Source Code

```
import javax.swing.*;
import java.awt.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.URL;
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        new MainMenu().open();
    }
}

class MainMenu extends JFrame {

    public MainMenu() {
        setTitle("SafeChat");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout());
        add(new GradientPanel(), BorderLayout.CENTER);
    }
```

```
    public void open() {
        setVisible(true);
    }
}

class GradientPanel extends JPanel {

    public GradientPanel() {
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.gridx = 0;
        gbc.gridy = GridBagConstraints.RELATIVE;
        gbc.anchor = GridBagConstraints.CENTER;

        JLabel welcomeLabel = new JLabel("Welcome to SafeChat!");
        welcomeLabel.setForeground(Color.WHITE);
        add(welcomeLabel, gbc);

        JButton createChatButton = new JButton("Create Chat");
        JButton joinChatButton = new JButton("Join Chat");

        styleButton(createChatButton);
        styleButton(joinChatButton);

        add(createChatButton, gbc);
        add(joinChatButton, gbc);

        createChatButton.addActionListener(e -> {
            String userName = promptUserName();
            if (userName != null && !userName.trim().isEmpty()) {
                new ChatScreen(userName).setVisible(true);
```

```
                setVisible(false);
            }
        });

        joinChatButton.addActionListener(e -> {
            String userName = promptUserName();
            String serverAddress = promptServerAddress();
            if (userName != null && !userName.trim().isEmpty() && serverAddress
!= null && !serverAddress.trim().isEmpty()) {
                new ClientScreen(userName, serverAddress).setVisible(true);
                setVisible(false);
            }
        });
    }

    private void styleButton(JButton button) {
        button.setBackground(new Color(50, 50, 50));
        button.setForeground(Color.WHITE);
        button.setFocusPainted(false);
        button.setBorderPainted(false);
    }

    private String promptUserName() {
        return JOptionPane.showInputDialog(this, "Enter your name:", "User
Name", JOptionPane.PLAIN_MESSAGE);
    }

    private String promptServerAddress() {
        return JOptionPane.showInputDialog(this, "Enter server address:",
"Server Address", JOptionPane.PLAIN_MESSAGE);
    }

    @Override
```

```
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        int width = getWidth();
        int height = getHeight();

        Color color1 = new Color(0, 0, 0);
        Color color2 = new Color(64, 64, 64);

        GradientPaint gp = new GradientPaint(0, 0, color1, 0, height, color2);
        g2d.setPaint(gp);
        g2d.fillRect(0, 0, width, height);
    }
}

class ChatScreen extends JFrame {
    private JTextArea chatArea;
    private JTextField inputField;
    private JButton sendButton;
    private ServerSocket serverSocket;
    private Set<PrintWriter> clientWriters = new HashSet<>();
    private String userName;

    public ChatScreen(String userName) {
        this.userName = userName;
        setTitle("Chat Room");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        chatArea = new JTextArea();
        chatArea.setEditable(false);
```

```
        JScrollPane scrollPane = new JScrollPane(chatArea);

        inputField = new JTextField();
        sendButton = new JButton("Send");

        JPanel inputPanel = new JPanel(new BorderLayout());
        inputPanel.add(inputField, BorderLayout.CENTER);
        inputPanel.add(sendButton, BorderLayout.EAST);

        add(scrollPane, BorderLayout.CENTER);
        add(inputPanel, BorderLayout.SOUTH);

        // Start the server in a separate thread
        new Thread(() -> {
            try {
                startServer();
            } catch (IOException e) {
                chatArea.append("Error starting server: " + e.getMessage() +
"\n");
            }
        }).start();

        sendButton.addActionListener(e -> {
            String message = inputField.getText().trim();
            if (!message.isEmpty()) {
                chatArea.append(userName + ": " + message + "\n");
                inputField.setText("");
                broadcastMessage(userName + ": " + message);
            }
        });
    }
```

```
    private void startServer() throws IOException {
        serverSocket = new ServerSocket(12345);

        String ipAddress = getPublicIpAddress();
        if (ipAddress != null) {
            chatArea.append("Server started at " + ipAddress + ":" +
serverSocket.getLocalPort() + "\n");
        } else {
            chatArea.append("Could not retrieve public IP address.\n");
        }

        while (true) {
            Socket clientSocket = serverSocket.accept();
            chatArea.append("Client connected: " +
clientSocket.getInetAddress() + "\n");

            PrintWriter writer = new
PrintWriter(clientSocket.getOutputStream(), true);
            clientWriters.add(writer);

            new ClientHandler(clientSocket, chatArea, clientWriters).start();
        }
    }

    private String getPublicIpAddress() {
        try {
            URL url = new URL("https://checkip.amazonaws.com");
            BufferedReader in = new BufferedReader(new
InputStreamReader(url.openStream()));
            return in.readLine();
        } catch (IOException e) {
            return null;
        }
    }
```

```
    private void broadcastMessage(String message) {
        for (PrintWriter writer : clientWriters) {
            writer.println(message);
        }
    }
}

class ClientHandler extends Thread {
    private Socket clientSocket;
    private JTextArea chatArea;
    private Set<PrintWriter> clientWriters;
    private BufferedReader in;
    private PrintWriter out;

    public ClientHandler(Socket socket, JTextArea chatArea, Set<PrintWriter>
clientWriters) {
        this.clientSocket = socket;
        this.chatArea = chatArea;
        this.clientWriters = clientWriters;

        try {
            in              =              new              BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(clientSocket.getOutputStream(), true);
        } catch (IOException e) {
            chatArea.append("Error   setting   up   client   streams:   "   +
e.getMessage() + "\n");
        }
    }

    @Override
    public void run() {
```

```
        try {
            String message;
            while ((message = in.readLine()) != null) {
                chatArea.append(message + "\n");
                broadcastMessage(message);
            }
        } catch (IOException e) {
            chatArea.append("Error communicating with client: " +
e.getMessage() + "\n");
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                chatArea.append("Error closing client socket: " +
e.getMessage() + "\n");
            }
        }
    }

    private void broadcastMessage(String message) {
        for (PrintWriter writer : clientWriters) {
            writer.println(message);
        }
    }
}

class ClientScreen extends JFrame {
    private JTextArea chatArea;
    private JTextField inputField;
    private JButton sendButton;
    private String userName;
    private String serverAddress;
    private Socket socket;
```

```
    private PrintWriter out;
    private BufferedReader in;

    public ClientScreen(String userName, String serverAddress) {
        this.userName = userName;
        this.serverAddress = serverAddress;
        setTitle("Chat Room - " + userName);
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        chatArea = new JTextArea();
        chatArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(chatArea);

        inputField = new JTextField();
        sendButton = new JButton("Send");

        JPanel inputPanel = new JPanel(new BorderLayout());
        inputPanel.add(inputField, BorderLayout.CENTER);
        inputPanel.add(sendButton, BorderLayout.EAST);

        add(scrollPane, BorderLayout.CENTER);
        add(inputPanel, BorderLayout.SOUTH);

        new Thread(() -> {
            try {
                connectToServer();
            } catch (IOException e) {
                chatArea.append("Error connecting to server: " + e.getMessage()
+ "\n");
            }
```

```
        }).start();

        sendButton.addActionListener(e -> {
            String message = inputField.getText().trim();
            if (!message.isEmpty()) {
                inputField.setText("");
                sendMessage(userName + ": " + message);
            }
        });
    }

    private void connectToServer() throws IOException {
        String[] addressParts = serverAddress.split(":");
        String host = addressParts[0];
        int port = Integer.parseInt(addressParts[1]);

        socket = new Socket(host, port);
        out = new PrintWriter(socket.getOutputStream(), true);
        in                =                new                BufferedReader(new
InputStreamReader(socket.getInputStream()));

        chatArea.append("Connected to server at " + serverAddress + "\n");

        new Thread(() -> {
            try {
                String message;
                while ((message = in.readLine()) != null) {
                    chatArea.append(message + "\n");
                }
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Server has disconnected",
"Server Disconnected", JOptionPane.ERROR_MESSAGE);
                System.exit(0);
```

```
            }
        }).start();
    }


    private void sendMessage(String message) {
        if (out != null) {
            out.println(message);
        }
    }
}
```