**CSE 322 (A2 + B2)**

**Offline Assignment 1**

**Topic**: Client-server application development using Socket Programming

**Task**: Implementation of both client and server sides of a complete chat application

1. **Requirement Specification:**

In this assignment, you are required to build a *complete chat* application on client-server architecture. You are required to write two programs (in whatever language you prefer that supports socket programming): (i) the client and (ii) the server. The client program will enable the users to chat and share files among themselves where all the communication will be coordinated by the server. Specific requirements for the task are as follows.

* On startup, the client program will prompt the user to sign up with a username and password. Upon receipt of the sign up feature the server will generate a <uname, password> pair for that user which will be stored at the server in some fashion(in a simple text file). The server will send a **“sign up and login successful”** to the client.
* If the user already has a valid username and password and then he can **login** to the server using the username, password pair.
* Upon receipt of the login information from a client, the server will check the credentials against the valid <uname, password> list stored in the server and if a valid match is found, a **“login successful”** message will be sent to the client.
* For a client there are two types of other clients: (1) Friend (2) Anonymous. Clients can see all the **active anonymous clients** by some commands or as a list in the UI. Clients can also see all of his **friends** and all of his **active friends** by the same manner. This list will be stored in the server in some fashion. The server will send this list to the client.
* A client can not chat or send file to an anonymous client. There should be a provision of sending **friend request** to an anonymous client. The server will send this request to the anonymous client.
* A client should be able to see the friend requests and he can either **reject/accept** the friend request. In case of reject/accept the server will send a notification to the client that sent the friend request and update the database of friends accordingly.
* Only the friends can chat among themselves. Now client A wishes to chat with client B(both are friends). There can be two cases: (1) client B is online, (2) client B is offline. If client B is online he will receive the message instantly. **However, if client B is offline, the message should be saved at the server at some fashion and will be delivered to client B as soon as client B is online.**
* The friend can share files among themselves. Now client A wishes to send a file to client B(both are friends). There can be two cases: (1) client B is online, (2) client B is offline. If client B is online he will receive the file instantly. **However, if client B is offline, the file should be saved at the server and will be delivered to client B as soon as client B is online.**
* Upon successful transfer of a file both client will receive a notification.
* There should also be provision for **log out**. Whenever a user logs out, the server will update the active client list properly.
* The server should keep provision for handling simultaneous connection from multiple clients. You must ensure that there is no inter-client dependency among client connections in server (such as blocking of one client by another client).

1. **Programming Issues**

* You must use socket programming in your implementation, both for client and server.
* You may use any programming language you wish (Java, C#, Python) as long as it supports socket abstraction to access OS’s native TCP service.
* Use object oriented programming. In that case, you will have at least two class files *client.class* and *server.class*. We will be running the client class multiple times in the same machine. Each client window will represent a separate user.
* You may take all the input and show output in console. Use of GUI is not a requirement. However, for this offline we expect some use of the GUI.
* Take care to handle exceptions. Unwanted action by client should not crash the server.

1. **Ethical Issues**

Since all of you will be doing the same assignment, experience tells us that there is high chance of copying. **Let us warn you that any case of plagiarism (copying) will be handled severely with nearly zero tolerance and may even result in suspension from the course irrespective of whether you were the server (source of code) or the client (who copied the code).**

1. **Submission**

For A2 : Submit it by 20th March 2:00AM in Moodle

For B2 : Submit it by 21st March 2:00AM in Moodle

No delay will be accepted.