A
Mini Project/ Summer Training Report
On

# Chatting Application

Submitted in partially fulfilment for the requirement for the award of the degree of

Bachelor of Technology

in

**Computer Science and Engineering**


Submitted by

**Nitesh Singh**
**(2202840100141)**

Under the Supervision of
**Mr. Bhadrinath Pratap Singh**



Department of Computer Science & Engineering

**UNITED INSTITUTE OF TECHNOLOGY (284)**

(Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow)


**Session-2024-2025**

# Vision of the Department

To be a center of excellence in the field of Computer Science and Engineering for producing talented engineers to ethically serve constantly changing needs of society and industry throughout their career and life.

# Mission of the Department

**M1.** Accomplish excellence with committed faculty by providing theoretical foundation and practical skills for solving complex engineering problems in the state-of-the-art trends in Computer science and allied disciplines.

**M2.** To foster skills and competency, generating novel ideas, entrepreneurship and model creations focused towards deep knowledge, interpersonal skills and leadership.

**M3.** To develop habitude of research among faculty and students in the area of Computer Science & Allied disciplines by providing the desired environment, for addressing the needs of industry and society.

**M4.** To mould the students with ethical principles in thoughts, expression and deeds.

# Index

# Chapter-1
# Introduction

- In the digital era, communication has evolved drastically, with texting and messaging becoming the most prevalent forms of interaction. Chatting applications, commonly referred to as messaging apps, enable users to communicate instantaneously across devices. These applications offer a broad range of features, including text messaging, multimedia sharing, voice and video calls, and group chats. In this project, we will explore the creation of a chatting application using JAVA and JAVA SWING programming language.

- This project aims to explore the development, features, and functionalities of a chatting application, analysing its core components, security measures, and potential use cases in the modern world. The project will also explore the technological stack involved in developing a basic messaging application and how these apps have transformed the way individuals and businesses communicate.

- It usually consists of two phase one is user interface and another is logic behind the code. We have focused here to make a Graphics based Chatting Application.

# Chapter-2
# Requirement

Hardware Requirements:

- Processor: Intel Pentium 4 or high.
- RAM: 4 GB or high.
- Graphics: Intel iRISxe.

Software Requirements:

- Operating System: Windows 7, 8,10 0r 11.
- Disk Space: 1 GB for installation and additional space for projects and libraries.
- Application: NetBeans IDE 22 or Visual Studio code.

# Chapter-3
# Conceptual Background

A **chatting application** is a software program that enables real-time communication between users through the exchange of text, audio, video, and images. In the context of this report, we focus on the development of a desktop-based chatting application using Java Swing. **Java Swing** is a part of the Java Foundation Classes (JFC) used for building Graphical User Interfaces (GUIs). The goal of this section is to provide a conceptual understanding of the fundamental principles behind a chatting application and how Java Swing is employed to implement the user interface for such applications.

At its core, a chatting application facilitates communication between users by enabling them to send and receive messages in real-time. To achieve this, a chatting application typically requires two key components:

- **User Interface (UI):** The visual elements that allow users to interact with the application, such as input fields, buttons, message windows, and other controls.
- **Backend Infrastructure:** The server-side components that handle message storage, user authentication, data transmission, and communication between different clients.

## Structure of the Chatting Application Using Java Swing

The general structure of a Java Swing-based chatting application involves several core functionalities, which are supported by the design of the GUI and underlying logic.

1. **Main Window (JFrame)**

    - The main window is created using the **JFrame** class, which acts as the primary container for the user interface.
    - This window will include a chat area (**JTextArea**), a text field (**JTextField**) for message input, and a send button (**JButton**).
    - The layout of the window is typically managed using layout managers such as **Border Layout** or **Flow Layout** for arranging components efficiently.

1. **Displaying Messages (JTextArea)**

    - A **JTextArea** component will be used to display messages in a scrollable area.
    - When a new message is received or sent, the message is appended to the text area.

- To manage long conversations, the **JScrollPane** is used to add scroll functionality to the **JTextArea**, ensuring that older messages can be accessed as the conversation grows.

1. **Message Input (JTextField)**

    - The user will type their message into a **JTextField** component, which is a single-line text box.
    - This text field will be located at the bottom of the window, with a "Send" button next to it.
    - The user can click the "Send" button or press Enter to submit the message.

2. **Sending Messages (JButton)**

    - The "Send" button is implemented using a **JButton** component. When the user clicks the button, the message in the **JTextField** is captured, added to the message history in the **JTextArea**, and sent to the server for delivery to the intended recipient.

3. **Event Handling**

    - Event-driven programming is central to Java Swing. The application responds to user actions, such as clicking the "Send" button or pressing Enter, through **ActionListener** or **KeyListener** events.
    - In this case, when a message is typed and the "Send" button is clicked, an event handler will process the message, update the UI, and send the message to the server for communication.

4. **Server Communication (Networking)**

    - While Swing handles the GUI, the actual communication between users requires **networking**. A **server** is needed to route the messages from one client to another.
    - **Sockets** are used to establish a connection between the client (the Java Swing application) and the server. Once a connection is established, messages can be sent back and forth between the client and server using **TCP/IP** sockets.
    - In this context, Java's **Socket** and **ServerSocket** classes are used to implement the communication layer, which allows real-time message exchange.

## 4. Flow of Message Exchange in the Application

Here’s a conceptual flow of how the chat application will work:

- **Step 1**: The user enters a message into the **JTextField**.
- **Step 2**: The user clicks the "Send" button (or presses Enter), triggering an event handler that processes the message.
- **Step 3**: The message is displayed in the **JTextArea** and sent to the server.
- **Step 4**: The server receives the message and sends it to the intended recipient (another client).
- **Step 5**: The recipient's client receives the message and displays it in their **JTextArea**.

This flow is repeated for every message exchanged between users.

# Chapter-4

# Implementation

## 4.1 Coding:

- **Client Server**

```
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/1.jpeg"));
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);
   //NAme
JLabel name = new JLabel("Nitesh");
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();
        }
    }
}
```

- **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;
    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);
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);
   }
});
//Image Icon
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);
//Name
JLabel name = new JLabel("Ajay");
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(200, 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 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();
      }
    }
}
```

## 4.2 Flowchart:


*Fig:4.2 Flowchart of Chatting Application.*

# Chapter-5
# User Interfaces

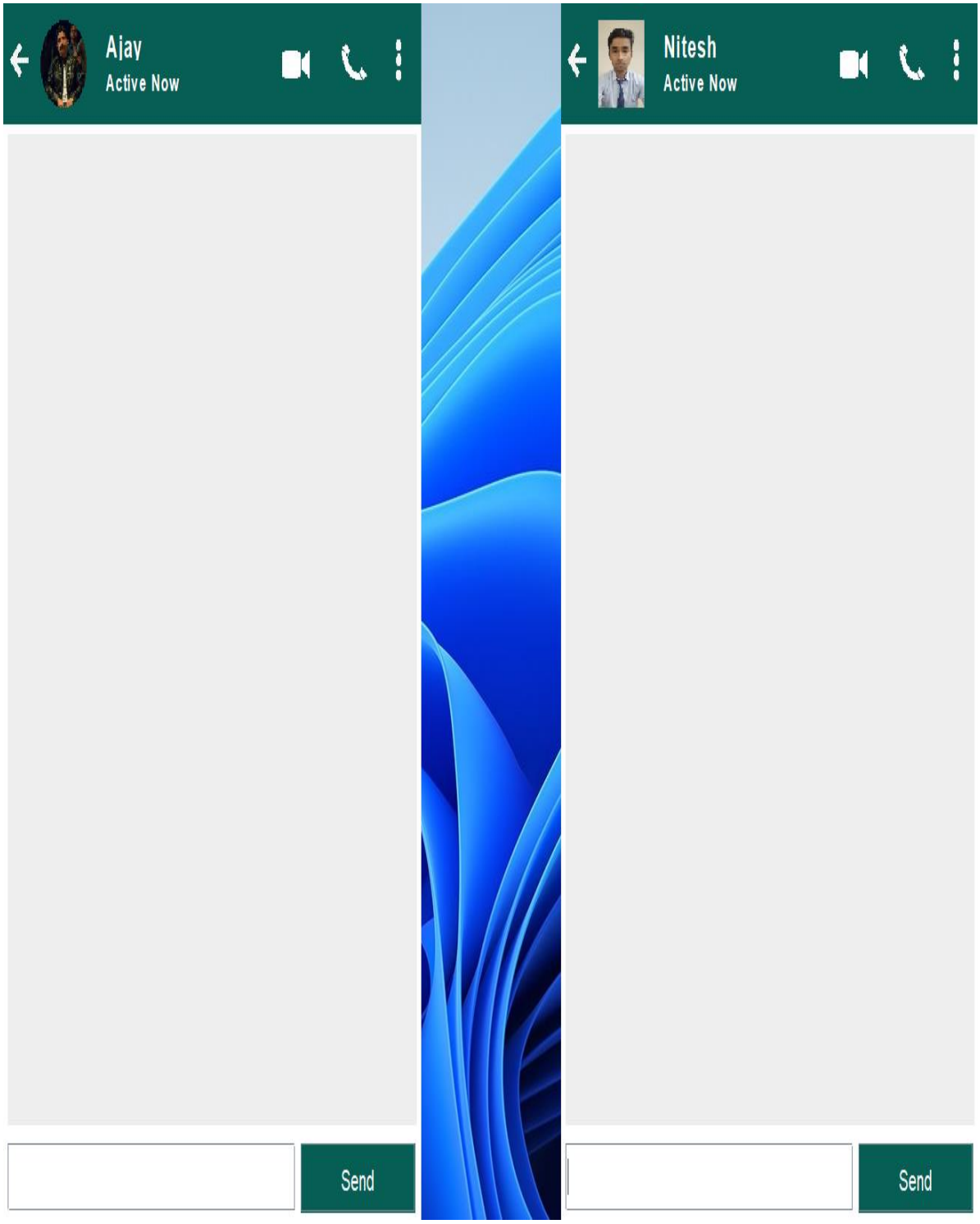


Fig:5.1 (Chatting Application main user interface)

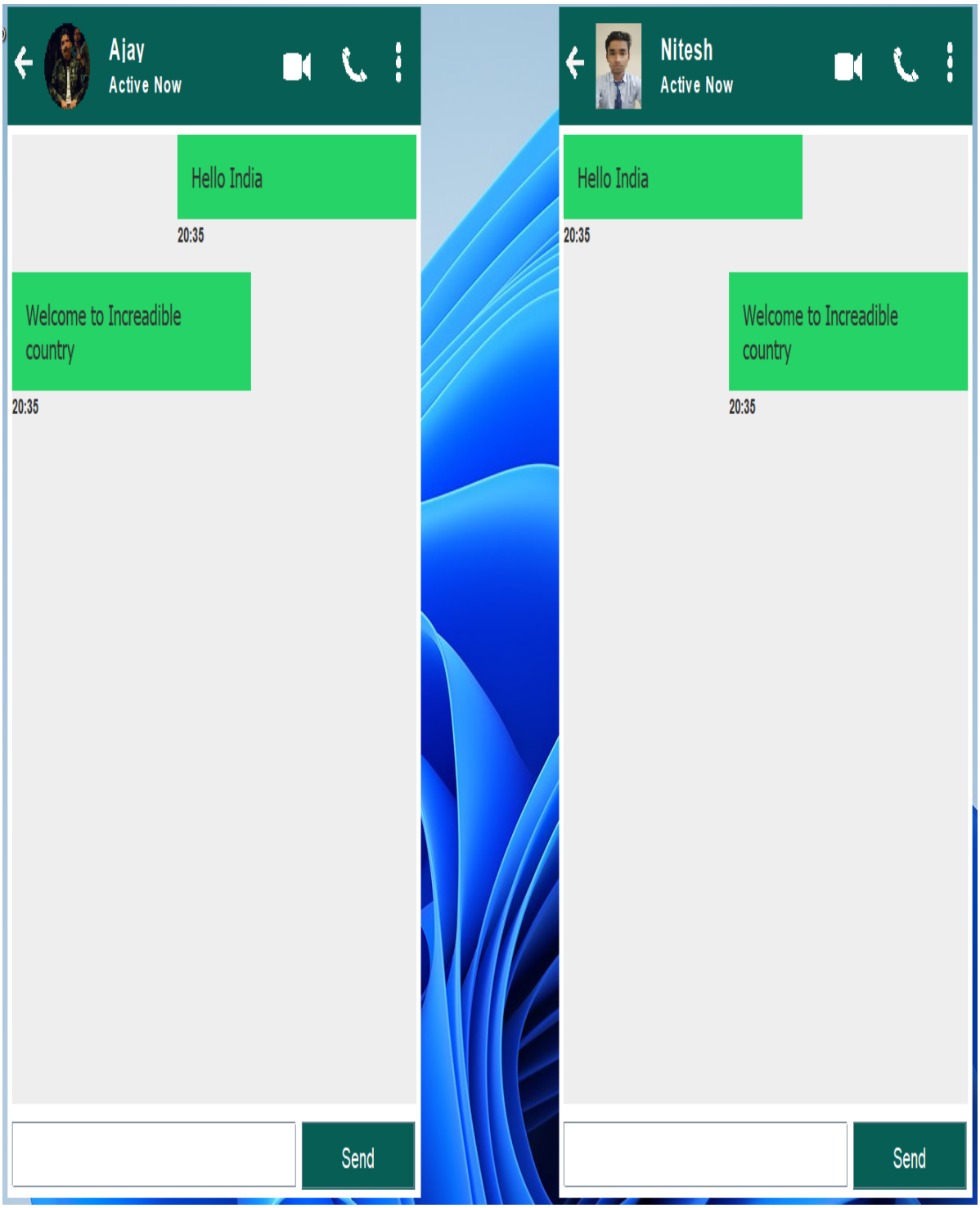


Fig:5.2 (Chatting application user interface 2)

In the interface 1 The main page is Displayed. while in the interface 2, the chatting through the text messages is displayed.

# Chapter-6

## Future Scope

We can customize the interface by adding additional features such as date, Video, Image, Audio, etc.

This project can focuses on developing a **basic messaging application** with key features such as:

- **User Authentication:** Users can register, log in, and manage their profiles.
- **Text Messaging:** Real-time text chat functionality for one-on-one communication.
- **Multimedia Sharing:** Users can send images, videos, and audio files.
- **Group Chat:** Functionality to create and participate in group conversations.
- **Notifications:** Users are notified of new messages.

The application will be built using commonly used technologies, focusing on the front-end and back-end components of the app. This project will cover both mobile and web-based platforms, providing a full-stack approach to development.

# References

**Journal Article Abstract (accessed from online database)**

Available: CODING NINJAS, https://www.codingninjas.com/studio/library/Chatting Application using Java.

Socket.io Documentation: https://socket.io/docs

**Books Single Author**

Kathy Walrath, Mary Campione, Alison Huml, and Sharon Zakhour, The JFC Swing Tutorial: A Guide to Constructing GUIs, Addison-Wesley Professional, 2 edition (March 5, 2004)

# Appendix

## Certificate: