MINOR PROJECT
on
"CHATTING APPLICATION"

submitted in partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

By

Rishab Mattoo
Enrollment No:- A50105220002

Under the guidance of

Under the guidance of
Dr. Priyanka Vashisht
Associate Professor

Department of Computer Science & Engineering Amity School of Engineering & Technology
AMITY UNIVERSITY GURUGRAM, HARYANA

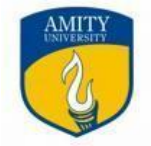
Department of Computer Science and Engineering

Amity School of Engineering and Technology

DECLARATION

I, **Rishab Mattoo**, student of B.Tech (Computer Science & Engineering) hereby declare that the project entitled **"Chatting Application"** which is submitted by me to department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University Haryana, in partial fulfilment of the requirement for the award of the degree of **Bachelors of Technology in Computer Science & Engineering**, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Date: **RISHAB MATTOO**



Department of Computer Science and Engineering

Amity School of Engineering and Technology

CERTIFICATE

This is to certify that the work in the project report entitled “Hangman Game Using JAVA” by **Rishab Mattoo** bearing **A50105220002** is a Bonafede record of project work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of ‘Bachelor of Technology” **VI** semester in the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon.

Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Date:

Dr. Priyanka Vashisht
Assistant Profesor
Computer Science & Engineering
ASET, Amity University Haryana

Head

Department of Computer Science & Engineering
Amity School of Engineering and Technology
Amity University Haryana, Gurgram

ACKNOWLEDGEMENT

I am presenting this project report entitled "**Chatting Application**"

I would like to acknowledge the open-source community for their valuable contributions. The collaborative spirit and sharing of knowledge have greatly enriched our application's functionality and features.

I would like to thank my guide **Dr. Priyanka Vashisht** for their constant support and guidance without whom the development of this project would not have been a success. In the last I would also like to thank all my colleagues who rendered their abilities to the completion of this project.

Project on the topic "Chatting Application" which also helped me in doing a lot of Research and gaining some precious knowledge.

Finally, I wish to express our appreciation to our parent for their love and support.

Rishab Mattoo

(Enrollment Number: A50105220002)

ABSTRACT

Teleconferencing or chatting refers to any kind of communication that offers a real-time transmission of messages from sender to the receiver. Chatting is a method of using technology to bring people and ideas together despite the geographical barriers. The technology to provide the chatting facility has been available for years, but the acceptance is quite recent. Analysis of chatting provides an overview of the technologies used, available features, functions, system of the architecture of the application, the structure of database of an Instant Messaging application: IChat (IC). The objective of IC application is to facilitate text messaging, group chatting option, data transfer without size restriction which is commonly seen in most of the messaging applications.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

"Chatting Application" is a desktop-based application.
A messaging app, usually referred to as a chatting app, is a piece of software that enables users to talk and send messages in real time. Although many contemporary messaging apps now enable extra capabilities like phone and video chatting, file sharing, and multimedia messaging, these programmes are made to promote text-based conversations.
Due to its accessibility, simplicity, and capacity to link individuals across various devices and locations, chat applications have been extremely popular in recent years. They now play a crucial role in both our personal and professional daily communication.

This client server chat application is based on java swing and used socket package. It is simple and easy and require only core java knowledge. I have taken this program from internet and modified a little bit to make it simpler and more elegant.
This application/program is a good example of using **java.io**, **java.net package** to create a chat application.

Chatting is a method of using technology to bring people and ideas "together" despite of the geographical barriers. The technology has been available for years but the acceptance 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 on the network. To start chatting client should get connected to server. We will focus on TCP and UDP socket connections which are a fundamental part of socket programming.
Keywords: *socket s, client-server, Java network programming-socket functions , Multicasting etc.*

Fig 1 Chatting Application Modes

1.1 Project Scope & Description

This project can be mainly divided into two modules:

1. Server
2. Client

This project is mainly depended on client/server model[1][2]. The client requests the server and server responses by granting the client's request.

The proposed system should provide both above features along with the followed ones:

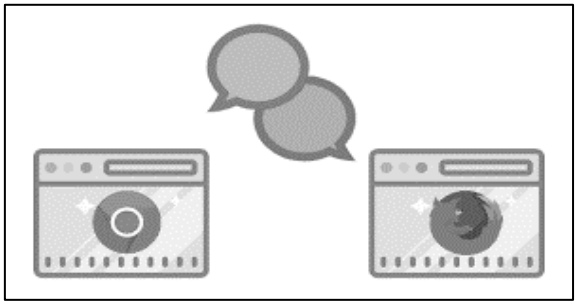


Fig 1.1 Client & Server

- 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.[1][2]

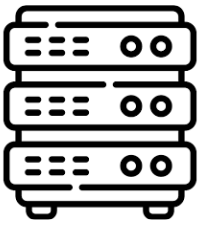


Fig 1.2 Server

- Client

The client module is the one that 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.[1][2]

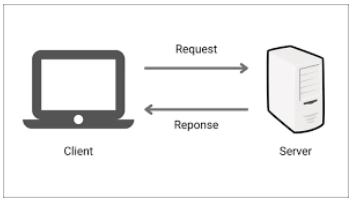


Fig 1.3 Client

1.2 Aim & Objective

The aim of this project is to express how we can implement a 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 SE language executed on a single stand-alone java across a network using loop back address concept.[3]

Objective:

1. Real-time Communication
2. User Engagement & Retention
3. Customization & personalization
4. Cross-platform Compatibility

CHAPTER 2

BACKGROUND STUDY

Figure 2.1 JAVA Symbol

2.1. About JAVA

In order to have as few implementation dependencies as feasible, Java is a high-level, class-based, object-oriented programming language. Because Java is a general-purpose programming language, compiled Java code can run on all platforms that accept Java without the need to recompile. This is what is meant by the phrase "write once, run anywhere." Regardless of the underlying computer architecture, Java applications are often compiled to bytecode that can run on any Java virtual machine (JVM). Although Java has fewer low-level features than either C or C++, it has syntax that is like each of them. Unlike most traditional compiled languages, the Java runtime has dynamic capabilities (such reflection and runtime code change).

According to GitHub, with 9 million developers as of 2019, Java was one of the most widely used programming languages, especially for client-server web applications.

2.2. JAVA SWING

Java Swing[4] tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etThe Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.c.

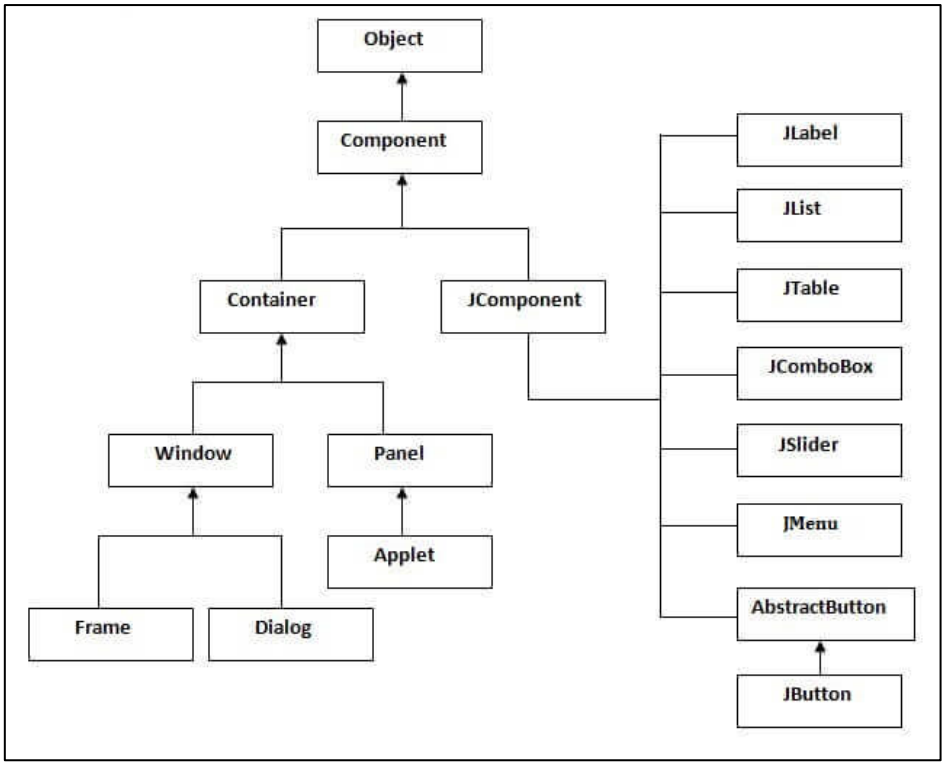


Figure 2.2 Hierarchy of JAVA Swing Class

2.3. JAVA SOCKET

Java Socket programming[5] is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

- Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

- ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

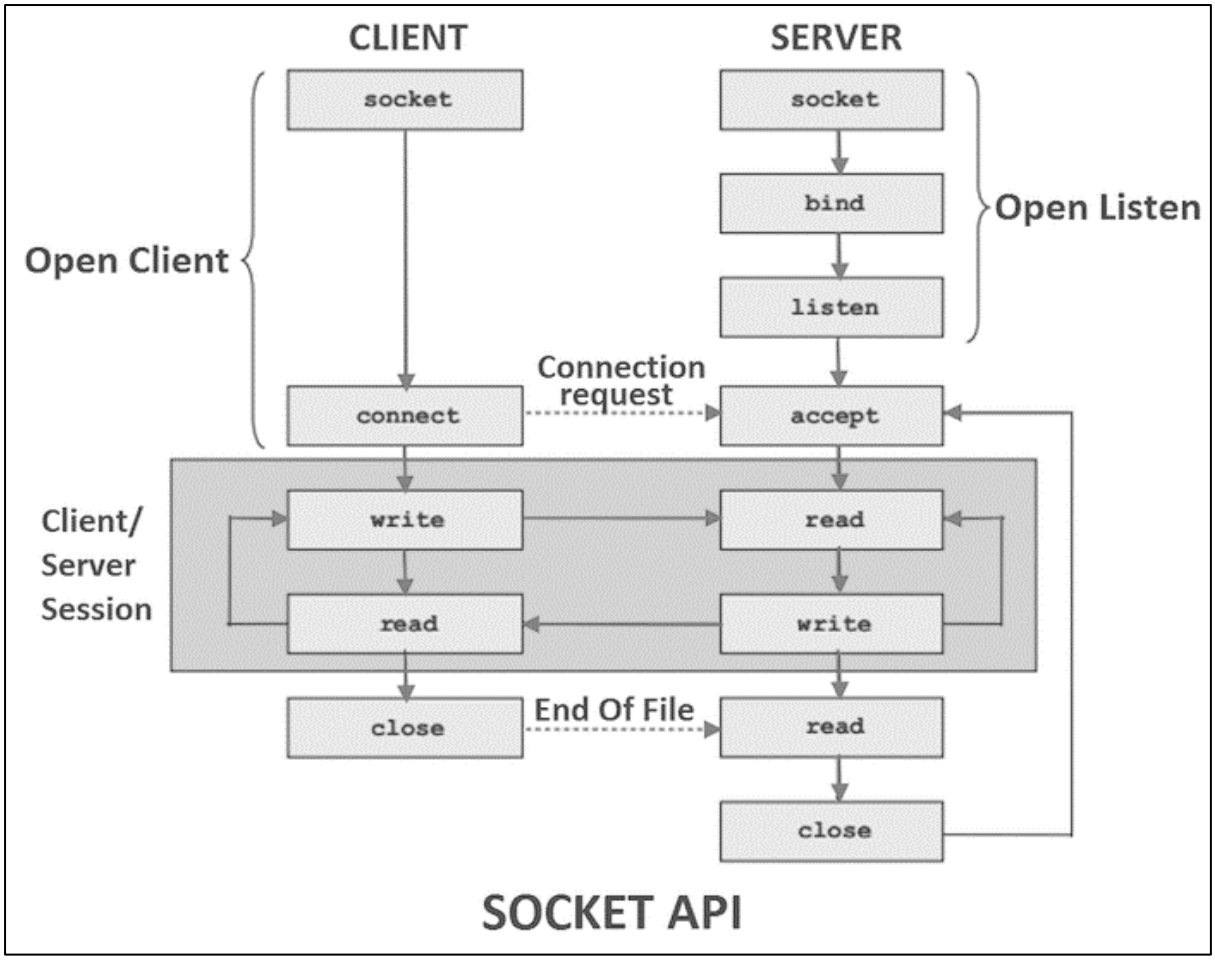


Figure 2.3 Socket API

CHAPTER 3
SYSTEM REQUIREMENT

3.1 Development Environment

1. Hardware Configuration

- Intel i5
- 8GB Ram

2. Software Configuration

- OS – Windows 10 pro
- Software used – Apache NetBeans IDE 17
- Development Environment: JAVA

CHAPTER 4

IMPLEMENTATION OF PROJECT

4.1. STEP FOR IMPLEMENTATION

- Set up GUI
- Create a server-side implementation
- Create a client-side implementation.
- Integrate the GUI with server and client
- Test the Application.

4.2. Brief of IMPLEMENTATION

1. Set up the GUI and start a fresh Java Swing project[3]

   Create the chat application's user interface using Swing elements including JFrame, JTextArea, JTextField, and JButton.

2. Establish a server-side implementation.[3]

   Make a new server-specific Java class. Create a ServerSocket object and assign it a certain port. To establish a connection with clients, create a socket object.

3. Implement a client-side solution.[3]

   For the client, make a new Java class. Create a client-facing GUI using Swing components. To establish a connection with the server, create a Socket object.

4. Integrate the GUI with the client and server[3]

   Include event listeners in the GUI elements, such as buttons. Put messages-sending and -receiving procedures into practise.

5. Run an application test.[3]

   Run the server programme on a particular port. Launch numerous client programmes and link them to the server. Messaging between clients should be tested.

4.3. TECHNOLOGY USED

- **Apache NetBeans IDE 17**

Apache NetBeans is an open-source integrated development environment (IDE) for Java, HTML, JavaScript, PHP, and other languages. It provides a range of features and tools to aid in software development, including code editing, debugging, and project management.

➢ **Features of Apache NetBeans IDE 17**

- ❖ Cross-Platform Support
- ❖ Language Support
- ❖ Project Management
- ❖ Code Editing & Refactoring
- ❖ Debugging & Profiling
- ❖ GUI Development
- ❖ Database Connectivity

Figure 4.3 Apache NetBeans logo

- **Socket Programming:** To establish communication channels between the client and server.
- **Java Swing:** To create the graphical user interface (GUI) for the chat application.

CHAPTER 5
UNDERSTANDING THE APPLICATION & RESULT

5.1 Understanding the Application

This project can be mainly divided into two modules:

1. Server
2. Client

This project is mainly depended on client/server model. The client requests the server and server responses by granting the client's request. The proposed system should provide both 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 client's requests. The client come and go 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.

1. THE SERVER SCREEN

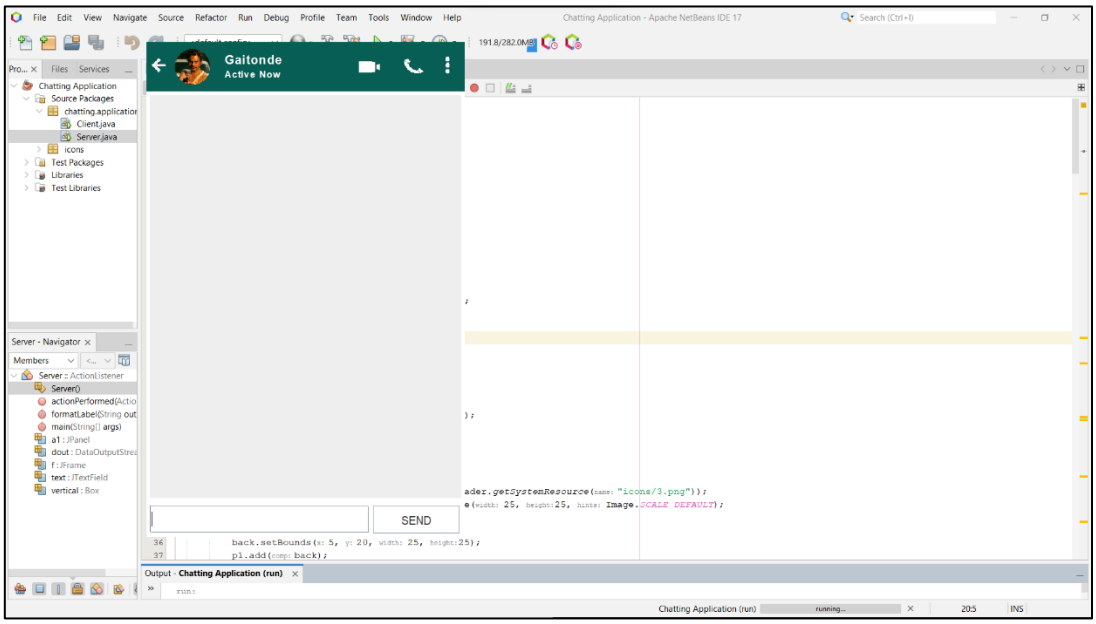


Figure 5.1.1 Server Screen

- **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.

2. THE CLIENT SCREEN

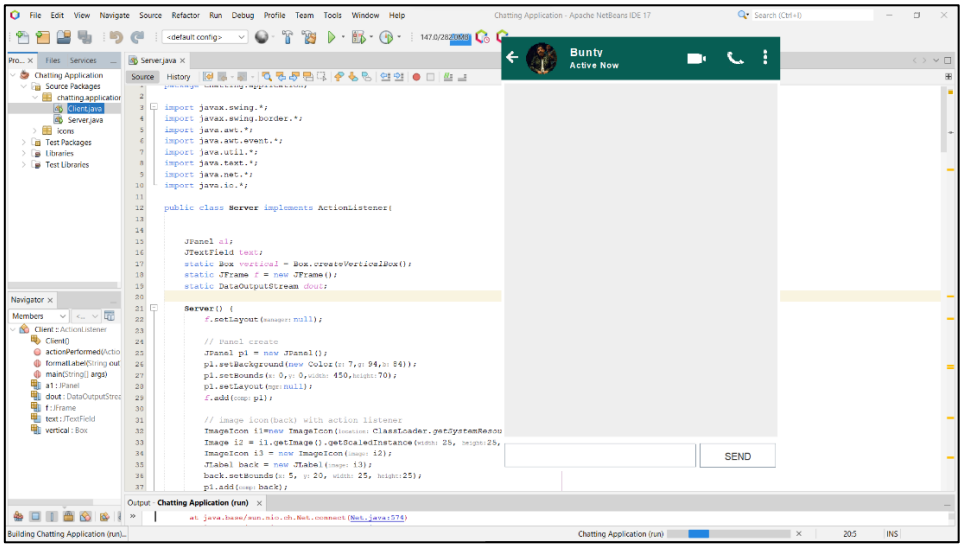


Figure 5.1.2 Client Screen

3. THE CLIENT SERVER SCREEN

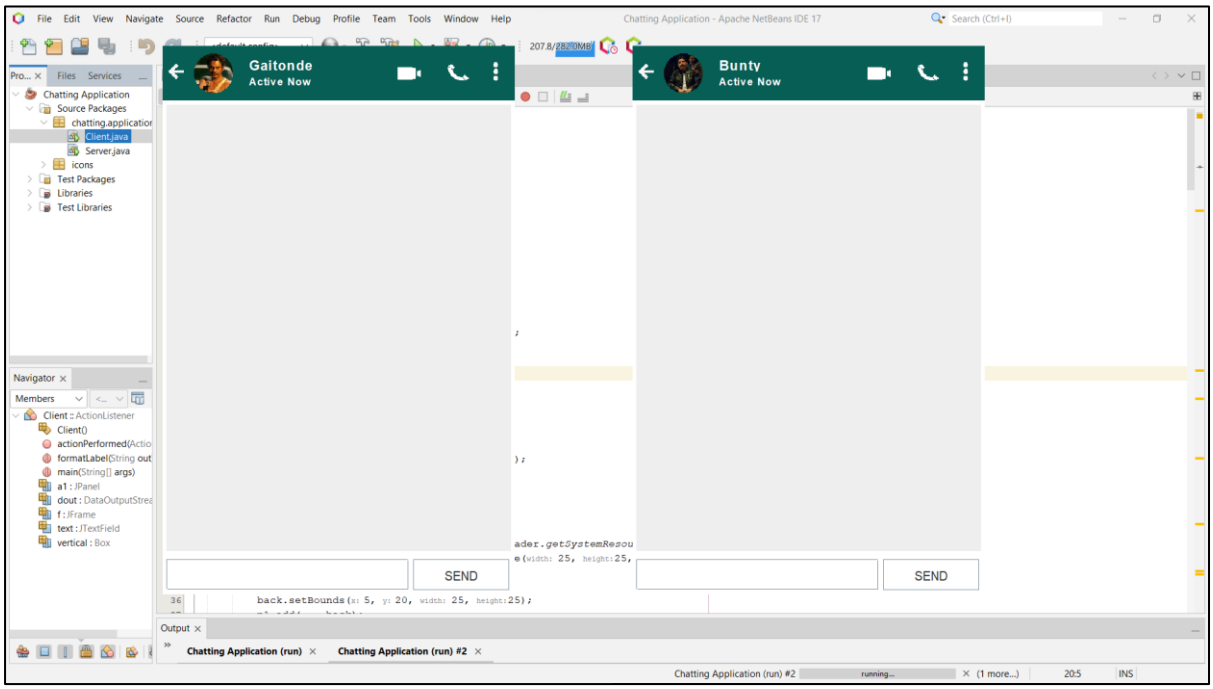


Figure 5.1.3 Client and Server Screen

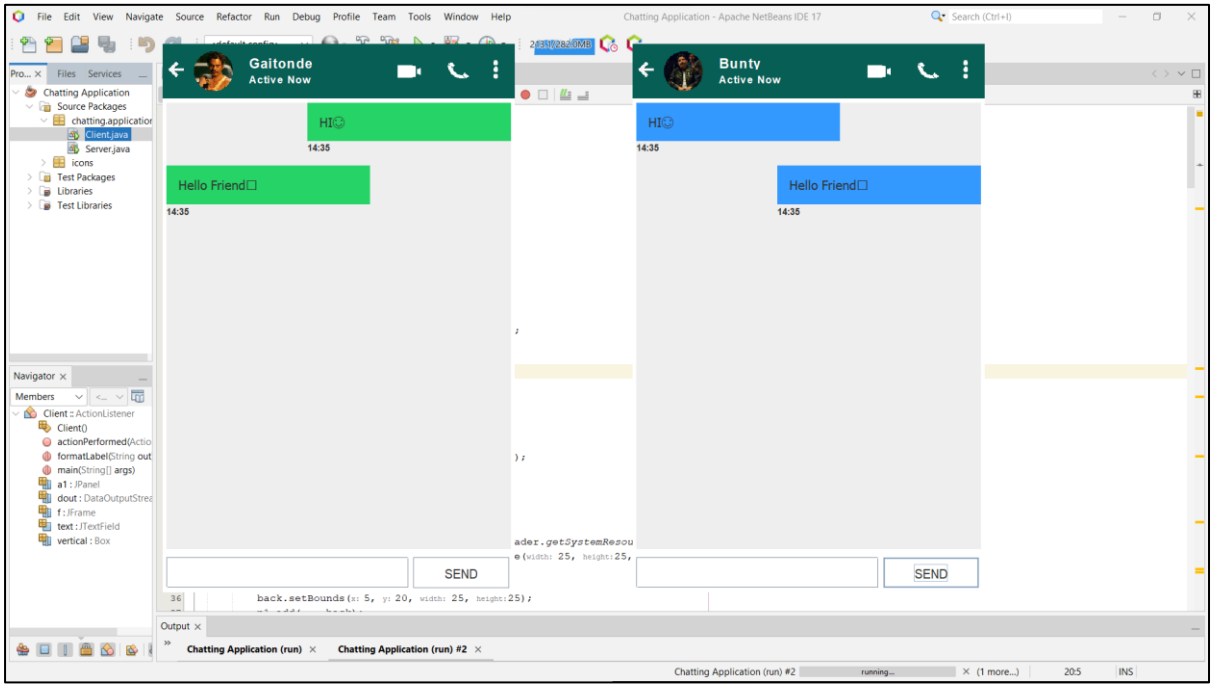


Figure 5.1.4 Communication process

Fig. 5.1.4 *Depict the Communication between two persons(Gaitonde & Bunty).*

4. USE OF EMOJI IN CHAT

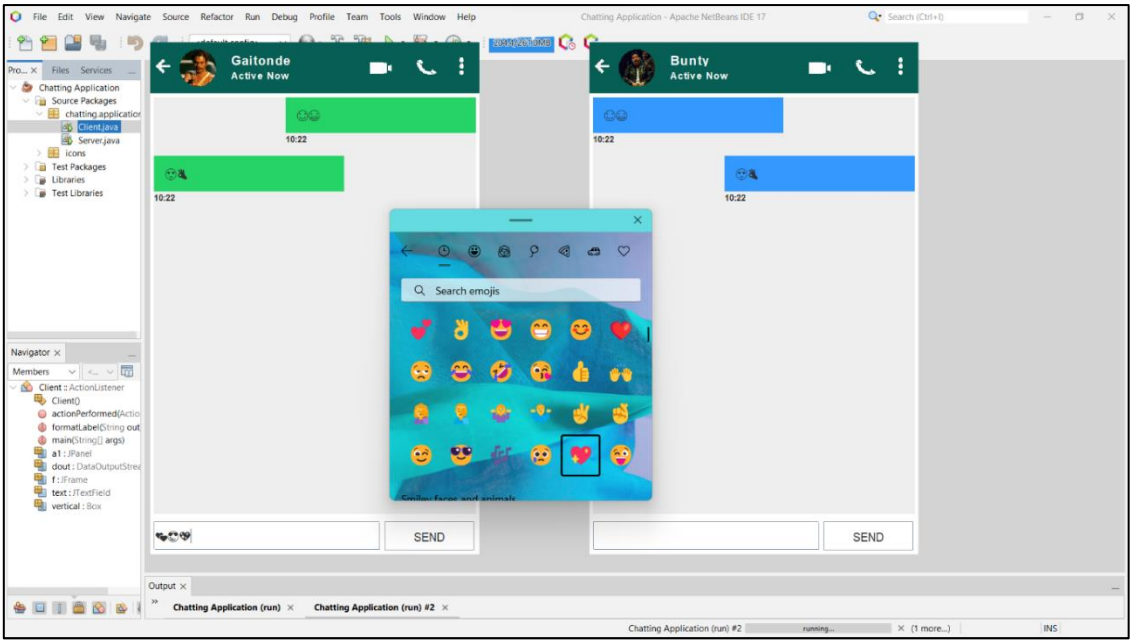


Figure 5.1.5 Use of Emoji

Fig. 5.1.5 *depict the use of Emoji in the chatting Application*

5. CUSTOMIZABLE TEXT COLOR BLOCK

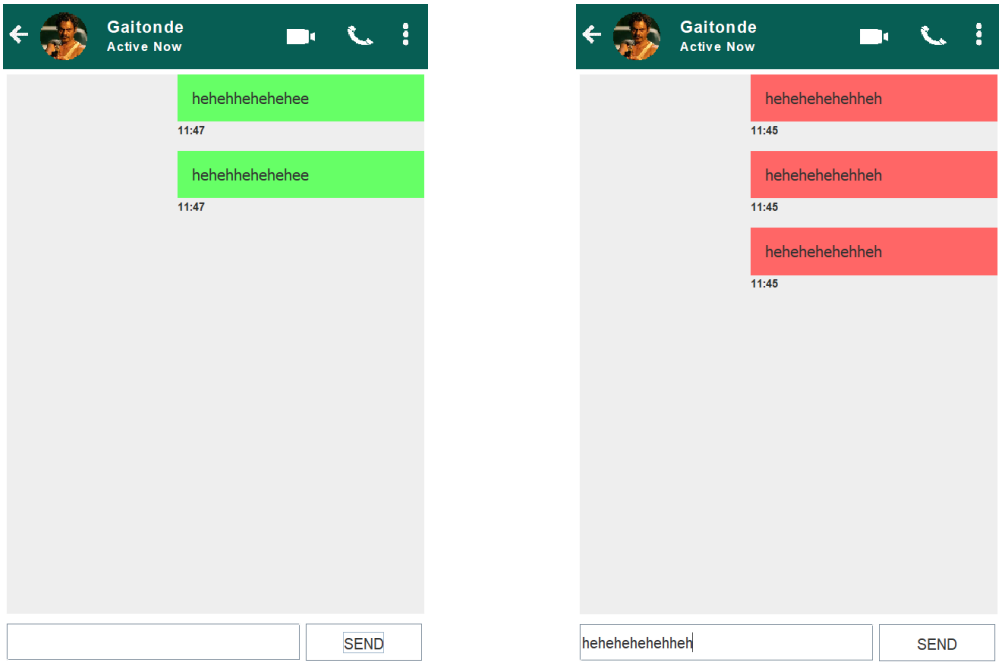


Figure 5.1.6 Customization of Text Box

Fig. 5.1.6 *depict the customization in the chatting Application.*

5.2 FUNCTIONALITIES OF THE APPLICATION

The following essential features are part of the chat application:-

1. **Establishment of the Connection:** Clients can connect to the server by entering its IP address and port number.
2. **Real-time text message exchange** is possible between clients. The chat window displays messages.
3. **Support for many clients**: The server can manage several client connections at once.
4. **User Interface**: The client application has a simple GUI that includes send/receive functions, a message input box.
5. **Error Handling**: To deal with connection drops, timeouts, and other network-related problems, the application includes error handling techniques.

5.3 PROBLEMS ENCOUNTERED

During the development , following challenges were encountered:

1. **Handling Multiple Clients**: Managing multiple client connections concurrently and ensuring synchronized message exchange between clients.
2. **GUI Design and Layout**: Designing an intuitive and user-friendly GUI for the client application that enables smooth interaction with the chat system.
3. **Error Handling and Network Issues**: Implementing error handling mechanisms to deal with network-related issues such as connection failures, timeouts, and data loss.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

I as a Java programmer have created network applications by utilising sockets and Web services.

These programmes are adaptable, effective, and simple to maintain for a broad clientele. The web-based conversation programme we created is distinctive in its features and, more significantly, is simple to customise. An effective and adaptable set of classes for developing network applications is offered by the java.net package. Typically, applications operating on client computers send requests to applications running on a server computer. These concern networking services that the transport layer offers. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the two most used transport protocols on the Internet.

TCP is a connection-oriented protocol that offers two computers a dependable data flow. 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.

6.2 FUTURE SCOPE

There is always a room for improvements in any software package, however good and efficient it may be done. But the most important thing should be flexible 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:

1. **Files transfer**: this will enable the user to send files of different formats to others via the chat- application.
2. **Voice chat**: this will enhance the application to a higher level where communication will be possible via voice calling as in telephone.
3. **Video chat**: this will further enhance the feature of calling into video communication.
4. **Gif** : this is also most important feature , that is mostly used by new generation.

REFERENCE

1. Henriyan, Diotra, et al. "Design and implementation of web based real time chat interfacing server." 2016 6th International Conference on System Engineering and Technology (ICSET). IEEE, 2016.

2. Kumar, T. S., et al. "INTERNET CHAT APPLICATION." International Journal of Advanced Research in Computer Science 12 (2021).

3. Gall, Ulrich, and Franz J. Hauck. "Promondia: a Java-based framework for real-time group communication in the Web." Computer networks and ISDN systems 29.8-13 (1997): 917-926.

4. https://www.javatpoint.com/java-swing

5. https://www.javatpoint.com/socket-programming

6. https://www.geeksforgeeks.org/a-group-chat-application-in-java/

7. https://www.java.com