King Abdul-Aziz University
Faculty of Computing and Information Technolog
Computer Science Department
CPCS 371-Computer Networks 1 Winter 2022

REMOTE HEALTH MONITORING SYSTEM (RHMS)

SIMULATING A REMOTE HEALTH MONITORING SYSTEM
FOR ELDERLY PATIENTS USING CLIENT-SERVER TCP SOCKETS

Project ID: Group 8

Instructor: Dr. Ohoud Alzamzami

Project team:

Name	ID	Email
Razan Arif Alamri	2006899	rmohammedalamri0001@stu.kau.edu.sa
Shatha Khalid Binmahfouz	2006687	sabinmahfouz@stu.kau.edu.sa
Ghada Eisa Fzia Alsulmi	1905190	galsulmi0005@stu.kau.edu.sa
Sarah Abdulhadi Mahdi Aljohani	1906525	saljohani0151@stu.kau.edu.sa
Waad Turki Magait Alharbi	2006198	wmagaitalharbi@stu.kau.edusa.
Shahad Mohammed Bafadhel	1906799	sbafadhel0001@stu.kau.edu.sa

Table of Contents

Table of Figures

1. Introduction

1.1. Client-Server applications

The Client-Server architecture consists of two parts: the client and the server. The client-server applications are distributed application frameworks that assign different responsibilities to servers and clients. A server is a machine that gives clients access to one or more services. Clients can acquire and uses those services from sever either through a network or within the same computer. Both server and client must follow the same protocol for communication when using a client-server system, which employs a request-response messaging pattern. [1]

1.2. Java TCP Sockets

TCP (Transmission Control Protocol) is one of the common main protocols of the Internet protocol suite. It is the most common transport layer protocol. The transport layer lies between the Application and Network Layers and utilizes TCP to deliver reliable services. It is a connection-oriented communications protocol that facilitates message exchanges between various devices over a network. TCP is used with the Internet Protocol (IP), which defines the method for exchanging data packets between computers [1]

1. The server creates a TCP socket via serverSocket using of client port number, then waits for the client requests.
2. The server accepts the client request by accept() method.
3. Then, server will wait for the client connection.
4. After that, the client creates a TCP socket using the server's IP address and port number, for a successful connection, the client's port number must match the server's port number.
5. I/O communication on both sides over sockets is done using the getOutputStream() and getInputStream() functions.

```
//Create Server Socket with specific port number.
  Socket  serverSocket = new ServerSocket(portForThisServer);
  clientSocket = serverSocket.accept();

// Create Server Socket    MedicalServer = new ServerSocket(port);
   Socket clientSocket = MedicalServer.accept();
```

1.3. Threads

A running program's route, the steps it takes, and the order in which those steps are taken are all referred to as a thread. A thread executes code for a specified set of inputs from its beginning place in an ordered, preset sequence. Multiple application processes can be executed in parallel using multiple threads, thus increasing application performance. [2]

```
// this thread makese to start execution
   new Thread(new Personal_Server_Handler(clientSocket)).start();
```

1.4. Selected GUI Elements

We have used java swing which is a lightweight GUI toolkit written completely in java to build an optimized window-based application. We have implemented GUI elements in five classes: Sensor_Client_Application, Personal_Server, Medical_Server, Start_Screen, and CloseApp. We have made use of a variety of elements, including labels, text fields, text areas, images, buttons, and sounds [3].

1.5. RHMS Application

RHMS is a Remote Health Monitoring System using Wireless Body Area Networks (WBAN). RHMS aims to improveg the quality of life of elderly patients with chronic diseasesand reducef medical carecosts forthem. It uses three kinds of sensors, namely, heart rate, oxygen saturatio,n and temperature, to monitor the health of elderly patients.

It consists of three main components:

1- Sensor Client Application.
2 - Personal Server.
3- Medical Server.

1.6. The Roles of Clients and Servers

RHMS consists of three components:

1- Personal Server:

It has two roles: a server and a client. It works as a server for the Sensor Client application that processes the sensor information received from sensors and decides whether the processed information is urgent or not based on certain conditions. Thus, if the processed information is urgent, it works as a client and sends this information to the medical server via TCP sockets.

2- The Medical Server:

It receives messages about the patient's condition from the personal server via TCP sockets. The medical server displays messages received from Caregiver's personal client. Then, it addresses the received information and displays the appropriate action to be taken by the caregiver.

3- The Sensor Client Application:

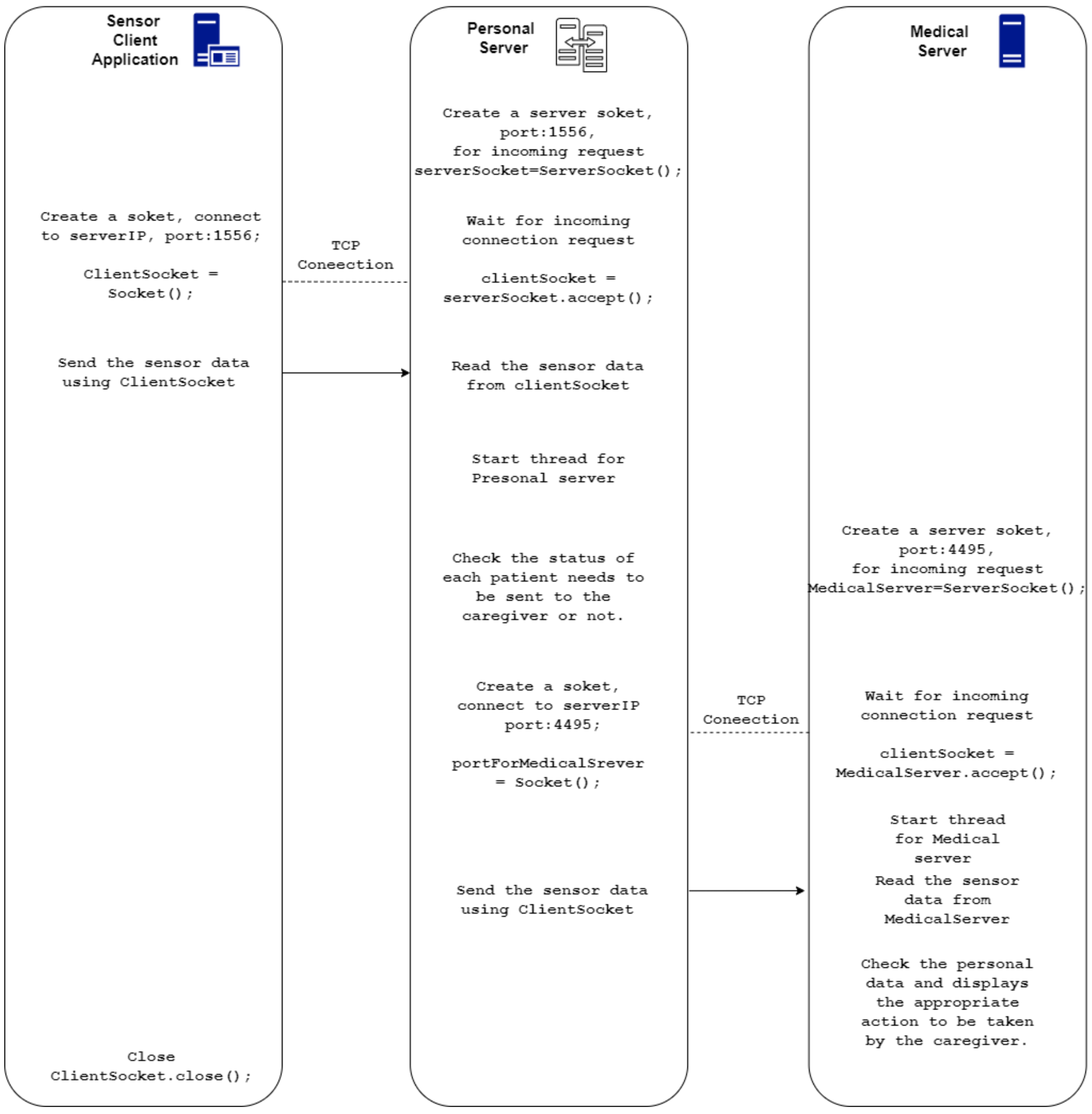
The main role of the sensor application is to generate, display, and send data every 5 seconds via TCP socket to the personal server to handle this data, and determine the status of each pulse(one reading at a time) separately. The user will enter the number of readings indicating in real time for sensors to be alive since that each reading takes exactly five seconds to send one packet of data(Temperature, Heart rate, Oxygen level). Therefore, the minimum number of readings is 12 which is equal to 60 seconds. Finally, the sensor application prints out the generated data on its screen.

1.7. Overview

In section 2, the patient Monitoring Application interaction diagram and pseudocode will be presented. In section 3, we will explain the RHMS implementation in detail. Snapshots of RHMS application output will be presented in section 4. Then, in section 5, we will discuss teamwork and responsibility assignments. Finally, section 6 is the conclusion of the report where we will summarize our project.

2. Patient Monitoring Application Interaction Diagram

2.1. Client-Server Interaction Diagram



Sensor Client Application
Personal Server
Medical Server
Create a server soket, port:1556, for incoming request serverSocket=ServerSocket();
Create a soket, connect to serverIP, port:1556;
ClientSocket = Socket();
TCP Coneection
Wait for incoming connection request
clientSocket = serverSocket.accept();
Send the sensor data using ClientSocket
Read the sensor data from clientSocket
Start thread for Presonal server
Create a server soket, port:4495, for incoming request MedicalServer=ServerSocket();
Check the status of each patient needs to be sent to the caregiver or not.
Create a soket, connect to serverIP port:4495;
portForMedicalSrever = Socket();
TCP Coneection
Wait for incoming connection request
clientSocket = MedicalServer.accept();
Start thread for Medical server
Send the sensor data using ClientSocket
Read the sensor data from MedicalServer
Check the personal data and displays the appropriate action to be taken by the caregiver.
Close ClientSocket.close();

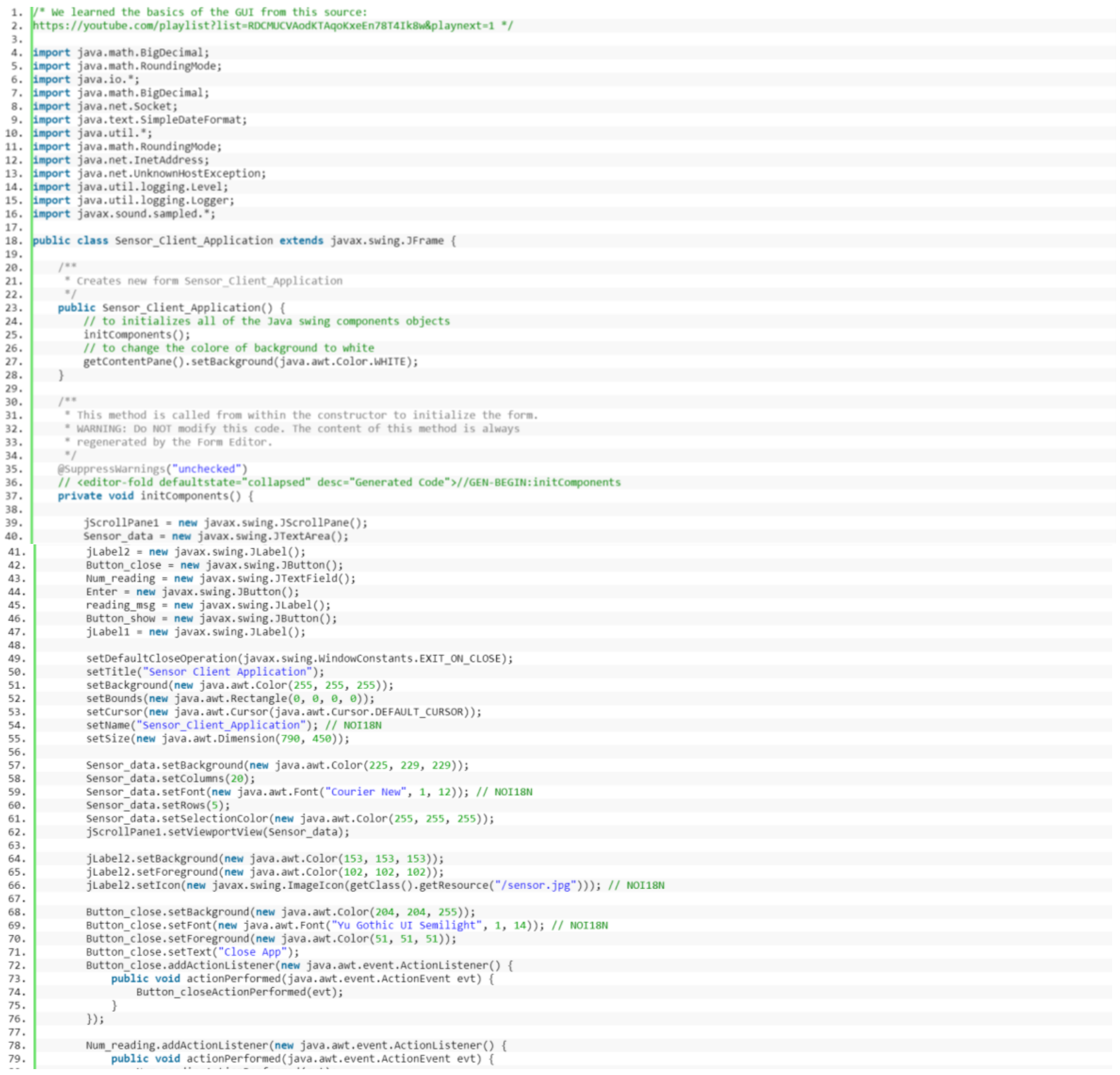
2.2. Pseudocode

```
Algorithm Server
//The following algorithm illustrate our client-server application of the networks
Application, consisting of two clients and a server connecting them.
Input🡐Reading //more than 12 readings
For 0: Input
        Create Sensor Client socket
        Generate sensor data randomly
        Send generated data to personal server
                // Personal Server Socket
                While(true)
                        Create TCP socket
                        Listening for client sensor to be connect
                        Accept connection from the client sensor
                        Start thread
                        Handling data came from client separately
                        Display Data
                                If data need to be send to the medical server
                                        Open socket
                                        Wait to be accept
                                        Data will send to medical server
                                                // Medical Server Socket
                                                While(true)
                                                        Create TCP socket
                                                        Listening for client Personal to be
                                                        connect
                                                        Accept connection from the Personal
                                                        client
                                                        Start thread
                                                        Handling data came from client
                                                        separately
                                                        Check personal data and display
                                                        appropriate message
                                Else
                                        Do nothing
                        Continue to Personal server
Close sensor client socket
```

3. RHMS Implementation

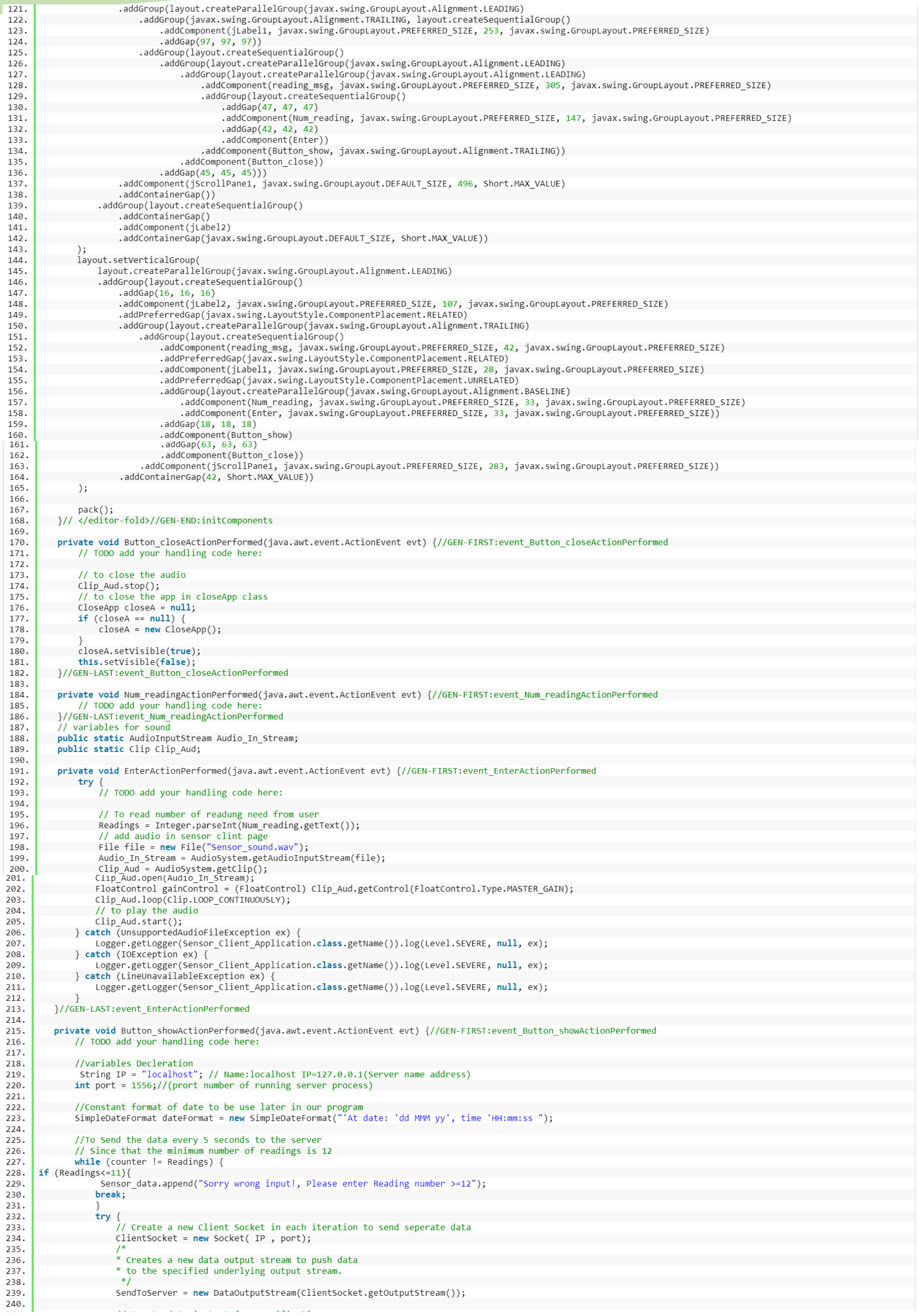
We have implemented RHMS using three classes: Sensor_Client_Application, Personal_Server, and Medical_Server. Note that Sensor_Client_Application class sends sensed data to Personal_Server class via a separate TCP socket every five seconds, thereby we have created a class called Personal_Server_Handler within Personal_Server class that handles each TCP socket separately, and undoubtedly Medical_Server class has a ServerHandler class too.

3.1. Sensor_Client_Application Class



```java
/* We learned the basics of the GUI from this source:
https://youtube.com/playlist?list=RDCMUCVAodKTAqoKxeEn78T4Ik8w&playnext=1 */

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.io.*;
import java.math.BigDecimal;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.*;
import java.math.RoundingMode;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.sound.sampled.*;

public class Sensor_Client_Application extends javax.swing.JFrame {

    /**
     * Creates new form Sensor_Client_Application
     */
    public Sensor_Client_Application() {
        // to initializes all of the Java swing components objects
        initComponents();
        // to change the colore of background to white
        getContentPane().setBackground(java.awt.Color.WHITE);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        Sensor_data = new javax.swing.JTextArea();
        jLabel2 = new javax.swing.JLabel();
        Button_close = new javax.swing.JButton();
        Num_reading = new javax.swing.JTextField();
        Enter = new javax.swing.JButton();
        reading_msg = new javax.swing.JLabel();
        Button_show = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Sensor Client Application");
        setBackground(new java.awt.Color(255, 255, 255));
        setBounds(new java.awt.Rectangle(0, 0, 0, 0));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setName("Sensor_Client_Application"); // NOI18N
        setSize(new java.awt.Dimension(790, 450));

        Sensor_data.setBackground(new java.awt.Color(225, 229, 229));
        Sensor_data.setColumns(20);
        Sensor_data.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        Sensor_data.setRows(5);
        Sensor_data.setSelectionColor(new java.awt.Color(255, 255, 255));
        jScrollPane1.setViewportView(Sensor_data);

        jLabel2.setBackground(new java.awt.Color(153, 153, 153));
        jLabel2.setForeground(new java.awt.Color(102, 102, 102));
        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/sensor.jpg"))); // NOI18N

        Button_close.setBackground(new java.awt.Color(204, 204, 255));
        Button_close.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 14)); // NOI18N
        Button_close.setForeground(new java.awt.Color(51, 51, 51));
        Button_close.setText("Close App");
        Button_close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Button_closeActionPerformed(evt);
            }
        });

        Num_reading.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```java
        jLabel2 = new javax.swing.JLabel();
        Button_close = new javax.swing.JButton();
        Num_reading = new javax.swing.JTextField();
        Enter = new javax.swing.JButton();
        reading_msg = new javax.swing.JLabel();
        Button_show = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Sensor Client Application");
        setBackground(new java.awt.Color(255, 255, 255));
        setBounds(new java.awt.Rectangle(0, 0, 0, 0));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setName("Sensor_Client_Application"); // NOI18N
        setSize(new java.awt.Dimension(790, 450));

        Sensor_data.setBackground(new java.awt.Color(225, 229, 229));
        Sensor_data.setColumns(20);
        Sensor_data.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        Sensor_data.setRows(5);
        Sensor_data.setSelectionColor(new java.awt.Color(255, 255, 255));
        jScrollPane1.setViewportView(Sensor_data);

        jLabel2.setBackground(new java.awt.Color(153, 153, 153));
        jLabel2.setForeground(new java.awt.Color(102, 102, 102));
        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/sensor.jpg"))); // NOI18N

        Button_close.setBackground(new java.awt.Color(204, 204, 255));
        Button_close.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 14)); // NOI18N
        Button_close.setForeground(new java.awt.Color(51, 51, 51));
        Button_close.setText("Close App");
        Button_close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Button_closeActionPerformed(evt);
            }
        });

        Num_reading.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Num_readingActionPerformed(evt);
            }
        });

        Enter.setBackground(new java.awt.Color(204, 204, 204));
        Enter.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 14)); // NOI18N
        Enter.setForeground(new java.awt.Color(51, 51, 51));
        Enter.setText("Enter");
        Enter.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                EnterActionPerformed(evt);
            }
        });

        reading_msg.setBackground(new java.awt.Color(67, 141, 177));
        reading_msg.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 18)); // NOI18N
        reading_msg.setForeground(new java.awt.Color(255, 255, 255));
        reading_msg.setText(" Plase enter number of reading: ");
        reading_msg.setName(""); // NOI18N
        reading_msg.setOpaque(true);

        Button_show.setBackground(new java.awt.Color(204, 204, 255));
        Button_show.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 18)); // NOI18N
        Button_show.setForeground(new java.awt.Color(51, 51, 51));
        Button_show.setText("Show Output Of Reading");
        Button_show.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Button_showActionPerformed(evt);
            }
        });

        jLabel1.setBackground(new java.awt.Color(0, 102, 153));
        jLabel1.setFont(new java.awt.Font("Calibri", 1, 14)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 0, 51));
        jLabel1.setText("(Note:minimum Number of readings is 12)");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```



```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 253, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(97, 97, 97))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(reading_msg, javax.swing.GroupLayout.PREFERRED_SIZE, 305, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGroup(layout.createSequentialGroup()
                                    .addGap(47, 47, 47)
                                    .addComponent(Num_reading, javax.swing.GroupLayout.PREFERRED_SIZE, 147, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(42, 42, 42)
                                    .addComponent(Enter))
                                .addComponent(Button_show, javax.swing.GroupLayout.Alignment.TRAILING))
                            .addComponent(Button_close))
                        .addGap(45, 45, 45)))
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 496, Short.MAX_VALUE)
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(16, 16, 16)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 107, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(reading_msg, javax.swing.GroupLayout.PREFERRED_SIZE, 42, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(Num_reading, javax.swing.GroupLayout.PREFERRED_SIZE, 33, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(Enter, javax.swing.GroupLayout.PREFERRED_SIZE, 33, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addComponent(Button_show)
                        .addGap(63, 63, 63)
                        .addComponent(Button_close))
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 283, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(42, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void Button_closeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Button_closeActionPerformed
        // TODO add your handling code here:

        // to close the audio
        Clip_Aud.stop();
        // to close the app in closeApp class
        CloseApp closeA = null;
        if (closeA == null) {
            closeA = new CloseApp();
        }
        closeA.setVisible(true);
        this.setVisible(false);
    }//GEN-LAST:event_Button_closeActionPerformed

    private void Num_readingActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Num_readingActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_Num_readingActionPerformed
    // variables for sound
    public static AudioInputStream Audio_In_Stream;
    public static Clip Clip_Aud;

    private void EnterActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_EnterActionPerformed
        try {
            // TODO add your handling code here:

            // To read number of readung need from user
            Readings = Integer.parseInt(Num_reading.getText());
            // add audio in sensor clint page
            File file = new File("Sensor_sound.wav");
            Audio_In_Stream = AudioSystem.getAudioInputStream(file);
            Clip_Aud = AudioSystem.getClip();
            Clip_Aud.open(Audio_In_Stream);
            FloatControl gainControl = (FloatControl) Clip_Aud.getControl(FloatControl.Type.MASTER_GAIN);
            Clip_Aud.loop(Clip.LOOP_CONTINUOUSLY);
            // to play the audio
            Clip_Aud.start();
        } catch (UnsupportedAudioFileException ex) {
            Logger.getLogger(Sensor_Client_Application.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Sensor_Client_Application.class.getName()).log(Level.SEVERE, null, ex);
        } catch (LineUnavailableException ex) {
            Logger.getLogger(Sensor_Client_Application.class.getName()).log(Level.SEVERE, null, ex);
        }
    }//GEN-LAST:event_EnterActionPerformed

    private void Button_showActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Button_showActionPerformed
        // TODO add your handling code here:

        //variables Decleration
         String IP = "localhost"; // Name:localhost IP=127.0.0.1(Server name address)
        int port = 1556;//(prort number of running server process)

        //Constant format of date to be use later in our program
        SimpleDateFormat dateFormat = new SimpleDateFormat("'At date: 'dd MMM yy', time 'HH:mm:ss ");

        //To Send the data every 5 seconds to the server
        // Since that the minimum number of readings is 12
        while (counter != Readings) {
if (Readings<=11){
             Sensor_data.append("Sorry wrong input!, Please enter Reading number >=12");
            break;
            }
            try {
                // Create a new Client Socket in each iteration to send seperate data
                ClientSocket = new Socket( IP , port);
                /*
                * Creates a new data output stream to push data
                * to the specified underlying output stream.
                 */
                SendToServer = new DataOutputStream(ClientSocket.getOutputStream());

```

```java
                // Create date instant for specific time
                Date date = new Date();

                Thread.sleep(5000); // wait five seconds

                TemperatureSensor = GenerateTemperatureData();
                Sensor_data.append(dateFormat.format(date) + ",sensed temperature is " + TemperatureSensor + "\n");
                SendToServer.writeDouble(TemperatureSensor); // send TemperatureSensor data to Personal Server

                HeartSensor = GenerateHeartData();
                Sensor_data.append(dateFormat.format(date) + ",sensed heart rate is " + HeartSensor + "\n");
                SendToServer.writeByte(HeartSensor); // send HeartSensor data to Personal Server

                OxygenSensor = GenerateOxygenData();
                Sensor_data.append(dateFormat.format(date) + ",sensed oxygen saturation is " + OxygenSensor + "\n");
                SendToServer.writeByte(OxygenSensor); // send OxygenSensor data to Personal Server

                Sensor_data.append("\n\n");

                counter += 1;

                SendToServer.close();// close data stream
                ClientSocket.close();// close client Socket

            } catch (IOException | InterruptedException ex) {
                Logger.getLogger(Sensor_Client_Application.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

    }//GEN-LAST:event_Button_showActionPerformed

    //Timer for whole task?  we have wrote temprorary number for the readings
    static int HeartSensor;
    static double TemperatureSensor;
    static int OxygenSensor;
    static Socket ClientSocket;
    static DataOutputStream SendToServer;
    /*This value determines how much program runs i.e
        if the program live 60 seconds then it needs to 12 readings to cover the time needed
        (Keep in mind the data are sending every five seconds).*/
    static int Readings = 0;
    static int counter = 0;

    public static void main(String args[]) throws Exception {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Sensor_Client_Application.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Sensor_Client_Application.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Sensor_Client_Application.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Sensor_Client_Application.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Sensor_Client_Application().setVisible(true);
            }
        });
    }

    // Generate randoms (Temperature-Heart-Oxygen)Data
    //(int)(Math.random()*(max-min+1)+min)
    public static double GenerateTemperatureData() {
        double GenerateValue = (double) (Math.random() * (41 - 36 + 1) + 36);

        BigDecimal TwoDigits = new BigDecimal(GenerateValue).setScale(2, RoundingMode.HALF_UP);
        double newValue = TwoDigits.doubleValue();

        return newValue;
    }

    public static int GenerateHeartData() {

        return (int) (Math.random() * (120 - 60 + 1) + 60);
    }

    public static int GenerateOxygenData() {

        return (int) (Math.random() * (100 - 60 + 1) + 60);
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton Button_close;
    private javax.swing.JButton Button_show;
    private javax.swing.JButton Enter;
    private javax.swing.JTextField Num_reading;
    private static javax.swing.JTextArea Sensor_data;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JLabel reading_msg;
    // End of variables declaration//GEN-END:variables
}
```

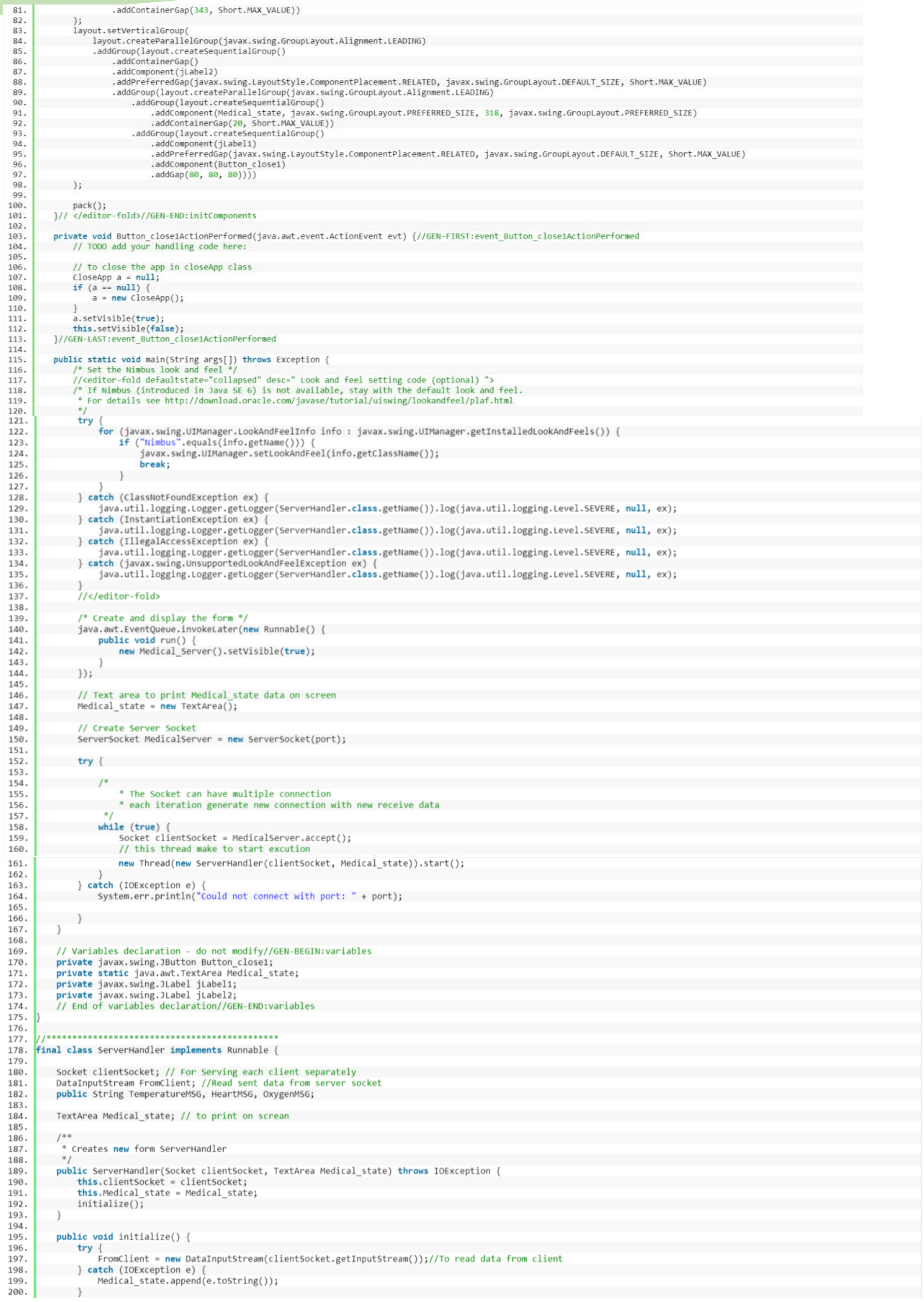
3.2. Personal_Server Class

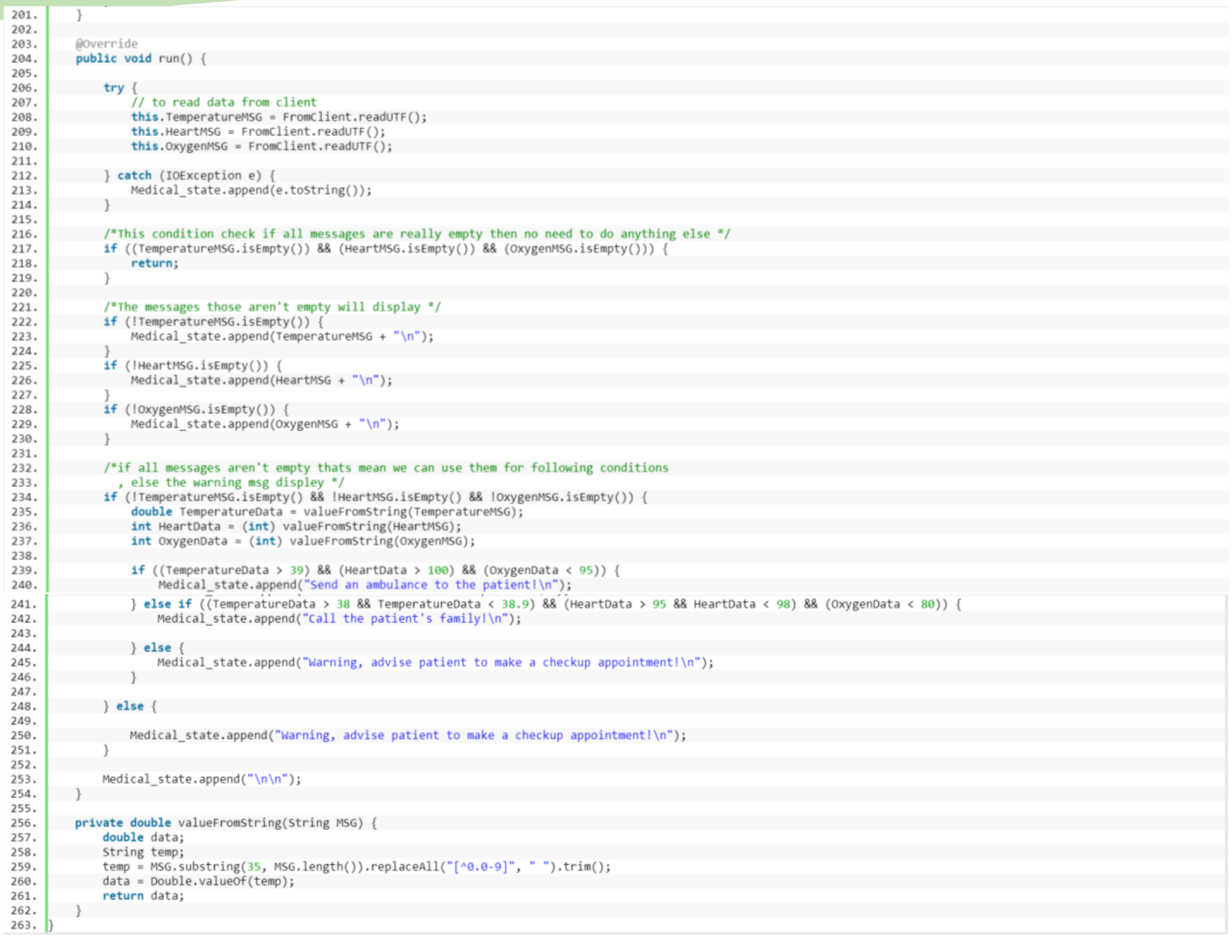
```java
import java.awt.TextArea;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Personal_Server extends javax.swing.JFrame {

    /**
     * Creates new form Personal_Server
     */
    public Personal_Server() {
        // to initializes all of the Java swing components objects
        initComponents();
        // to change the colore of background to white
        getContentPane().setBackground(java.awt.Color.WHITE);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jColorChooser1 = new javax.swing.JColorChooser();
        Button_close = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        Personal_state = new java.awt.TextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Personal State");
        setName("Personal State"); // NOI18N

        Button_close.setBackground(new java.awt.Color(204, 204, 255));
        Button_close.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 14)); // NOI18N
        Button_close.setForeground(new java.awt.Color(51, 51, 51));
        Button_close.setText("Close App");
        Button_close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Button_closeActionPerformed(evt);
            }
        });

        jLabel2.setBackground(new java.awt.Color(255, 255, 255));
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/state.jpg"))); // NOI18N

        jLabel1.setBackground(new java.awt.Color(67, 141, 177));
        jLabel1.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 24)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setText(" Personal State");
        jLabel1.setName(""); // NOI18N
        jLabel1.setOpaque(true);

        Personal_state.setBackground(new java.awt.Color(225, 229, 229));
        Personal_state.setFont(new java.awt.Font("Courier New", 0, 12)); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel2))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 224, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequentialGroup()
                                .addContainerGap()
                                .addComponent(Button_close, javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(Personal_state, javax.swing.GroupLayout.DEFAULT_SIZE, 550, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(0, 17, Short.MAX_VALUE)
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addGap(232, 232, 232)
                        .addComponent(Button_close)
                        .addGap(24, 24, 24))
                    .addComponent(Personal_state, javax.swing.GroupLayout.PREFERRED_SIZE, 318, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void Button_closeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Button_closeActionPerformed
        // TODO add your handling code here:

        // to close the app in closeApp class
        CloseApp closeA = null;
        if (closeA == null) {
            closeA = new CloseApp();
        }
        closeA.setVisible(true);
        this.setVisible(false);
    }//GEN-LAST:event_Button_closeActionPerformed

    // This Srever work with two faces, server(recieve msg) and client(send msg)
    //Server Socket decleration
    static ServerSocket serverSocket;
    static int portForThisServer = 1556;
    static Socket clientSocket;//For accept client connection

    public static void main(String args[]) throws Exception {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Personal_Server_Handler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Personal_Server_Handler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Personal_Server_Handler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Personal_Server_Handler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        // Text area to print Personal_state data on screen
        Personal_state = new TextArea();

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Personal_Server().setVisible(true);
            }
        });

        try {
            //Create Server Socket with specific port number.
            serverSocket = new ServerSocket(portForThisServer);

            /*The Socket can have multiple connection
                each iteration generate new connection with new sensor data*/
            while (true) {

                clientSocket = serverSocket.accept();
                // this thread make to start excution
                new Thread(new Personal_Server_Handler(clientSocket, Personal_state)).start();

            }

        } catch (IOException e) {

            System.err.println("Could not connect with port: " + portForThisServer);
            System.exit(1);

        }

    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton Button_close;
    private static java.awt.TextArea Personal_state;
    private javax.swing.JColorChooser jColorChooser1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    // End of variables declaration//GEN-END:variables
}

//**********************************************
class Personal_Server_Handler implements Runnable {
    // to run the applcation on multiple device
    InetAddress addr =InetAddress.getByName("10.24.155.65");
    String hostName = addr.getHostName();

    Socket ForMedicalServer;// For Sending dangerous cases to Medical Server
 // String IP = "localhost";
    int portForMedicalSrever = 4495;

    Socket ForClientSocket;//The data that come from client to be process

    public DataInputStream RecievesFromClient;// for receiving data
```

```java
    public DataOutputStream SendToServer2; // for sending data to

    TextArea Personal_state; // to print on screen

    /**
     * Creates new form Personal_Server_Handler
     */
    public Personal_Server_Handler(Socket connectedSocket, TextArea Personal_state) throws IOException {
        this.ForClientSocket = connectedSocket;
        this.Personal_state = Personal_state;
        initialize();

    }

    private void initialize() {
        try {
            this.ForMedicalServer = new Socket(hostName, portForMedicalSrever);
            this.RecievesFromClient = new DataInputStream(ForClientSocket.getInputStream());
            this.SendToServer2 = new DataOutputStream(ForMedicalServer.getOutputStream());
        } catch (IOException e) {
            Personal_state.append(e.toString());
        }
    }

    @Override
    public void run() {

        // Variables declaration (need to print Personal_state)
        double TemperatureData = 0;
        int HeartData = 0;
        int OxygenData = 0;
        String msg1 = "", msg2 = "", msg3 = "";

        SimpleDateFormat dateFormat = new SimpleDateFormat("'At date: 'dd MMM yy', time 'HH:mm:ss ");

        try {

            Date date = new Date();
            msg1 = dateFormat.format(date);
            // to read data from sensor
            TemperatureData = RecievesFromClient.readDouble();

            /*Case 1: if the Temperature Data > 38 then data will send to the server
                  to take the appropriate action otherwise it's normal */
            if (TemperatureData > 38) {
                msg1 += ", Temperature is high " + TemperatureData;
                SendToServer2.writeUTF(msg1);  // msg1 is is sent to the Medical Server
                Personal_state.append(msg1 + ". An alert message is sent to the Medical Server.\n");

            } else {
                Personal_state.append(msg1 + ", Temperature is normal\n");
                SendToServer2.writeUTF("");

            }
            // to read data from sensor
            HeartData = RecievesFromClient.read();
            msg2 = dateFormat.format(date);

            /*Case 1: if the Heart Data > 100 || Heart Data <60 then data will send to the server
                  to take the appropriate action otherwise it's normal */
            if (HeartData > 100) {
                msg2 += ", Heart rate is above normal " + HeartData;
                SendToServer2.writeUTF(msg2);
                Personal_state.append(msg2 + ". An alert message is sent to the Medical Server.\n");

            } else if (HeartData < 60) {
                msg2 += ", Heart rate is below normal " + HeartData;
                SendToServer2.writeUTF(msg2);// msg2 is is sent to the Medical Server
                Personal_state.append(msg2 + ". An alert message is sent to the Medical Server.\n");
            } else {
                Personal_state.append(msg2 + ", Heart rate is normal\n");
                SendToServer2.writeUTF("");

            }
            // to read data from sensor
            OxygenData = RecievesFromClient.read();
            msg3 = dateFormat.format(date);

            /*Case 1: if the OxygenData<75 then data will send to the server
                  to take the appropriate action otherwise it's normal */
            if (OxygenData < 75) {
                msg3 += ", Oxygen saturation is low " + OxygenData;
                SendToServer2.writeUTF(msg3);// msg3 is is sent to the Medical Server
                Personal_state.append(msg3 + ". An alert message is sent to the Medical Server.\n");
            } else {

                Personal_state.append(msg3 + ", Oxygen Saturation is normal\n");
                SendToServer2.writeUTF("");
            }

            /* <Note: We sent an empty msg to server when data is normal
                    to be sure about the order of data that received  from server side>*/
            Personal_state.append("\n\n");

        } catch (IOException e) {
            Personal_state.append(e.toString());

        }

    }
}
```

3.3. Medical_Server Class

```java
import java.awt.TextArea;
import java.io.DataInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Medical_Server extends javax.swing.JFrame {

    // prort number
    private static final int port = 4495;

    /**
     * Creates new form Medical_Server
     */
    public Medical_Server() {
        // to initializes all of the Java swing components objects
        initComponents();
        // to change the colore of background to white
        getContentPane().setBackground(java.awt.Color.WHITE);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        Button_close1 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        Medical_state = new java.awt.TextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Medical State");
        setName("Medical_Server"); // NOI18N
        setPreferredSize(new java.awt.Dimension(794, 450));

        Button_close1.setBackground(new java.awt.Color(204, 204, 255));
        Button_close1.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 14)); // NOI18N
        Button_close1.setForeground(new java.awt.Color(51, 51, 51));
        Button_close1.setText("Close App");
        Button_close1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Button_close1ActionPerformed(evt);
            }
        });

        jLabel1.setBackground(new java.awt.Color(67, 141, 177));
        jLabel1.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 24)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setText(" Medical State");
        jLabel1.setName(""); // NOI18N
        jLabel1.setOpaque(true);

        jLabel2.setBackground(new java.awt.Color(255, 255, 255));
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/medical.jpg"))); // NOI18N

        Medical_state.setBackground(new java.awt.Color(225, 229, 229));
        Medical_state.setFont(new java.awt.Font("Courier New", 0, 12)); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 220, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(Button_close1, javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(Medical_state, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)
```


```java
                .addContainerGap(343, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(Medical_state, javax.swing.GroupLayout.PREFERRED_SIZE, 318, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(20, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(Button_close1)
                        .addGap(80, 80, 80))))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void Button_close1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Button_close1ActionPerformed
        // TODO add your handling code here:

        // to close the app in closeApp class
        CloseApp a = null;
        if (a == null) {
            a = new CloseApp();
        }
        a.setVisible(true);
        this.setVisible(false);
    }//GEN-LAST:event_Button_close1ActionPerformed

    public static void main(String args[]) throws Exception {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ServerHandler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ServerHandler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ServerHandler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ServerHandler.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Medical_Server().setVisible(true);
            }
        });

        // Text area to print Medical_state data on screen
        Medical_state = new TextArea();

        // Create Server Socket
        ServerSocket MedicalServer = new ServerSocket(port);

        try {

            /*
                * The Socket can have multiple connection
                * each iteration generate new connection with new receive data
             */
            while (true) {
                Socket clientSocket = MedicalServer.accept();
                // this thread make to start excution
                new Thread(new ServerHandler(clientSocket, Medical_state)).start();
            }
        } catch (IOException e) {
            System.err.println("Could not connect with port: " + port);

        }
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton Button_close1;
    private static java.awt.TextArea Medical_state;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    // End of variables declaration//GEN-END:variables
}

//**********************************************
final class ServerHandler implements Runnable {

    Socket clientSocket; // For Serving each client separately
    DataInputStream FromClient; //Read sent data from server socket
    public String TemperatureMSG, HeartMSG, OxygenMSG;

    TextArea Medical_state; // to print on screan

    /**
     * Creates new form ServerHandler
     */
    public ServerHandler(Socket clientSocket, TextArea Medical_state) throws IOException {
        this.clientSocket = clientSocket;
        this.Medical_state = Medical_state;
        initialize();
    }

    public void initialize() {
        try {
            FromClient = new DataInputStream(clientSocket.getInputStream());//To read data from client
        } catch (IOException e) {
            Medical_state.append(e.toString());
        }
```



```java
    }

    @Override
    public void run() {

        try {
            // to read data from client
            this.TemperatureMSG = FromClient.readUTF();
            this.HeartMSG = FromClient.readUTF();
            this.OxygenMSG = FromClient.readUTF();

        } catch (IOException e) {
            Medical_state.append(e.toString());
        }

        /*This condition check if all messages are really empty then no need to do anything else */
        if ((TemperatureMSG.isEmpty()) && (HeartMSG.isEmpty()) && (OxygenMSG.isEmpty())) {
            return;
        }

        /*The messages those aren't empty will display */
        if (!TemperatureMSG.isEmpty()) {
            Medical_state.append(TemperatureMSG + "\n");
        }
        if (!HeartMSG.isEmpty()) {
            Medical_state.append(HeartMSG + "\n");
        }
        if (!OxygenMSG.isEmpty()) {
            Medical_state.append(OxygenMSG + "\n");
        }

        /*if all messages aren't empty thats mean we can use them for following conditions
          , else the warning msg displey */
        if (!TemperatureMSG.isEmpty() && !HeartMSG.isEmpty() && !OxygenMSG.isEmpty()) {
            double TemperatureData = valueFromString(TemperatureMSG);
            int HeartData = (int) valueFromString(HeartMSG);
            int OxygenData = (int) valueFromString(OxygenMSG);

            if ((TemperatureData > 39) && (HeartData > 100) && (OxygenData < 95)) {
                Medical_state.append("Send an ambulance to the patient!\n");
            } else if ((TemperatureData > 38 && TemperatureData < 38.9) && (HeartData > 95 && HeartData < 98) && (OxygenData < 80)) {
                Medical_state.append("Call the patient's family!\n");

            } else {
                Medical_state.append("Warning, advise patient to make a checkup appointment!\n");
            }

        } else {

            Medical_state.append("Warning, advise patient to make a checkup appointment!\n");
        }

        Medical_state.append("\n\n");
    }

    private double valueFromString(String MSG) {
        double data;
        String temp;
        temp = MSG.substring(35, MSG.length()).replaceAll("[^0.0-9]", " ").trim();
        data = Double.valueOf(temp);
        return data;
    }
}
```

4. RHMS Application Run Snapshots

4.1. Run of the program on one machine

Case 1: 12 readings

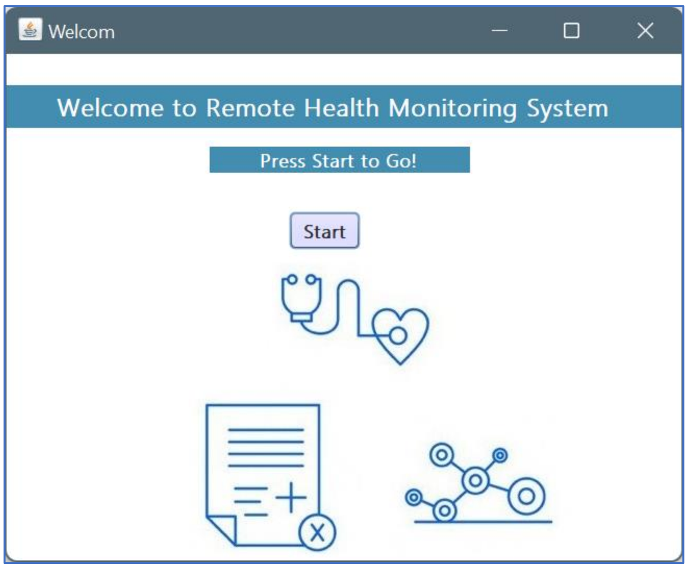


Figure 1: *Start Screen*

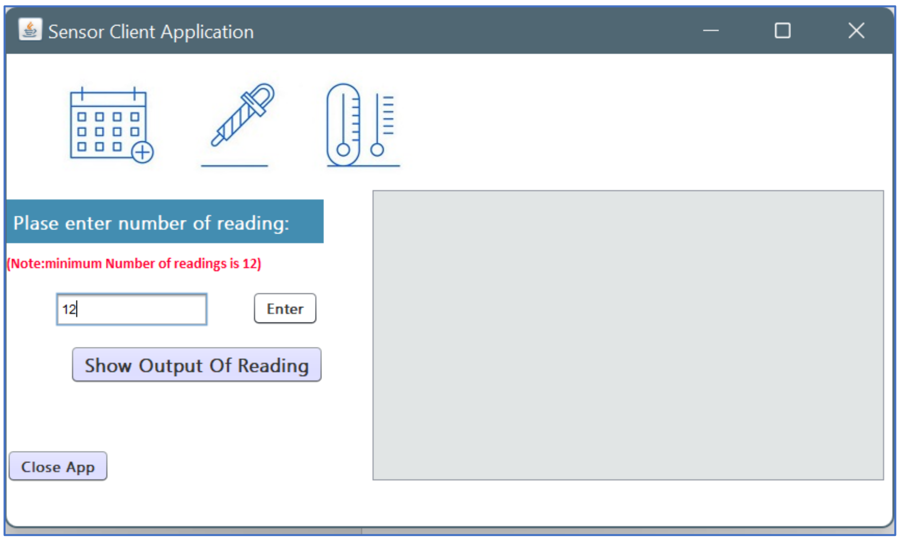
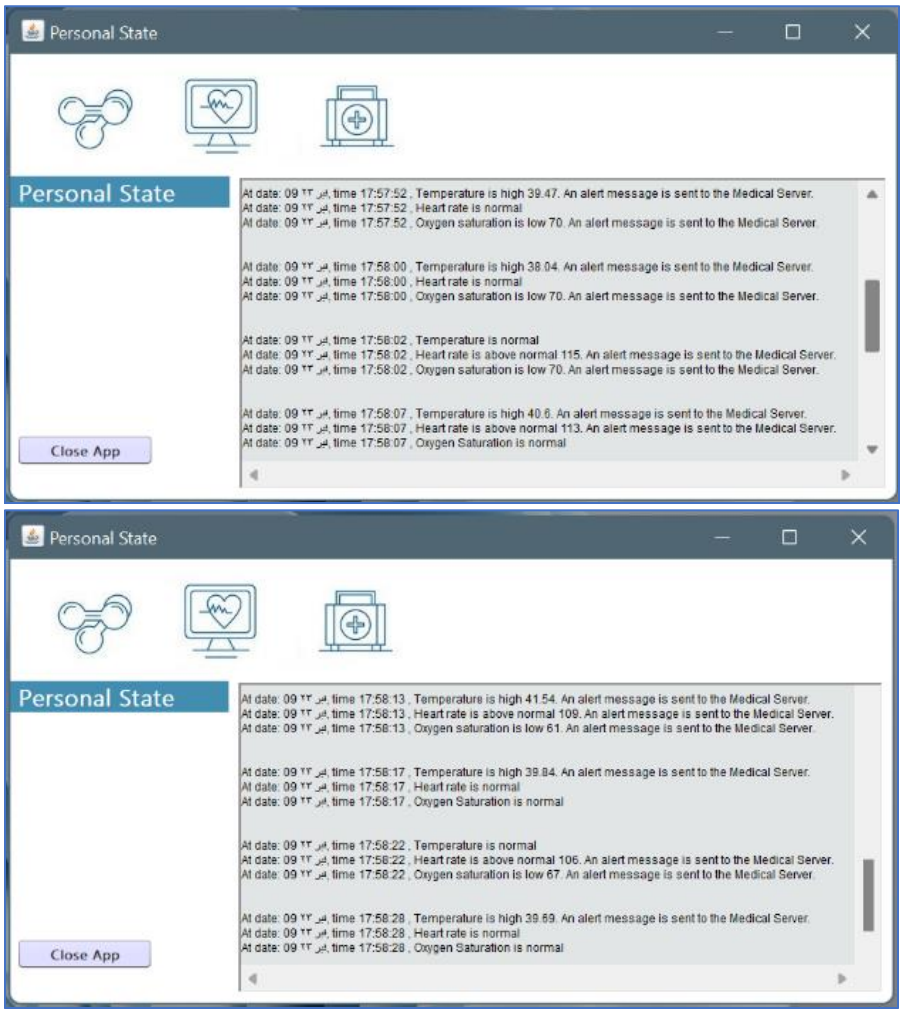

Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

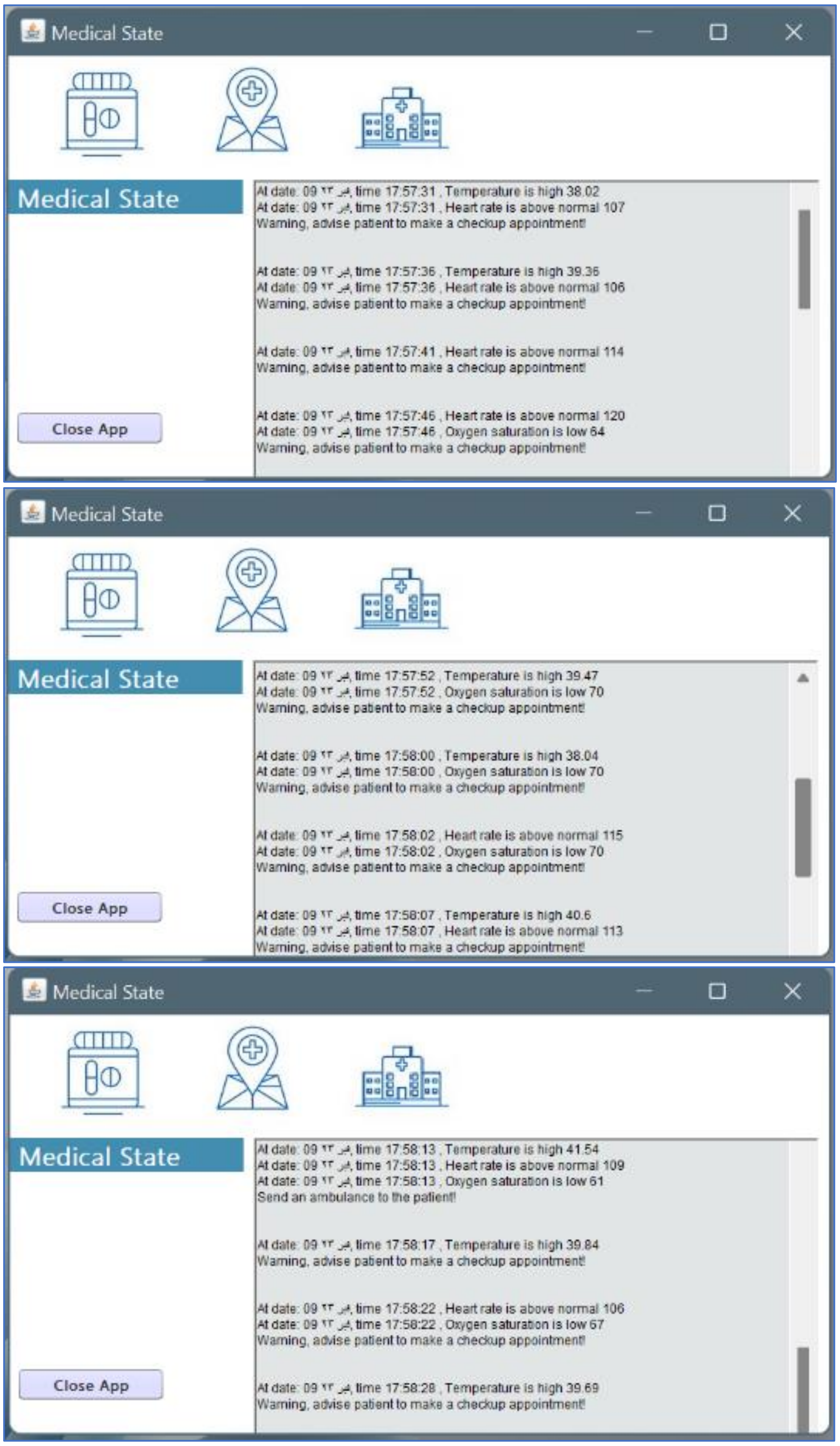
Figure 2: *Sensor Client Application (Enteringana acceptable number of readings)*



```
At date: 09 ٢٣ ير, time 17:57:52 , Temperature is high 39.47. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:57:52 , Heart rate is normal
At date: 09 ٢٣ ير, time 17:57:52 , Oxygen saturation is low 70. An alert message is sent to the Medical Server.


At date: 09 ٢٣ ير, time 17:58:00 , Temperature is high 38.04. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:00 , Heart rate is normal
At date: 09 ٢٣ ير, time 17:58:00 , Oxygen saturation is low 70. An alert message is sent to the Medical Server.


At date: 09 ٢٣ ير, time 17:58:02 , Temperature is normal
At date: 09 ٢٣ ير, time 17:58:02 , Heart rate is above normal 115. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:02 , Oxygen saturation is low 70. An alert message is sent to the Medical Server.


At date: 09 ٢٣ ير, time 17:58:07 , Temperature is high 40.6. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:07 , Heart rate is above normal 113. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:07 , Oxygen Saturation is normal
```

```
At date: 09 ٢٣ ير, time 17:58:13 , Temperature is high 41.54. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:13 , Heart rate is above normal 109. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:13 , Oxygen saturation is low 61. An alert message is sent to the Medical Server.


At date: 09 ٢٣ ير, time 17:58:17 , Temperature is high 39.84. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:17 , Heart rate is normal
At date: 09 ٢٣ ير, time 17:58:17 , Oxygen Saturation is normal


At date: 09 ٢٣ ير, time 17:58:22 , Temperature is normal
At date: 09 ٢٣ ير, time 17:58:22 , Heart rate is above normal 106. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:22 , Oxygen saturation is low 67. An alert message is sent to the Medical Server.


At date: 09 ٢٣ ير, time 17:58:28 , Temperature is high 39.69. An alert message is sent to the Medical Server.
At date: 09 ٢٣ ير, time 17:58:28 , Heart rate is normal
At date: 09 ٢٣ ير, time 17:58:28 , Oxygen Saturation is normal
```

Figure 3: *Personal Server Output*

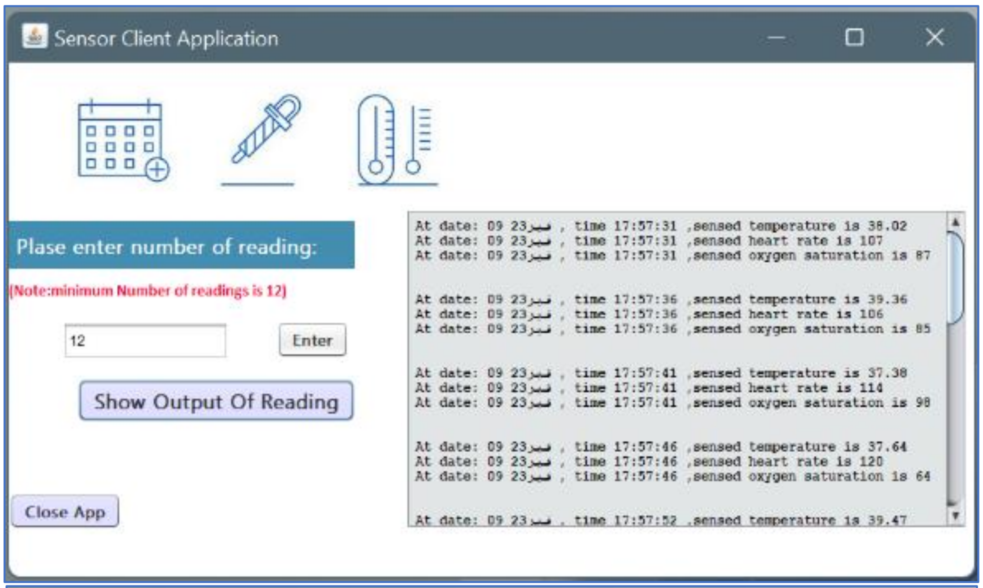


```
At date: 09 ٢٣ بر, time 17:57:31 , Temperature is high 38.02
At date: 09 ٢٣ بر, time 17:57:31 , Heart rate is above normal 107
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:57:36 , Temperature is high 39.36
At date: 09 ٢٣ بر, time 17:57:36 , Heart rate is above normal 106
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:57:41 , Heart rate is above normal 114
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:57:46 , Heart rate is above normal 120
At date: 09 ٢٣ بر, time 17:57:46 , Oxygen saturation is low 64
Warning, advise patient to make a checkup appointment!
```

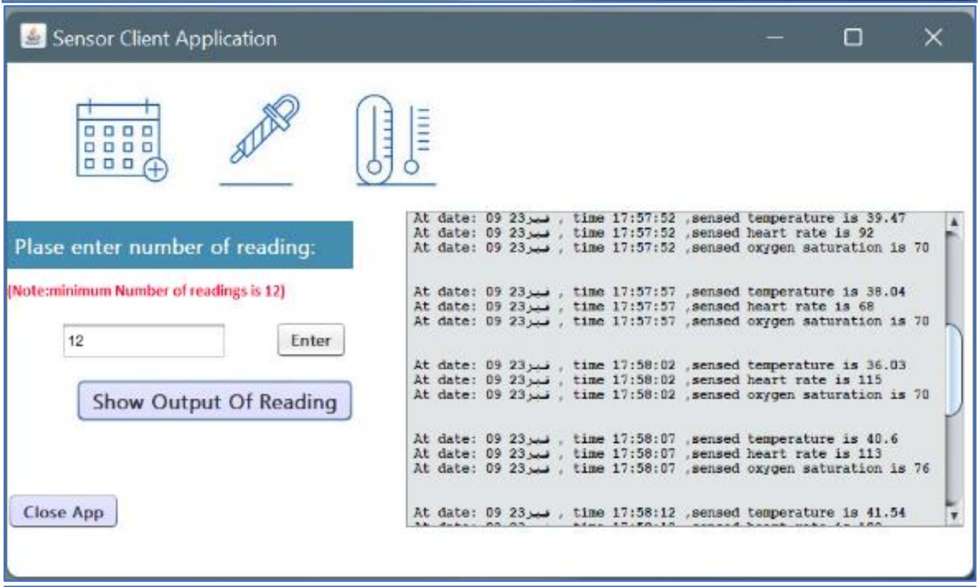
```
At date: 09 ٢٣ بر, time 17:57:52 , Temperature is high 39.47
At date: 09 ٢٣ بر, time 17:57:52 , Oxygen saturation is low 70
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:58:00 , Temperature is high 38.04
At date: 09 ٢٣ بر, time 17:58:00 , Oxygen saturation is low 70
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:58:02 , Heart rate is above normal 115
At date: 09 ٢٣ بر, time 17:58:02 , Oxygen saturation is low 70
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:58:07 , Temperature is high 40.6
At date: 09 ٢٣ بر, time 17:58:07 , Heart rate is above normal 113
Warning, advise patient to make a checkup appointment!
```

```
At date: 09 ٢٣ بر, time 17:58:13 , Temperature is high 41.54
At date: 09 ٢٣ بر, time 17:58:13 , Heart rate is above normal 109
At date: 09 ٢٣ بر, time 17:58:13 , Oxygen saturation is low 61
Send an ambulance to the patient!


At date: 09 ٢٣ بر, time 17:58:17 , Temperature is high 39.84
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:58:22 , Heart rate is above normal 106
At date: 09 ٢٣ بر, time 17:58:22 , Oxygen saturation is low 67
Warning, advise patient to make a checkup appointment!


At date: 09 ٢٣ بر, time 17:58:28 , Temperature is high 39.69
Warning, advise patient to make a checkup appointment!
```

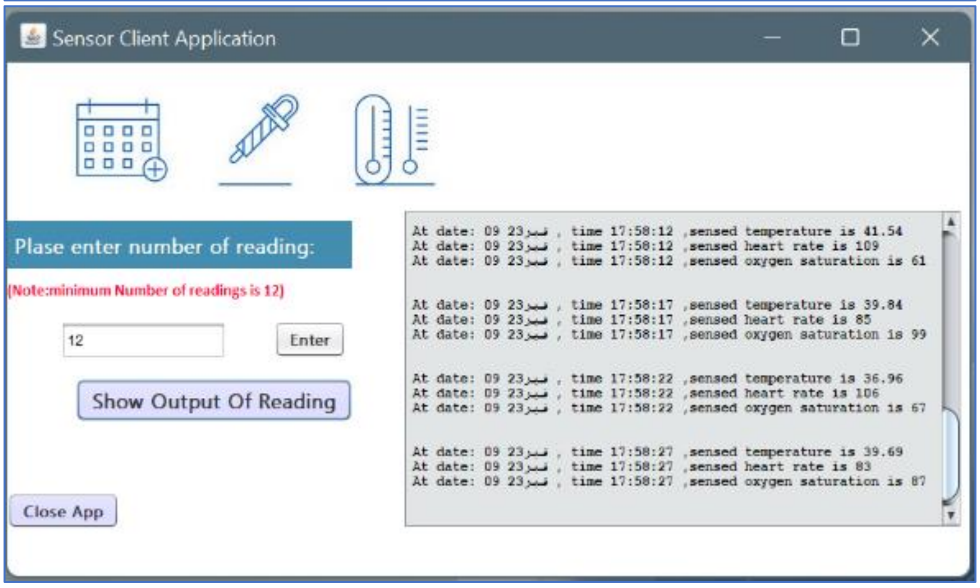
Figure 4: *Medical Server Output*


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

```
At date: 09 23قبر , time 17:57:31 ,sensed temperature is 38.02
At date: 09 23قبر , time 17:57:31 ,sensed heart rate is 107
At date: 09 23قبر , time 17:57:31 ,sensed oxygen saturation is 87


At date: 09 23قبر , time 17:57:36 ,sensed temperature is 39.36
At date: 09 23قبر , time 17:57:36 ,sensed heart rate is 106
At date: 09 23قبر , time 17:57:36 ,sensed oxygen saturation is 85


At date: 09 23قبر , time 17:57:41 ,sensed temperature is 37.38
At date: 09 23قبر , time 17:57:41 ,sensed heart rate is 114
At date: 09 23قبر , time 17:57:41 ,sensed oxygen saturation is 98


At date: 09 23قبر , time 17:57:46 ,sensed temperature is 37.64
At date: 09 23قبر , time 17:57:46 ,sensed heart rate is 120
At date: 09 23قبر , time 17:57:46 ,sensed oxygen saturation is 64


At date: 09 23قبر , time 17:57:52 ,sensed temperature is 39.47
```


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

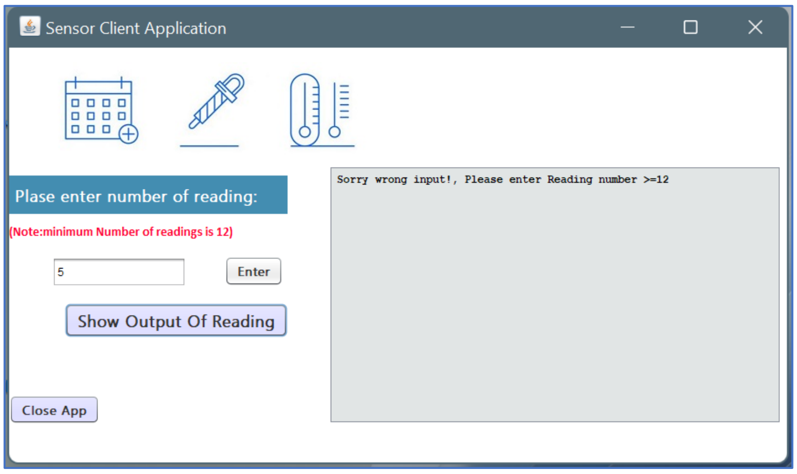
```
At date: 09 23قبر , time 17:57:52 ,sensed temperature is 39.47
At date: 09 23قبر , time 17:57:52 ,sensed heart rate is 92
At date: 09 23قبر , time 17:57:52 ,sensed oxygen saturation is 70


At date: 09 23قبر , time 17:57:57 ,sensed temperature is 38.04
At date: 09 23قبر , time 17:57:57 ,sensed heart rate is 68
At date: 09 23قبر , time 17:57:57 ,sensed oxygen saturation is 70


At date: 09 23قبر , time 17:58:02 ,sensed temperature is 36.03
At date: 09 23قبر , time 17:58:02 ,sensed heart rate is 115
At date: 09 23قبر , time 17:58:02 ,sensed oxygen saturation is 70


At date: 09 23قبر , time 17:58:07 ,sensed temperature is 40.6
At date: 09 23قبر , time 17:58:07 ,sensed heart rate is 113
At date: 09 23قبر , time 17:58:07 ,sensed oxygen saturation is 76


At date: 09 23قبر , time 17:58:12 ,sensed temperature is 41.54
```


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

```
At date: 09 23قبر , time 17:58:12 ,sensed temperature is 41.54
At date: 09 23قبر , time 17:58:12 ,sensed heart rate is 109
At date: 09 23قبر , time 17:58:12 ,sensed oxygen saturation is 61


At date: 09 23قبر , time 17:58:17 ,sensed temperature is 39.84
At date: 09 23قبر , time 17:58:17 ,sensed heart rate is 85
At date: 09 23قبر , time 17:58:17 ,sensed oxygen saturation is 99


At date: 09 23قبر , time 17:58:22 ,sensed temperature is 36.96
At date: 09 23قبر , time 17:58:22 ,sensed heart rate is 106
At date: 09 23قبر , time 17:58:22 ,sensed oxygen saturation is 67


At date: 09 23قبر , time 17:58:27 ,sensed temperature is 39.69
At date: 09 23قبر , time 17:58:27 ,sensed heart rate is 83
At date: 09 23قبر , time 17:58:27 ,sensed oxygen saturation is 87
```

Figure 5: *Sensor Client Application Output*

Case 2: 5 readings


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
5
Enter
Show Output Of Reading
Close App

```
Sorry wrong input!, Please enter Reading number >=12
```

Figure 6: *Sensor Client Application (Entering a nonacceptable number of readings)*

4.2. Run of the program on different machines

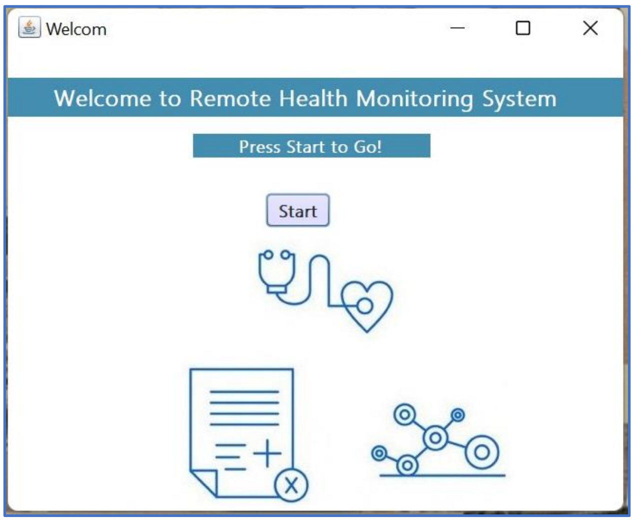


Figure 7: DeviceA start screen.

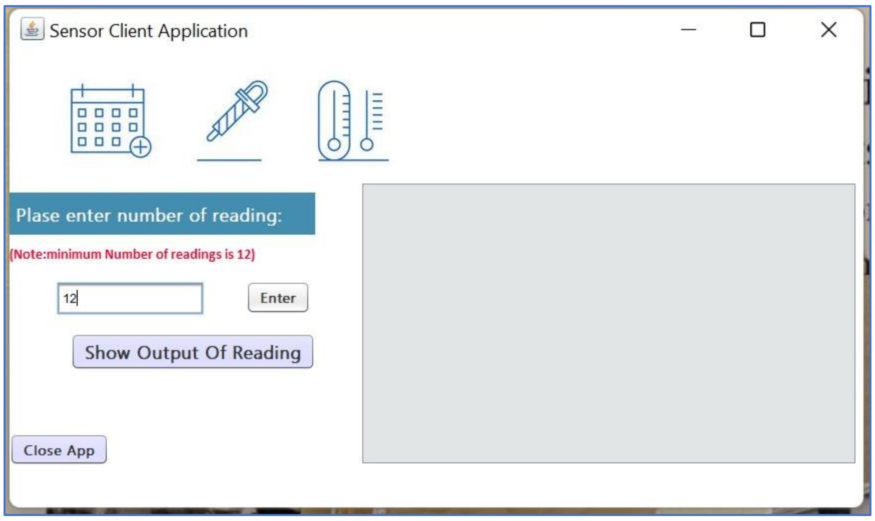
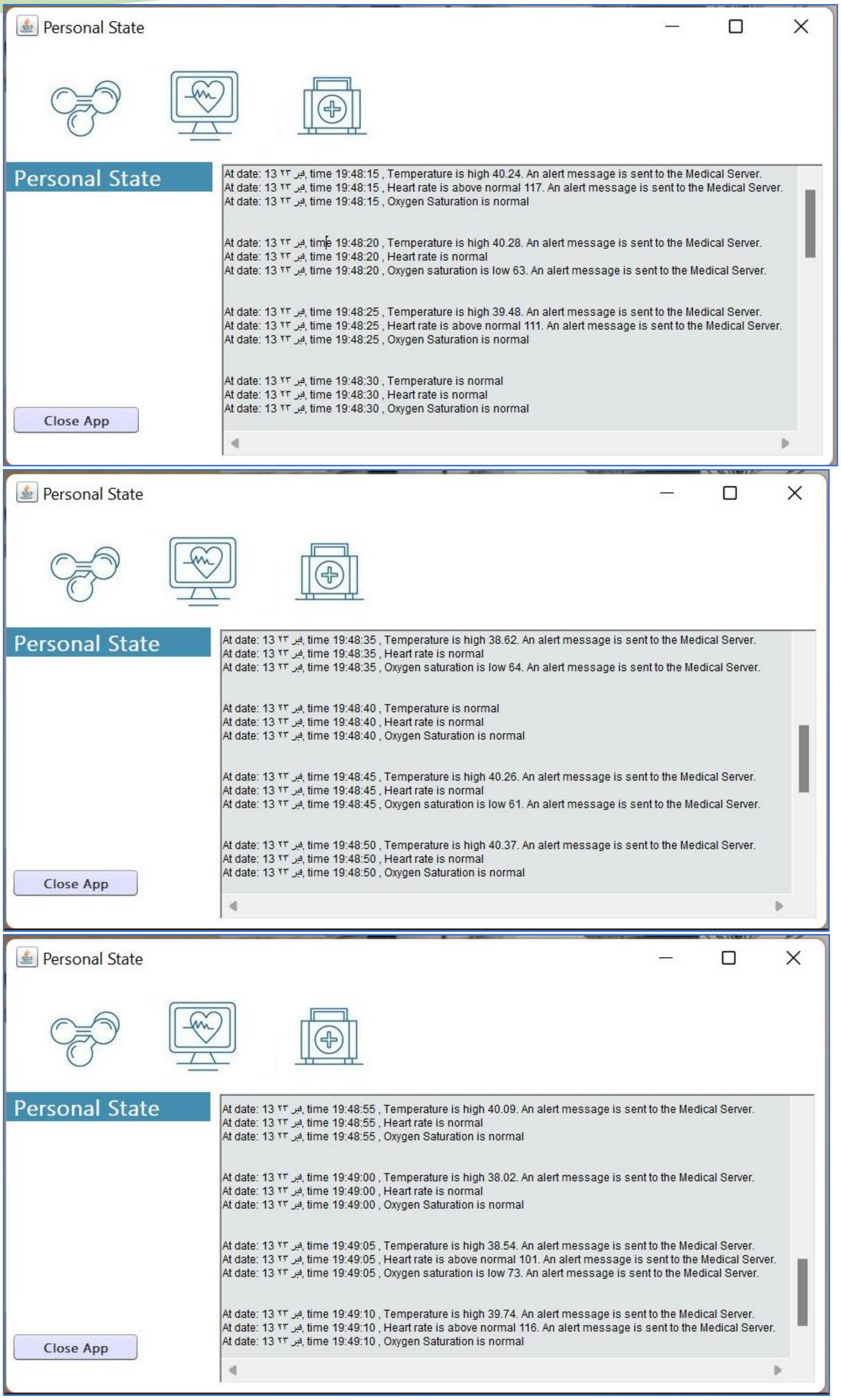

Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

Figure 8: DeviceA Sensor Client Application (Entering a number of readings)

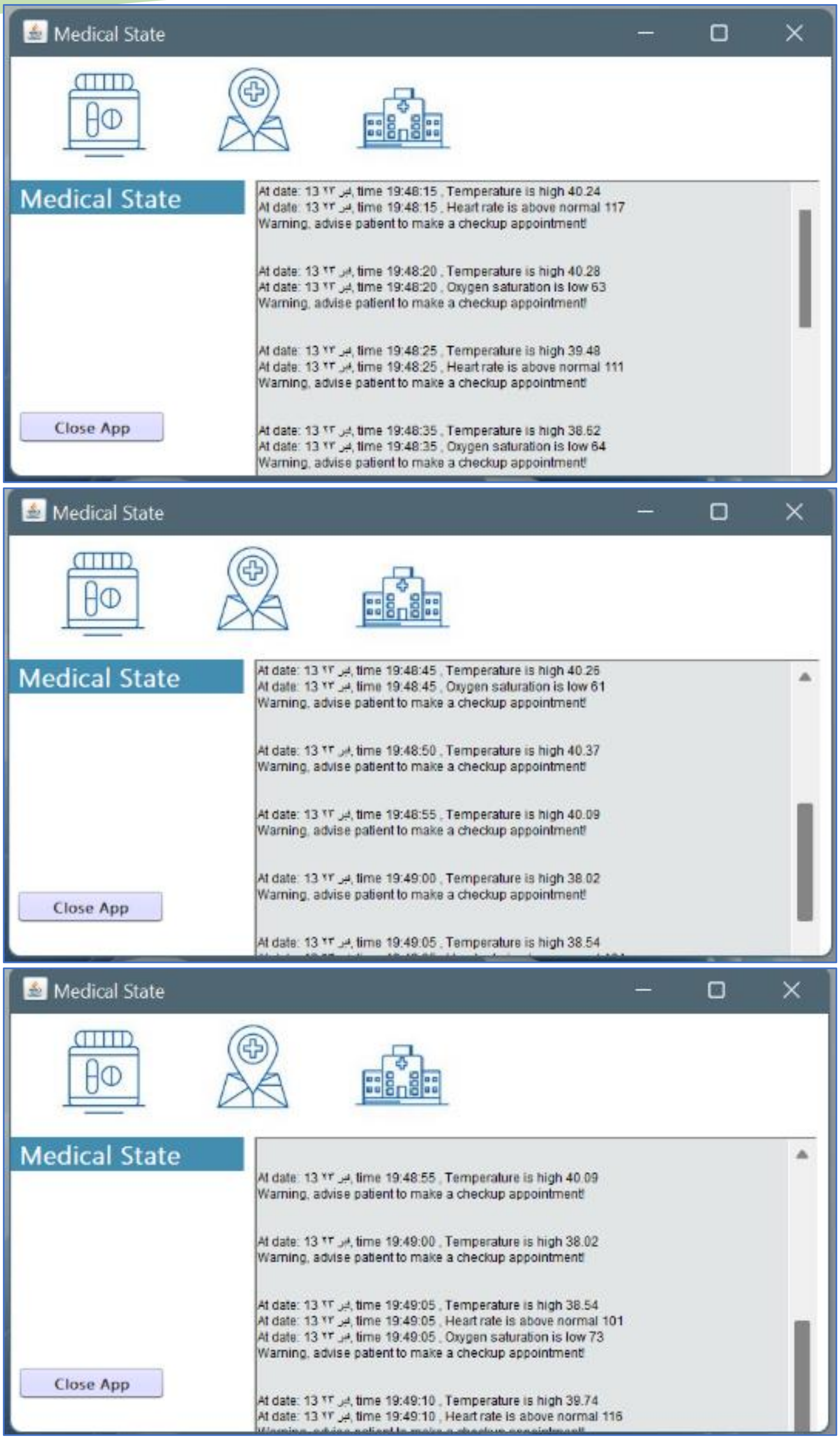


```
At date: 13 ٢٣ فبر, time 19:48:15 , Temperature is high 40.24. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:15 , Heart rate is above normal 117. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:15 , Oxygen Saturation is normal


At date: 13 ٢٣ فبر, time 19:48:20 , Temperature is high 40.28. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:20 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:20 , Oxygen saturation is low 63. An alert message is sent to the Medical Server.


At date: 13 ٢٣ فبر, time 19:48:25 , Temperature is high 39.48. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:25 , Heart rate is above normal 111. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:25 , Oxygen Saturation is normal


At date: 13 ٢٣ فبر, time 19:48:30 , Temperature is normal
At date: 13 ٢٣ فبر, time 19:48:30 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:30 , Oxygen Saturation is normal
```

```
At date: 13 ٢٣ فبر, time 19:48:35 , Temperature is high 38.62. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:35 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:35 , Oxygen saturation is low 64. An alert message is sent to the Medical Server.


At date: 13 ٢٣ فبر, time 19:48:40 , Temperature is normal
At date: 13 ٢٣ فبر, time 19:48:40 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:40 , Oxygen Saturation is normal


At date: 13 ٢٣ فبر, time 19:48:45 , Temperature is high 40.26. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:45 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:45 , Oxygen saturation is low 61. An alert message is sent to the Medical Server.


At date: 13 ٢٣ فبر, time 19:48:50 , Temperature is high 40.37. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:50 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:50 , Oxygen Saturation is normal
```

```
At date: 13 ٢٣ فبر, time 19:48:55 , Temperature is high 40.09. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:48:55 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:48:55 , Oxygen Saturation is normal


At date: 13 ٢٣ فبر, time 19:49:00 , Temperature is high 38.02. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:49:00 , Heart rate is normal
At date: 13 ٢٣ فبر, time 19:49:00 , Oxygen Saturation is normal


At date: 13 ٢٣ فبر, time 19:49:05 , Temperature is high 38.54. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:49:05 , Heart rate is above normal 101. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:49:05 , Oxygen saturation is low 73. An alert message is sent to the Medical Server.


At date: 13 ٢٣ فبر, time 19:49:10 , Temperature is high 39.74. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:49:10 , Heart rate is above normal 116. An alert message is sent to the Medical Server.
At date: 13 ٢٣ فبر, time 19:49:10 , Oxygen Saturation is normal
```

Figure 9: DeviceA Personal Server Output

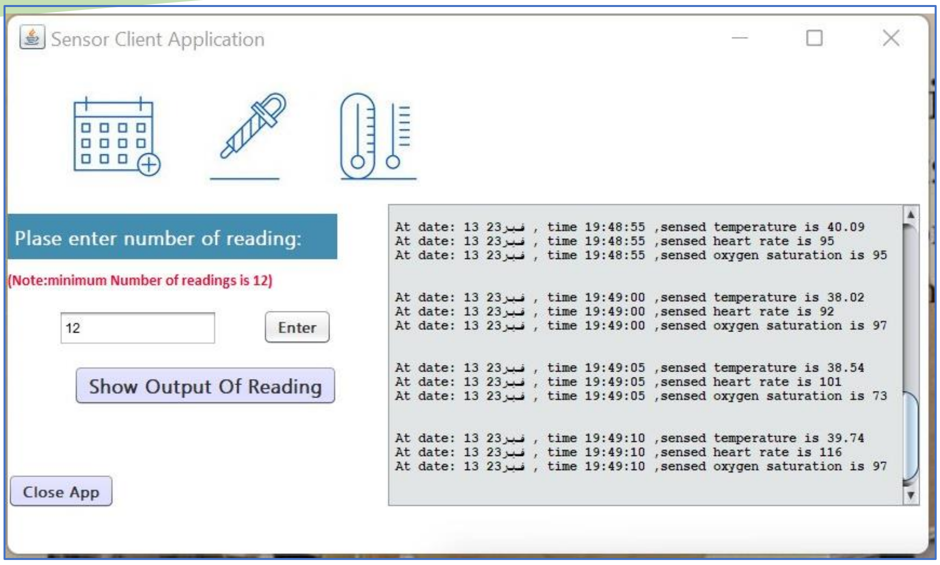


```
At date: 13 ٢٣ بر, time 19:48:15 , Temperature is high 40.24
At date: 13 ٢٣ بر, time 19:48:15 , Heart rate is above normal 117
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:48:20 , Temperature is high 40.28
At date: 13 ٢٣ بر, time 19:48:20 , Oxygen saturation is low 63
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:48:25 , Temperature is high 39.48
At date: 13 ٢٣ بر, time 19:48:25 , Heart rate is above normal 111
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:48:35 , Temperature is high 38.62
At date: 13 ٢٣ بر, time 19:48:35 , Oxygen saturation is low 64
Warning, advise patient to make a checkup appointment!
```

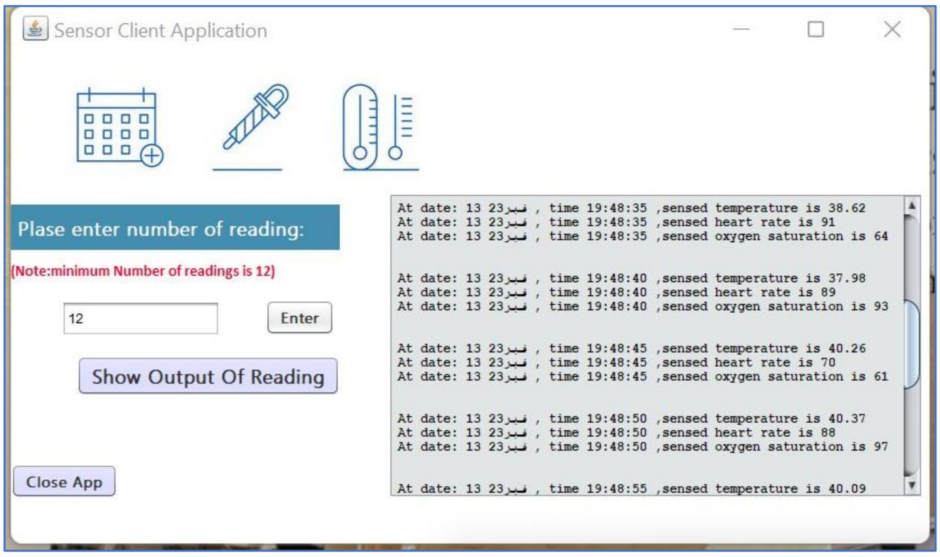
```
At date: 13 ٢٣ بر, time 19:48:45 , Temperature is high 40.25
At date: 13 ٢٣ بر, time 19:48:45 , Oxygen saturation is low 61
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:48:50 , Temperature is high 40.37
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:48:55 , Temperature is high 40.09
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:49:00 , Temperature is high 38.02
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:49:05 , Temperature is high 38.54
```

```
At date: 13 ٢٣ بر, time 19:48:55 , Temperature is high 40.09
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:49:00 , Temperature is high 38.02
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:49:05 , Temperature is high 38.54
At date: 13 ٢٣ بر, time 19:49:05 , Heart rate is above normal 101
At date: 13 ٢٣ بر, time 19:49:05 , Oxygen saturation is low 73
Warning, advise patient to make a checkup appointment!


At date: 13 ٢٣ بر, time 19:49:10 , Temperature is high 39.74
At date: 13 ٢٣ بر, time 19:49:10 , Heart rate is above normal 116
```

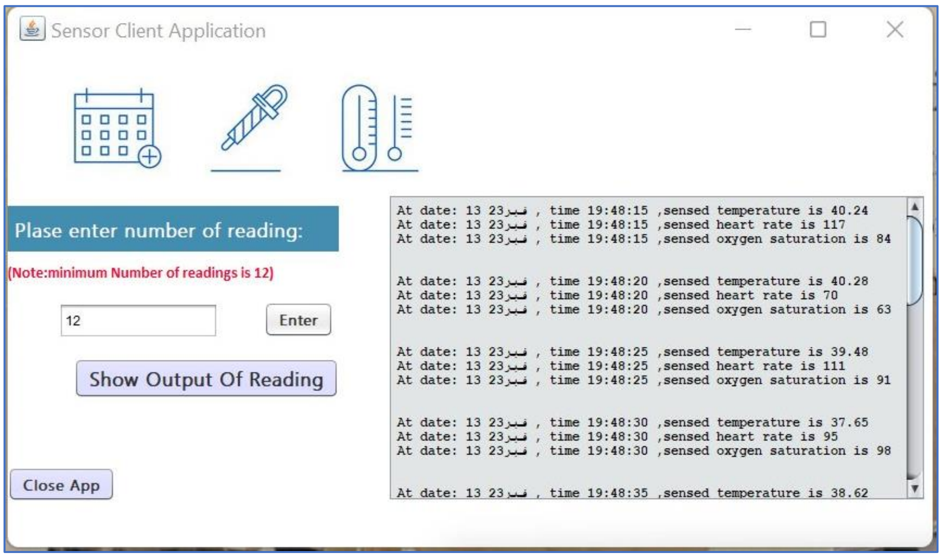
Figure 10: DeviceB Medical Server Output


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

```
At date: 13 23فبر , time 19:48:55 ,sensed temperature is 40.09
At date: 13 23فبر , time 19:48:55 ,sensed heart rate is 95
At date: 13 23فبر , time 19:48:55 ,sensed oxygen saturation is 95


At date: 13 23فبر , time 19:49:00 ,sensed temperature is 38.02
At date: 13 23فبر , time 19:49:00 ,sensed heart rate is 92
At date: 13 23فبر , time 19:49:00 ,sensed oxygen saturation is 97


At date: 13 23فبر , time 19:49:05 ,sensed temperature is 38.54
At date: 13 23فبر , time 19:49:05 ,sensed heart rate is 101
At date: 13 23فبر , time 19:49:05 ,sensed oxygen saturation is 73


At date: 13 23فبر , time 19:49:10 ,sensed temperature is 39.74
At date: 13 23فبر , time 19:49:10 ,sensed heart rate is 116
At date: 13 23فبر , time 19:49:10 ,sensed oxygen saturation is 97
```


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

```
At date: 13 23فبر , time 19:48:35 ,sensed temperature is 38.62
At date: 13 23فبر , time 19:48:35 ,sensed heart rate is 91
At date: 13 23فبر , time 19:48:35 ,sensed oxygen saturation is 64


At date: 13 23فبر , time 19:48:40 ,sensed temperature is 37.98
At date: 13 23فبر , time 19:48:40 ,sensed heart rate is 89
At date: 13 23فبر , time 19:48:40 ,sensed oxygen saturation is 93


At date: 13 23فبر , time 19:48:45 ,sensed temperature is 40.26
At date: 13 23فبر , time 19:48:45 ,sensed heart rate is 70
At date: 13 23فبر , time 19:48:45 ,sensed oxygen saturation is 61


At date: 13 23فبر , time 19:48:50 ,sensed temperature is 40.37
At date: 13 23فبر , time 19:48:50 ,sensed heart rate is 88
At date: 13 23فبر , time 19:48:50 ,sensed oxygen saturation is 97


At date: 13 23فبر , time 19:48:55 ,sensed temperature is 40.09
```


Sensor Client Application
Plase enter number of reading:
(Note:minimum Number of readings is 12)
12
Enter
Show Output Of Reading
Close App

```
At date: 13 23فبر , time 19:48:15 ,sensed temperature is 40.24
At date: 13 23فبر , time 19:48:15 ,sensed heart rate is 117
At date: 13 23فبر , time 19:48:15 ,sensed oxygen saturation is 84


At date: 13 23فبر , time 19:48:20 ,sensed temperature is 40.28
At date: 13 23فبر , time 19:48:20 ,sensed heart rate is 70
At date: 13 23فبر , time 19:48:20 ,sensed oxygen saturation is 63


At date: 13 23فبر , time 19:48:25 ,sensed temperature is 39.48
At date: 13 23فبر , time 19:48:25 ,sensed heart rate is 111
At date: 13 23فبر , time 19:48:25 ,sensed oxygen saturation is 91


At date: 13 23فبر , time 19:48:30 ,sensed temperature is 37.65
At date: 13 23فبر , time 19:48:30 ,sensed heart rate is 95
At date: 13 23فبر , time 19:48:30 ,sensed oxygen saturation is 98


At date: 13 23فبر , time 19:48:35 ,sensed temperature is 38.62
```

Figure 11: DeviceA Sensor Client Application Output

5. Teamwork and Lessons Learned

5.1. Planning and Coordinating

We divided up the tasks among us; three students worked on the report and reviewed it, while the other three worked on the code implementation We were holding regular meetings to go through task ideas, solutions, and implementation. However, we all cooperated and supported one another in completing the tasks.

Name of students	Tasks
Shahad Mohammed Bafadhel	Introduction and conclusion.
Waad Turki Megat Alharbi	RHMS implementation, Application run snapshots and Teamwork and Lessons learned
Shatha Khalid Binmahfouz	Patient Monitoring Application interaction diagram and Teamwork and Lessons learned
Razan Arif Alamri	Coding and GUI implementation.
Ghada Eisa Fzia Alsukmi	Coding and GUI implementation.
Sarah Abdulhadi Mahdi Aljohani	Coding and GUI implementation.

5.2. Difficulties

This project gave us the opportunity to implement practically our knowledge about crucial Network topics which are client-server architecture and TCP sockets. We also implemented new concepts and tools in programming via java which is multi-threading and GUI elements.

5.3. Learning Outcomes

Throughout the project, we have faced some challenges. In particular, the implementation of multi-client, threads, and GUI elements was difficult at the beginning because it was our first time to deal with those concepts. Yet, after searching on the Internet, we found helpful sources on YouTube that guide us to get over the problem.
The Internet is awesome.

6. Conclusion

Elderly patients deserve to live a better lifestyle and RHMS can help them live happier and healthier. So far, we discussed some points about client-server application, and TCP((Transmission Control Protocol), which is the most common transport layer protocol between server and client. Next, we talked about threads which are the steps performed and the order in which the steps are performed by a running program. After that, we reviewed java swing which is a lightweight GUI toolkit and used its variety of elements, including labels, text fields, text areas, images, buttons, and sounds.

We also discussed the RHMS and its purpose as well as the roles used for both the client and the servers. Moreover, we gave an overview of the whole report. We drew Client-Server Interaction Diagram with pseudocode too. Furthermore, sections three and four shows a list of the application code and different cases snapshots of our application. Finally, section five is discussing teamwork and lessons learned.

7. References

[1] Kurose, James F., and Keith W. Ross. *Computer Networking*. 6th ed. Pearson Education, 2012.

[2] Liang, Y. Daniel. Introduction to Java programming: comprehensive version. Pearson Education, 2015.

[3] Deitel, P., & Deitel, H. (2011). Java How to program. Prentice Hall Press.