King Abdulaziz University

Faculty of Computing and Information Technology

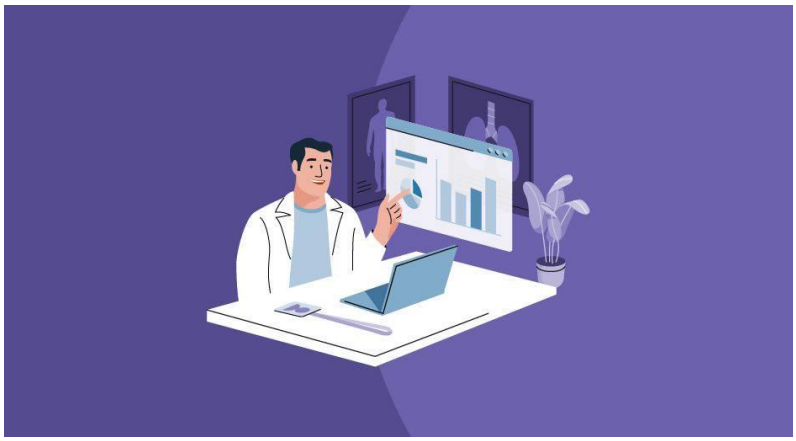
Jeddah, Saudi Arabia

Course Project Report:

Simulating a Remote Health Monitoring System for elderly patients using client-server TCP Sockets

CPCS 371- Computer Network I

Group 7



Name	ID
Rahaf Atteh Albrakati	1906995
Abrar Ramadan Alkamajani	1907027
Roaa Abdullah Alzhrani	2005863
Hanin Suleiman Omer Alhaj	2010269
Shaima Bashammakh	1914892
Alaa Emad Alhamzi	2010304

Table of Contents

1. Introduction

1.1 client-server applications

A common software design architecture called client-server divide program into client-side and server-side. The client is the application that runs on the end-user computer. The client sends requests to the server to obtain data or content. Responses to client requests are handled by the server-side program. They can converse either directly on the same computer or through a network. Applications that are running on different machines must be physically connected typically through a network (LAN, WAN, or Internet).

1.2 Java TCP sockets

Network communication between applications is established and maintained according to the Transmission Control Protocol (TCP) standard. TCP works with the Internet Protocol (IP), which specifies how computers send data packets to each other. Java.net packaging provides the classes necessary to implement networking applications. Java.net.sockets, are establishing fundamental mechanisms for bidirectional data transmission. Java.net.ServerSocket listens for network requests to arrive. Based on the request, it executes some operation before possibly returning a result to the requester.

1- The server uses a server socket to construct a TCP socket with the port number of the client, then waits for client requests.
2- Using accept, the server acknowledges a client's request on the same port.
3- The server then waits for a client connection.
4- To establish a connection successfully, the client constructs a TCP socket with the server's IP address and a port number that must match the server's port number.
5- If the connection is successful, step 5 is to start communicating.
6- Through sockets, the getOutputStream() and getInputStream() functions are utilized for I/O communication on both sides.

1.3. GUIs Elements

In our application, we created the RHMS interface using the Swing library for all servers and clients. To alert the user, a dialog window is used. A set of UI components are used to dynamically update the user interface on a JFrame. Also, the main components that we used are textboxes, labels, and buttons.

1.4 Remote Health Monitoring System(RHMS)

Several older patients have chronic illnesses. In our project, we are going to implement a simulation of a typical Remote Health Monitoring System (RHMS) which uses Wireless Body Area Networks (WBAN) to enable remote monitoring of patients with chronic illnesses. RHMS is based on the client-server architecture. Heart rate, oxygen saturation, and temperature are the three types of sensors that the system you must implement takes into account.

1.5 Role of the clients and servers in this application

The RHMS consists of (3) main components sensor(client), personal play roles (client and server), and medical (server)

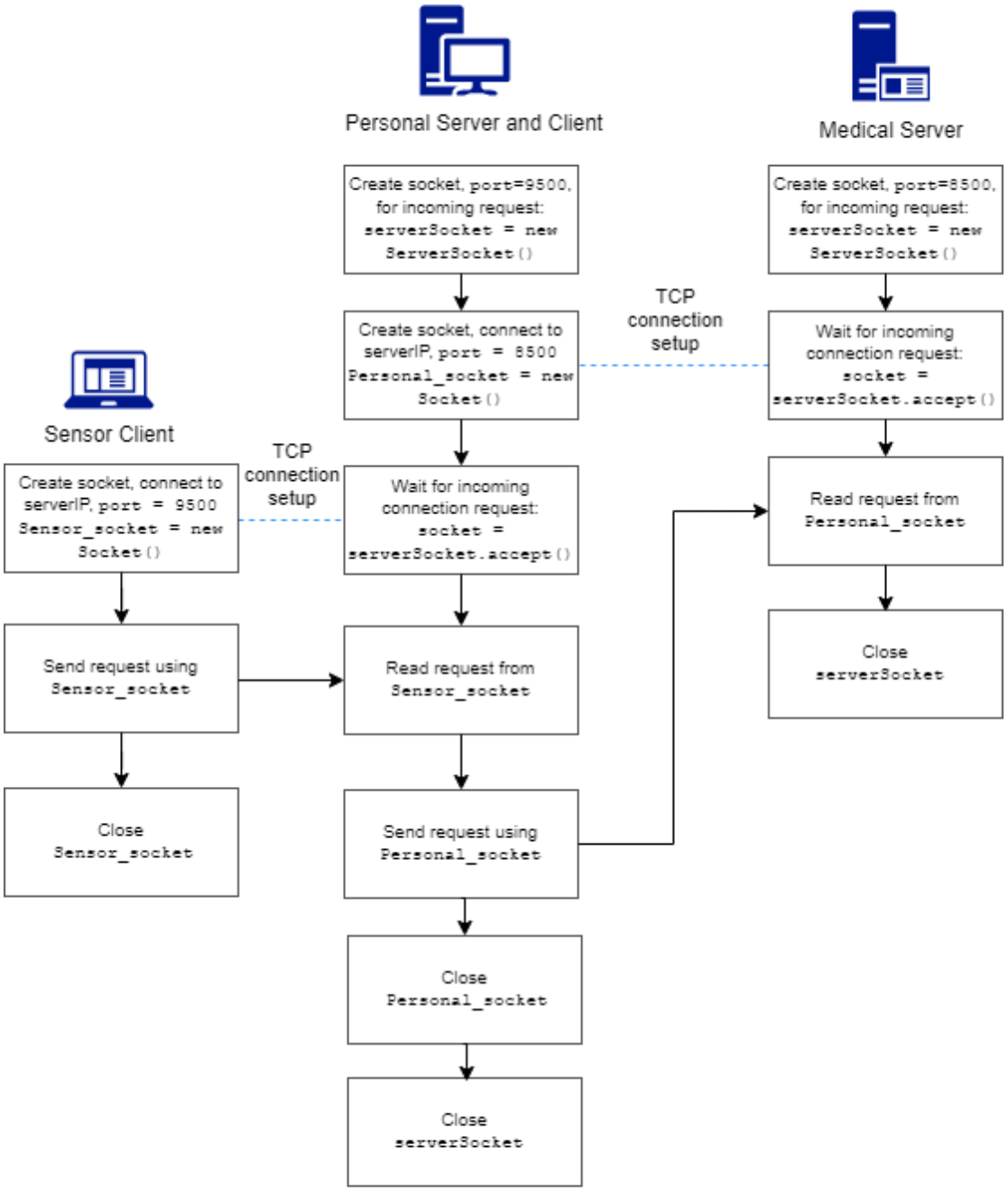
1- The Sensor Client Application gathers sensor data and transmits it to the Personal Server's server side.
2- The personal Server, which performs two functions for the Sensor Client Application, analyzes the sensor data received and chooses whether send it to the caregiver or not.
3- The client of the Personal sends the required messages to the Medical Server if the processed information needs to be provided.
4- The Medical Server receives messages about the patient's status from the personal. The Medical Server displays the received messages and displays the appropriate action to be taken by the caregiver.

1.6 overview

The following sections in this report will cover the RHMS interaction diagram in section 2. After that, we will show a list of the application code and different case snapshots in sections 3-4. Furthermore, we will be discussed in section 5 work division, coordination, and managing our teamwork together. Finally, we will conclude in section 6 by summarizing our project.

2. Patient Monitoring Application interaction diagram

2.1 Client-Server Interaction Diagram



Interaction diagram

2.2 Pseudocode

2.2.1 Sensor Application Pseudocode

ALGORITM Sensor Application

// This algorithm demonstrates the sensor application class for this system

Create a connection socket to connect with the Personal Server

start ← current time

end ← start + 60 * 1000

While start < end:

    //the values here is generated randomly

    Read temperature values from the user

    // the values here is generated randomly

    Read Heart Rate values from the user

    // the values here is generated randomly

Read Oxygen values from the user

Display the sensed data

Send the sensed data to Personal Server

End while loop.

Close Client Socket.

2.2.2 Personal Server Pseudocode

ALGORITM Personal server

// This algorithm demonstrates the medical server class for this system

Create a listening TCP socket

while true:

Wait for sensor client to establish connection

Accept connection from sensor client as sensorSocket

while true:

Read the temperature values from the Sensor application

Read the heart rate values from the Sensor application

Read the oxygen level values from the Sensor application

If normal conditions happened:

Display the received values from the sensors class

else

if temperature > 38:

Display an alert message

Send the received values to the medical class

if the heart rate > 100:

Display an alert message

Send the received values to the medical class

if the heart rate < 60:

Display an alert message

Send the received values to the medical class

if the oxygen saturation is < 75:

Display an alert message

Send the received values to the medical class

End if

if the client (sensor) sent 12 reading

the server (personal) will close the TCP

break

end if

end while loop

end while loop

2.2.3 Medical Server Pseudocode

ALGORITM Medical server

// this algorithm is demonstrates the medical server class for this system

Create a listening TCP socket

while true:

Wait for Personal client to establish connection

Accept connection from Personal client as PersonalSocket

while true:

If the client (personal) sent 12 reading and the value of "EndOrNo" was "end"

the server (medical) will close the TCP connection

End If

Read the temperature values from the personal server

Read the heart rate values from the personal server

Read the oxygen level values from the personal server

Display the received values from the personal class

If abnormal condition happened:

Choose on the appropriate action

Display the output on the caregiver interface

End If

End while loop

End while loop

3. RHMS Implementation

3.1 Server and clients code

3.1.1 Sensor class

```
//CPCS 371 - Project 2 - Group 7

import javax.swing.JOptionPane;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;

public class Sensors extends javax.swing.JFrame {

    public Sensors() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        //display message dialog contain the following message
        JOptionPane.showMessageDialog(null, "Welcome to the sensor client.");
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JScrollPane jScrollPane1;
    private static javax.swing.JTextArea textBox;
    // End of variables declaration

public static void main(String args[]) throws IOException {

        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(Sensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Sensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
            java.util.logging.Logger.getLogger(Sensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Sensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }

        new Sensors().setVisible(true);

        //start a class
         //create a socket for a client (sensors)
        Socket client_socket = new Socket("localhost", 9500);

        //create a dataOutputStream to send a request/data to a server (personal)
        DataOutputStream request_to_Server = new DataOutputStream(client_socket.getOutputStream());

        //create currentTimeMillis objects
        long start = System.currentTimeMillis();
        long end = start + 60 * 1000; //60 because of 60 seconds = 1 min as required

        // variables
        //counter used to know if we reach 12 inputs/values
        int counter = 1;
        //variables for temperature, heart rate and oxygen saturation
        int tem, oxg, heartRate = 0;


        //create date and time objects
        //time
        DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("HH:mm:ss");
        //date
        DateTimeFormatter myFormatObj2 = DateTimeFormatter.ofPattern("dd MMM yy");



        //while loop until 60 seconds
        while (System.currentTimeMillis() < end) {

            //Generate random numbers for temperature, heart rate and oxygen saturation
            tem = (int) (Math.random() * (41 - 36 + 1) + 36);
            heartRate = (int) (Math.random() * (120 - 50 + 1) + 50);
            oxg = (int) (Math.random() * (100 - 60 + 1) + 60);

            //send counter value to server (personal)
            //send counter to know when the TCP connection is closed
            //if counter = 12 -> the TCP connection will close
            request_to_Server.writeUTF(counter + "");
```

```
            //send temperature, heart rate and oxygen saturation values to server (personal)
            request_to_Server.writeUTF(tem + "");
            request_to_Server.writeUTF(heartRate + "");
            request_to_Server.writeUTF(oxg + "");


            //create and generate time and date
            //print the time and date + temperature, heart rate and oxygen saturation values

            //create the date
            String date = LocalDate.now().format(myFormatObj2);

            //create the first time (for temperature)
            String time1 = LocalTime.now().format(myFormatObj);
           //print in GUI using textBox
            textBox.append("\nAt date: " + date + ", time: " + time1+ ", sensed temperature is " + tem);
            //stop 1 second
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ie) {
            }


             //create the second time (for heart rate)
            String time2 = LocalTime.now().format(myFormatObj);
            //print in GUI using textBox
            textBox.append("\nAt date: " + date + ", time: " + time2 + ", sensed heart rate is " + heartRate);

            //stop 1 second
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ie) {
            }


             //create the third time (for oxygen saturation)
            String time3 = LocalTime.now().format(myFormatObj);
            //print in GUI using textBox
            textBox.append("\nAt date: " + date + ", time: " + time3 +", sensed oxygen saturation is " + oxg + "\n");


             //stop 3 seconds
             try {
                Thread.sleep(3000);
            } catch (InterruptedException ie) {
            } //After this stop: the heart rate and oxygen saturation values ware generated every 5 seconds
```

```
                //increase the counter
                counter++;
            }//end loop

            //close socket
            client_socket.close();


        }//end main

}//end class
```

3.1.2 Personal class

```
//CPCS 371 - Project 2 - Group 7

import javax.swing.JOptionPane;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;

public class Personal extends javax.swing.JFrame {

    public Personal() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        //display message dialog contain the following message
        JOptionPane.showMessageDialog(null, "Welcome to the personal client/server.");

    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JScrollPane jScrollPanel;
    private static javax.swing.JTextArea textBox;
    // End of variables declaration

    public static void main(String args[]) throws IOException {

        new Personal().setVisible(true);

        //As server

        //create a server socket (personal)
        ServerSocket server_socket = new ServerSocket(9500);
```

```
// ---------------------------- As client -------------------------
//create a socket for client (personal)
Socket client_socket = new Socket("localhost", 1500);

//create a dataOutputStream to send a request to a server
DataOutputStream request_to_Server = new DataOutputStream(client_socket.getOutputStream());

// ---------------------------- As client -------------------------


//loop to make a server always ON to reaceive any TCP connection
while (true) {

    // As server
    //create a TCP Connection - The handshaking step
    Socket socket = server_socket.accept();
    //create a datainputstream to recieve a data/message from a client
    DataInputStream Request_from_client = new DataInputStream(socket.getInputStream());


    String Temp = "";
    int intTemp;
    String Heart = "";
    int intHeart;
    String Oxg = "";
    int intOxg;

    //loop to make a TCP connection is persistent
    while (true) {

        //take a data from a client (sensor)
        //this variable contain counter value that send from clinet(sensor)
        String recieved_data = Request_from_client.readUTF();

        //recieve temperature, heart rate and oxygen saturation values
        Temp = Request_from_client.readUTF();
        Heart = Request_from_client.readUTF();
        Oxg = Request_from_client.readUTF();


        //create and generate time and date
        //create the date and the time objects
        DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("HH:mm:ss");
        DateTimeFormatter myFormatObj2 = DateTimeFormatter.ofPattern("dd MMM yy");

        //create the date
        String date = LocalDate.now().format(myFormatObj2);

        //create the first time (for temperature)
        String time1 = LocalTime.now().format(myFormatObj);
        //stop 1 second
        try {
        Thread.sleep(1000);
    } catch (InterruptedException ie) {
    }

        //create the second time (for heart rate)
        String time2= LocalTime.now().format(myFormatObj);
        //stop 1 second
        try {
        Thread.sleep(1000);
    } catch (InterruptedException ie) {
    }

        //create the third time (for oxygen saturation)
        String time3 = LocalTime.now().format(myFormatObj);



        //convert string to integer
        intTemp = Integer.parseInt(Temp);
        intHeart = Integer.parseInt(Heart);
        intOxg = Integer.parseInt(Oxg);



        //comparison the values of temperature, heart rate and oxygen saturation
        if (intTemp <= 38 && !(intHeart > 100 || intHeart < 60) && intOxg >= 75) {
            //print in GUI using textBox
            textBox.append("At date: " + date + "AT time: "+ time1 + ", Temperature is normal " + intTemp);
            textBox.append("\nAt date: " + date + "AT time: "+ time2 + ", Heat rate is normal " + intHeart);
            textBox.append("\nAt date: " + date + "AT time: "+ time3 +", Oxygen saturation is normal " + intOxg+"\n\n");
        } else {

            //send to counter value to server (medical)
            request_to_Server.writeUTF(recieved_data);
            //send this value to server (medical) to inform the server not to close the TCP connection
            request_to_Server.writeUTF("not end");
```

```
                //1- check temperature value and comparison
                //firstly print and then send message to the server (medical)
                if (intTemp <= 38) {
                 request_to_Server.writeUTF("At date: "+date + "AT time: "+ time1 +", Temperature is normal " + intTemp);
                 //print in GUI using textBox
                    textBox.append("At date: " + date + "AT time: "+ time1 +", Temperature is normal " + intTemp );
                } else if (intTemp > 38) {
                 request_to_Server.writeUTF("At date: "+date + "AT time: "+ time1 +", Temperature is high " + intTemp);
                 //print in GUI using textBox
                    textBox.append("At date: " + date + "AT time: "+ time1 +", Temperature is high " + intTemp + ". An alert message is sent to the Medical Server. " );

                }

                //2- check heart rate value and comparison
                //firstly print and then send message to the server (medical)
                if (intHeart > 100 || intHeart < 60) {
                    if (intHeart > 100) {
                    request_to_Server.writeUTF("At date: "+date + "AT time: "+ time2 +", Heart rate is above normal " + intHeart);
                    //print in GUI using textBox
                        textBox.append("\nAt date: " + date + "AT time: "+ time2 +", Heat rate is above normal " + intHeart + ". An alert message is sent to the Medical Server. " );

                    } else if (intHeart < 60) {
                    request_to_Server.writeUTF("At date: "+date + "AT time: "+ time2 +", Heart rate is below normal " + intHeart);
                    //print in GUI using textBox
                        textBox.append("\nAt date: " + date + "AT time: "+ time2 +", Heat rate is below normal " + intHeart + ". An alert message is sent to the Medical Server. " );

                    }
                } else {
                 request_to_Server.writeUTF("At date: "+date + "AT time: "+ time2 +", Heart rate is normal " + intHeart);
                 //print in GUI using textBox
                    textBox.append("\nAt date: " + date + "AT time: "+ time2 +", Heart rate is normal " + intHeart );

                }

                //3- check oxygen saturation value and comparison
                //firstly print and then send message to the server (medical)
                if (intOxg >= 75) {
                request_to_Server.writeUTF("At date: "+date + "AT time: "+ time3 +", Oxygen saturation is normal " + intOxg);
                //print in GUI using textBox
                    textBox.append("\nAt date: " + date + "AT time: "+ time3 +", Oxygen saturation is normal " + intOxg +"\n\n");
                } else if (intOxg < 75) {
                request_to_Server.writeUTF("At date: "+date + "AT time: "+ time3 +", Oxygen saturation is low " + intOxg);
                //print in GUI using textBox
                    textBox.append("\nAt date: " + date + "AT time: "+ time3 +", Oxygen Saturations is low " + intOxg + ". An alert message is sent to the Medical Server. \n\n");

                }
```

```
                    }
                }

                //if the client (sensor) sent "12", the server will close the TCP connection
                if (recieved_data.equalsIgnoreCase("12")) {

                    // ---------------------------- As client ----------------------------
                     //send 12 to server (medical) to close TCP connection
                    request_to_Server.writeUTF("12");
                     //send this value to server (medical) to close TCP connection
                    request_to_Server.writeUTF("end");
                    //close socket of a client (personal)
                     client_socket.close();
                    // ---------------------------- As client ----------------------------

                    //close TCP connection of server (personal)
                    socket.close();
                    break;
                } //end if

               }//end loop
            }//end loop

    }//end main

}//end class
```

3.1.3 Medical class

```
//CPCS 371 - Project 2 - Group 7

import java.io.*;
import java.net.*;
import javax.swing.JOptionPane;

public class Medical extends javax.swing.JFrame {

    public Medical() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        //display message dialog contain the following message "Welcome to the medical server."
        JOptionPane.showMessageDialog(null, "Welcome to the medical server.");
    }

    // Variables declaration - do not modify
    private javax.swing.border.EmptyBorder emptyBorder1;
    private javax.swing.JButton jButton1;
    private javax.swing.JColorChooser jColorChooser1;
    private javax.swing.JColorChooser jColorChooser2;
    private javax.swing.JColorChooser jColorChooser3;
    private javax.swing.JScrollPane jScrollPane1;
    private static javax.swing.JTextArea textBox;
    // End of variables declaration

    public static void main(String args[]) throws IOException {

        Medical m = new Medical();
        m.setVisible(true);

        //create a server socket for (medical)
         ServerSocket server_socket = new ServerSocket(1500);
```

```
// make server on
while (true) {

  // Create a TCP Connection - The handshaking step
  Socket socket = server_socket.accept();
  // Create a datainputstream to recieve a data/message from a client
  DataInputStream Request_from_client = new DataInputStream(socket.getInputStream());


  //variables for temperature, heart rate and oxygen saturation
   String Temp = "";
   String Heart = "";
   String Oxg = "";

  //loop to make a TCP connection is persistent
   while (true) {

      //take a value of counter from a client (personal)
      String recieved_data = Request_from_client.readUTF();
      //used if we want to close the TCP connection of the server (medical)
      //if the value was "end" -> close TCP connection
      String EndOrNo = Request_from_client.readUTF();

       //if the client (personal) sent "12" && the value of "EndOrNo" was "end" -> the server (medical) will close the TCP connection
      if (recieved_data.equalsIgnoreCase("12")&& EndOrNo.equalsIgnoreCase("end")) {
          //close TCP connection.
          socket.close();
          break;
      } //end if



      //recienve messages from client (personal)
       Temp = Request_from_client.readUTF();
       Heart = Request_from_client.readUTF();
       Oxg = Request_from_client.readUTF();

       //print the recieved messages (messages from personal)
       textBox.append(Temp + ".");
       textBox.append("\n" + Heart + ".");
       textBox.append("\n" + Oxg + ".");
```

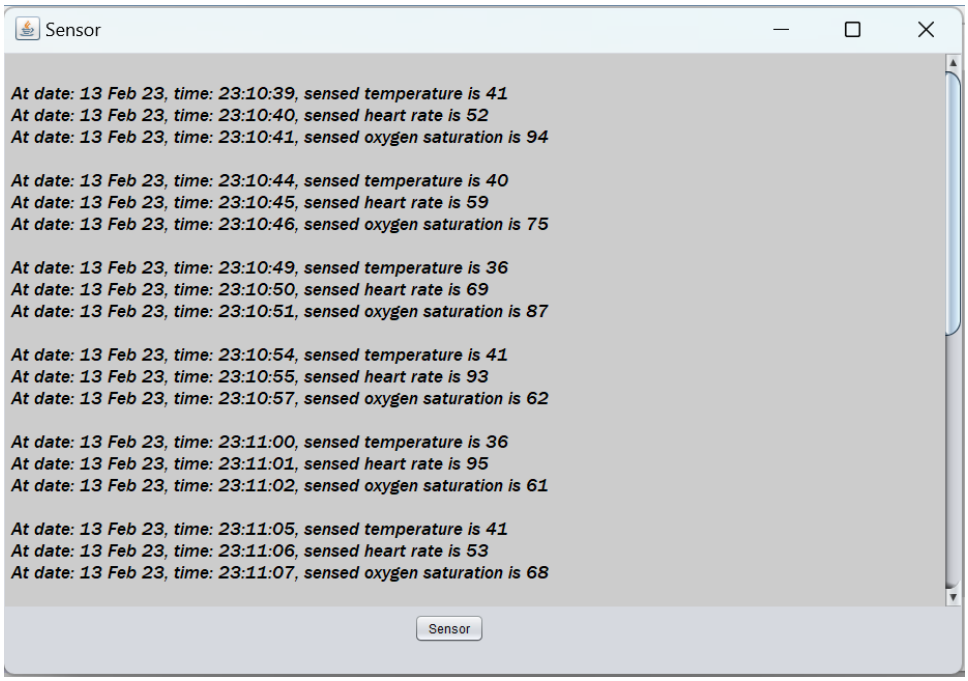
```
                //split the string to take the temperature value
                String[] tem=Temp.split("[ ]");
                //the last part of splitting ( it is a temperature value )
                String temp= tem[tem.length-1];
                //convert the string to the double
                double temperature= Double.valueOf(temp);


              //split the string to take the heart rate value
                String[] Heatrate=Heart.split("[ ]");
                //the last part of splitting ( it is a heart rate value )
                String heart= Heatrate[Heatrate.length-1];
                //convert the string to the int
                int heart_rate= Integer.valueOf(heart);

               //split the string to take the oxygen saturations value
                String[] oxygen=Oxg.split("[ ]");
                //the last part of splitting ( it is an oxygen saturations value )
                String oxygen1= oxygen[oxygen.length-1];
                //convert the string to the int
                int oxygen_saturations= Integer.valueOf(oxygen1);



               //comparison the values of temperature, heart rate and oxygen saturation to determine the output (what to print?)

               //1- first case
               if( temperature>39 && heart_rate>100 && oxygen_saturations<95 ){
                   //text box -> use display output in GUI
                   textBox.append("\nACTION: Send an ambulance to the patient! \n\n");
                   }

               //2- second case
                else if ( (temperature>=38&&temperature<=38.9) && (heart_rate>=95&&heart_rate<=98) && oxygen_saturations<80 ) {
                    //text box -> use display output in GUI
                   textBox.append("\nACTION: Call the patient's family! \n\n");
                }

               //3- third case
               else{
                    //text box -> use display output in GUI
                   textBox.append("\nACTION: Warning, advise patient to make a checkup appointment! \n\n");
               }


            }//end loop
        }//end loop

    }//end main

}//end class
```

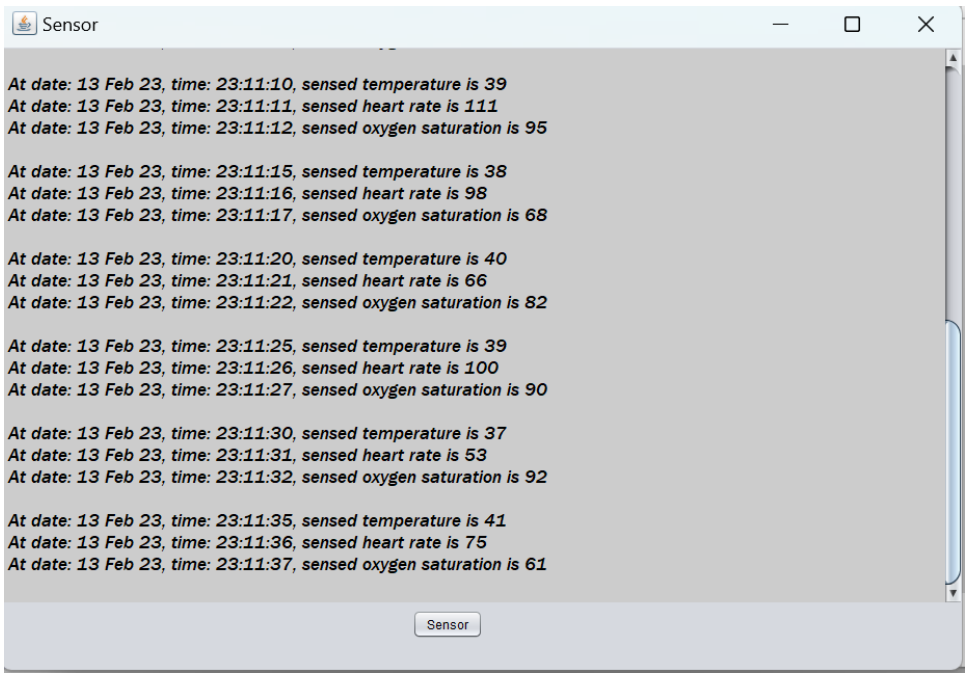
4. RHMS Application Run snapshots

4.1 One machine

4.1.1 Sensor class:

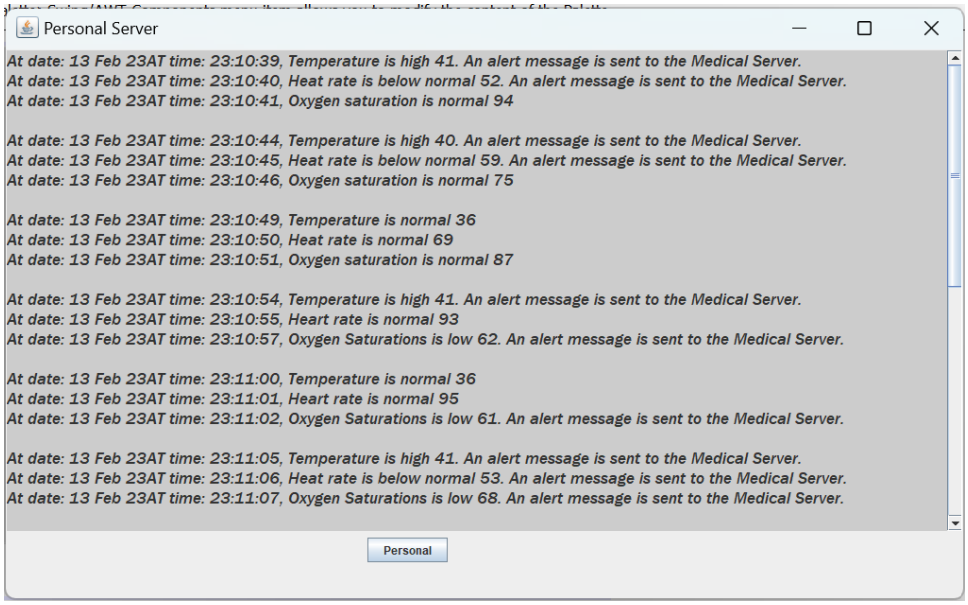
Generate different 12 value of temperature, heart rate and oxygen and send it to class personal.

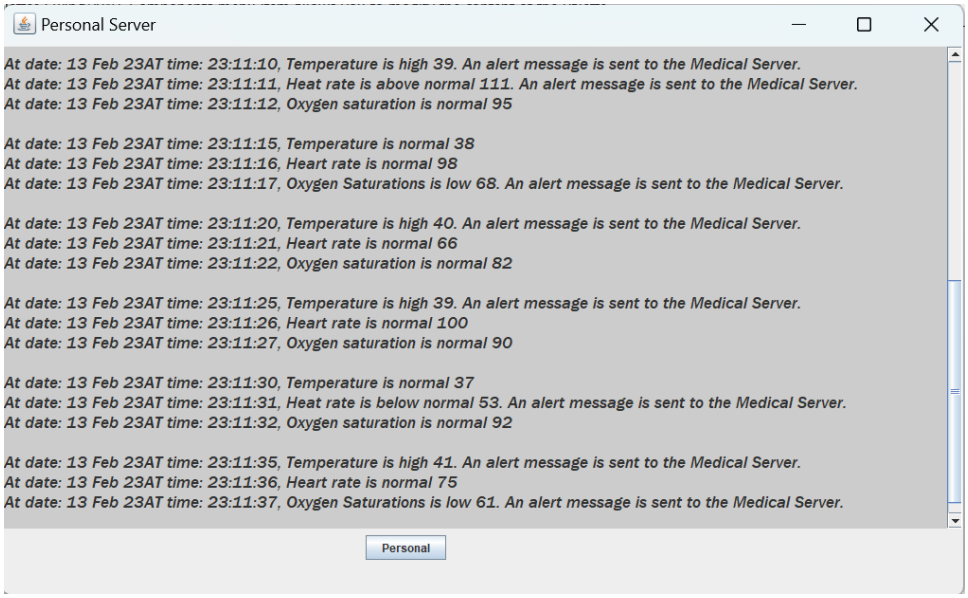




4.1.2 Personal class:

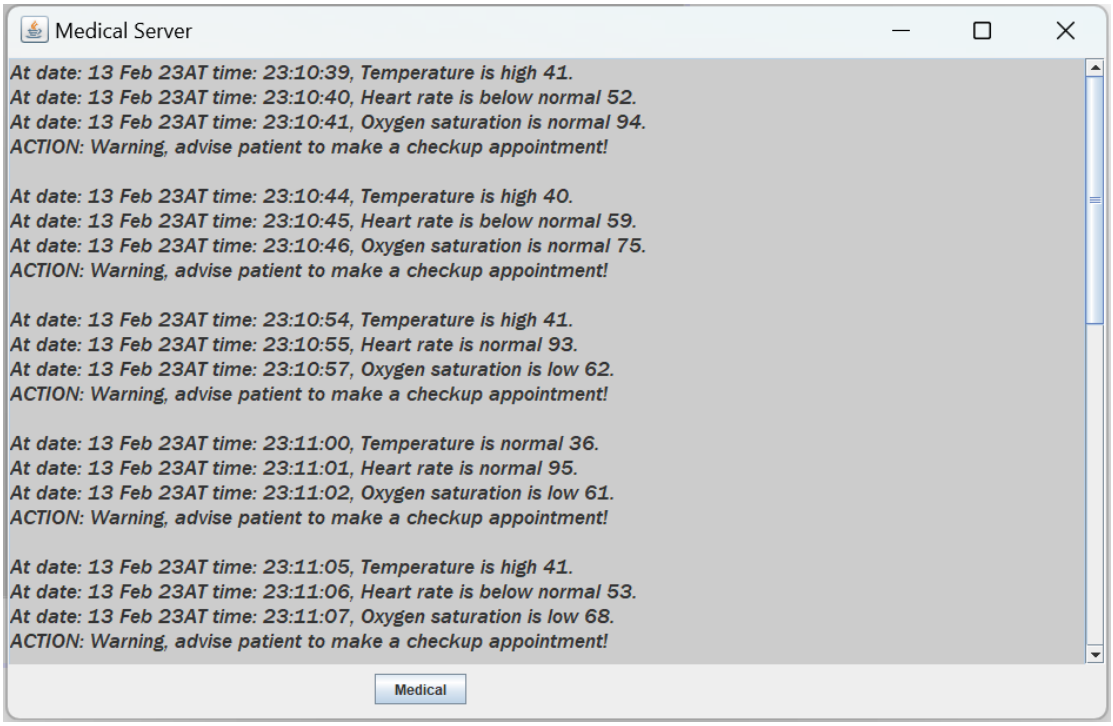
Receive different 12 value of temperature, heart rate and oxygen and send it to class medical.

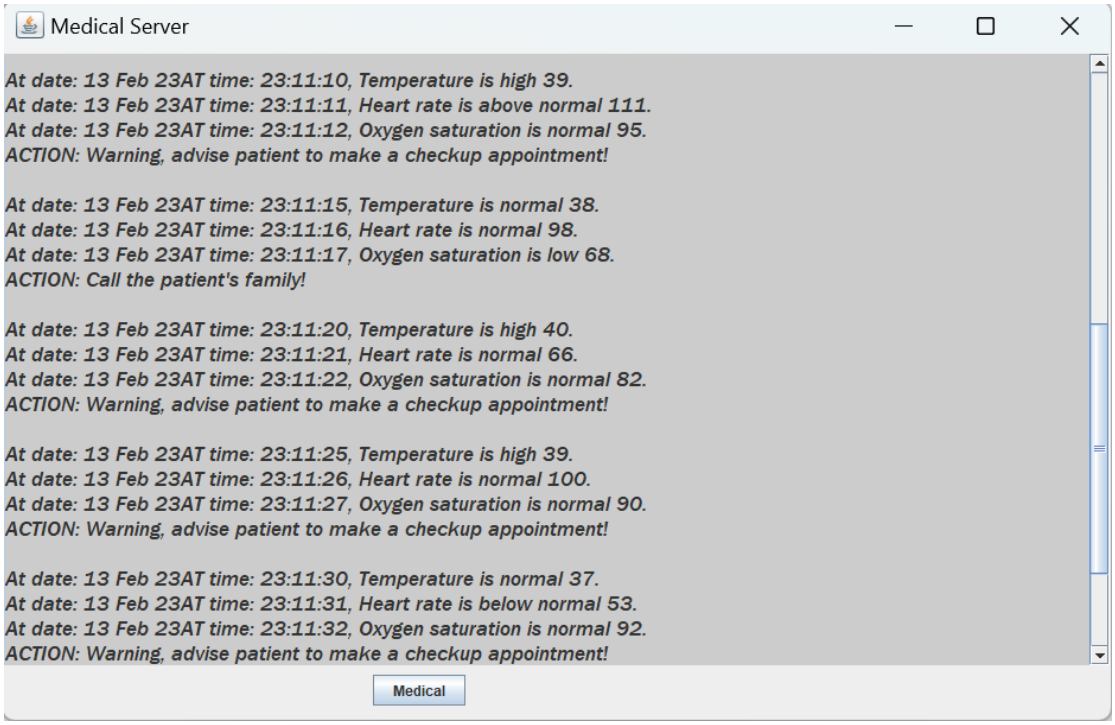


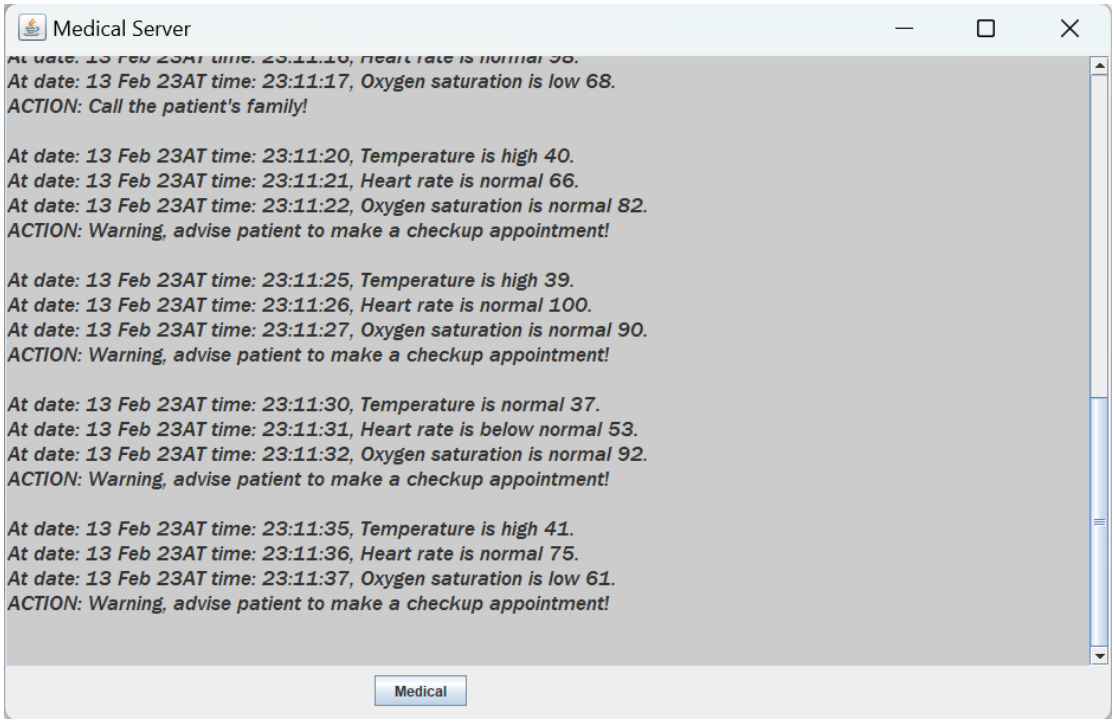


4.1.3 Medical class:

Receive different 12 value of temperature, heart rate and oxygen with messages and send it to class medical.

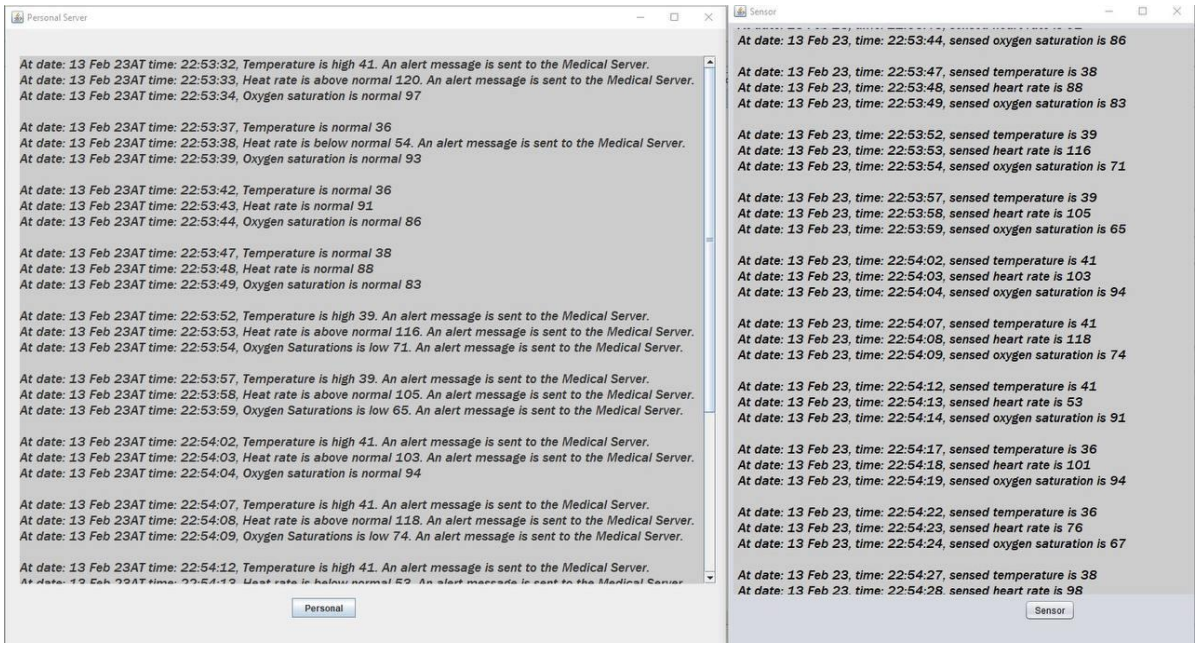


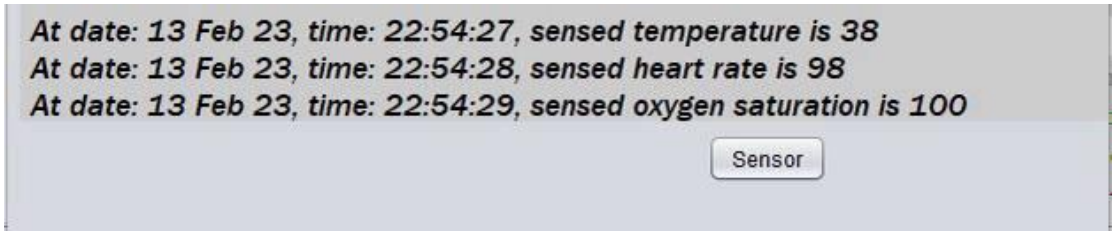


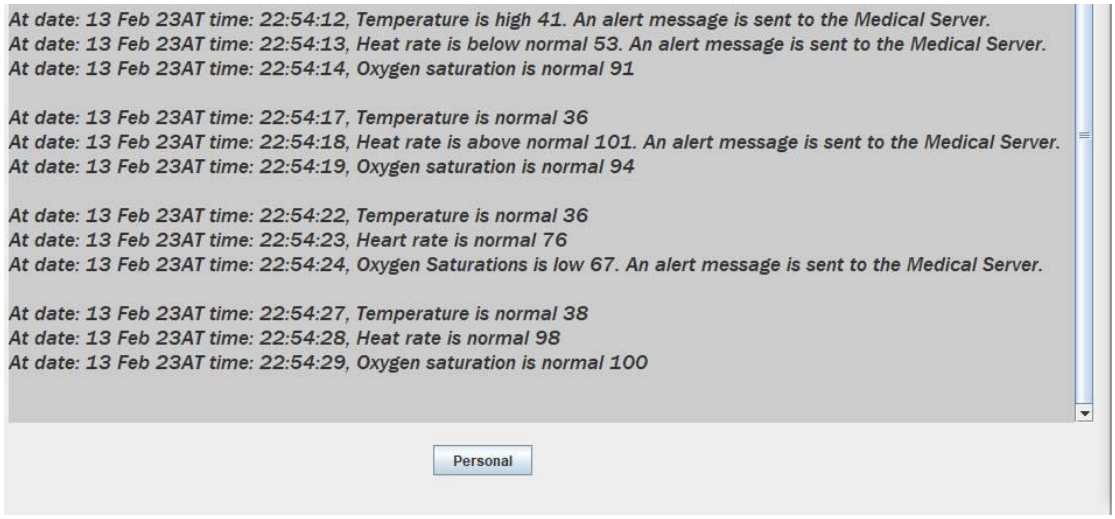


4.2 Two machines

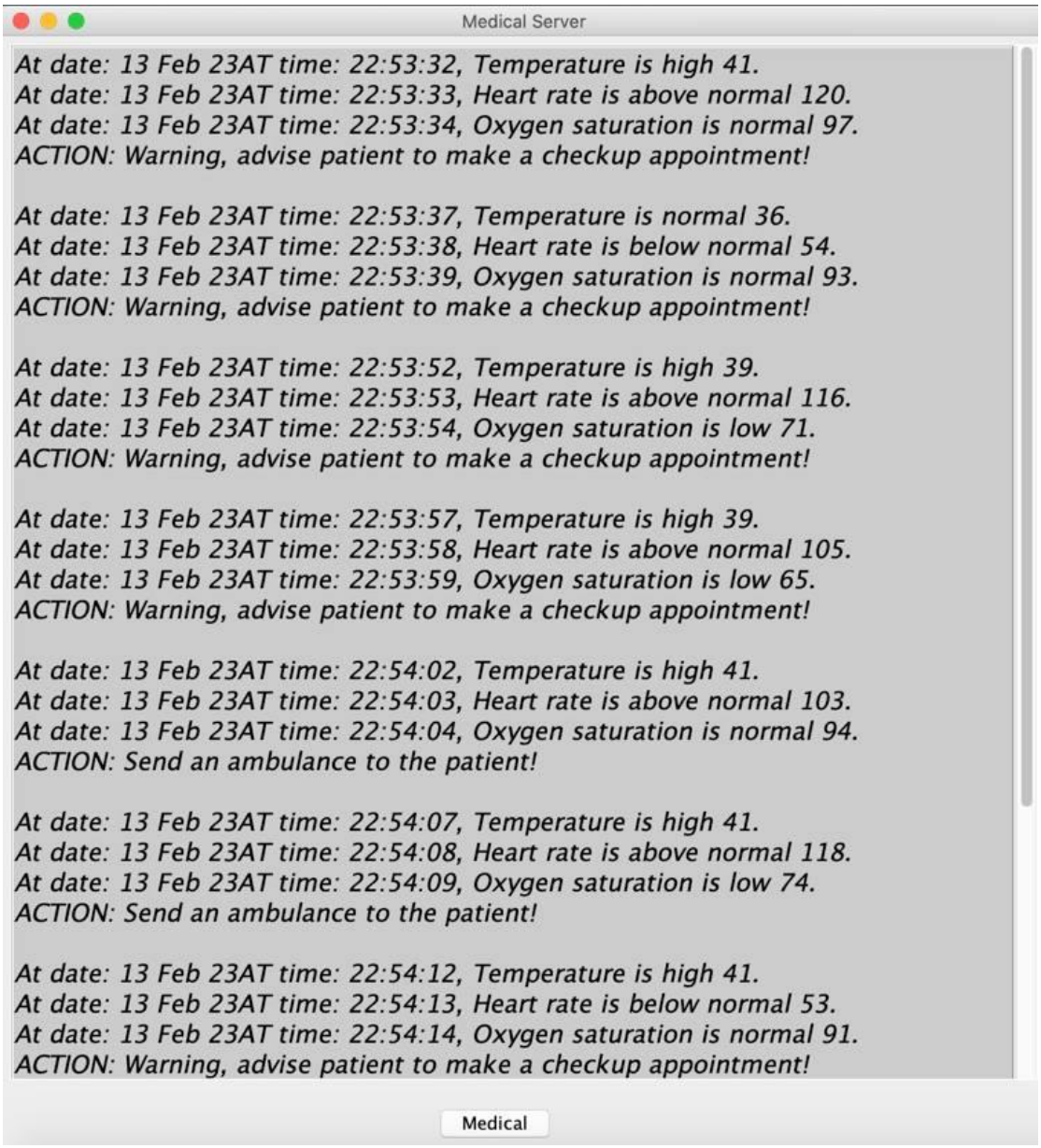
4.2.1 Sensor and personal class:

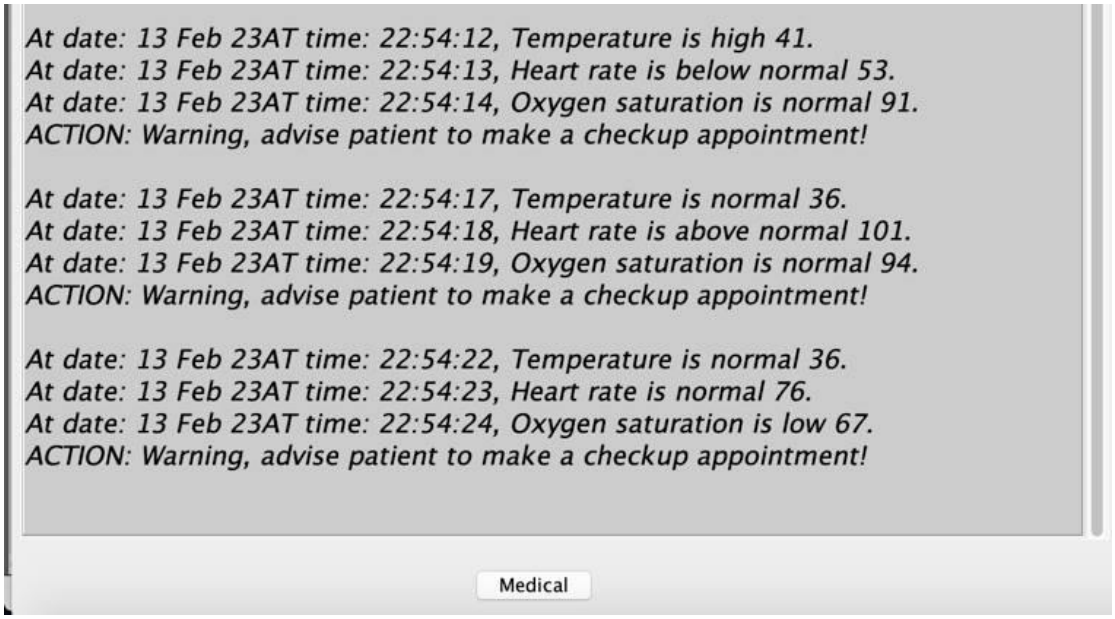



At date: 13 Feb 23, time: 22:54:27, sensed temperature is 38
At date: 13 Feb 23, time: 22:54:28, sensed heart rate is 98
At date: 13 Feb 23, time: 22:54:29, sensed oxygen saturation is 100
Sensor


At date: 13 Feb 23AT time: 22:54:12, Temperature is high 41. An alert message is sent to the Medical Server.
At date: 13 Feb 23AT time: 22:54:13, Heat rate is below normal 53. An alert message is sent to the Medical Server.
At date: 13 Feb 23AT time: 22:54:14, Oxygen saturation is normal 91
At date: 13 Feb 23AT time: 22:54:17, Temperature is normal 36
At date: 13 Feb 23AT time: 22:54:18, Heat rate is above normal 101. An alert message is sent to the Medical Server.
At date: 13 Feb 23AT time: 22:54:19, Oxygen saturation is normal 94
At date: 13 Feb 23AT time: 22:54:22, Temperature is normal 36
At date: 13 Feb 23AT time: 22:54:23, Heart rate is normal 76
At date: 13 Feb 23AT time: 22:54:24, Oxygen Saturations is low 67. An alert message is sent to the Medical Server.
At date: 13 Feb 23AT time: 22:54:27, Temperature is normal 38
At date: 13 Feb 23AT time: 22:54:28, Heat rate is normal 98
At date: 13 Feb 23AT time: 22:54:29, Oxygen saturation is normal 100
Personal

4.2.2 Medical class:




At date: 13 Feb 23AT time: 22:54:12, Temperature is high 41.
At date: 13 Feb 23AT time: 22:54:13, Heart rate is below normal 53.
At date: 13 Feb 23AT time: 22:54:14, Oxygen saturation is normal 91.
ACTION: Warning, advise patient to make a checkup appointment!
At date: 13 Feb 23AT time: 22:54:17, Temperature is normal 36.
At date: 13 Feb 23AT time: 22:54:18, Heart rate is above normal 101.
At date: 13 Feb 23AT time: 22:54:19, Oxygen saturation is normal 94.
ACTION: Warning, advise patient to make a checkup appointment!
At date: 13 Feb 23AT time: 22:54:22, Temperature is normal 36.
At date: 13 Feb 23AT time: 22:54:23, Heart rate is normal 76.
At date: 13 Feb 23AT time: 22:54:24, Oxygen saturation is low 67.
ACTION: Warning, advise patient to make a checkup appointment!
Medical

5. Teamwork and Lessons Learned

5.1. work Division

Student Name	Individual Tasks	Shared Tasks
Abrar Ramadan	Coding Sensor class, write report	Coding the communication between the three classes.
Rahaf Albrakati	Coding Sensor class, write report	
Roaa Alzahrani	Coding Medical class, write report	
Alaa Emad	Coding Personal class, GUI Implementation	
Shaimaa Abdullah	Coding Medical class, GUI Implementation	
Hanin Suleiman	Coding Personal class, GUI Implementation	

5.2. Planning and Coordination

First, we had a meeting to discuss the project and understand it well. After that, we started working on the code. First, we established communication between the classes, and then we divided the work so that every two students work on one class of the code. Then we held a meeting to discuss ideas for the classes and solve the problems, then we divided the rest of the tasks so that 3 students work on the report and 3 other students work on the GUI. However, we all worked and helped each other to finish the project.

5.3. difficulties

The difficulties we faced are:

1- The Personal class which is play 2 roles client and server at the same time.
2- GUI designing and connecting it with source code.

We overcome these difficulties by researching and watching multiple tutorials on YouTube.

5.4. Learning outcomes

- Understand the basics of client-server Internet applications and how to code them.
- Understand the basics of Transmission Control Protocol (TCP) and how to code it.
- Designing GUI and linking it to the source code.

6. Conclusion

To conclude, in this project we implemented a client-server application by using different java concepts like sockets and different GUI elements. we provided the interaction diagram and take snapshots of code implementation and the output to demonstrate the client-server interaction.

7. References

1- *Chapter 2*. Developing Client/Server Applications. (n.d.). Retrieved February 13, 2023, from https://www.microfocus.com/documentation/net-express/nx31books/sgdevt.htm

2- *What is a socket?* What Is a Socket? (The Java™ Tutorials > Custom Networking > All About Sockets). (n.d.). Retrieved February 13, 2023, from https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

3- YouTube. (2017, December 13). *الواجهة الرسومية بالجافا - درس 1: إنشاء أول برنامج ذو واجهة رسومية Gui*. YouTube. Retrieved February 13, 2023, from https://www.youtube.com/watch?v=-Rh8lVnouDA