CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Project Name: Chatting Application using Java

This project focuses on developing a simple yet functional real-time chat application using Java. The application employs a client-server architecture, enabling multiple users to communicate via text messages over a local network.

Submitted by:

Tanisha Jain

24MCI10047

24MAM 3/B

Submitted To:

Tanya Sagwal

Github Link:

Acknowledgement

I would like to express my sincere gratitude to Tanya Sagwal, for their invaluable guidance, support, and encouragement throughout the development of this "Real-Time Chat Application" project. Their expertise and insights were instrumental in the successful completion of this work.

I also extend my thanks to Department UIC and Chandigarh University for providing the necessary resources and facilities.

Finally, I would like to acknowledge the support of my peers and family, whose encouragement and understanding made this project possible.

<u>**Certificate**</u>

This is to certify that project entitled “Real time Chat application” has been successfully completed by Tanisha Jain

The project is a Bonafide work and fulfils the requirements for the Masters of Computers Application and course code – 24CAP-652 conducted during my Second semester 2025 academic Session

The project demonstrates a comprehensive understanding of Java network programming GUI development and client-server architecture. The student has successfully implemented a real time chat application, showcasing their ability to apply theoretical knowledge to practical application.

Supervisor Name: Tanya Sagwal

Department: University Institute of Computing

Institution name: Chandigarh University

Date: 30/03/2025

INDEX

FILE STURCTURE SUGGESTION

```
Chat application
|------src/
     |---- chatting/application/
     |----server.java
     |-----client java
|------icons/-
     |----1.png
    |-----2.png
    |----3.png
    |---3 icon.png
|----docs/
```

- ➢ Acknowledgement
- ➢ Certificate
- ➢ Abstract
- ➢ Introduction
- ➢ Objective
- ➢ Aim
- ➢ Tasks
- ➢ Learning outcomes

ABSTRACT

This project develops a real-time chat application enabling users to exchange text messages over a local network. The application comprises a server and a client component, built using Java Swing for the graphical user interface and Java Sockets for network communication. The server manages client connections and message distribution, while the client provides a user-friendly interface for sending and receiving messages. The application demonstrates fundamental concepts of client-server architecture, multithreading, and GUI programming. It offers a simple yet functional platform for instant messaging, suitable for small-scale communication scenarios

INTRODUCTION

Instant messaging has become an integral part of modern communication. This project aims to create a basic real-time chat application, providing users with a platform for instant text-based communication. The application utilizes Java's networking capabilities to establish a client-server communication model.

The primary goal is to develop an application that allows multiple clients to connect to a central server and exchange messages in real-time. This project focuses on the implementation of a simple, functional chat application, emphasizing the core concepts of network programming and GUI development.

The application is designed for local area networks (LANs), showcasing the basic principles of client-server communication and real-time data exchange.

OBJECTIVE

Project Title: Real-Time Chat Application - Detailed Overview

1. **Project Context and Motivation**

- ➢ **Rationale:**
  - Instant messaging has revolutionized communication, enabling rapid and efficient exchange of information.
  - Understanding network programming and real-time communication is crucial for developers in today's interconnected world.
  - This project provides a practical learning experience in implementing fundamental networking concepts using Java.
- ➢ **Motivation:**

- To gain hands-on experience in client-server architecture.
- To develop proficiency in Java Sockets and Java Swing.
- To create a functional application that demonstrates real-time communication.
- To understand how data flows through networks.

2. **Project Goals and Objectives**

- ➢ **Primary Goal**:
  - To develop a fully functional, real-time chat application that enables multiple users to communicate over a local network.

Specific Objectives:

- Implement a robust server application capable of handling multiple client connections.
- Design and develop a user-friendly client application with a graphical interface.
- Establish seamless communication between clients and the server using Java Sockets.
- Ensure real-time message delivery and display.
- Implement message timestamps for accurate communication tracking.
- Develop a well-structured and easy to read code base.
- To properly handle exceptions.

3. **System Architecture**

- **Client-Server Model**:
- The application follows a client-server architecture, where a central server manages communication between multiple clients.
- The server acts as a message broker, receiving messages from clients and distributing them to other connected clients.

Components:

Server:

- Listens for client connections on a specified port.
- Manages client connections using multithreading.
- Receives messages from clients and broadcasts them to all connected clients.
- Handles client disconnections gracefully.

Client:

- Establishes a connection to the server.
- Provides a graphical interface for sending and receiving messages.
- Displays messages in a chat-like format with timestamps.

AIM

The aim of this project is to provide a functional and educational tool for understanding network programming and GUI development in Java. By creating a simple chat application, we aim to:

- Gain practical experience in client-server communication.
- Enhance skills in Java Swing for GUI design.
- Understand the use of Java Sockets for real-time data transfer.
- Develop a working application that demonstrates real-time messaging

Task To be Done

The project involves the following tasks:

1. **Server Development:**
    - Create a server application using Java Sockets to listen for client connections.
    - Implement multithreading to handle multiple client connections simultaneously.
    - Manage message distribution from one client to all connected clients.

2. **Client Development:**
    - Develop a client application with a user-friendly GUI using Java Swing.
    - Implement functionality to send and receive text messages.
    - Display messages in a chat-like interface with timestamps.
    - Establish a connection to the server using Java Sockets.

3. **GUI Design:**
    - Design a clean and intuitive user interface for both the server and client applications.
    - Implement message display and input areas.
    - Include visual cues for connection status and message timestamps.

4. **Network Communication:**
    - Use Java Sockets to establish and maintain connections between clients and the server.
    - Implement data streams for sending and receiving messages.
    - Handle exceptions for network issues.

5. **Testing and Debugging:**
    - Test the application for functionality and stability.
    - Debug any issues related to network communication or GUI display.
    - Ensure proper message delivery and handling of multiple clients.

CODING

Chatting Application

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
this template
 */
package chatting.application;

/**
 *
 * @author jaint
 */
public class ChattingApplication {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Server side

```
package chatting.application;
```

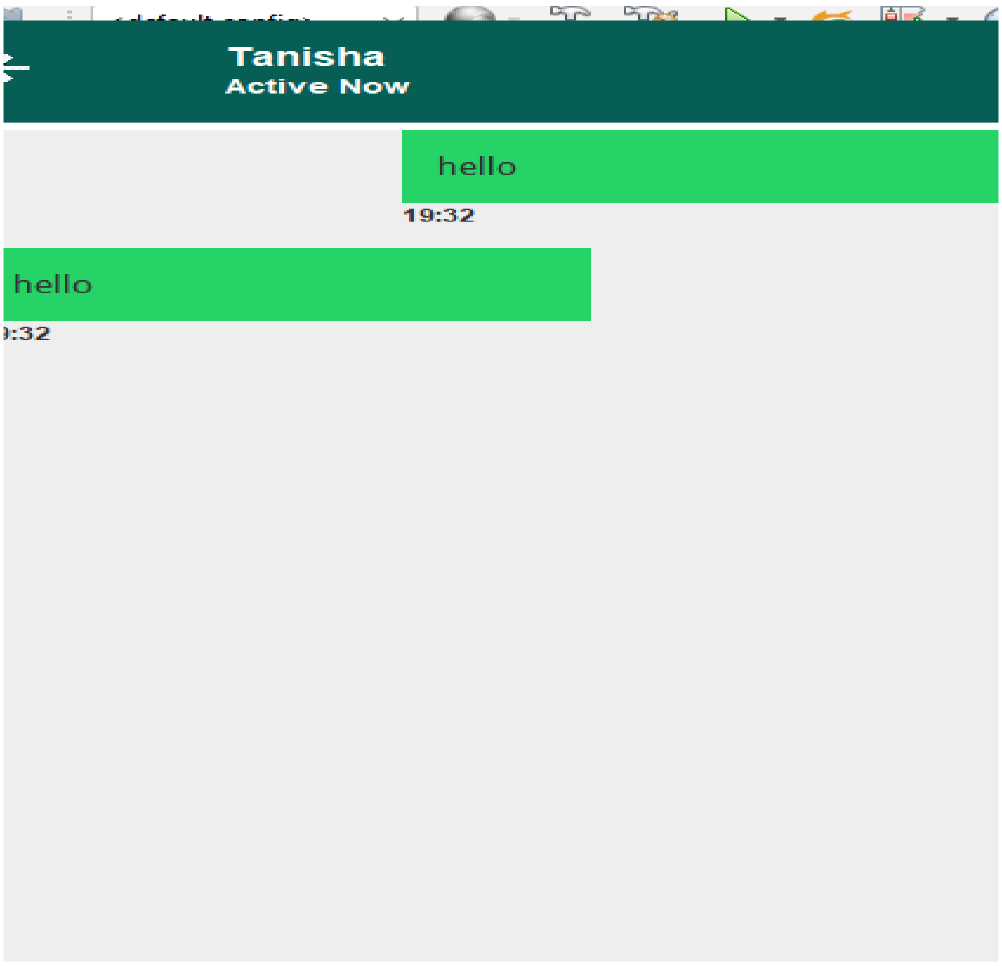
```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.net.*;
import java.io.*;

public class Server implements ActionListener {
    JTextField text;
    JPanel a1;
    JButton send;
    static Box vertical = Box.createVerticalBox();
    static JFrame f = new JFrame();
    static DataOutputStream dout;

    Server() {
        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);
        JLabel back = new JLabel(new ImageIcon(i2));
        back.setBounds(5, 20, 25, 25);
        p1.add(back);

        back.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ae) {
                System.exit(0);
            }
        });

        JLabel name = new JLabel("Tanisha");
        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);

        a1 = new JPanel();
        a1.setBounds(5, 75, 440, 570);
        a1.setLayout(new BorderLayout());
        f.add(a1);

        text = new JTextField();
        text.setBounds(5, 655, 310, 40);
        text.setFont(new Font("SAN_SERIF", Font.PLAIN, 16));
        f.add(text);

        send = new JButton("Send");
        send.setBounds(320, 655, 123, 40);
        send.setBackground(new Color(7, 94, 84));
        send.setForeground(Color.WHITE);
        send.setFont(new Font("SAN_SERIF", Font.PLAIN, 16));
        send.addActionListener(this);
        f.add(send);

        f.setSize(450, 700);
        f.setLocation(200, 50);
        f.setUndecorated(true);
        f.getContentPane().setBackground(Color.WHITE);
        f.setVisible(true);
    }

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

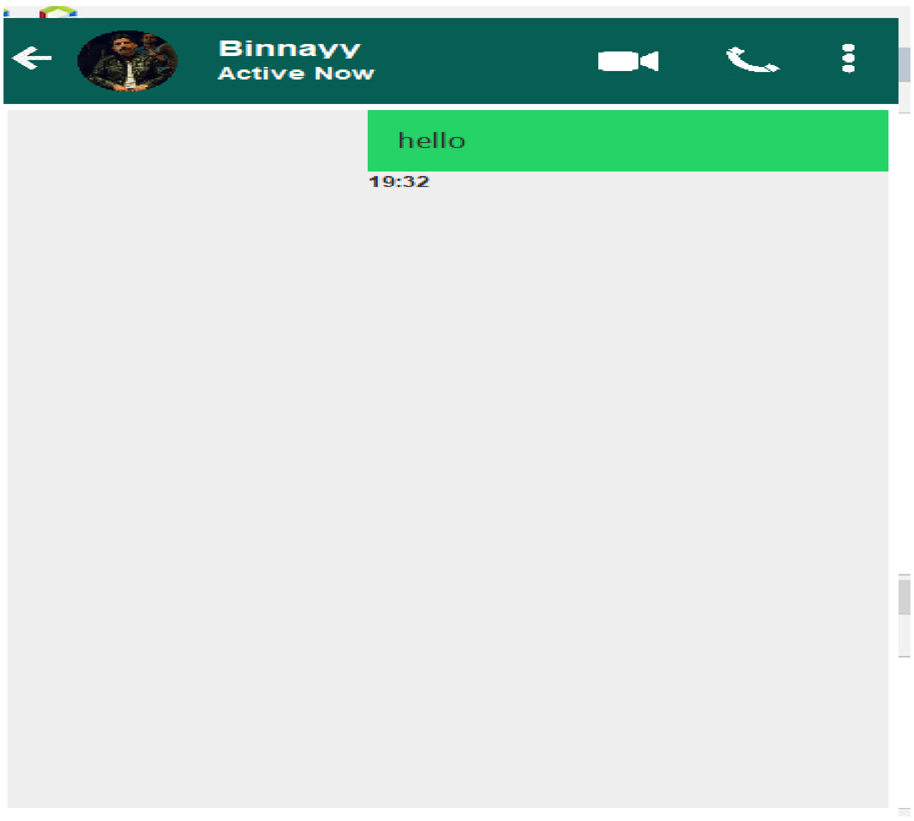
```
            JPanel p2 = formatLabel(out);

            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);
            a1.revalidate();
            a1.repaint();

            dout.writeUTF(out);
            text.setText("");
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
    }

    public static JPanel formatLabel(String out) {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        JLabel output = new JLabel("<html><p style=\"width: 150px\">" + out +
  "</p></html>");
        output.setFont(new Font("Tahoma", Font.PLAIN, 16));
        output.setBackground(new Color(37, 211, 102));
        output.setOpaque(true);
        output.setBorder(new EmptyBorder(15, 15, 15, 50));

        panel.add(output);

        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
        JLabel time = new JLabel(sdf.format(cal.getTime()));
        panel.add(time);

        return panel;
    }

    public static void main(String[] args) {
        new Server();

        try {
            ServerSocket skt = new ServerSocket(6001);
            while (true) {
                Socket s = skt.accept();
                DataInputStream din = new DataInputStream(s.getInputStream());
                dout = new DataOutputStream(s.getOutputStream());

                while (true) {
                    String msg = din.readUTF();
                    JPanel panel = formatLabel(msg);

                    JPanel left = new JPanel(new BorderLayout());
                    left.add(panel, BorderLayout.LINE_START);
                    vertical.add(left);
                    f.validate();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Client side

```
package 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 Client implements ActionListener {

    JTextField text;
    static JPanel a1;
    static Box vertical = Box.createVerticalBox();

    static JFrame f = new JFrame();

    static DataOutputStream dout;

    Client() {

        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);
        back.setBounds(5, 20, 25, 25);
        p1.add(back);

        back.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ae) {
                System.exit(0);
            }
        });

        ImageIcon i4 = new
ImageIcon(ClassLoader.getSystemResource("icons/2.png"));
        Image i5 = i4.getImage().getScaledInstance(50, 50,
Image.SCALE_DEFAULT);
        ImageIcon i6 = new ImageIcon(i5);
        JLabel profile = new JLabel(i6);
        profile.setBounds(40, 10, 50, 50);
        p1.add(profile);

        ImageIcon i7 = new
ImageIcon(ClassLoader.getSystemResource("icons/video.png"));
        Image i8 = i7.getImage().getScaledInstance(30, 30,
Image.SCALE_DEFAULT);
        ImageIcon i9 = new ImageIcon(i8);
        JLabel video = new JLabel(i9);
        video.setBounds(300, 20, 30, 30);
        p1.add(video);

        ImageIcon i10 = new
ImageIcon(ClassLoader.getSystemResource("icons/phone.png"));
        Image i11 = i10.getImage().getScaledInstance(35, 30,
Image.SCALE_DEFAULT);
        ImageIcon i12 = new ImageIcon(i11);
        JLabel phone = new JLabel(i12);
        phone.setBounds(360, 20, 35, 30);
        p1.add(phone);
        ImageIcon i13 = new
ImageIcon(ClassLoader.getSystemResource("icons/3icon.png"));
        Image i14 = i13.getImage().getScaledInstance(10, 25,
Image.SCALE_DEFAULT);
        ImageIcon i15 = new ImageIcon(i14);
        JLabel morevert = new JLabel(i15);
        morevert.setBounds(420, 20, 10, 25);
        p1.add(morevert);

        JLabel name = new JLabel("Binnayy");
        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);

        a1 = new JPanel();
        a1.setBounds(5, 75, 440, 570);
        f.add(a1);

        text = new JTextField();
        text.setBounds(5, 655, 310, 40);
        text.setFont(new Font("SAN_SERIF", Font.PLAIN, 16));
        f.add(text);

        JButton send = new JButton("Send");
        send.setBounds(320, 655, 123, 40);
        send.setBackground(new Color(7, 94, 84));
        send.setForeground(Color.WHITE);
        send.addActionListener(this);
        send.setFont(new Font("SAN_SERIF", Font.PLAIN, 16));
        f.add(send);

        f.setSize(450, 700);
        f.setLocation(800, 50);
        f.setUndecorated(true);
        f.getContentPane().setBackground(Color.WHITE);

        f.setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        try {
            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);

            dout.writeUTF(out);

            text.setText("");

            f.repaint();
            f.invalidate();
            f.validate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static JPanel formatLabel(String out) {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        JLabel output = new JLabel("<html><p style=\"width: 150px\">" + out +
"</p></html>");
        output.setFont(new Font("Tahoma", Font.PLAIN, 16));
        output.setBackground(new Color(37, 211, 102));
        output.setOpaque(true);
        output.setBorder(new EmptyBorder(15, 15, 15, 50));

        panel.add(output);

        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");

        JLabel time = new JLabel();
        time.setText(sdf.format(cal.getTime()));

        panel.add(time);

        return panel;
    }

    public static void main(String[] args) {
        new Client();

        try {
            Socket s = new Socket("127.0.0.1", 6001);
            DataInputStream din = new DataInputStream(s.getInputStream());
            dout = new DataOutputStream(s.getOutputStream());

            while(true) {
                a1.setLayout(new BorderLayout());
                String msg = din.readUTF();
                JPanel panel = formatLabel(msg);
                JPanel left = new JPanel(new BorderLayout());
                left.add(panel, BorderLayout.LINE_START);
                vertical.add(left);

                vertical.add(Box.createVerticalStrut(15));
                a1.add(vertical, BorderLayout.PAGE_START);

                f.validate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

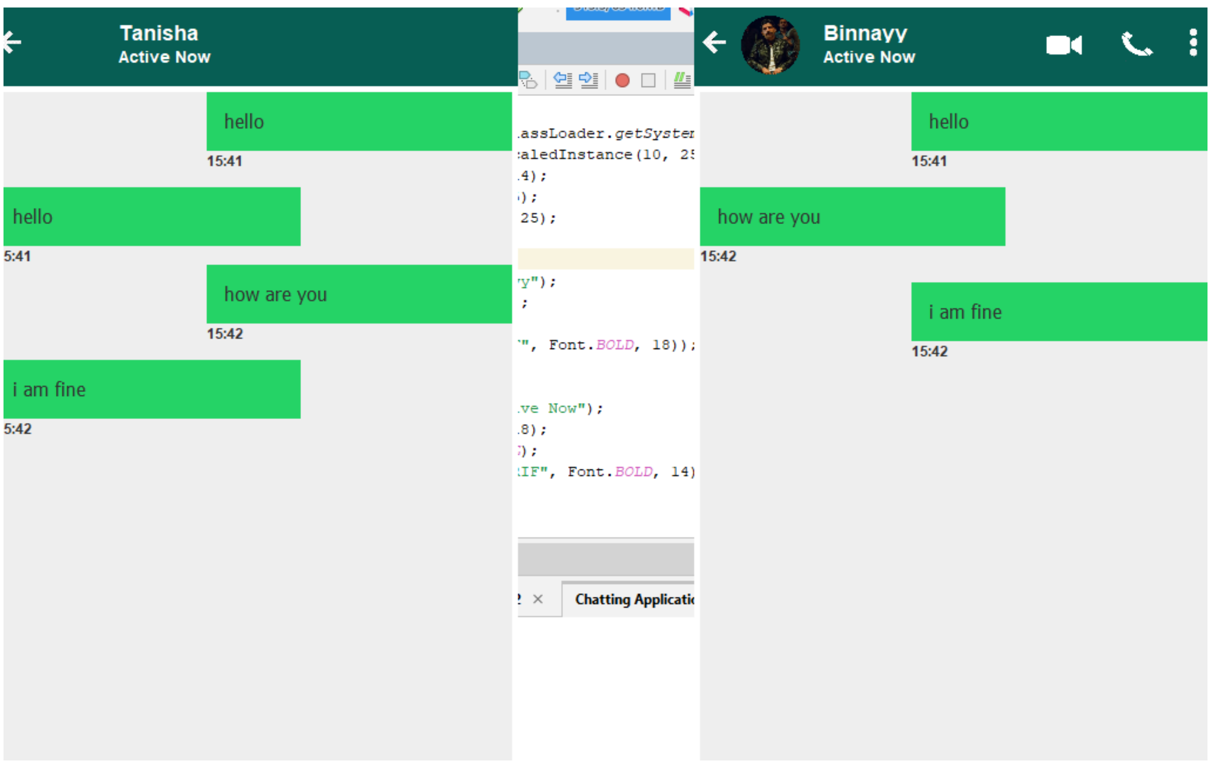
Output


Tanisha
Active Now
hello
19:32
hello
:32



hello
19:32

Chatting



hello
15:41

hello
5:41

how are you
15:42

i am fine
5:42

```
.assLoader.getSyster
:aledInstance(10, 2!
.4);
);
 25);

'y");
;
'", Font.BOLD, 18));

.ve Now");
.8);
);
.IF", Font.BOLD, 14)
```

hello
15:41

how are you
15:42

i am fine
15:42

Learning Outcomes:

Upon completion of this project, the following learning outcomes will be achieved:

- Understanding of client-server architecture and network programming.
- Proficiency in using Java Sockets for network communication.
- Ability to design and implement user interfaces using Java Swing.
- Knowledge of multithreading for handling concurrent connections.
- Experience in handling real-time data exchange.
- Practical skills in testing and debugging network applications.
- Enhanced understanding of Input and Output streams.