Project Report

on

CHATTING APPLICATION

https://github.com/HIMU-2001/CN_Project

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OF

HERITAGE INSTITUTE OF TECHNOLOGY

SUBMITTED TO:

Mohuya Byabartta Kar

SUBMITTED BY:

Himanshu Rai, 2051266
Srinjoy Ray, 2051073
Shreya Mazumdar, 2051060
Moitreyo Datta, 2051058
Dev Kumar Gupta, 2051057
Sanhati Kundu, 2051050
Aditya Kant, 2051043

HERITAGE INSTITUTE OF TECHNOLOGY

Chowbaga Road, Anandapur, Mundapara,
Kolkata, West Bengal 700107

DECLARATION

We declare that the project entitled **CHATTING APPLICATION** is our own work conducted under the supervision of **Mohuya Byabartta Kar**, Department of Computer Science & Engineering, **HERITAGE INSTITUTE OF TECHNOLOGY, Kolkata.**

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our professor **Mohuya Byabartta Kar** as well as our H.O.D **Subhashis Majumder** who gave us the golden opportunity to do this wonderful project on the topic Chatting Application which also helped us in doing a lot of research and gaining some precious knowledge.

Finally we wish to express our appreciation to our parents for their love and support.

CONTENTS

ABSTRACT

Real-time communication through teleconferencing or chatting allows messages to be transmitted instantly from the sender to the receiver, bridging geographical barriers and bringing people and ideas closer together with the help of technology.

Although the technology to enable chatting has been available for some time, it has only recently gained widespread acceptance. By analyzing the various features, functions, and system architecture of an Instant Messaging application such as IChat(IC), we can gain a comprehensive understanding of the technology behind chatting. The main objective of IC is to enable text messaging, group chatting, and data transfer without any size restrictions, which is a common feature in most messaging applications.

INTRODUCTION

Chatting Application is a desktop based application. This client server chat application is based on Java Swing and uses Socket packages. It's simple and easy and requires only core java knowledge.
This application/program is a good example of using java.io, java.net package to create a
chat application. A beginner of java language, who is familiar with these packages can benefit.

Chatting is a method of using technology to bring people and ideas “together” despite the geographical barriers. The technology has been available for years but the acceptance of it was quite recent. Our project is an example of a multiple client chat server.
It is made up of 2 applications: the client application, which runs on the user’s Pc and server application,which runs on any PC locally. To start chatting, the client should get connected to the server. We will focus on TCP and UDP socket connections which are a fundamental part of socket programming.

MAIN OBJECTIVE

The aim of this project is to express how we can implement a simple chat application between a server and a client. The application is a desktop based application and is implemented using Swing and AWT. The project is developed in Java language executed on a single stand-alone java across a network using loopback address concept.

Application consists of two programs:

- Server
- Client

Server

The server module of the application waits for the client to connect to it. Then if connection is granted a client interacts, communicates and connects to the server, it can mutually communicate with the server. The duty of the server is to let clients exchange the messages.

Client

The client module is the one that the utilizer sends requests to the server. Utilizer utilizes the client as the means to connect to the server. Once he establishes the connection, he can communicate to the connected server.

REQUIREMENTS

User Interface

The user interface required to be developed for the system should be user friendly and attractive.

There are two sets of Java APIs for graphics programming: AWT (Abstract Windowing Toolkit) and Swing.

- AWT API was introduced in JDK 1.0. Most of the AWT components have become obsolete and should be replaced by newer Swing components.
- Swing API, a much more comprehensive set of graphics libraries that enhances the AWT, was introduced as part of Java Foundation Classes (JFC) after the release of JDK 1.1 . JFC consists of Swing, Java2D, Accessibility, Internationalization, and Pluggable Look-and-Feel Support APIs. JFC was an add-on to JDK 1.1 but has been integrated into core Java since JDK 1.2

OPERATIONAL CONCEPTS AND SCENARIOS

Operation of the application based on the inputs given by the user:

- When the run button is clicked then the chat form is initialized with a connection between the host and the client machine.

  Note: server must be started first before a client starts.

- Contains a rich textbox which send messages from one user to another
- Contains a textbox for messages to be written that is sent across the network.
- Contains a Send button

  When the sent button is clicked, in the background, the text in the textbox is encoded and sent as a packet over the network to the client machine. Here this message is decoded and is shown in the rich textbox.

PROJECT SCOPE

This project can be mainly divided into two modules:

1. Server

2. Client

This project is mainly dependent on the client/server model. The client requests the server and server responses by granting the client's request. The proposed system should provide both of the above features along with the followed ones:

Server

A server is a computer program that provides services to other computer programs (and their users) in the same or other computers. The computer that a server program runs in is also frequently referred to as a server. That machine may be a dedicated server or used for other purposes as well.
Example Server, Database, Dedicated, Fileserver, Proxy Server, Web Server. The server is always waiting for the client's requests. The client comes and goes down but the server remains the same.

A server application normally listens to a specific port waiting for connection requests from a client. When a connection request arrives, the client and the server establish a dedicated connection over which they can communicate. During the connection process, the client is assigned a local port number, and binds a socket to it. The client talks to the server by writing to the socket and gets information from the server by reading from it. Similarly, the server gets a new local port number (it needs a new port number so that it can continue to listen for connection requests on the original port). The server also binds a socket to its local port and communicates with the client by reading from and writing to it. The client and the server must agree on a protocol, that is, they must agree on the language of the information transferred back and forth through the socket. Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

- **THE SERVER SCREEN**

CLIENT

On the client site the client knows the hostname of the machine on which the server is running and the port number on which the server is listening.

To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

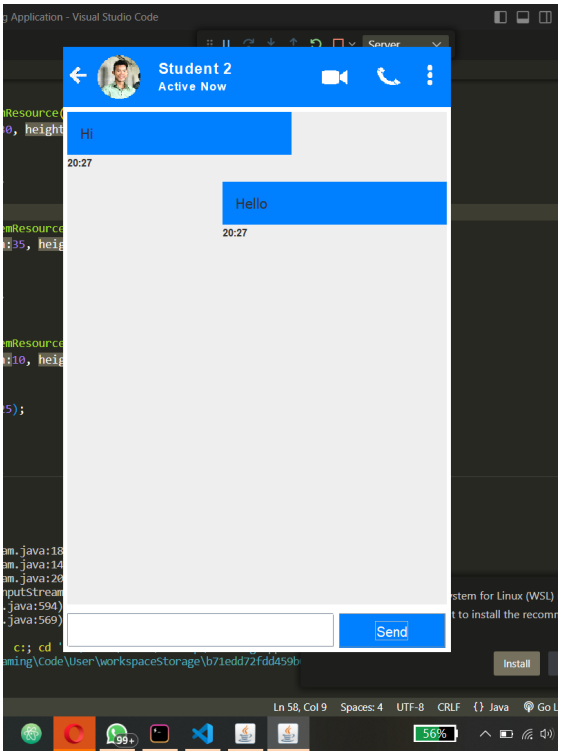
The model used for this project is the

Single server –Single client model.

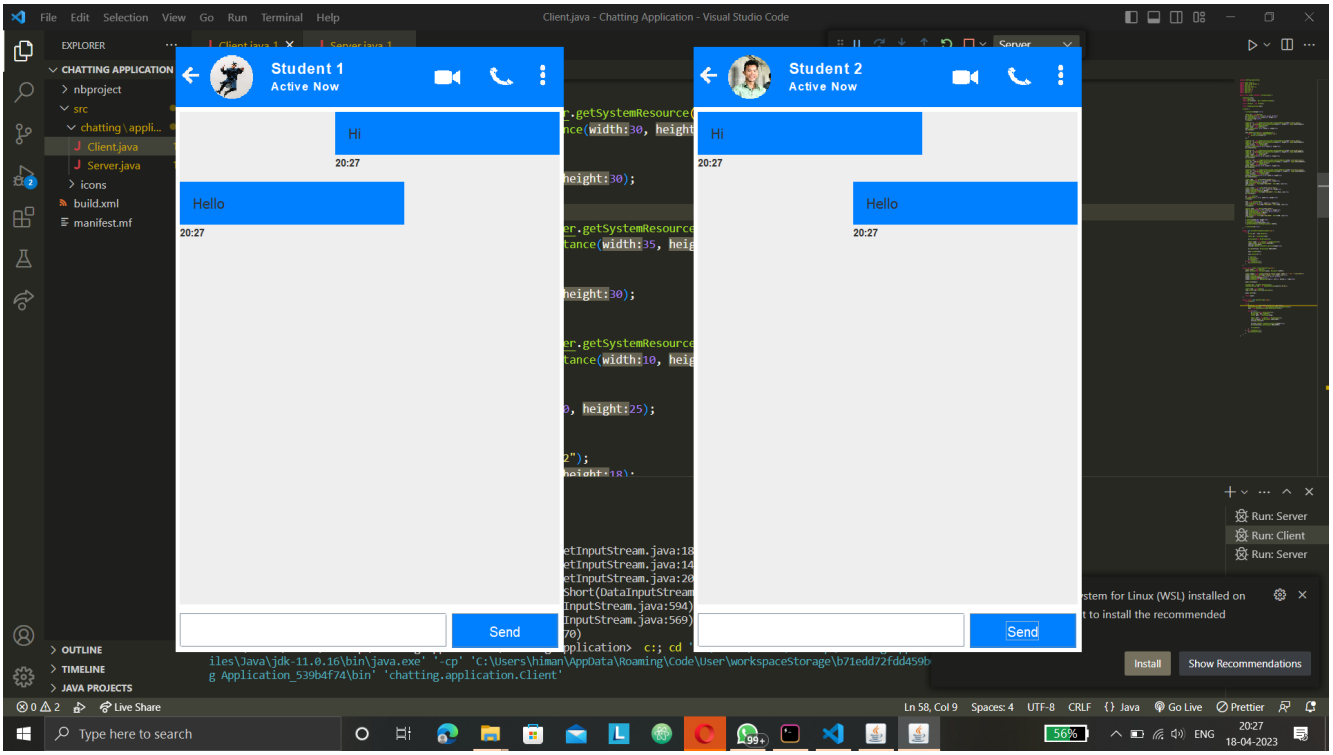
The following specification must be implemented:

- The server and client must be two separate programs.

- # THE CLIENT SCREEN

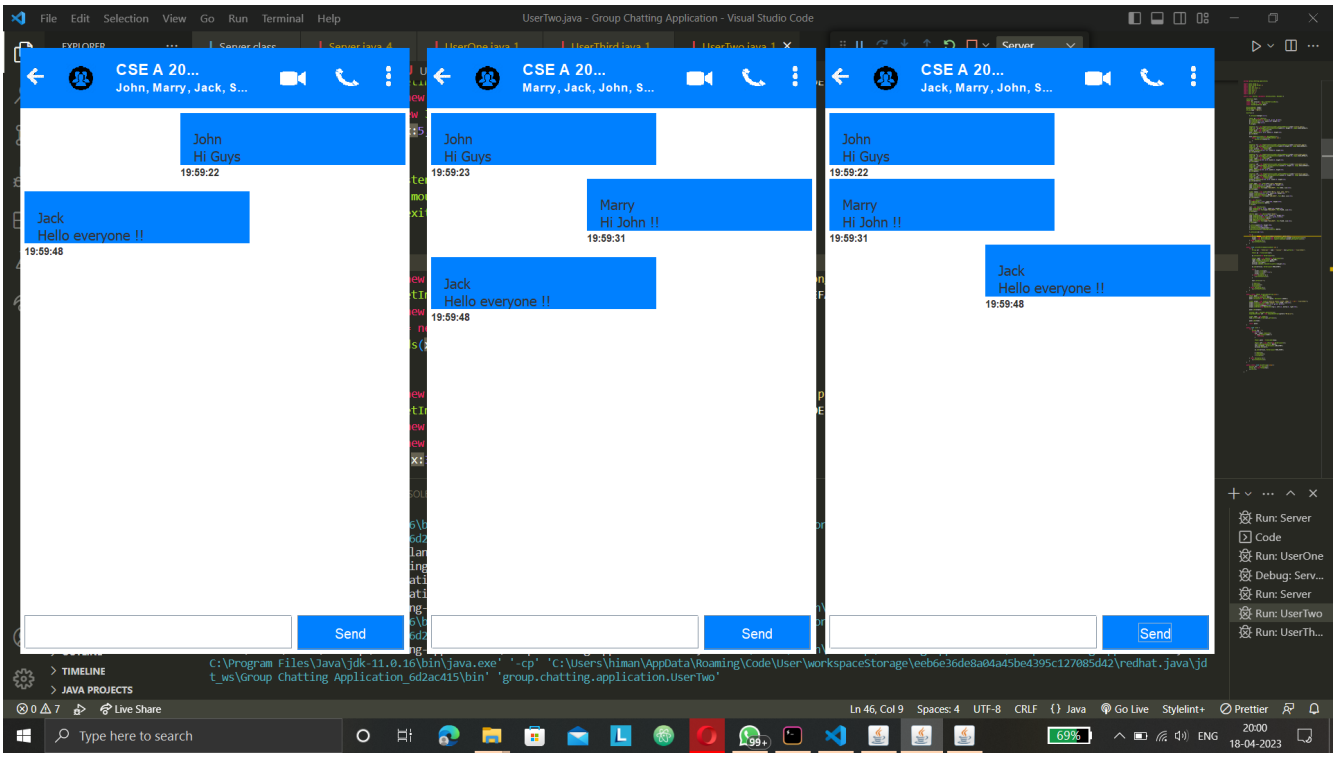


- # THE CLIENT SERVER SCREEN



GROUP CHAT

In a Java program, a simple group chat feature can be implemented using Socket programming. The program involves a server that listens for incoming connections from clients, and multiple clients that connect to the server. When a client sends a message to the server, the server broadcasts the message to all connected clients. Socket programming allows for real-time communication between multiple clients and the server, making it a useful tool for implementing a group chat feature in a Java program.



FUTURE WORK

There is always room for improvements in any software package, however good and efficient it may be done. But the most important thing should be flexibility to accept further modification. Right now we are just dealing with text communication.

In future this software may be extended to include features such as:

Files transfer: this will enable the user to send files of different formats to others via the chat application.

Voice chat: this will enhance the application to a higher level where communication will be possible via voice calling as in telephone.

Video chat: this will further enhance the feature of calling into video communication.

CONCLUSION

We developed network applications in Java by using sockets and threads concepts.

This software is portable, efficient, and easily maintainable for a large number of clients.

Our developed chatting software is unique in its features and more importantly easily customizable. The java.net package provides a powerful and flexible set of classes for implementing network applications. Typically, programs running on client machines make requests to programs on a server Machine. These involve networking services provided by the transport layer. The most widely used transport protocols on the Internet are TCP (Transmission control Protocol) and UDP (User Datagram Protocol).
TCP is a connection-oriented protocol providing a reliable flow of data between two computers. On the other hand, UDP is a simpler message-based connectionless protocol which sends packets of data known as datagrams from one computer to another with no guarantees of arrival.