# CSCCI4211 Project Documentation

## Features

1. Account operations. The author used a text file "userList.txt" in the same folder as server to keep a record of all registered users and their passwords. This idea ensures that these records will not get lost after the execution of program. Note that writing this text file manually using text editors may cause unexpected error – because mistyped spaces or line breaks may make the server to parse the file incorrectly.
2. Message sending and file transfer. To achieve concurrency of data passing, this project utilizes POSIX threads, i.e. Pthreads to deal with multi-threaded programming. On the server side, different requests (script commands) from the client will be allocated to different threads and be executed simultaneously. So, the "DELAY" commands are critical to the ordering of script commands. For the scripts without DELAYs, the actual execution of commands may be uncertain. On the client side, this author, generally, separates the sending and receiving operations into different threads. Consequently, the client can send requests to the server while receiving files or messages in the meantime. "Mutex Lock" is also used to achieve thread synchronization at some points.
3. Group chat functionalities. Unlike the accounts, group information in this project will not be remembered in disk after the execution of file. So, offline communication is not implemented in this project. The program used a 1-D simulated "2-D array" "groupMembers" to store all the groups with their member conditions. Also, 5 other commands are added to the program:

| Command | Description |
|---|---|
| ADDGROUP_[group_name] | Add chat group |
| REMOVEGROUP_[group_name] | Remove chat group |
| JOINGROUP_[group_name] | Join chat group |
| LEAVEGROUP_[group_name] | Leave chat group |
| SENDGROUP_[group_name]_[msg] | Send message to the members of group |

4. Another “command” – “INIT” is not intended to be written in script. The author treated it “like” other commands – a “INIT” request is sent to the server from the client to initialize the connection. This is very important is this project because multithreading is used among different requests of different users. So, it will be a must for the server to be able to tell which client does each request belong to. This is exactly what “INIT” do, it helps the client to fetch a unique “onlineID” from the server. This ID will be attached to further transmissions from the client so that the server can recognize it. Note that ending the client execution by force (e.g. press Ctrl + C in terminal) my result in the consequence that the corresponding thread in server keeping waiting for the client. The server will not get crashed because of this since its other threads work properly. But it is still recommended to use LOGOUT command to end the execution of client – computing resources in the server will not get wasted.

## Compile Instructions

1. Suppose the running environment is Ubuntu, you may need to first install the Pthreads library on your machine if you do not have one. Run the following two commands in the terminal:
   ```
   sudo apt-get install glibc-doc
   sudo apt-get install manpages-posix manpages-posix-dev
   ```

2. Then you can use the following commands to compile C++ source code:
   ```
   g++ server.cpp -o server -lpthread
   g++ client.cpp -o client -lpthread
   ```

3. The file to record registered accounts – “userList.txt” should be in the same folder as the server. And by default, the “send file” instructions – SENDF and SENDF2 will send the files to the folder will the client’s executable file lies in.