A Project Report

On

Chatting Application by Java Socket Programming

Submitted in partial fulfillment of the

requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Department of Computer Science and Engineering

Session 2023-24
In
Bachelor of Technology

Submitted By – BT4362

Anubhav Yadav – 20scse1010034
Manvendra Singh – 20scse1010029

Under The Supervision of
Ms. Ambika Gupta
Assistant Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

November 2023

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"Chatting Application by Java Socket Programming"** in partial fulfilment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted to the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out from AUGUST 2023 to DECEMBER 2023 under the supervision of **Ms. Ambika Gupta**, Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering, Galgotias University, Greater Noida.

Anubhav Yadav – 20scse1010034
Manvendra Singh – 20scse1010029

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name

Ms. AMBIKA GUPTA (ASSISTANT PROFESSOR)

CERTIFICATE

This is to certify that Project Report entitled "**Chatting Application by Java Socket Programming**." which is submitted by Anubhav Yadav and Manvendra Singh in partial fulfilment of the requirement for the award of degree B. Tech. in Department of Computer Science and Engineering, Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Program Chair **Signature of Dean**

Date: Nov, 2023

Place: Greater Noida

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Ast. Professor Ms. Ambika Gupta Department of Computer Science & Engineering, Galgotias University, Greater Noida, India *for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.*

We also take the opportunity to acknowledge the contribution of Professor (Dr.) ………., Head, Department of Computer Science & Engineering, Galgotias University, Greater Noida, India *for his full support and assistance during the development of the project.*

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:

Name :

Roll No.:

Date :

Signature:

Name :

Roll No.:

Date :

ABSTRACT

The aim of our project is to design and develop a chatting application with the help of JAVA Socket programming that also lets the user chat with another person or creating any group and chat with multiple people at the same time. The project will follow an agile methodology that consists of several iterations of planning, analysis, design, implementation, testing, and deployment. The expected outcome of this project is a fully functional and user-friendly chatting application that can be accessed from any web browser. The application will allow users to creating and join chat rooms, send and receive text messages and manage their contacts and groups, and create room for data transfer as well. This application is kind of an IC application that stands for Instant Chatting Application which connects users directly from person to person or via a group. This application will be evaluated by conducting user testing and feedback sessions, as well as measuring various performance metrics. The project will also document the design decisions, implementation details, testing results, and future work of the application. This application shows the use of JAVA Socket and Swing programming as well.

TABLE OF CONTENTS

hhhh

List Of Figures

Acronyms

ICA	Interconnected Chatting Application
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
JDK	Java Development Toolkit
IDE	Integrated Development Environment
AWT	Abstract Window Toolkit

CHAPTER: 1

Introduction to Project

1.1) Overview

Chatting has become a major part of people's lives nowadays. We can directly connect to our family and friends or with anyone without having to call them and we can easily convey our message to them. Chatting is a part of the technology that connects people and brings them closer regardless of how far they are by removing any geographical barriers. Communication that provides a real-time transmission of written text-type messages from one person’s device to another person’s device is called Chatting. Chatting has been there for many years but the recent advancement in the technology sector and availability of the Internet at very low prices in India have become major causes of the sudden trend in Chatting. The basic features of any chatting application are person-to-person chat and group chat. A person-to-person chat lets the user chat directly to another contact. A group chat however lets the user create a room and add different people to that room and have a conversation with them all at the same time. Our project aims to do the same. Our project intent is to create a server through which all of the contacts are connected and chatting takes place by the clients. It is a desktop-based Web application. Our project is built up by two applications combined.

1. **Client-Side Application**: This application runs on the user's computer.
2. **Server-Side Application**: This application can run on any computer which is connected to the network.

First, the client side should get connected to the internet and then connect to the server side. There are two major parts of the JAVA Socket programming i.e. TCP and UDP.

CHAPTER: 2

Literature Survey/Comparative Study:

We did the proper literature survey and studied about different-different proposed systems and already built systems. In the already available system, there are very advanced chatting applications like WhatsApp, Viber, and Telegram.
There are many proposed systems as well that use different technologies that are more advanced and somewhat better than these.

Some of them are listed below.

1. The first one is by Ashutosh Kumar and Atul Singh. Their research paper discusses the development of a chat application based on Java multi-threading and network concepts, allowing for private and public messaging and resource sharing. The proposed application aims to create a decentralized system to enhance robustness and security, avoiding the reliance on centralized databases.[1]

2. The second one is by Mr. Waghule Sourabh Rajesh. There web-based chat application developed in this project is reliable, and secure, and offers potential for future enhancements. These chat applications allow for both private and public messaging, as well as the sharing of resources such as files, images, and videos.[2]

3. The third one is by R. Gayathri. The article discusses the importance of chat applications in today's technological world. It highlights the development of a chat system based on Java multithreading and network concepts, allowing for private and public messaging as well as file sharing. The article also mentions the use of Node's WebSocket and REST API components, JSP and servlet technology, and the integration of Natural Language Processing and Machine Learning for improved chat application performance and security.

It further explores the challenges of unethical practices in chat applications and proposes a model for profanity detection. The article concludes with the mention of a real-time chat application developed using MERN stack development and the increasing popularity of end-to-end networking in chat programs.[3]

CHAPTER: 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1) OBJECTIVE:

The main objective of this project is to create a minimal and simple chatting application with group chatting function between a client and a server. This project is to be developed in JAVA language which is executed on a stand-alone java which is single through a network by using the loop back address concept.

This application has two modules which are as follows:

1. Server

2. Client

3.2) Required Tools and Libraries

- Programming Language – JAVA
- Socket Programming
- Swing
- JDK
- Netbeans IDE
- AWT

3.3) Interface Requirements

User Interface

A user interface, often abbreviated as UI, is the point of interaction between a user and a computer program, device, or system. It encompasses the elements and mechanisms that allow users to communicate with and control the software or hardware. User interfaces are critical for making technology accessible and user-friendly.

The design of a user interface is crucial for user experience (UX) and usability. A well-designed UI should be intuitive, efficient, and responsive to the user's needs, making it easier for users to interact with and control the software or hardware. It includes the layout of elements, colour schemes, typography, and the overall user experience design.

We can apply Java API by following two sets:

AWT: AWT, or Abstract Window Toolkit, is a part of Java's standard library used for creating graphical user interfaces (GUIs) in Java applications. It was one of the original GUI libraries provided by Java and is a fundamental part of the Java Foundation Classes (JFC).

AWT provides a set of classes and methods that allow Java developers to create windows, dialogs, buttons, text fields, and other graphical components for their applications. It is a platform-dependent library, meaning that the appearance and behaviour of AWT components may vary from one operating system to another to match the look and feel of the underlying platform.

Here are some key points about AWT in Java:

1. Platform Independence: While AWT provides platform-specific components, it attempts to make GUI programming platform-independent by abstracting some of the underlying platform details. However, this can lead to some variations in how components look and behave on different platforms.

2. Lightweight and Heavyweight Components: AWT components can be categorized into two types: heavyweight and lightweight. Heavyweight components are platform-specific and are drawn by the operating system. Lightweight components are drawn by Java itself, making them more flexible but potentially less integrated with the native platform.

SWING: Swing is a set of GUI (Graphical User Interface) components and a library in Java that provides a rich set of tools and components for building desktop applications with graphical user interfaces. Swing is part of the Java Foundation Classes (JFC) and was introduced to improve upon the older Abstract Window Toolkit (AWT) for creating graphical user interfaces in Java.

Swing has been a popular choice for developing desktop applications in Java for many years due to its flexibility, platform independence, and extensive feature set. While Swing is still a viable option, JavaFX, introduced in Java 8, has gained prominence for its modern and rich features and is often considered a more suitable choice for building Java desktop applications. JavaFX is designed to work seamlessly with Swing, allowing developers to use the best of both worlds if needed.

Here are some of its features:

1. Rich Set of Components: Swing provides a wide range of GUI components, including buttons, labels, text fields, text areas, lists, tables, trees, sliders, progress bars, dialogs, and more. Swing also includes more advanced components like tabbed panes, internal frames, and tables for creating feature-rich desktop applications.

2. Layout Managers: Swing includes a variety of layout managers, such as BorderLayout, FlowLayout, GridLayout, and others, to help arrange components within containers in a flexible and responsive manner.

Chapter: 4

SYSTEM DESIGN

4.1) Concept and Proposed System:

The concept of this project is to create a minimal and simple chatting application with a group chat function between a client and a server. This project is to be developed in JAVA language which is executed on a stand-alone Java which is single through a network by using the loopback address concept.

4.2) Design:

4.2.1) For a personal one-to-one chat:

The main input of this application is given by the user itself:

- Since it is a desktop-based web application to use the application concept one must install any JAVA IDE and have JAVA installed.

- The user must click on the Run button then a connection is established between the user and the server.

- For the connection to be established, the user must start the server before starting the client system.

- After connecting with the user there pops up a frame that contains the user name, back button, and calling and video calling buttons.

- It also contains a text box where a user writes the message and besides that, there is a send button

- For the user to send a message user must click on the send button then it's sent on the network and received by the user.

- When the message is sent then in the background the message is encoded as a packet and decoded when received by the user.

4.2.2) For a group one-to-many chat:

- All things will remain the same but there will be multiple clients in case of one.
- Users must start the server first then all the clients and then they can chat with all of them.
- If one sends the message it can be read by all the group members and each one of them can reply to each one of them.

Chapter:5

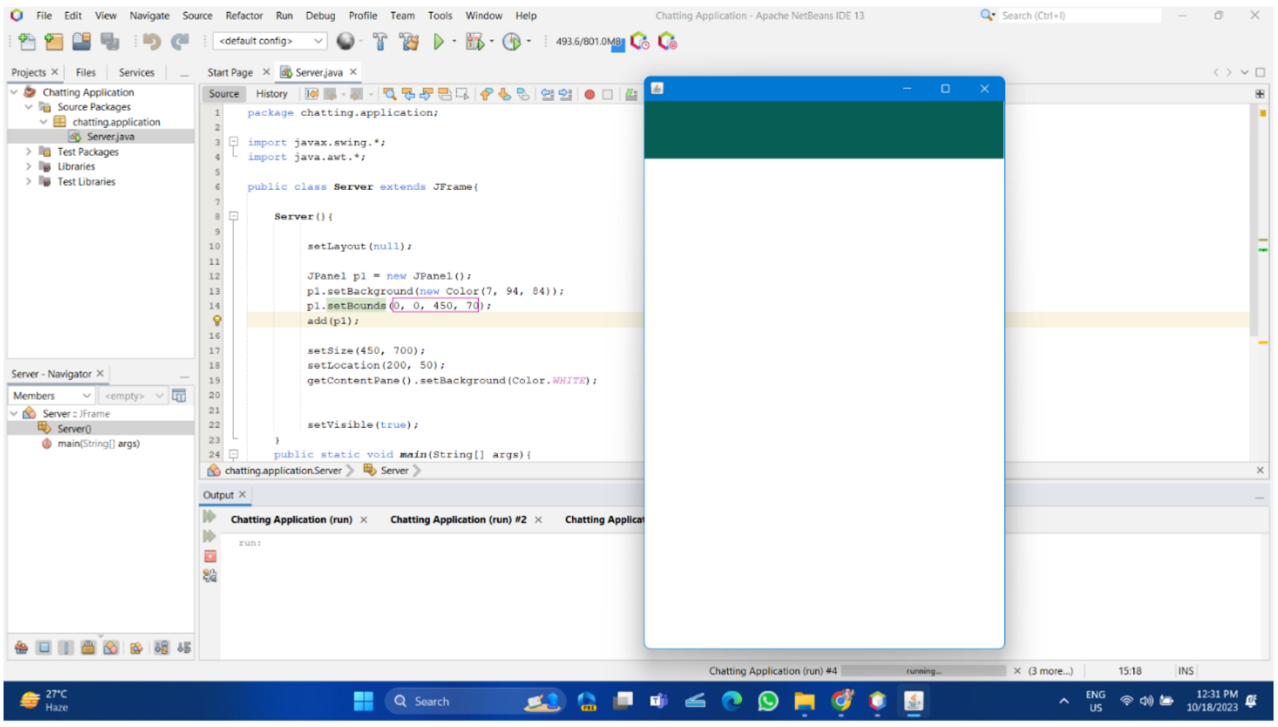
IMPLEMENTATION AND RESULT

5.1) Implementation:

Firstly we need to create the frame that displays the message.

To create the frame we need to import the swing and AWT package.

We create a class called server and create a main function that holds the Server constructor. In that constructor, we create the panel and set its dimensions and colours in the setBackground(Color. WHITE);



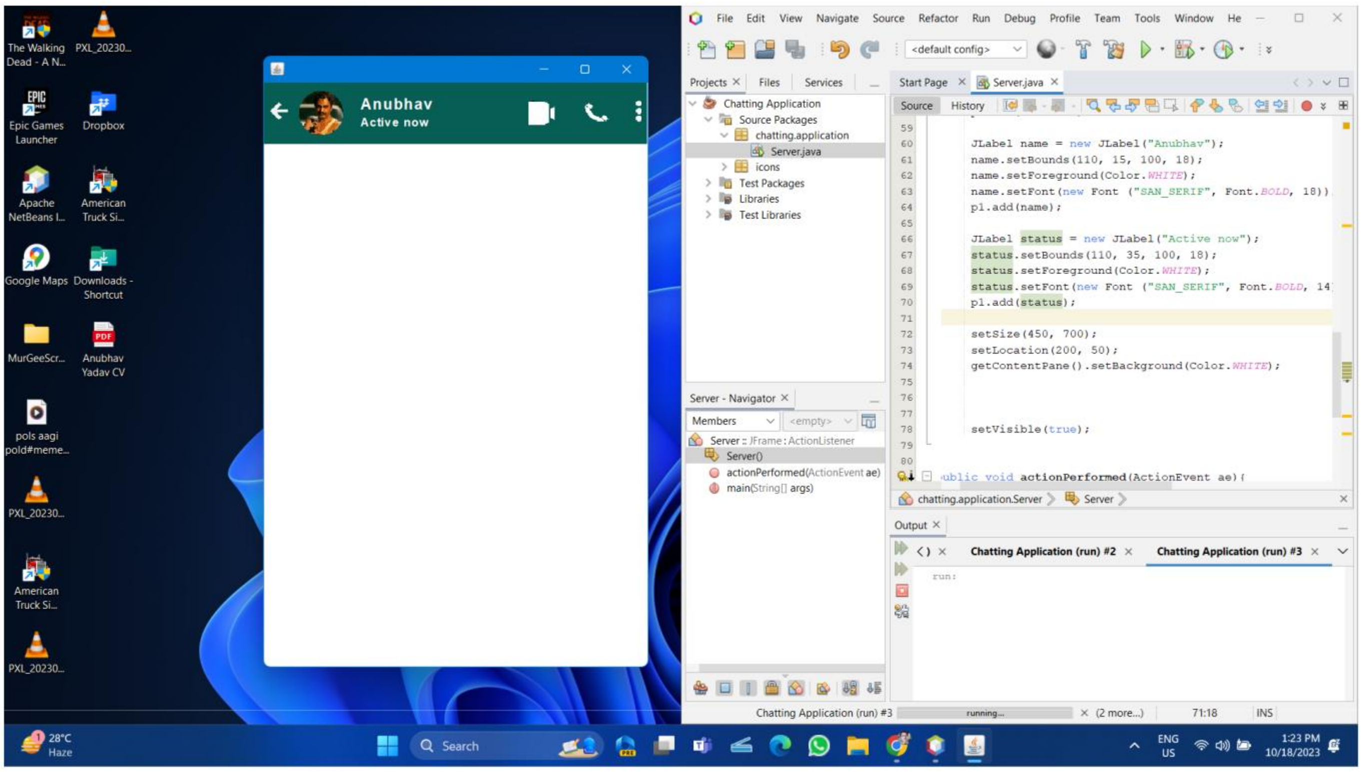
```
package chatting.application;

import javax.swing.*;
import java.awt.*;

public class Server extends JFrame{

    Server(){

        setLayout(null);

        JPanel p1 = new JPanel();
        p1.setBackground(new Color(7, 94, 84));
        p1.setBounds(0, 0, 450, 70);
        add(p1);

        setSize(450, 700);
        setLocation(200, 50);
        getContentPane().setBackground(Color.WHITE);


        setVisible(true);
    }
    public static void main(String[] args){
```

```
run:
```

Fig 1. Creating Frame

After creating the frame we need to add the button user name ,user status and user photo.

So we will create a JLabel and add the user name, and its boundaries and then we will create the status to make it visible on the frame we will add the setVisible function to true so that it will display the changes on the panel.



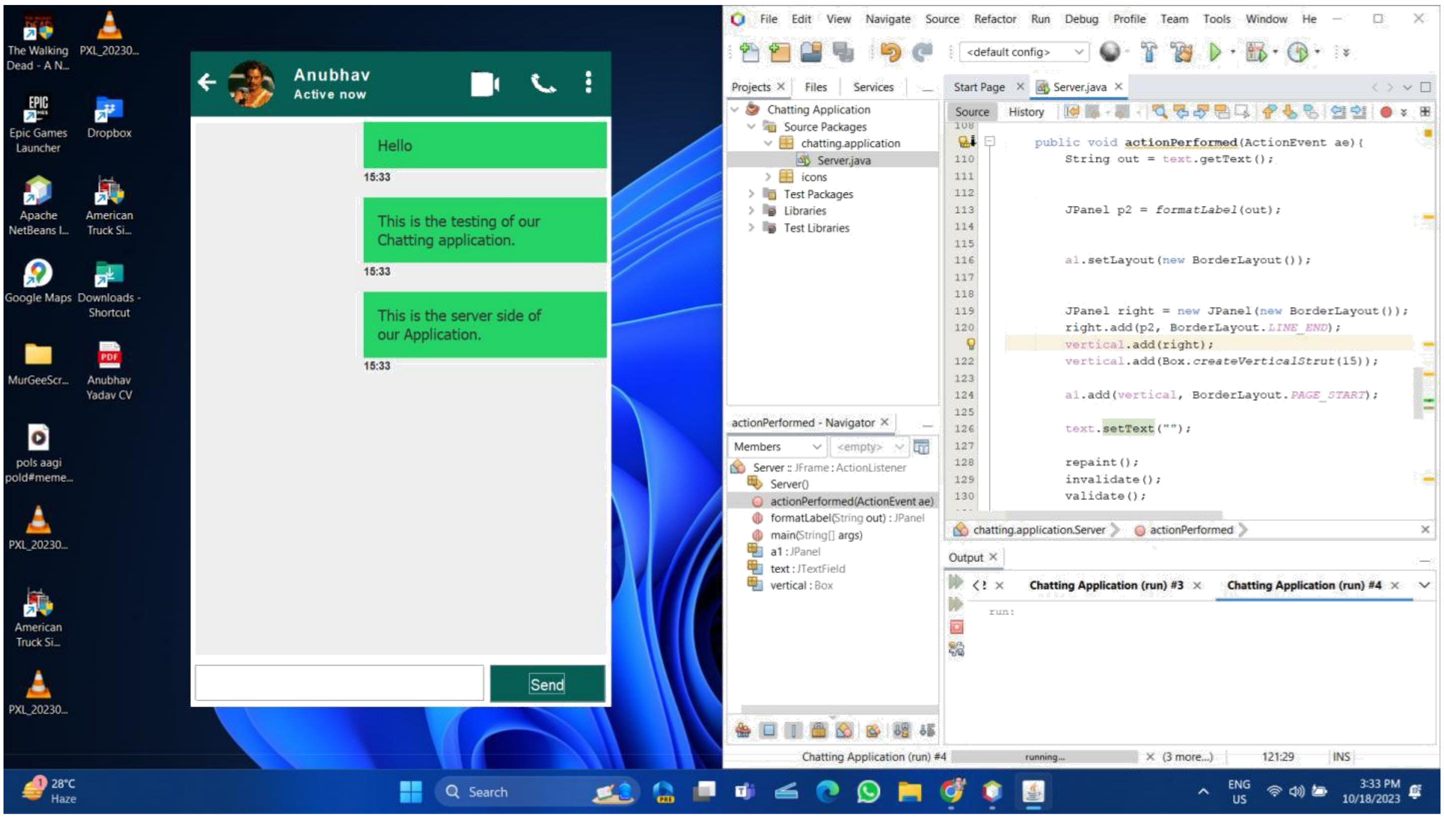
```
        JLabel name = new JLabel("Anubhav");
        name.setBounds(110, 15, 100, 18);
        name.setForeground(Color.WHITE);
        name.setFont(new Font ("SAN_SERIF", Font.BOLD, 18))
        p1.add(name);

        JLabel status = new JLabel("Active now");
        status.setBounds(110, 35, 100, 18);
        status.setForeground(Color.WHITE);
        status.setFont(new Font ("SAN_SERIF", Font.BOLD, 14
        p1.add(status);

        setSize(450, 700);
        setLocation(200, 50);
        getContentPane().setBackground(Color.WHITE);



        setVisible(true);


    ublic void actionPerformed(ActionEvent ae){
```

```
run:
```

Fig 2. Adding Symbols

After that, we added a send button and set its dimensions and create a send button and we have to add a function that shifts all the sent messages to the right of the panel. That is done by the vertical.add(right) function and we have to create a function that removes the text from the text box after the message has been sent so we create a text.setText(“”); function.

We have also added the time in the format of HH: MM format which determines when the text has been sent and is imported by the JAVA Util class.



Hello
15:33
This is the testing of our Chatting application.
15:33
This is the server side of our Application.
15:33

```
    public void actionPerformed(ActionEvent ae){
        String out = text.getText();


        JPanel p2 = formatLabel(out);


        a1.setLayout(new BorderLayout());


        JPanel right = new JPanel(new BorderLayout());
        right.add(p2, BorderLayout.LINE_END);
        vertical.add(right);
        vertical.add(Box.createVerticalStrut(15));

        a1.add(vertical, BorderLayout.PAGE_START);

        text.setText("");

        repaint();
        invalidate();
        validate();
```

```
run:
```

Fig 3. Adding sending functions

After that, we create the client with the same code and then we connect the client to the server via socket programming.

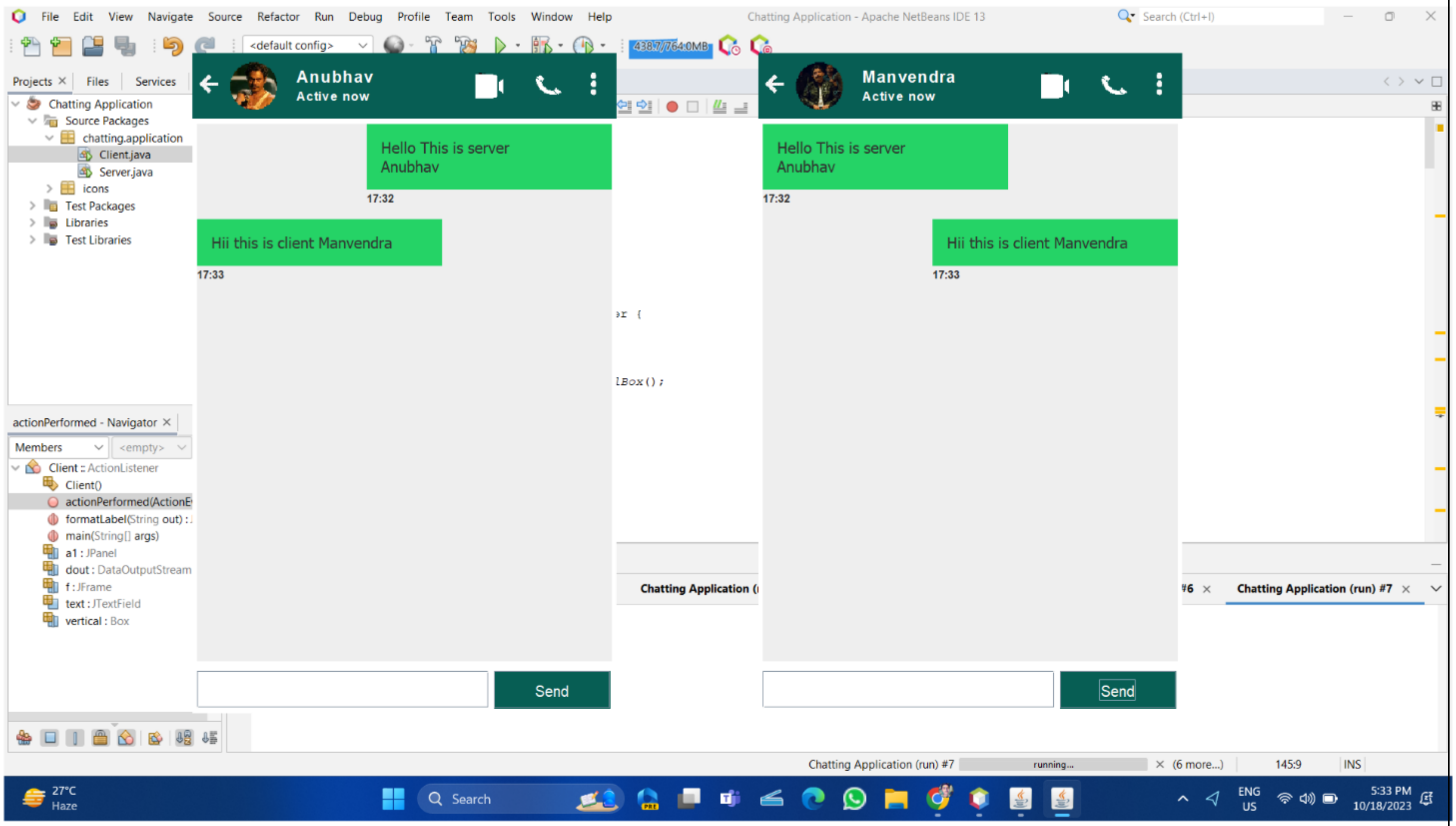
A server is a computer or a software program that provides services, resources, or functionality to other computers, known as clients, over a network. Servers play a crucial role in networked computing by handling requests and delivering data or services to clients.

Servers can run different operating systems, including Linux, Windows, and others, depending on the specific server's role and requirements. They typically have higher processing power, memory, and storage capacity than client computers to handle the demands of serving multiple clients simultaneously. Servers are a fundamental component of the modern networked computing environment, enabling various online services and applications to function.

A server application often monitors a certain port while waiting for client connection requests.

The client and server create a specific connection via which they can communicate when a connection request comes in. A local port is given to the client during connecting. Binding a socket to that number. By writing to the socket, the client communicates with the server and receives reading data from the server to obtain information. The server also receives a new local port number (it needs a new port number in order to keep monitoring the original port for connection requests. port).

Additionally, the server binds a socket to a local port and uses it to read from and write to the client. The protocol, or the language in which information is exchanged back and forth across the connection, must be agreed upon by the client and the server. A server typically has a socket that is connected to a certain port number and runs on a particular machine. The server is passively waiting for a client to request a connection while listening to the socket.



Hello This is server Anubhav
17:32
Hii this is client Manvendra
17:33

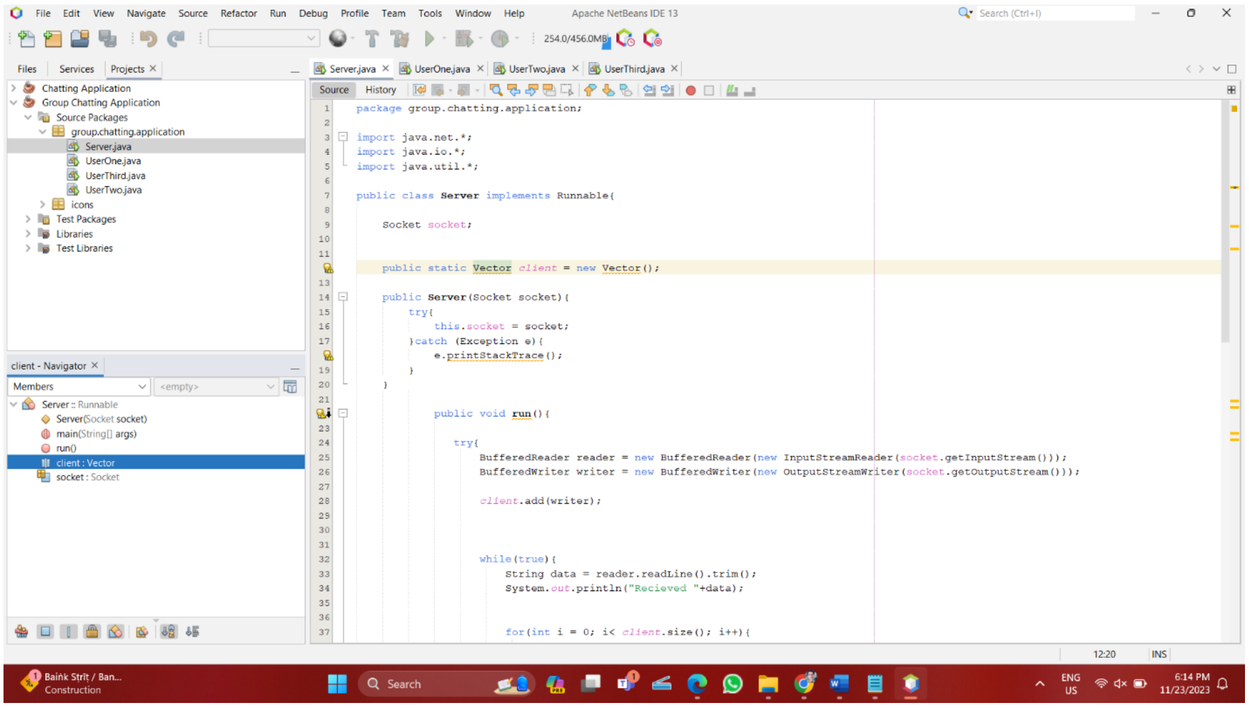
Hello This is server Anubhav
17:32
Hii this is client Manvendra
17:33

Fig 4. Creating Client and connecting it via a socket

We will take the frame from the previous code and implement it in the same code for Group chatting Application. Here same code will be applicable for the frame and we connect it to the server class through Socket Programming.

Same Algorithm goes for group chatting there we create a server and connects all the clients to the server via socket.

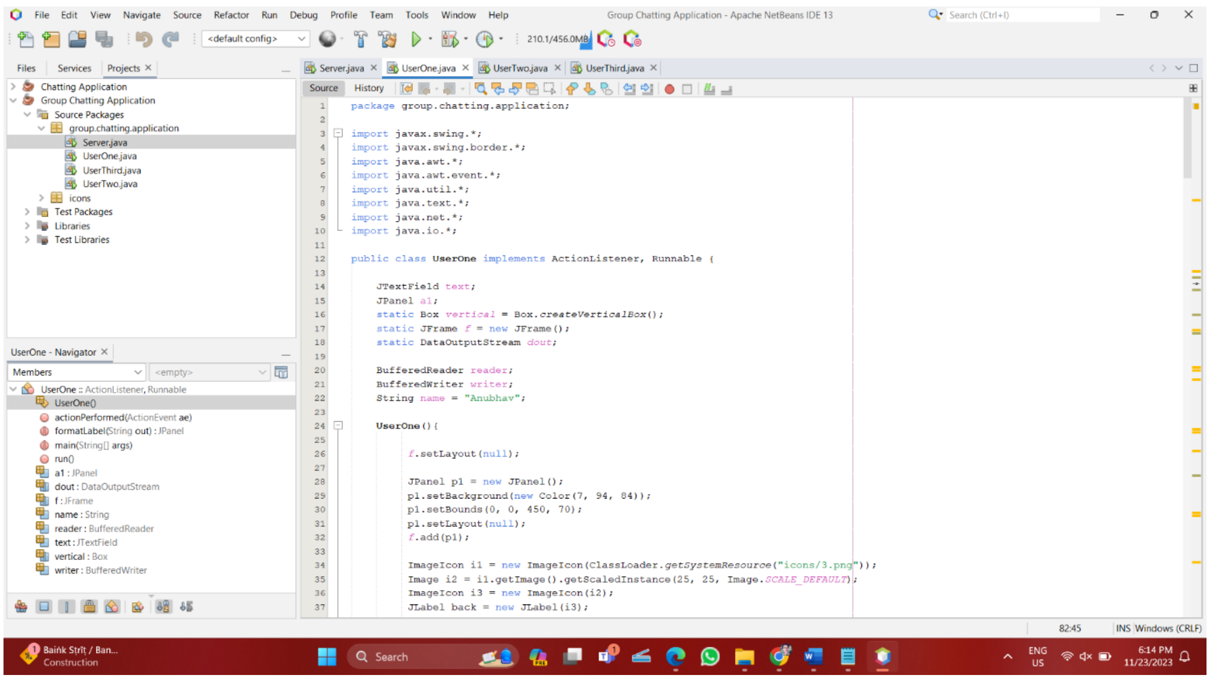
First of all we create The server class :-



```java
package group.chatting.application;

import java.net.*;
import java.io.*;
import java.util.*;

public class Server implements Runnable{

    Socket socket;


    public static Vector client = new Vector();

    public Server(Socket socket){
        try{
            this.socket = socket;
        }catch (Exception e){
            e.printStackTrace();
        }
    }

            public void run(){

               try{
                   BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                   BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

                   client.add(writer);



                   while(true){
                       String data = reader.readLine().trim();
                       System.out.println("Recieved "+data);


                       for(int i = 0; i< client.size(); i++){
```

Fig 5. Server Class

Then we create UserOne with the same code as previous use

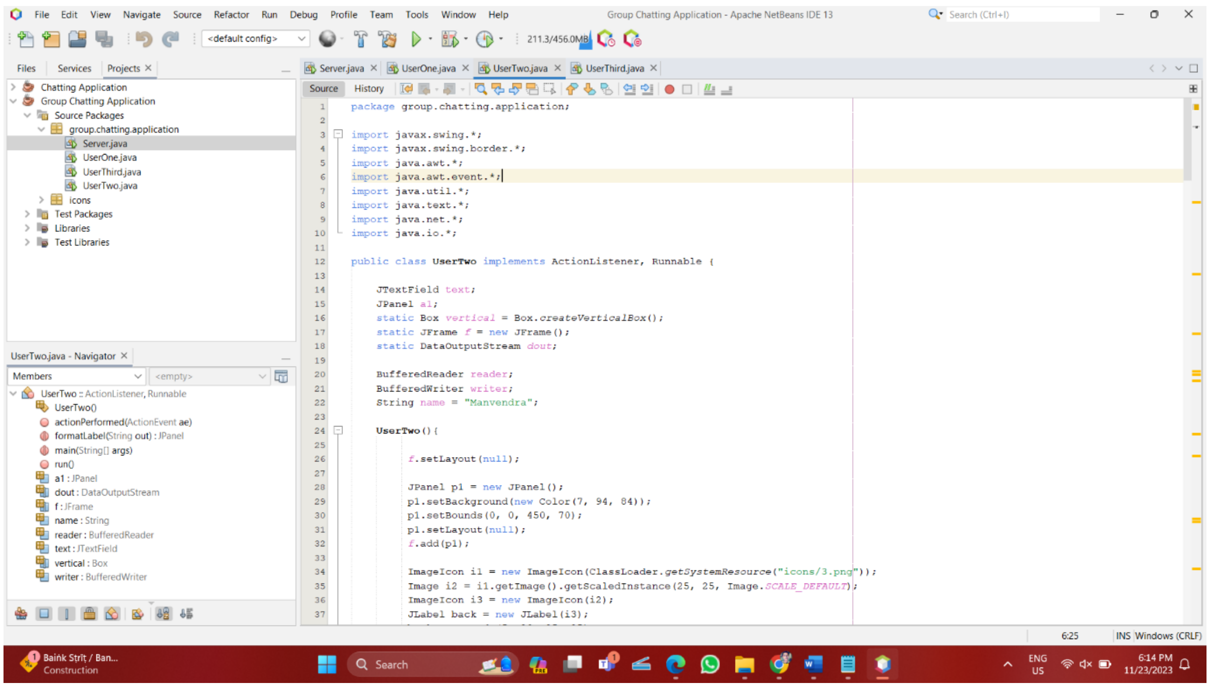


```java
package group.chatting.application;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.net.*;
import java.io.*;

public class UserOne implements ActionListener, Runnable {

    JTextField text;
    JPanel a1;
    static Box vertical = Box.createVerticalBox();
    static JFrame f = new JFrame();
    static DataOutputStream dout;

    BufferedReader reader;
    BufferedWriter writer;
    String name = "Anubhav";

    UserOne(){

        f.setLayout(null);

        JPanel p1 = new JPanel();
        p1.setBackground(new Color(7, 94, 84));
        p1.setBounds(0, 0, 450, 70);
        p1.setLayout(null);
        f.add(p1);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/3.png"));
        Image i2 = i1.getImage().getScaledInstance(25, 25, Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel back = new JLabel(i3);
```

Fig 6. UserOne Class

In this we name the first user and uploaded its profile picture.

Then we create UserTwo with the same code as previous use

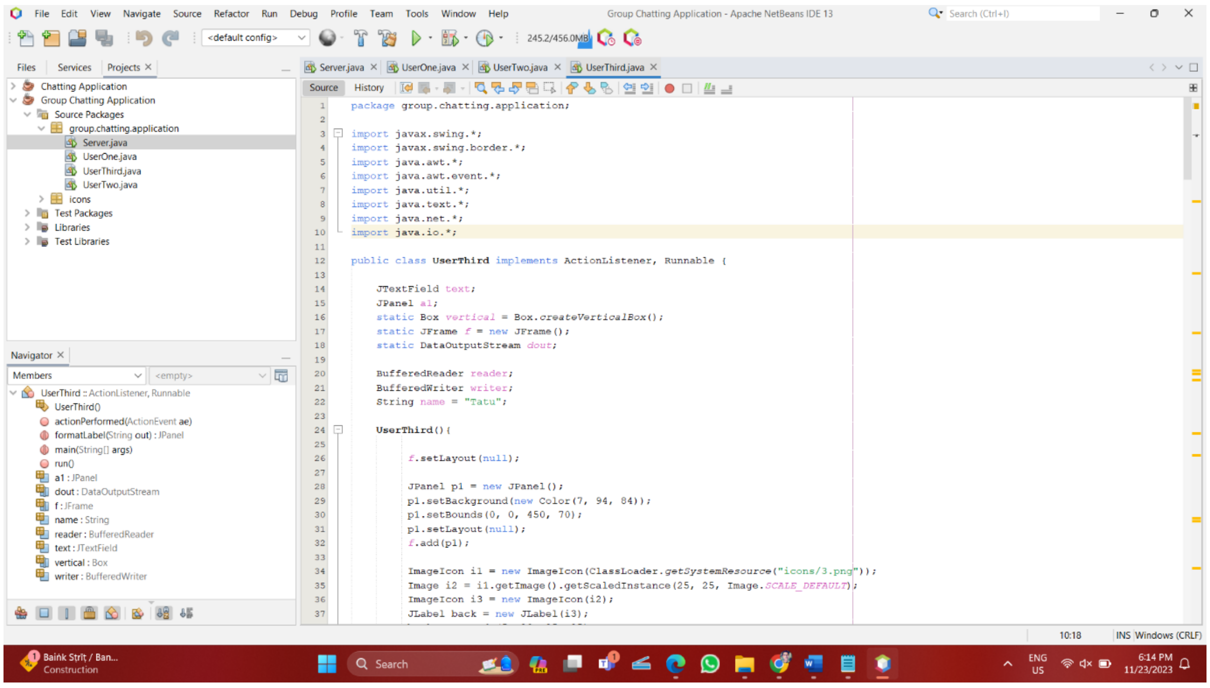


```java
package group.chatting.application;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.net.*;
import java.io.*;

public class UserTwo implements ActionListener, Runnable {

    JTextField text;
    JPanel a1;
    static Box vertical = Box.createVerticalBox();
    static JFrame f = new JFrame();
    static DataOutputStream dout;

    BufferedReader reader;
    BufferedWriter writer;
    String name = "Manvendra";

    UserTwo(){

        f.setLayout(null);

        JPanel p1 = new JPanel();
        p1.setBackground(new Color(7, 94, 84));
        p1.setBounds(0, 0, 450, 70);
        p1.setLayout(null);
        f.add(p1);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/3.png"));
        Image i2 = i1.getImage().getScaledInstance(25, 25, Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel back = new JLabel(i3);
```

Fig 7. UserTwo Class

In this we name the second user and uploaded its profile picture.

Then we create UserThird with the same code as previous use.



```java
package group.chatting.application;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.net.*;
import java.io.*;

public class UserThird implements ActionListener, Runnable {

    JTextField text;
    JPanel a1;
    static Box vertical = Box.createVerticalBox();
    static JFrame f = new JFrame();
    static DataOutputStream dout;

    BufferedReader reader;
    BufferedWriter writer;
    String name = "Tatu";

    UserThird(){

        f.setLayout(null);

        JPanel p1 = new JPanel();
        p1.setBackground(new Color(7, 94, 84));
        p1.setBounds(0, 0, 450, 70);
        p1.setLayout(null);
        f.add(p1);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/3.png"));
        Image i2 = i1.getImage().getScaledInstance(25, 25, Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel back = new JLabel(i3);
```

Fig 8. UserThird Class

In this we name the third user and uploaded its profile picture.

Code Explanation :

Java code in Server Class represents a basic implementation of a simple server for a group chat application using sockets.

1. Server Class :

1.1. Server Class Overview:

a) **Constructor:**

The Server class has a constructor that takes a Socket as a parameter. This constructor is used to initialize the socket field with the provided socket.

b) **run Method:**

The class implements the Runnable interface, suggesting that instances of this class can be executed in a separate thread.
The run method is the main logic for handling communication with a connected client.

1.2. Vector for Client Writers:

The Vector named client is a collection used to store BufferedWriter objects. Each BufferedWriter is associated with a client and is used to write messages to that client.

1.3. run Method Details:

a) Buffered Readers and Writers:

BufferedReader and BufferedWriter are used to read from and write to the client's socket stream, respectively.

b) Client Registration:

Each new client's BufferedWriter is added to the client vector, effectively registering the client for future broadcasts.
Infinite Loop for Receiving Messages:

The while(true) loop continuously listens for messages from the connected client using reader.readLine().

c) Message Broadcasting:

When a message is received from a client, it is trimmed and printed.
The code then iterates through all the registered clients (BufferedWriter objects) in the client vector and sends the received message to each one using bw.write(data) and bw.flush().

1.4. Main Method:

a) **Server Socket Initialization:**

The main method starts by initializing a ServerSocket on port 2003.

b) **Infinite Loop for Accepting Clients:**

The server enters an infinite loop where it accepts incoming client connections using s.accept().
For each accepted connection, a new Server instance is created, and its run method is executed in a separate thread.

1.5. Explanation:

The server continuously listens for incoming client connections.
Each connected client has its own thread (Server instance) to handle communication independently.
Messages sent by any client are broadcasted to all connected clients.
The Vector client is used to keep track of connected clients and their BufferedWriter objects.

1.6. Potential Improvements:

Exception handling should be enhanced for robustness.
Security considerations, such as encryption, should be addressed for secure communication.

2. User Class:

This Java code represents the implementation of a basic graphical user interface (GUI) for a chat application. The application has a chat window for sending and receiving messages.

2.1. UserOne Class Overview:

Attributes:

- JTextField text: A text field for typing messages.
- JPanel a1: A panel for displaying the conversation.
- static Box vertical: A vertical box for organizing message panels.
- static JFrame f: The main frame of the GUI.
- static DataOutputStream dout: A data output stream for writing data to the server.
- BufferedReader reader: A buffered reader for reading data from the server.
- BufferedWriter writer: A buffered writer for writing data to the server.
- String name: The name of the user ("Anubhav" in this case).
- Constructor: Initializes the GUI components, connects to the server using a socket, and initializes the reader and writer.

2.2. GUI Initialization:

The GUI includes a header panel (p1) with user information, a conversation panel (a1) for displaying messages, and an input area (text and send button) for typing and sending messages.

2.3. ActionPerformed Method:

Invoked when the "Send" button is clicked.
Constructs an HTML-formatted string (out) representing the user's message.
Creates a formatted panel (p2) for displaying the message in the conversation panel.
Appends the formatted panel to the vertical box (vertical).
Writes the message to the server using the writer and clears the input text field.

2.24. FormatLabel Method:

Creates a JPanel with formatted labels for displaying messages.
The method takes the formatted message string (out) as input and creates a JLabel with a specified font and background color.
Adds the message label and a timestamp label to the panel.

2.5. Run Method (implements Runnable):

Continuously reads messages from the server using the reader.
Constructs a formatted panel (panel) for displaying the received message.
Appends the panel to the vertical box (vertical).
Updates the conversation panel and the main frame to reflect the new messages.

2.6. Main Method:

Creates an instance of the UserOne class.
Starts a new thread (t1) to run the run method continuously for reading messages from the server.

2.7. GUI Styling:

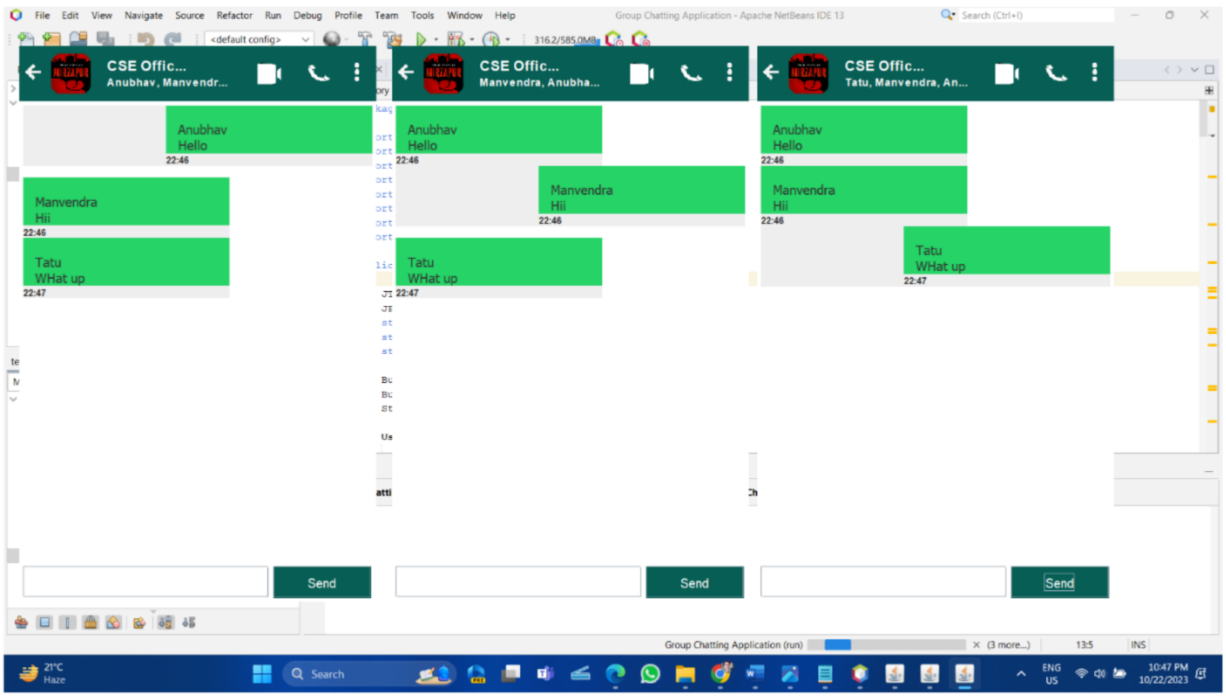
Icons are used for profile pictures and various actions.
Stylish formatting is applied to the message panels, including background colors and borders.

2.8. Notes:

The application assumes a specific message format for distinguishing the user's own messages from others' messages.
The Calendar and SimpleDateFormat are used to display timestamps for messages.

5.2 RESULT

The result of our project is a web-based desktop chat application that provides the facility to chat with other people and share some of our best memories with them.



CSE Offic...
Anubhav, Manvendr...

Anubhav
Hello
22:46

Manvendra
Hii
22:46

Tatu
WHat up
22:47

CSE Offic...
Manvendra, Anubha...

Anubhav
Hello
22:46

Manvendra
Hii
22:46

Tatu
WHat up
22:47

CSE Offic...
Tatu, Manvendra, An...

Anubhav
Hello
22:46

Manvendra
Hii
22:46

Tatu
WHat up
22:47

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Group Chatting Application - Apache NetBeans IDE 13
Search (Ctrl+I)
<default config>
Send
Send
Send
Group Chatting Application (run)
21°C Haze
Search
ENG US
10:47 PM 10/22/2023

Fig 9. Result

Chapter:6

FUTURE SCOPE AND CONCLUISION

6.1 FUTURE SCOPE

Regardless of how well and efficiently anything is done, there is always space for improvement. The most crucial component, however, should be adaptable enough to welcome additional alteration. We just use text messages to communicate at the moment. Future updates to this software could add features like:

File transfer: With this feature, users can transmit files in various formats to others using the chat program.

Voice chat will take the program to a new level by enabling voice calling, much as on a telephone, for conversation.

Video chat will improve the ability to call into video conversation.

We also intend to connect our app with JDBC so that we can store the chats by adding a query in the database so that we can perform chat analysis on it as well.

6.2 CONCLUSION:

.

Java programmers have created network applications by using sockets, threads, 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, simple to use, customizable. 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 commonly 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 more straightforward message-based connectionless protocol that transmits datagrams—packets of information that are sent from one computer to another—with no assurances that they will arrive.

REFERENCES:

[1]. Research paper on Group chatting Application by Ashutosh Kumar and Atul Singh 2019

[2]. Group Chat Application by Mr. Waghule Sourabh Rajesh - June 2022 International Journal of Advanced Research in Science Communication and Technology

[3]. Multi User Chat Application by R. Gayathri - June 2020-International Journal of Engineering and Advanced Technology

[4]. Judith C. Simon, Charles J. Campbell. "Java" , Wiley, 2004

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

[6]. https://www.geeksforgeeks.org/socket-programming-in-java/

[7]. Java Swing Modernization Approach Complete Abstract Representation based on Static and Dynamic Analysis by Zineb Gotti July 2016 11th International Conference on Software Engineering and Applications