**Summary**

This project implements Network Communication. In this we used Sockets for communication between the processes. The code for this project is written in Java language. The client and server demonstrate a message posting system. The server maintains all the messages posted by the clients. The client can retrieve the messages and view them. The function of the client and server and the what we have done in each part is mentioned below.

**Client**: The client accepts a machine name and port number to connect to as command line arguments. It connects to that server and prompts for send the user’s name. It presents the user a menu of choices like, displaying the names of all known users, displaying names of all currently connected user, sending a text message to a particular user, sending a text message to all connected users, sending a text messages to all known users, getting the messages sent to that user from other users and an option to exit. It interacts with the server to support the menu choices. It asks the user for next choice until the user wants to exit. Each client transaction interacts with the server. The clients won’t communicate directly with each other. The clients have privileges to use the system and they don’t require a password. A known user is any user who has connected during the server session. A message sent to an unknown user makes him/her known.

There are mainly three functions in the Client program. They are main function, communicate function and listenSocket function. The main function is for creating a new client for each of the client trying to login. It reads the server name and the host name and tries to connect to that host using the server name and the port number.

The listenSocket function is used for creation of a socket for communication between the server and the client. The communicate function is used for communication between the client and the server. It provides exchange of messages between the client and server and servers the all the tasks needed by the user. It displays all the options that are available to the user and whenever that runs in a loop until the user wants to exit. If the user selects the menu choice which is invalid, then it throws an error. Also, it handles all the exceptions that may occur during the execution of the client program.

**Server and Server thread**: The server accepts a port number as a command line argument. It accepts connections from the clients and create a new thread for each client. It also stores the messages sent to each user. It terminates with control-C. The server thread accepts and processes requests from the client. It adds the user’s name to the list of known users and provides mutual exclusion protection needed for the data structures that store the messages. It sends only the minimal data required by the client. It doesn’t send the menu or other UI text.

The server supports multiple different clients at the same time. But, if two clients have the same name it won’t allow them to connect at the same time. There is a maximum of 100 known users and a maximum of 10 messages for each user and each message can be maximum of 80 characters. When a user gets their messages, then those are removed from the server. When the server exists, all the messages it is storing are lost.

The server class consists of distinct functions. The main function creates a socket server running a particular port which is given as the command line argument, starts and runs until Control-C command s used. The listen socket function is for creating a socket so that the server can listen to the client and serve the required functionality. The finalize function is used for closing the socket and if it is unable to close the socket then it handles the exception. The Server class creates one new thread for each new client and starts servicing it.

The database class is used for creating objects which contains sender name, receiver name, message content and time. The database holds list of all messages, current users and known users and time stamps. The ClientWorker class is responsible for all the work done by the server. The run method which is in the ClientWorker class takes care of the validations, messages exchanges between the client and server. It has all the logic written in it. It handles all exceptions that may occur while execution of the server. The server class also provides mutual exclusion.

There are three semaphores used for mutual exclusion.

mutex1: It protects the known user list i.e., when there are two users trying to access the known users list at a time, then it allows only one user to access the list thus provides mutual exclusion.

mutex2: It protects the connected user list i.e., when there are two users trying to access the connected users list at a time, then it allows only one user to access the list thus provides mutual exclusion.

mutex3: It protects the messages database i.e., when there are two users trying to access the database of messages at a time, then it allows only one user to access the database thus provides mutual exclusion.

The messages exchanged between the client and server, that is each message sent, purpose of that message and the data contained in it are provided in the design document.

**Special Rules considered while designing**:

These are not mentioned in the project document. These are considered while designing the system.

1. The user won’t be able to send a message to him/herself.

2. When the message length is more than 80 characters, then this system trims the message and sends the first 80 characters to the designated user.

3. When a user gets more than 10 messages, he/she can be able to see the recent 10 messages. He/she can view the previous remaining messages in the next attempt of getting messages, if any, in the same server session.

**Challenges faced and Results:**

It is quite interesting to work on sockets for the communication. There are some challenges we faced while dealing with the project. They are restricting the number of known users to certain limit and displaying the recent 10 messages to the user, and creating database for messages, these are challenging. Dealing with the semaphores for mutual exclusion of the server threads. In this project, we learnt how to use to sockets for communication and they are useful for communication between processes of same machine or across processes of different machines.