jesus Serna, Paras Sharma, and Lisa Rodriguez), and the remaining members of the class. The UberChat Software system will be a server-client chat system on the Linux operating system. The users, or clients, will access a centralized server where they will then be able to communicate with every client that has also accessed the server.

Use case 1: Join a chatroom



Use Case ID:	1		
Use Case Name:	Join a chatroom		
Created By:	David Benepe	Last Updated By:	David Benepe
Date Created:	10/10/17	Date Last Updated:	10/10/17
Actors:	User		
Description:	The user shall use a currently not-in-use nickname and join an existing chatroom, where they will be able to chat with everyone in it.		
Preconditions:	A chatroom already exists.		
Postconditions:	The nickname that the user will use is no longer available if they are online.		
Normal Flow:	1.) The user shall request a nickname to use as their identity. 2.) The user will then join one or more of the existing chatrooms. 3.) The user can then chat with anyone in the chatroom until they exit the program.		
Alternative Flows:	• If a nickname is already taken, then one or more underscores will be added to their requested nickname.		
Exceptions:	N/A		
Includes:	N/A		
Priority:	High		
Frequency of Use:	As needed		
Business Rules:	N/A		
Special Requirements:	Linux PC		
Assumptions:	The server and client programs are both running without any issues.		

Use case 2: Create a chatroom



Use Case ID:	2		
Use Case Name:	Create a chatroom		
Created By:	David Benepe	Last Updated By:	David Benepe
Date Created:	10/10/17	Date Last Updated:	10/17/17
Actors:	User		
Description:	The user will create a new chatroom.		
Preconditions:	N/A		
Postconditions:	A new chatroom will be available for anyone to enter.		
Normal Flow:	1.) The user enters the server with a unique nickname. 2.) The user clicks on the "Create a room" button and will then enter a unique name for the chat room. 3.) A new chatroom is created and is added to the list of chatrooms.		
Alternative Flows:	• If the user enters a room name that is already exists, then they will be prompted to enter another name.		
Exceptions:	N/A		
Includes:	N/A		
Priority:	High		
Frequency of Use:	As needed		
Business Rules:	N/A		
Special Requirements:	Linux PC		
Assumptions:	The server and client programs are both running without any issues.		

UBERCHAT REQUIERMENTS LIST

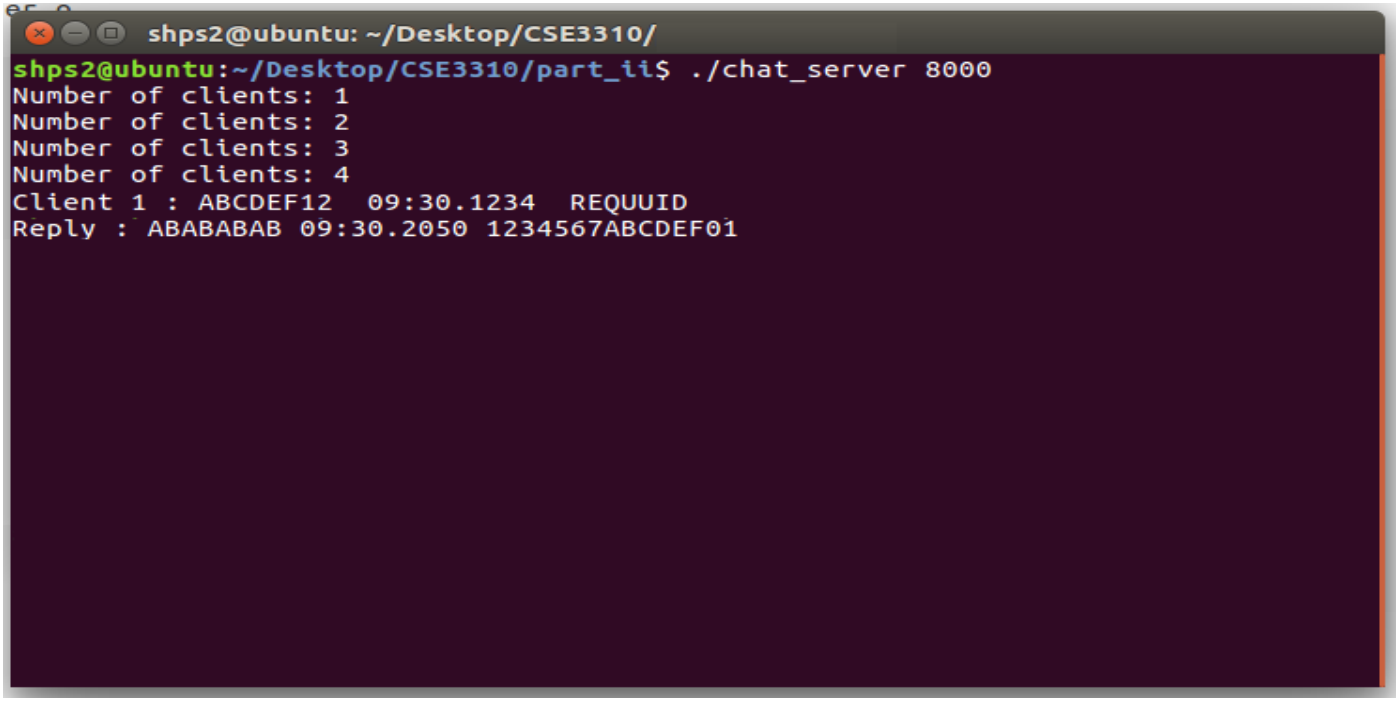
	Non-Functional	Functional	
1	✓		Uberchat server shall support a minimum of 10 clients at any given time.
2	✓		Uberchat shall be written in C++11.
3	✓		Uberchat server shall be able to send, via TCP, a message to all clients in less than 100ms from receiving the message from a client.
4	✓		Uberchat shall use TCP for all communication to and from server and clients.
5	✓		Uberchat shall communicate in a requests and response format between the client and server respectably.
6	✓		Uberchat shall add a number, in consecutive order, to any name that needs to be modifies for uniqueness.
7	✓		Uberchat shall use a checksum to verify all data integrity.
8	✓		Uberchat shall use CRC-32 and boost to ensure interoperability.
9	✓		Uberchat shall use ASCII for all TCP data trasfers between clients and server.
10	✓		Uberchat shall have a CRC-32 checksum and a time included in every request from the clients and every reply from the server.
11	✓		Uberchat shall not allow the use of commas (",") in any name including but not limited to chatroom names and nicknames.
12	✓		Uberchat shall use comma delimited lists for all lists including but not limited to user list and chatroom list.
13	✓		Uberchat shall use double quotes (" ") for any spaces and delimiters to be used as part of a name/string.
14	✓		Uberchat shall use a NULL ("/0") character to terminate every command and reply as well as any strings.
15		✓	Uberchat shall have a button in the GUI which will trigger a command called "REQUUID" to execute.
16		✓	Uberchat shall make use of a command named "REQUUID" which will trigger the server to reply with a unique UUID.
17	✓		Uberchat shall not allow any client to connect/join any chatroom without a UUID.

18		✓	Uberchat shall have a button in the GUI which will trigger a command called "NICK" to execute.
19		✓	Uberchat shall make use of a command named "NICK" which will trigger the server to store all subsequent characters, up until the terminating character, and associate that nickname with the clients UUID.
20		✓	Uberchat shall echo back the stored string associated with the client that sent the command.
21		✓	Uberchat shall compare the clients nickname against the list of used nicknames and modify the nickname, if nessesary, to make it unique. This will be done everytime a user changes or creates a new nick.
22		✓	Uberchat shall have a button in the GUI which will trigger a command called "REQCHATROOMS" to execute.
23		✓	Uberchat shall make use of a command named " REQCHATROOMS" which will trigger the server to send, via TCP, a list of all available chatrooms if any.
24		✓	Uberchat shall have a button in the GUI which will trigger a command called "NAMECHATROOM" to execute.
25		✓	Uberchat shall make use of a command named "NAMECHATROOM" which will trigger the server to store all subsequent charaters, up until the terminating charater, and create a chatroom named after the stored string.
26		✓	Uberchat shall add newly created chatrooms to the chatroom list upon executing the "NAMECHATROOM" command.
27		✓	Uberchat shall compare the stored string against the list of chatrooms and modify the string if nessesary, to make it unique.
28		✓	Uberchat server shall echo the stored string (chatroom name) back to the client that requested the "NAMECHATROOM" command.
29		✓	Uberchat shall have a button in the GUI which will trigger a command called "SENDTEXT" to execute.
30		✓	Uberchat shall make use of a command named "SENDTEXT" that will trigger the server to store all subsequent characters, up until the terminating charater and echo the stored string to all available clients in the chatroom.

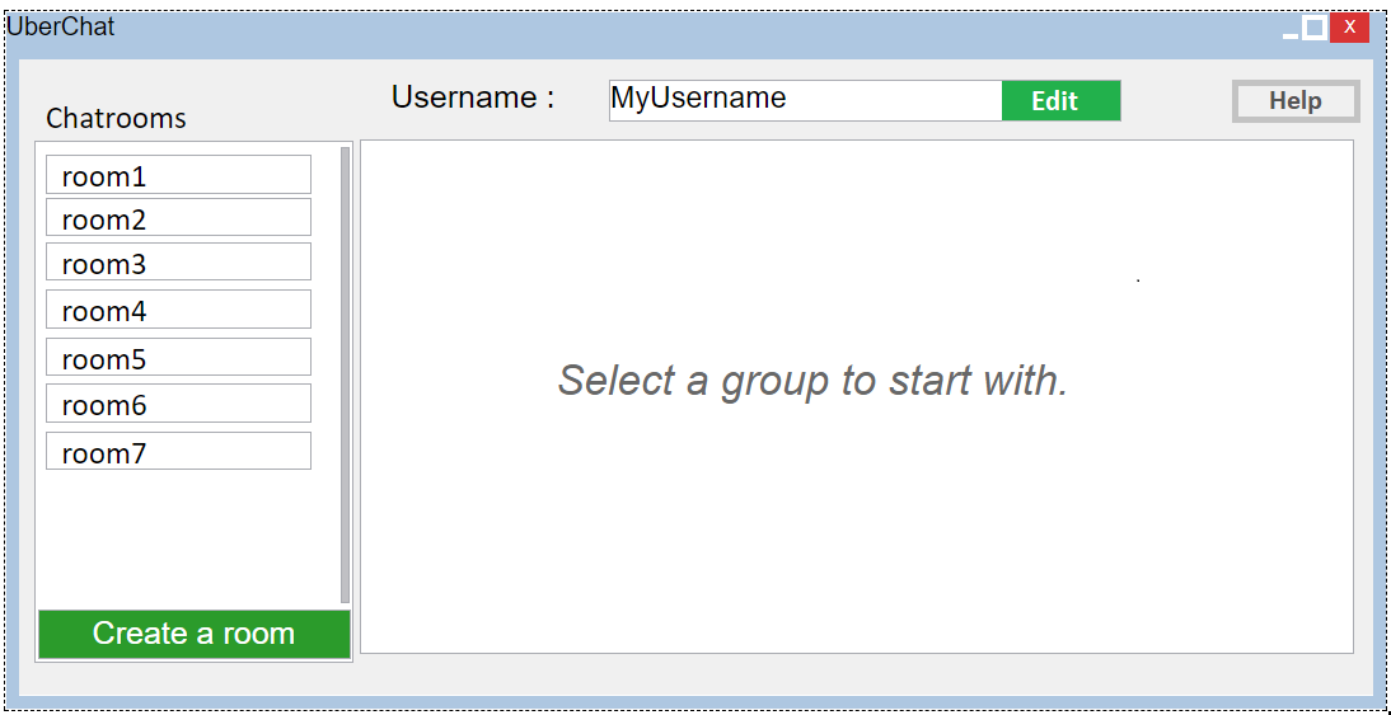
31		✓	Uberchat shall echo the count/number of characters, including any spaces and excluding the terminating charater, in the stored string back to the client that requested the "SENDTEXT" command.
32		✓	Uberchat shall have a button in the GUI which will trigger a command called "REQTEXT" to execute.
33		✓	Uberchat shall make use of a command named "REQTEXT" that will trigger the server to echo all text in the buffer back to the client that requested the "REQTEXT" command.
34	✓		Uberchat server shall have a 3,000 character buffer
35	✓		Uberchat shall store strings in the buffer using the following format: <UUID (of original sender)> <SPACE> <TEXT (in form of string)>
36		✓	Uberchat shall have a button in the GUI which will trigger a command called "REQUSERS" to execute.
37		✓	Uberchat shall make use of a command named "REQUSERS" that will trigger the server to send, via TCP, a list of all UUID and the nicks attached to them within the chatroom. This may be an empty list.
38		✓	Uberchat shall have a button in the GUI which will trigger a command called "JOINCHATROOM" to execute.
39		✓	Uberchat shall make use of a command named "JOINCHATROOM" which will trigger the server to store all subsequent charaters, up until the terminating character, and compare the stored string to the list of available chatrooms.
40		✓	Uberchat server shall link the client who requested the "JOINCHATROOM" command to the requested chatroom, if there is a match between the stored string and a chatroom name within the availabale chatroom list.
41		✓	Uberchat shall echo back to the client who requested the "JOINCHATROOM" command, the stored string (chatroom name) upon completing the command.
42	✓		Uberchat shall format every request from the client as follows: <CHECKSUM> <TIME> <MAJOR COMMAND> <OPTIONAL ARGUMENTS>
43	✓		Uberchat shall use GMT accurate to 1/10th of a millisecond for all time stamps.

User Interface Sketch

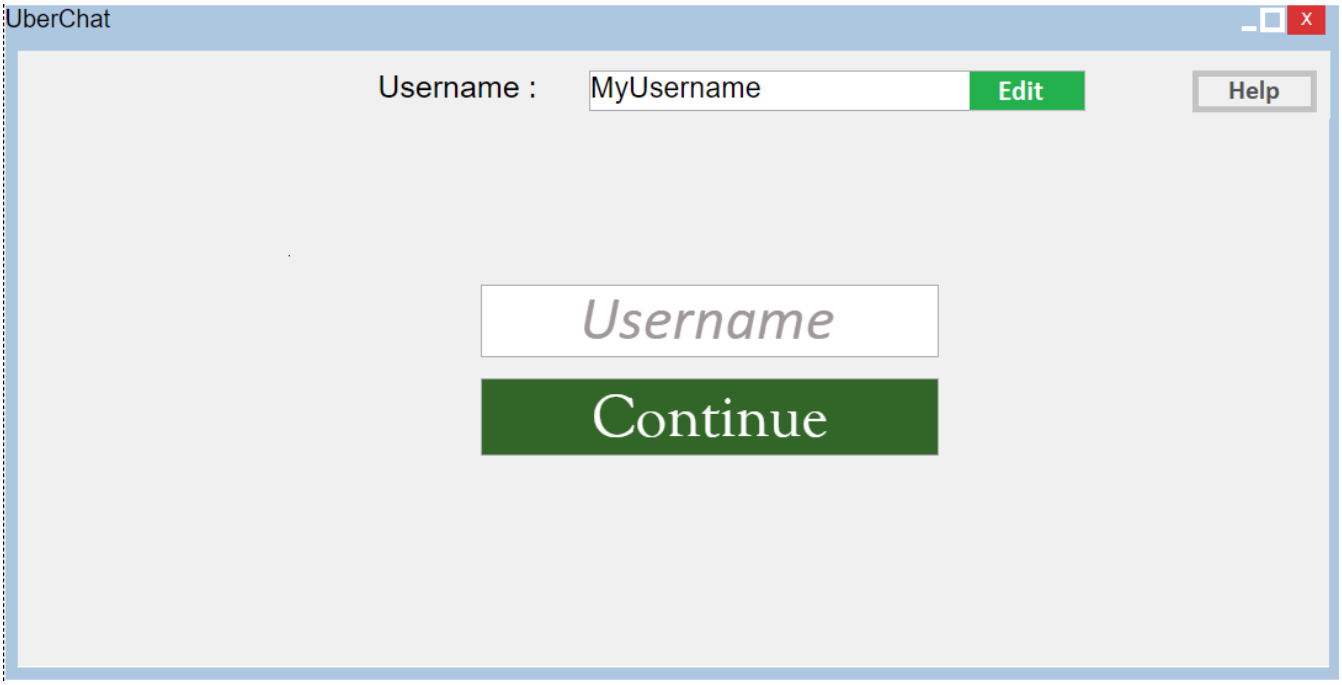
a. Server Main Page (i.e. its going to be in shell)



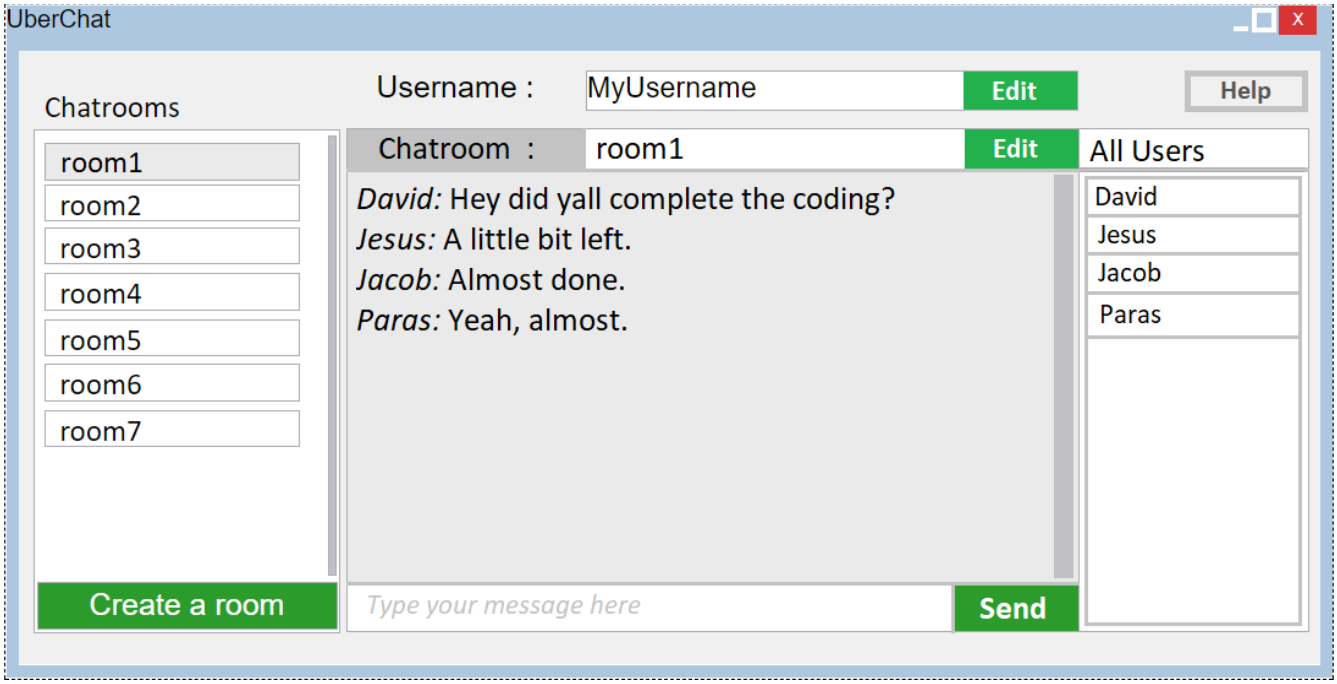
b. Client Main Page



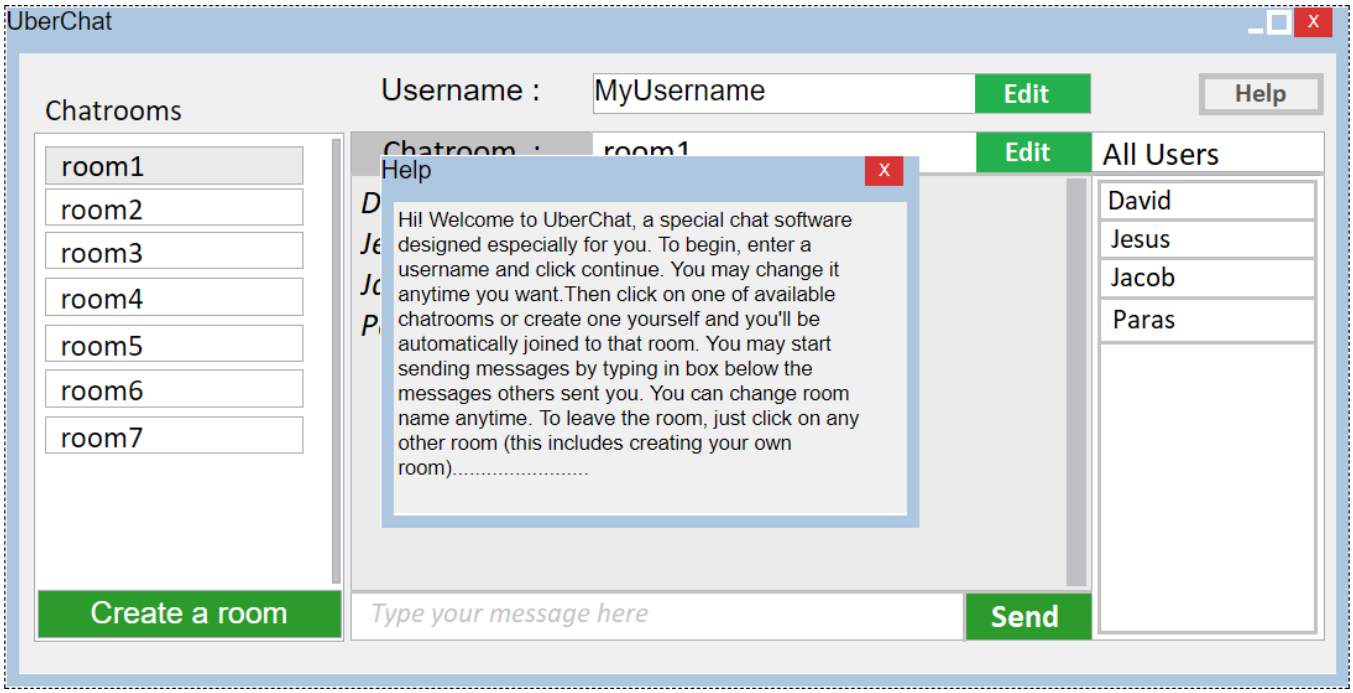
c. Client Login/Username Select



d. Client Room Select



e. Client Help



System Models

a. Context Model

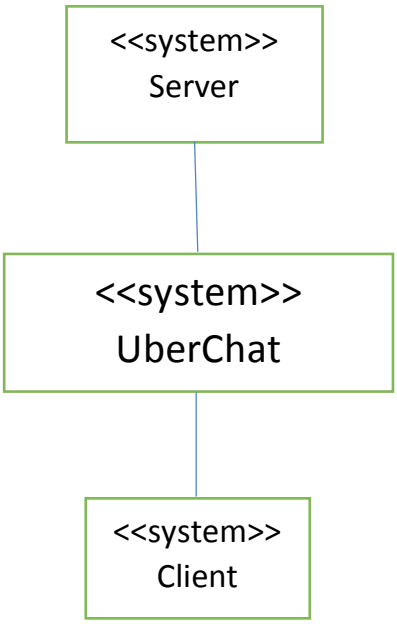


Fig: Context Diagram of UberChat

b. Class Model

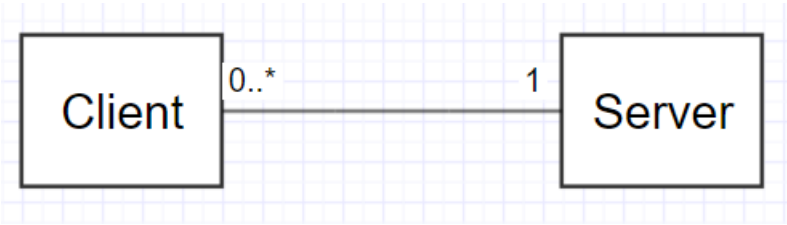


Fig: Class Model Diagram of UberChat

c. Sequence Model

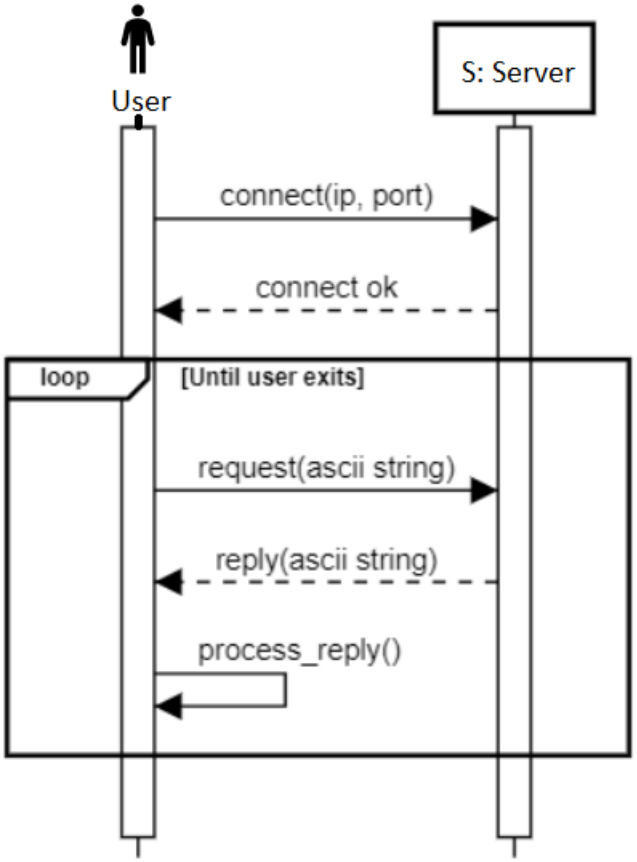


Fig: Sequence Model Diagram of UberChat

* request(string) can be anything from selecting/changing username, selecting/creating/changing chatroom, request for messages updates (uploads and downloads), etc. that a client sends to server.

* reply(string) can be anything like validating username, updating messages, etc. that server gives to client.

* process_reply() is a tool that says what to do with the response from the server such as printing out the new messages.

d. Use Case Model

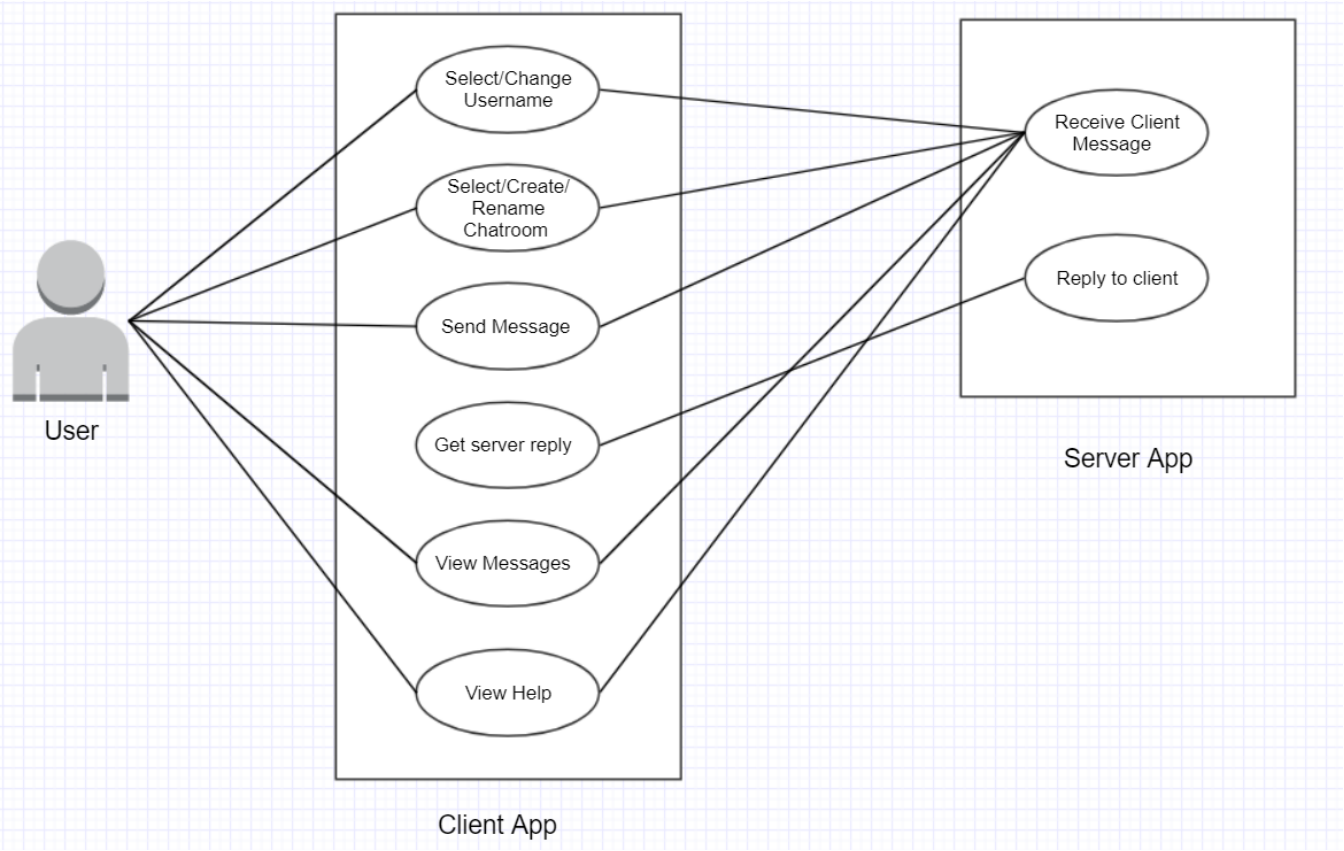


Fig: Use-Case Diagram of UberChat