King Abdulaziz University

DEPARTMENT OF COMPUTER SCIENCE

Faculty of Computing and Information Technology

CPCS 371, Winter Semester 2022

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

Member	ID	Section
Areej Suleman	1916786	B2
Ebtihaj Alnaqeeb	2011859	B1
Mai Khalil	2010295	B2
Sarah Abukhammas	2006235	B1
Shahad Bin Kulaib	2005339	B2
Raghad Alghamdi	2006357	B1

Group number: 5

Instructors: Dr. Etimad Fadel and Dr. Ohoud Alzamzami

Table Of Contents

List of figures:

List of tables:

1. Introduction

1.1 Client-server application

A client-server application is a popular **software design architecture** that breaks software down abstractly into client and server components.

- A client-side application (or simply client) runs on the end-user computer; it provides the user-interface (UI) that handles how the application feels, looks like, and interacts with end-user. The application may consume temporary and local storage on the user's computer (computer device).
- In server-side applications, requests are received from clients, and logic is used to return appropriate data to the clients. Application programming interfaces (APIs) are usually used instead of user interfaces. Moreover, the server often includes a database, which stores all application data permanently.

1.2 JAVA TCP sockets

A TCP socket is a reliable way to communicate between two processes in different hosts. TCP socket is a connection-oriented protocol, that means the connection is required to be established before the data begin to be exchanged between the two end-points which are the server and client.

First of all, we must create the Server socket that will wait for the client's request.

// Creating the server socket to communicate with the client & the port No. should be the same.

```
server_socket = new ServerSocket(2014);
```

Then the establishment will be done in several steps starting with the client:

1- Then the client creates the TCP socket that contains the address of the server and the port number of the socket and sends it to the server socket that waits for any request.

```
Socket   client_socket = new Socket("localhost", 2014);
```

2- The server receives the requested message from the client, and accepts the TCP connection using the accept() method that will create a new socket for this client's communication.

```
socket = server_socket.accept();
```

3- After exchanging data, the client will close its socket and the server will close the socket that the accept method creates. However, the server socket will always be on.

```
client_socket.close();  the client socket

socket.close();   the client's server socket
```

1.3 Selected GUI elements

Java includes libraries and classes to provide support for Graphical User Interface objects, the one we have used in our program is called JFrame. JFrame is a class of the *javax.swing* package that is extended by *java.awt.frame*. JFrame class has various methods which can be used to design the GUI interface. Here are some of the GUI elements available in java that we have used in our code when creating our RHMS application:

- Labels
- Images
- Icons
- Panels
- Text field

1.4 RHMS application

Remote Health Monitoring System (RHMS) is a client-server-based application that helps elderly patients who suffer from chronic diseases. It uses the Wireless Body Area Networks (WBAN) which helps monitoring the patients remotely. RHMS application consists of three major components:

- The sensor client application that has three sensors to read the values of oxygen saturation, heart rate and temperature of the patient and send it to personal server.
- The personal server that acts as a server and a client. Its server side receives the sensed data from the sensor application and checks the data to decide if it should be sent to the medical server. On the other hand, the client side is responsible for sending messages to the medical server if the sensed data are not normal.
- The medical server receives messages from the personal server, displays them to the caregiver in addition to checking the data to deciding what actions need to be taken.

1.5 Role of clients and server

The client's role is to generate specific data representing the heart rate, oxygen, and temperature of the patient and send these data to the personal server through TCP sockets.

The personal server plays two roles. It acts as a server with the sensor client to accept the sensor client's connection, receive the sensed data and display these data on the application interface, and check if it applies to the different cases to decide if the data need to be sent to the medical server or not. The second role is to act as a client when the sensed data need to be sent to the medical server to send the appropriate messages to the medical server.

The Medical Server's role is to accept the personal client's connection and receive the data sent by the personal server and display it with the appropriate action needed.

1.6 Report's overview

In section 2 of this report, we will demonstrate the client- server interaction through a diagram and write a pseudocode to demonstrate the logic behind our RHMS (Remote Health Monitoring System) application. In section 3 we will list the actual code of our application and in section 4 we will show the snapshots of the actual run of our application. Moving on to section 5 where we will list how we distributed the work amongst all team members as well as how we coordinate the work, some of the difficulties we encountered as well as the lessons we have learned. Finally, in section 6 we will end our report with a conclusion that summarizes the whole journey of building the RHMS application.

2 Patient Monitoring Application interaction diagram

2.1 Client-server interaction diagram

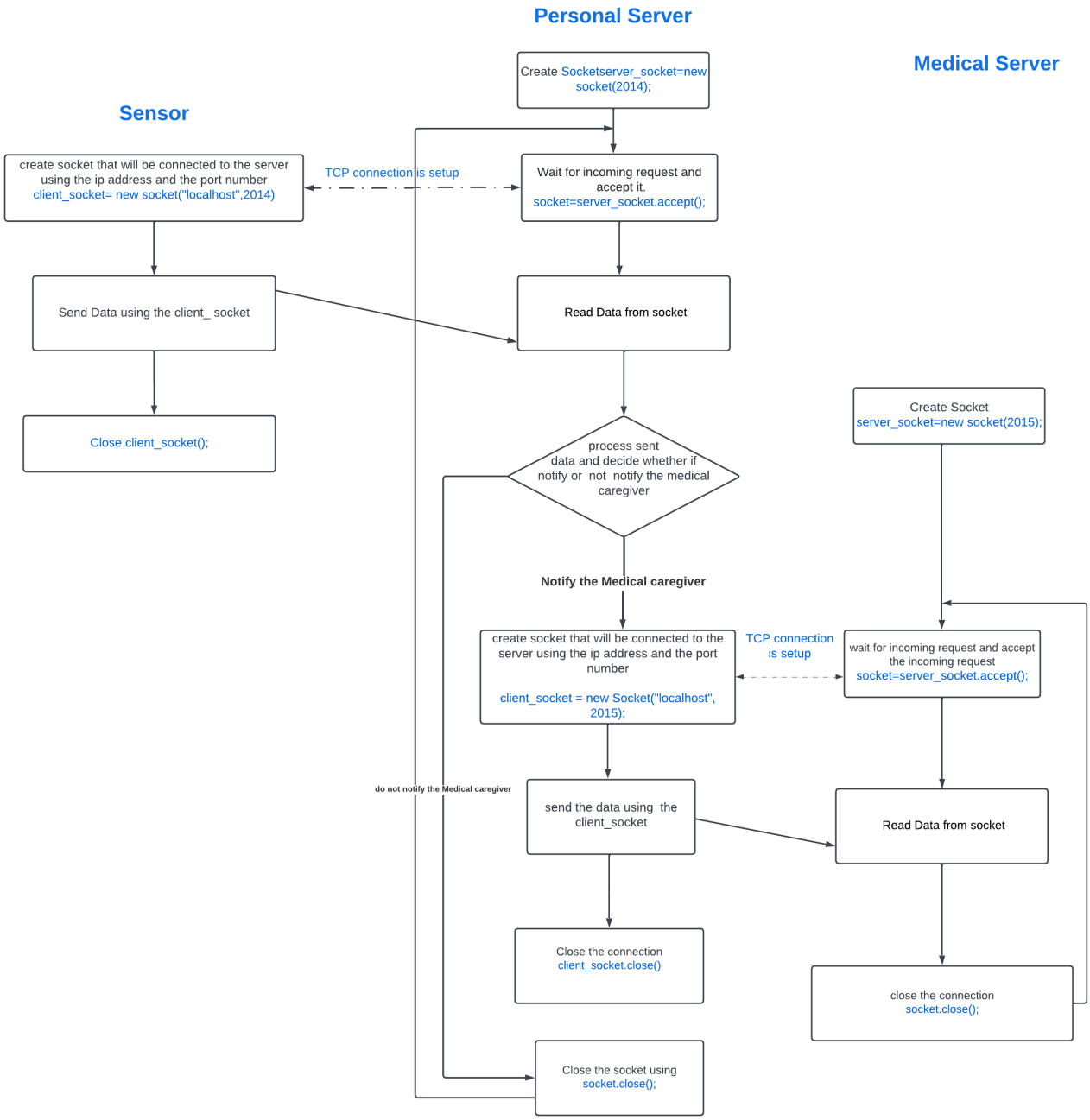


Figure 1:Client-server interaction diagram

Our program consists of a client sensor application that interacts with the personal server using a socket to send information to it. The personal server will accept the sent data via a server socket which establishes the TCP connection between the two. The personal server will then process the data and decides whether or not to send the processed data to the medical server via a socket, which will then initiate the role of the client side of the personal server, as seen in figure 1. After the personal server has decided to notify the medical server , the data is sent to the medical server via a socket , and the medical server will then accept the data coming from the personal server using its server socket, establishing a TCP connection between the two, all shown in the figure above.

2.2 Pseudocode

Sensor Client Application

```
CLASS Sensor_Client_Application
   main function
      PRINT "Enter the duration of the connection time in seconds: (duration must be greater than or equal 60s) : "
      TAKE the duration from the user

      WHILE duration is less than 60
         PRINT "duration must be greater than or equal 60s, try again:  "
         TAKE the duration from the user
      END WHILE

      CONVERT duration from seconds to milliseconds by multiplying it by 1000
      CREATE client_socket
      SET host name to "localhost"
      SET port number to 2014

      CREATE outputStreamWrite
      CREATE bufferedWriter

      WRITE duration to server using bufferedWriter

      SET startTime to currentTimeMillis

      CREATE dateFormat
      SET dateFormat to "yyyy-MM-dd 'time' HH:mm:ss"

      WHILE true
         IF currentTimeMillis is greater than startTime plus duration
            BREAK
         END IF

         SEND 36 and 41 to the getRandomValue function and SET the returned value to temprature
         PRINT the temprature with currentTimeMillis and date
         WRITE temprature to server using bufferedWriter
         FLUSH the bufferedWriter
         WAIT for 1 seconds

         IF currentTimeMillis is greater than startTime plus duration
            BREAK
         END IF

         SEND 50 and 120 to the getRandomValue function and SET the returned value to heartRate
         PRINT the heartRate with currentTimeMillis and date
         WRITE heartRate to server using bufferedWriter
         FLUSH the bufferedWriter
         WAIT for 1 seconds

         IF currentTimeMillis is greater than startTime plus duration
            BREAK
         END IF

         SEND 60 and 100 to the getRandomValue function and SET the returned value to oxygen
         PRINT the oxygen with currentTimeMillis and date
         WRITE oxygen to server using bufferedWriter
         FLUSH the bufferedWriter
         WAIT for 5 seconds

         IF currentTimeMillis is greater than startTime plus duration
            BREAK
         END IF
      END WHILE

      CLOSE client_socket
      CLOSE outputStreamWriter
      CLOSE bufferedWriter

   END main function

   function getRandomValue (Arg1,Arg2)
      GENERATE randomValue with range from Arg1 to Arg2
      RETURN randomValue
   END getRandomValue function

END Sensor_Client_Application CLASS
```

Personal Server

```
CLASS Personal_Server
   main function

      INITIALIZE socket and SET it to null
      INITIALIZE client_socket and SET it to null
      INITIALIZE inputStreamReader and SET it to null
      INITIALIZE outputStreamWriter and SET it to null
      INITIALIZE bufferedReader and SET it to null
      INITIALIZE bufferedWriter and SET it to null
      INITIALIZE server_socket and SET it to null

      CREATE server_socket
      SET port number to 2014

      ACCEPT the connction by server_socket and ASSIGN it to socket

      SET startTime to currentTimeMillis
      PRINT "Server is ON and wait for data..."

      DECLARE oxygen
      DECLARE temprature
      DECLARE heartRate
      SET timeout TO 0

      WHILE TRUE
         PRINT "New client with new connection is established.\nwaiting for data..."
         CREATE inputStreamReader
         CREATE outputStreamWriter
         CREATE bufferedReader
         CREATE bufferedWriter
         READ duration from client using bufferedReader
         CREATE dateFormat
         SET dateFormat to "yyyy-MM-dd 'time' HH:mm:ss"
         TRY
            WHILE timeout is equal to zero
               READ temprature from client using bufferedReader

               IF temprature is equal to -1
                  BREAK
               END IF
```

```
               IF temprature is less than 38
                  PRINT the currentTimeMillis and Temperature and " Temprature is normal"
               END IF

               ELSE
                  PRINT the currentTimeMillis and Temperature and " Temperature is high. An alert message is sent to the Medical Server."
               END ELSE

               WAIT for 1 second

               READ heartRate from client using bufferedReader

               IF heartRate is equal to -1
                  BREAK
               END IF

               IF heartRate is less than or equal 100 and greater than or equal 60
                  PRINT the currentTimeMillis and heartRate and " heartRate is normal"
               END IF

               ELSE IF heartRate is less than 60
                  PRINT the currentTimeMillis and heartRate and " Heart rate is below normal. An alert message is sent to the Medical Server"
               END ELSE

               ELSE
                  PRINT the currentTimeMillis and heartRate and" Heart rate is above normal. An alert message is sent to the Medical Server"
               END ELSE
               WAIT for 1 second

               READ oxygen from client using bufferedReader

               IF oxygen is equal to -1
                  BREAK
               END IF

               IF oxygen is greater than 75
                  PRINT the currentTimeMillis and oxygen and " Oxygen saturation is normal"
               END IF
```

```
            ELSE
               PRINT the currentTimeMillis and oxygen and " Oxygen Saturations is low. An alert message is sent to the Medical Server. "
            END ELSE

            IF temprature is greater than 38 or heartRate is greater than 100 or heartRate is less than 60 or oxygen is less than 75
               CREATE client_socket
               SET host name to "localhost"
               SET port number to 2023

               CREATE outputStreamWriter1
               CREATE bufferedWriter1

               WRITE temprature to medical server using bufferedWriter1
               FLUSH the bufferedWriter1

               WRITE heartRateto medical server using bufferedWriter1
               FLUSH the bufferedWriter1

               WRITE oxygen to medical server using bufferedWriter1
               FLUSH the bufferedWriter1

               CLOSE client_socket
            END IF
            WAIT for 5 seconds
         END WHILE
      END TRY
      CATCH ioException
         CLOSE socket
         CLOSE inputStreamReader
         CLOSE outputStreamWriter
         CLOSE bufferedReader
         CLOSE bufferedWriter
      END CATCH
      PRINT "Waiting for another connection..."
   END WHILE
  END main function
END Personal_Server CLASS
```

Medical Server

```
CLASS Medical_Server
   main function
      INITIALIZE socket and SET it to null
      INITIALIZE inputStreamReader and SET it to null
      INITIALIZE bufferedReader and SET it to null
      INITIALIZE server_socket and SET it to null
      CREATE server_socket
      SET port number to 2023
      PRINT "Server is ON and wait for data..."
      DECLARE temprature
      DECLARE heartRate
      DECLARE oxygen
      TRY
         WHILE TRUE
            ACCEPT the connction by server_socket and ASSIGN it to socket
            CREATE inputStreamReader
            CREATE bufferedReader

            READ temprature from personal server using bufferedReader
            READ heartRate from personal server using bufferedReader
            READ oxygen from personal server using bufferedReader

            CREATE dateFormat
            SET dateFormat to "yyyy-MM-dd 'time' HH:mm:ss"

            SET timeout TO 0
            WHILE timeout is equal to zero
               IF temprature is greater than 39 and heartRate is greater than 100 and oxygen is less than 95
                  PRINT the currentTimeMillis and "Temprature is high"
                  PRINT the currentTimeMillis and "Heart Rate is above normal"
                  PRINT the currentTimeMillis and "oxygen saturation is low"
                  PRINT "Send an ambulance to the patient!"
               END IF

               ELSE IF temprature is greater than  or equal 38 and temprature is less than or equal 38.9 and heartRate is greater
than or equal 95 and heartRate is less than or equal 98 and oxygen is less than 80
                  PRINT the currentTimeMillis and "Temprature is normal" and temprature
                  PRINT the currentTimeMillis and "Heart Rate is normal" and heartRate
                  PRINT the currentTimeMillis and "oxygen saturation is normal" and oxygen
                  PRINT "Call the patient's family!"
               END ELSE IF
```

```
            ElSE
               IF temprature is greater than  or equal 39
                   PRINT the currentTimeMillis and "Temprature is high" and temprature
               END IF

               ELSE IF temprature is less than 37
                  PRINT the currentTimeMillis and "Temprature is low" and temprature
               END ELSE IF

               ELSE
                  PRINT the currentTimeMillis and "Temprature is normal" and temprature
               END ELSE


               IF heartRate is less than 60
                  PRINT the currentTimeMillis and "Heart Rate is  below normal" and heartRate
               END IF

               ELSE IF heartRate is greater than 100
                  PRINT the currentTimeMillis and "Heart Rate is above normal" and heartRate
               END ELSE IF

               ELSE
                  PRINT the currentTimeMillis and "Heart Rate is normal" and heartRate
               END ELSE


               IF oxygen is less than 95
                  PRINT the currentTimeMillis and "oxygen saturation is low" and "oxygen
               END IF

               ELSE IF oxygen is greater than 100
                  PRINT the currentTimeMillis and "oxygen saturation is high" and "oxygen
               END ELSE IF

               ELSE
                  PRINT the currentTimeMillis and "oxygen saturation is normal" and "oxygen
               END ELSE

               PRINT "Warning, advise patient to make a checkup appointment!"
            END ELSE
```

```
               SET timeout to -1
            END WHILE
         END WHILE
      END TRY

      CATCH ioException
         CLOSE socket
         CLOSE inputStreamReader
         CLOSE bufferedReader
      END CATCH
   END main function
END Medical_Server CLASS
```

3 RHMS implementation

3.1 Code

There are three classes in our code: Sensor Client Application, Personal Server, and Medical Server. **The Sensor Client Application**: collects sensed data and sends it to the Personal Server via TCP sockets. **The Personal Server**: has two roles. It acts as a server for the **Sensor Client Application** that processes the received sensor information and decides whether or not the information has to be sent via a client socket to the **Medical Server**. The medical server receives the sensed data from the client side of the Personal Server. Patient status messages are received from the **Personal Server** and displayed to the caregiver by the **Medical Server after**. Also, it processes the received information and displays the appropriate action to be taken by the caregiver.

The Sensor Client Application Class

```
19  public class Sensor extends javax.swing.JFrame {
20      /**
21       * Creates new form Sensor
22       */
23      public Sensor() {
24          initComponents();
25      }
26      //Create a duration variable for the specified duration of the connection.
27      static int duration;
28      @SuppressWarnings("unchecked")
29
30      //Needed variables for Image icon in the Gui interface screen.
        static String imageName = null;
32      static ImageIcon Temprature = new ImageIcon( filename: "Temperature.png");
33      static ImageIcon HeartRate = new ImageIcon( filename: "HeartRate.png");
34      static ImageIcon OxygenSaturation = new ImageIcon( filename: "OxygenSaturation.png");
```

```
35
36      Generated Code
116
        private void TextOxygenActionPerformed(java.awt.event.ActionEvent evt) {
118
119     }
120
        private void TextHeartActionPerformed(java.awt.event.ActionEvent evt) {
122
123     }
124
        private void Duration1ActionPerformed(java.awt.event.ActionEvent evt) {
126
127     }
128
        private void TextTemprature1ActionPerformed(java.awt.event.ActionEvent evt) {
130
131     }
132
```

```
133     public static void main(String args[]) throws IOException, InterruptedException {
134
135         //Creating a scanner object.
136          Scanner input = new Scanner( source: System.in);
137         //Printing a message to ask the user to enter the duration of the connection.
138         System.out.print( s: "Enter the duration of the connection time in seconds: (duration must be greater than or equal 60s) : ");
139         System.out.println();
140         auration = input.nextInt();
141         //Integer.parseInt(Duration1.getText());
142         //Loop for asking the user to enter the duration again if it is smaller than 60 seconds.
143          while(auration < 60){
144             //Printing a message.
145             System.out.println( x: "duration must be greater than or equal 60s, try again:  ");
146             System.out.println();
147             auration = input.nextInt();
148         }//The end of loop.
149
150          //Setting the panel
151          new Sensor().setVisible( b: true);
152
153         //Display the input duration in the GUI.
154         Duration1.setText(" "+auration);
155
156         //Converting duration from seconds to milliseconds.
157         auration = auration * 1000;
```

```
159             //Creating a socket to connect to the server using the same port number.
160             //these two lines of code will be used when running this program on different machines
161
162             /*InetAddress addresses = InetAddress.getByName("192.186.100.10");
163              String hostName = addresses.getHostName();*/
164             Socket client_socket = new Socket( host: "localhost",  port: 2014);
165
166             //Creating the output stream writer to write the messages.
167             OutputStreamWriter outputStreamWriter = new OutputStreamWriter( out: client_socket.getOutputStream());
168
169             //Creatng the buffered writer and associating it with the output stream writer.
170             BufferedWriter bufferedWriter = new BufferedWriter( out: outputStreamWriter);
171
172             //Sending the duration of the connection to the server.
173             bufferedWriter.write((int) auration);
174
175             //Initializing the starting time.
176             long startTime = System.currentTimeMillis();
177
178             //Creating a date and identifying the format of the date.
179             SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd 'time' HH:mm:ss");
```

```
        //Infinie while loop for the connection between the client and the server.
        while(true){

            //Checking if the time is out(sepecified duration of the connection is done)to stop the connection.
            if(System.currentTimeMillis() > startTime + duration) {
                break;
            }
            //-----------------------------------------------------------------
            //Generating a random value for the temprature between 36 and 41.
            int temprature = getRandomValue( min: 36,  max: 41);
            //Display the icon of Temperature.
            imageName = "Temprature";
            imageOfTemperature.setIcon( icon: Emprature);
            //Printing the temprature.
            TextTemprature1.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " sensed temperature is " + temprature+"\n");
            //Sending the value of the temprature to the server.
            bufferedWriter.write( c: temprature);
            //Flushing the buffered writer.
            bufferedWriter.flush();

            //To wait for 1 second before sending the value of the temprature, we will use a thread.
            Thread.sleep( millis: 1000);

            //Checking if the time is out to stop the connection.
            if(System.currentTimeMillis() > startTime + duration) {
                break;
            }
```

```
            //-----------------------------------------------------------------
            //Generating a random value for the heart rate between 50 and 120.
            int heartRate = getRandomValue( min: 50,  max: 120);
            //Display the icon of Heart rate.
            imageName = "HeartRate";
            imageOfHeartRate.setIcon( icon:HeartRate);
            //Printing the heart rate.
            TextHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " sensed heart rate is  " + heartRate+"\n");
            //Sending the value of the heart rate to the server.
            bufferedWriter.write( c: heartRate);
            //Flushing the buffered writer.
            bufferedWriter.flush();

            //To wait for 1 second before sending the value of the heartrate, we will use a thread.
            Thread.sleep( millis: 1000);

            //Cheching if the time is out(sepecified duration of the connection is done)to stop the connection.
            if(System.currentTimeMillis() > startTime + duration) {
                break;
            }
            //-----------------------------------------------------------------
            //Generating a random value for the oxygen saturation between 60 and 100.
            int oxygen = getRandomValue( min: 60,  max: 100);
            //Display the icon of Oxygen saturation.
            imageName = "OxygenSaturation_1";
            imageOfOxygenSaturation.setIcon( icon:OxygenSaturation);
            //Printing the oxygen.
            TextOxygen.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " sensed oxygen saturation is " + oxygen+"\n");
```

```
236             //Sending the value of the oxygen saturation to the server.
237             bufferedWriter.write( c: oxygen);
238             //Flushing the buffered writer.
239             bufferedWriter.flush();
240
241             //To wait for 5 seconds before sending the next block of data, we will use a thread..
242             Thread.sleep( millis: 5000);
243
244             //Checking if the time is out(sepecified duration of the connection is done)to stop the connection.
245             if(System.currentTimeMillis() > startTime + auration) {
246                 break;
247             }
248             //-----------------------------------------------------------------
249             //To distinguish between each block of sensed data, print a blank line.
250             System.out.println( x: "");
251             //To reset all the texts in the GUI screen.
252             textTemprature1.setText( t: null);
253             textHeart.setText( t: null);
254             textOxygen.setText( t: null);
255
256         } //The end of while loop.
```

```
258         //Closing the socket and all of the in/output streams and buffers.
259         if (client_socket != null) {
260             client_socket.close();
261         }
262         if (outputStreamWriter != null) {
263             outputStreamWriter.close();
264         }
265         if (bufferedWriter != null) {
266             bufferedWriter.close();
267         }
268     }//The end of the main method.
269     //-----------------------------------------------------------------------------
270     //Header of getRandomValue method that generates a random value within the given range.
271     public static int getRandomValue(int min, int max) throws InterruptedException {
272         // Generating a random value and returning it.
273         int randomValue = (int) (min + Math.random() * ((max - min) + 1));
274         return randomValue;
275     }//The end of the getRandomValue method.
276 //-------------------------------------------------------------------------------------------
```

```
276     //-----------------------------------------------------------------------------------------------------
277         // Variables declaration - do not modify
278         static javax.swing.JLabel Background;
279         static javax.swing.JTextField Duration1;
280         static javax.swing.JTextField TextHeart;
281         static javax.swing.JTextField TextOxygen;
282         static javax.swing.JTextField Textimprature1;
283         static javax.swing.JLabel imageOfHeartRate;
284         static javax.swing.JLabel imageOfOxygenSaturation;
285         static javax.swing.JLabel imageOfTemperature;
286         static javax.swing.JLabel jLabel1;
287         private javax.swing.JLabel jLabel2;
288         static javax.swing.JPanel jPanel1;
289         // End of variables declaration
290     }// End of the class
```

The Personal Server

```
10  package Personal_Server;
11  //All needed imports go to here :
12  import java.awt.Image;
13  import java.io.*;
14  import java.net.*;
15  import java.text.SimpleDateFormat;
16  import javax.swing.ImageIcon;
17  //-----------------------------------------------------------------------------
18  public class Personal_GUI extends javax.swing.JFrame {
19
20      public Personal_GUI() {
21          initComponents();
22      }
23
24      /**
25       * This method is called from within the constructor to initialize the form.
26       * WARNING: Do NOT modify this code. The content of this method is always
27       * regenerated by the Form Editor.
28       */
29      @SuppressWarnings("unchecked")
30
31      static String imageName = null;
32      //Create a Temperature image.
33      static ImageIcon Temperature = new ImageIcon( filename: "Temperature.png");
34      //Create a Heart Rate image.
35      static ImageIcon HeartRate = new ImageIcon( filename: "HeartRate.png");
36      //Create an Oxygen stauration image.
37      static ImageIcon Oxygen = new ImageIcon( filename: "Oxygen.png");
38      Generated Code
```

```
129
    private void oxgTextActionPerformed(java.awt.event.ActionEvent evt) {
131
    }
132
133
    private void heartRTextActionPerformed(java.awt.event.ActionEvent evt) {
135
    }
136
137
    private void connectionActionPerformed(java.awt.event.ActionEvent evt) {
139
    }
140
141
    private void resultActionPerformed(java.awt.event.ActionEvent evt) {
143
    }
144
```

```
145
146 public static void main(String args[]) throws IOException, InterruptedException {
147
148         new Personal_GUI().setVisible( b: true);
149
150     //Initailizing a socket.
151     Socket socket = null;
152
153     //Socket to connect to the medical server.
154     Socket client_socket = null;
155
156     //Initailizing an input stream reader.
157     InputStreamReader inputStreamReader = null;
158
159     //Initailizing an output stream writer.
160     OutputStreamWriter outputStreamWriter = null;
161
162     //Initailizing a buffered reader.
163     BufferedReader bufferedReader = null;
164
165     //Initailizing a buffered writer.
166     BufferedWriter bufferedWriter = null;
167
168     //Initailizing a server socket that waits for requests to come over the network.
169     ServerSocket server_socket = null;
170
171     //Creating the server socket to communicate with the client, & the port No. should be the same.
172     //Listen on a certain port No. for connections.
173     server_socket = new ServerSocket( port: 2014);
174
```

```
175         //The setReuseAddress method allows the socket to be bound even though a previous connection is in a timeout state.
176         server_socket.setReuseAddress( on: true);
177
178         //The starting time.
            long startTime = System.currentTimeMillis();
180
181         //Printing a message.
182         connection.setText( t: "Server is ON and wait for data...");
183
184         //Declaring variabales to be read by socket.
185         int oxygen ;
186         int temprature;
187         int heartRate;
188
189         //Initailizing the timeout variable with 0 to start connection with 1 client.
190         int timeout = 0;
```

```
192         //While loop to create new socket and accept the connection with the client.
193         while (true) {
194
195             //Communication done ,Handshaking.
196             socket = server_socket.accept();
197
198             //Printing a message.
199             System.out.println( x: "New client with new connection is established.\nwaiting for data...");
200             System.out.println();
201
202             //Creating the input stream reader to read the messages.
203             inputStreamReader = new InputStreamReader( in: socket.getInputStream());
204
205             //Creating the output stream writer to write the messages.
206             outputStreamWriter = new OutputStreamWriter( out: socket.getOutputStream());
207
208             //Creatng the buffered reader and associating it with the input stream reader.
209             bufferedReader = new BufferedReader( in: inputStreamReader);
210
211             //Creatng the buffered writer and associating it with the output stream writer.
212             bufferedWriter = new BufferedWriter( out: outputStreamWriter);
213
214             //Reading the duration of the connection set by the client.
                int duration = bufferedReader.read();
216
217             //Creating a date and identifying the format of the date.
218             SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd 'time' HH:mm:ss");
```

```
220             try {
221
222                 //While loop to receive and read data sent by the client. It ends when the connection time is out.
223                 while (timeout == 0) {
224
225                     //Reading the value of the temprature sent by client.
226                     temprature = bufferedReader.read();
227
228                     //Checking if the buffer is empty (there is no data to read).
229                     if (temprature == -1) {
230                         break;
231                     }
232
233                     //Case where the Temprature is normal.
234                     if (temprature < 38) {
235                         tempText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Temprature is normal");
236                     }
237                     //Case where the Temprature is above the normal.
238                     else {
239                         tempText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Temperature is high " + temprature);
240                         imageName = "Temperature";
241                         image.setIcon( icon: Temperature);
242                         result.setText( t: " An alert message is sent to the Medical Server.");
243                     }
```

```
245                     //To wait for 1 second before reading the next int, we will use a thread.
246                     Thread.sleep( millis: 1000);
247                     //-----------------------------------------------------------------
248
249                     //Reading the value of the heart rate sent by client.
250                     heartRate = bufferedReader.read();
251
252                     //Checking if the buffer is empty (there is no data to read).
253                     if (heartRate == -1) {
254                         break;
255                     }
256                     //Case where the heartRate is normal.
257                     if (heartRate <= 100 && heartRate >= 60) {
258                         heartText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is normal");
259                     }
260                     //Case where the heartRate is below the normal.
261                     else if (heartRate < 60) {
262                         heartText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart rate is below normal " + heartRate);
263                         result.setText( t: " An alert message is sent to the Medical Server.");
264                         imageName = "HeartRate";
265                         image.setIcon( icon: HeartRate);
266                     } //Case where the heartRate is above the normal.
267                     else {
268                         heartText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart rate is above normal " + heartRate );
269                         result.setText( t: " An alert message is sent to the Medical Server.");
270                         imageName = "HeartRate";
271                         image.setIcon( icon: HeartRate);
272                     }
273
```

```
274                 //To wait for 1 second before reading the next int, we will use a thread.
                    Thread.sleep( millis: 1000);
276                 //-----------------------------------------------------------------
277
278                 //Reading the value of the oxygen saturation sent by client.
279                 oxygen = bufferedReader.read();
280
281                 //Checking if the buffer is empty (there is no data to read).
282                 if (oxygen == -1) {
283                     break;
284                 }
285                 //Case where the Oxygen saturation is normal.
286                 if (oxygen > 75) {
287                     oxgText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Oxygen saturation is normal");
288                     imageName = "Oxygen";
289                     image.setIcon( icon: Oxygen);
290                 }
291                 //Case where the Oxygen saturation is below the normal.
292                 else {
293                     oxgText.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Oxygen Saturations is low " + oxygen);
294                     imageName = "Oxygen";
295                     image.setIcon( icon: Oxygen);
296                     result.setText( t: " An alert message is sent to the Medical Server.");
297                 }
298                 //-----------------------------------------------------------------
```

```
299                     /*
300                     When the sensed data needs to be sent to the medical server to take serious action,
301                     we must establish a connection with the medical server by
302                     opening the client side of the personal server.
303                      */
304                     //The cases were the client side of the personal server is opened :
305                     if (temprature > 38 || heartRate > 100 || heartRate < 60 || oxygen < 75) {
306
307                         //Create the socket with medical server.
308                         //these two lines of code will be used when running this program on different machines
309
310                         /*InetAddress addresses = InetAddress.getByName("192.186.100.10");
311                          String hostName = addresses.getHostName();*/
312                         client_socket = new Socket( host: "localhost",  port: 6062);
313                         OutputStreamWriter outputStreamWriter1 = new OutputStreamWriter( out: client_socket.getOutputStream());
314                         BufferedWriter bufferedWriter1 = new BufferedWriter( out: outputStreamWriter1);
315
316                         //Send the data to the medical server.
317                         bufferedWriter1.write( c: temprature);
318                         bufferedWriter1.flush();
319
320                         bufferedWriter1.write( c: heartRate);
321                         bufferedWriter1.flush();
322
323                         bufferedWriter1.write( c: oxygen);
324                         bufferedWriter1.flush();
325
326                         //Close the client socket after sending the data to the medical server.
327                         client_socket.close();
328                     }
```

```
329                     //-----------------------------------------------------------------
330                     //To wait for 5 seconds before reading the next block of data,we will use a thread.
                        Thread.sleep( millis: 5000);
332                     //To distinguish between each block of sensed data, print a blank line.
333                     System.out.println( x: "");
334                     //To reset all the texts in the GUI screen.
335                     temp1ext.setText( t: null);
336                     nearTK1ext.setText( t: null);
337                     oxg1ext.setText( t: null);
338
339                 }//The end of inner while loop.
340             }//The end of try.
341             catch (IOException exp) {
342                 //Closing the socket and all of the in/output streams and buffers.
343                 socket.close();
344                 inputStreamReader.close();
345                 outputStreamWriter.close();
346                 bufferedReader.close();
347                 bufferedWriter.close();
348
349             }//The end of the catch.
350
351             //Printing a message.
352             System.out.println( x: "Waiting for another connection...");
353         }//The end of outer while loop.
354     }//The end of the main method.
```

```
355 //-----------------------------------------------------------------------------------------------
356     // Variables declaration - do not modify
357     static javax.swing.JTextField connection;
358     static javax.swing.JTextField nearTK1ext;
359     static javax.swing.JLabel image;
360     static javax.swing.JLabel jLabel4;
361     static javax.swing.JLabel jLabel5;
362     static javax.swing.JLabel jLabel6;
363     private javax.swing.JLabel jLabel7;
364     private javax.swing.JPanel jPanel1;
365     static javax.swing.JTextField oxg1ext;
366     static javax.swing.JTextField result;
367     static javax.swing.JTextField temp1ext;
368     // End of variables declaration
369 }//The end of the class.
```

The Medical Server

```
10  package Medical_Server;
11  //All needed imports go to here :
12  import java.io.BufferedReader;
13  import java.io.IOException;
14  import java.io.InputStreamReader;
15  import java.net.ServerSocket;
16  import java.net.Socket;
17  import java.text.SimpleDateFormat;
18  import javax.swing.ImageIcon;
19  //------------------------------------------------------------------------
20  public class Medical_Server extends javax.swing.JFrame {
21
22      /**
23       * Creates new form MedicalServer
24       */
25      public Medical_Server() {
26          initComponents();
27      }
28
29      @SuppressWarnings("unchecked")
30      Generated Code
90
91      //Needed variables for Image icon in the Gui interface screen.
92      static String imageName = null;
93      static ImageIcon ambulance = new ImageIcon( filename: "ambulance.jpg");
94      static ImageIcon callFamily = new ImageIcon( filename: "callFamily.jpg");
95      static ImageIcon appointment = new ImageIcon( filename: "appointment.jpg");
96
```

```
    private void TextActionActionPerformed(java.awt.event.ActionEvent evt) {

    }

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

        new Medical_Server().setVisible( b: true);

        //Initailizing a socket.
        Socket socket = null;

        //Initailizing an input stream reader.
        InputStreamReader inputStreamReader = null;

        //Initailizing a buffered reader.
        BufferedReader bufferedReader = null;

        //Initailizing a server socket that waits for requests to come over the network.
        ServerSocket server_socket = null;

        //Creating the server socket to communicate with the client, & the port No. should be the same.
        //Listen on a certain port No. for connections.
        server_socket = new ServerSocket( port: 6062);

        //Printing a message.
        serverISON.setText( t: "Server is ON and wait for data...");

        //Declaring variabales to be read the by socket.
        int temprature;
        int heartRate;
        int oxygen;
```

```
        try {
            while (true) {

                //Communication done ,Handshaking.
                socket = server_socket.accept();

                //Printing a message.
                conncetionEsablished.setText( t: "New connection with Personal Server is established, here are the received data: ");

                //Creating the input stream reader to read the messages.
                inputStreamReader = new InputStreamReader( in: socket.getInputStream());

                //Creatng the buffered reader and associating it with the input stream reader.
                bufferedReader = new BufferedReader( in: inputStreamReader);

                //Reading data sent by the client.
                temprature = bufferedReader.read();
                heartRate = bufferedReader.read();
                oxygen = bufferedReader.read();
```

```
150             //Creating a date and identifying the format of the date.
151             SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd 'time' HH:mm:ss");
152
153             //Initailizing the timeout variable with 0 to start connection with 1 client.
154             int timeout = 0;
155
156             //While loop to receive and read data sent by the client. It ends when the connection time is out.
157             while (timeout == 0) {
158
159                 /*
160                 Cases of medical server, these are the message that are sent to the caregiver of the patient.
161                 The medical server will first process the sent data of the patient and based on
162                 the 3 cases will send the caregiver a message to show them the proper action to
163                 take care of the patint.
164                 */
165                 //1st case.
166                 if (temprature > 39 && heartRate > 100 && oxygen < 95) {
167                     textTemprature.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Tempruture is high " + temprature);
168                     // Printing a message.
169                     textHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is above normal " + heartRate);
170                     // Printing a message.
171                     textOxagyn.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " oxygen saturation is low " + oxygen);
172                     //Printing a message.
173                     textAction.setText( t: "Send an ambulance to the patient!");
174                     //Create an image and display it.
175                     imageName = "ambulance";
176                     image.setIcon( icon:ambulance);
177                 }
```

```
178             //---------------------------------------------------------------
179             //2nd case.
180             else if ((temprature >= 38 && temprature <= 38.9) && (heartRate >= 95 && heartRate <= 98) && oxygen < 80) {
181                 //Printing a message.
182                 textTemprature.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Tempruture is normal " + temprature);
183                 //Printing a message.
184                 textHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is normal " + heartRate);
185                 //Printing a message.
186                 textOxagyn.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " oxygen saturation is low " + oxygen);
187                 //Printing a message.
188                 textAction.setText( t: "Call the patient's family!");
189                 //Create an image and display it.
190                 imageName = "callFamily";
191                 image.setIcon( icon:callFamily);
192             }
193             //---------------------------------------------------------------
194             //3rd case.
195             else {
196                 if(temprature >= 39){
197                 //Printing a message.
198                 textTemprature.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Tempruture is high " + temprature);
199                 }
200                 else if(temprature < 37){
201                 //Printing a message.
202                 textTemprature.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Tempruture is low " + temprature);
203                 }
204                 else{
205                 //Printing a message.
206                 textTemprature.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Tempruture is normal " + temprature);
207                 }
```

```
                if(heartRate < 60){
                //Printing a message.
                 textHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is below normal " + heartRate);
                }
                else if(heartRate > 100){
                //Printing a message.
                 textHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is above normal " + heartRate);
                }
                else{
                //Printing a message.
                 textHeart.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " Heart Rate is normal " + heartRate);
                }
                if(oxygen < 95 ){
                //Printing a message.
                 textOxagyn.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " oxygen saturation is low " + oxygen);
                }
                else if(oxygen > 100){
                //Printing a message.
                 textOxagyn.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " oxygen saturation is high " + oxygen);
                }
                else{
                //Printing a message.
                 textOxagyn.setText("At date: " + formatter.format( obj: System.currentTimeMillis()) + " oxygen saturation is normal " + oxygen);
                }
                //Printing a message.
                 textAction.setText( t: "Warning, advise patient to make a checkup appointment!");
                //Create an image and display it.
                imageName = "appointment";
                image.setIcon( icon:appointment);
            }
```

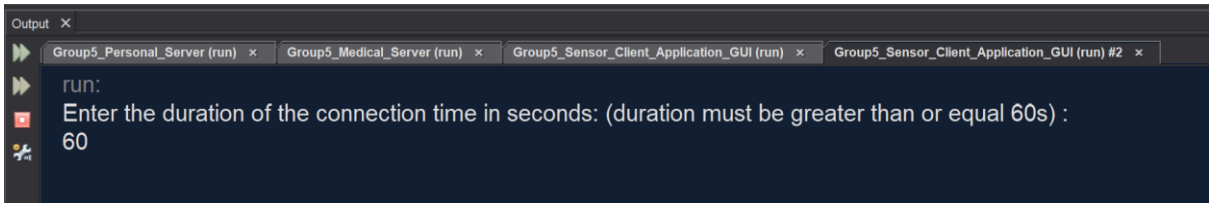
```
                    //---------------------------------------------------------------
                    //To distinguish between each block of sensed data, print a blank line.
                    System.out.println();
                    //To end the connection with the client,change the value of the timeout variable to 1.
                    timeout = -1;
                }//The end of inner loop.
            }//The end of outer loop.
        }//The end of try.
        catch (IOException exp) {
        //Closing the socket and all of the in/output streams and buffers.
            socket.close();
            inputStreamReader.close();
            bufferedReader.close();
        }//The end of catch.
    }//The end of the main method.
//-----------------------------------------------------------------------------------------------
    // Variables declaration - do not modify
    static javax.swing.JTextField textAction;
    static javax.swing.JTextField textHeart;
    static javax.swing.JTextField textOxagyn;
    static javax.swing.JTextField textTemprature;
    static javax.swing.JLabel background;
    static javax.swing.JTextField connectionEstablished;
    static javax.swing.JLabel image;
    static javax.swing.JPanel jPanel1;
    static javax.swing.JTextField serverIsOn;
    // End of variables declaration
}//The end of the class.
```

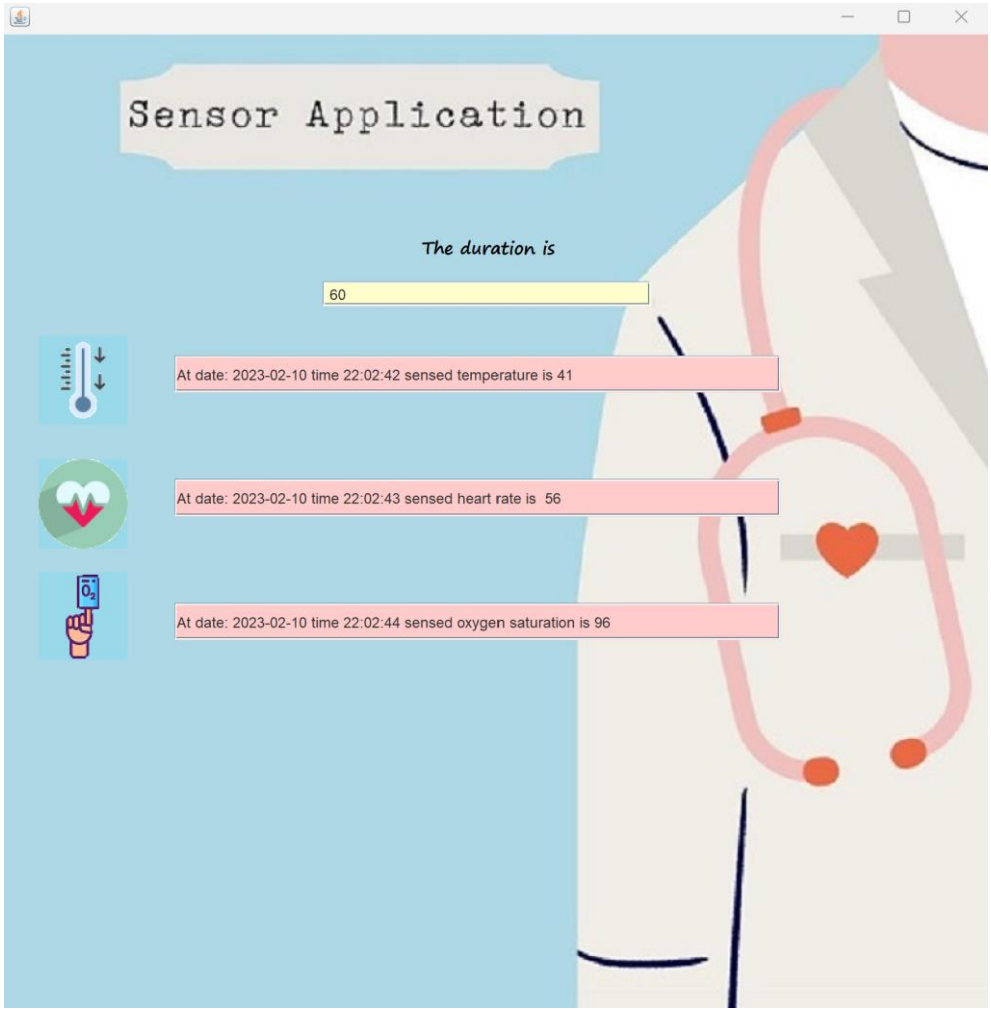
4. RHMS Application run snapshots

4.1 Sample run on one machine.

- Sample run of the Client Side:

First the system will ask for the duration of measuring the temperature, heartrate and oxygen in the client side. Then the generated data will be shown in the GUI of the client and sent to the personal server.



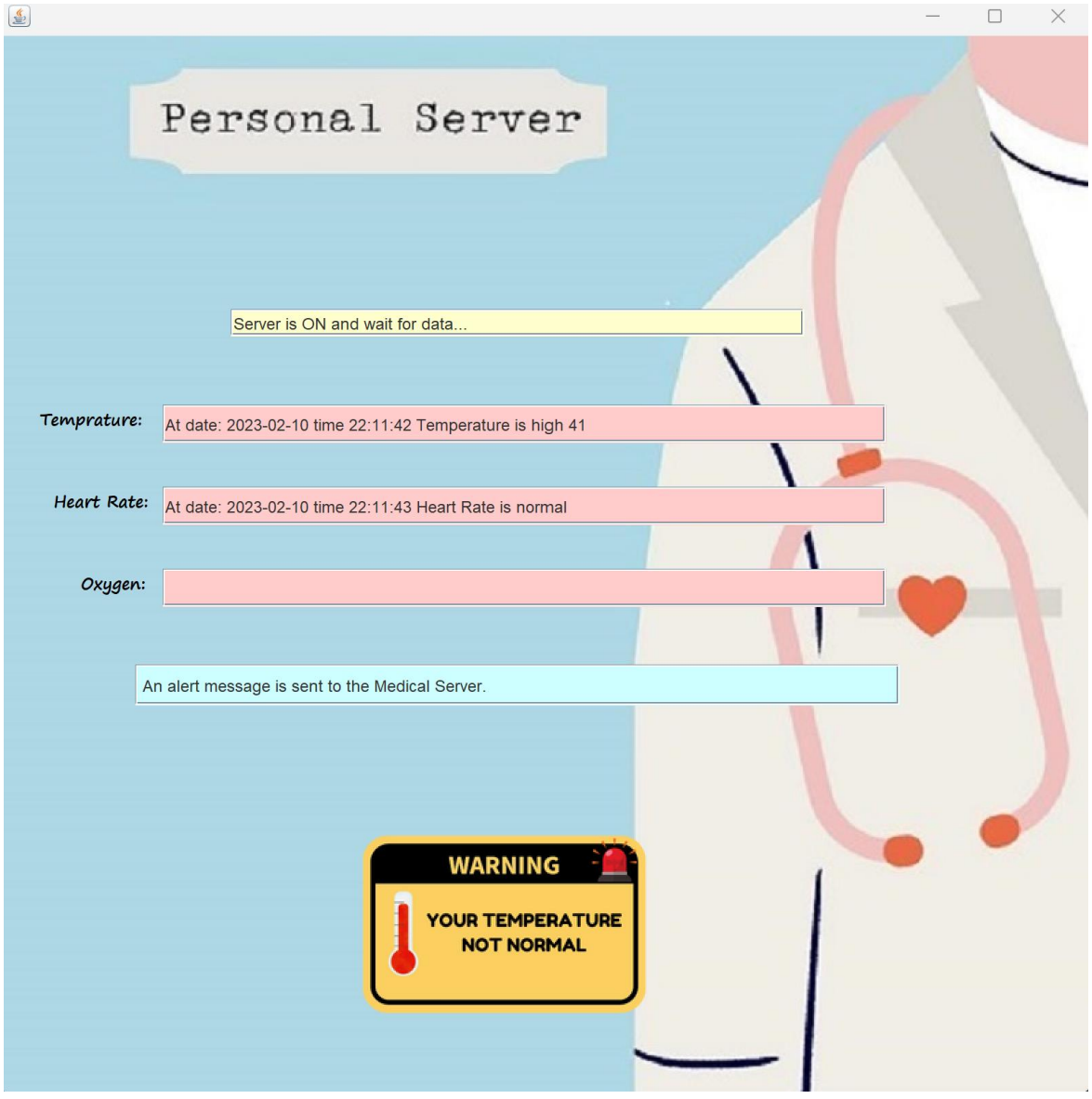


- Sample run of the Personal Server:

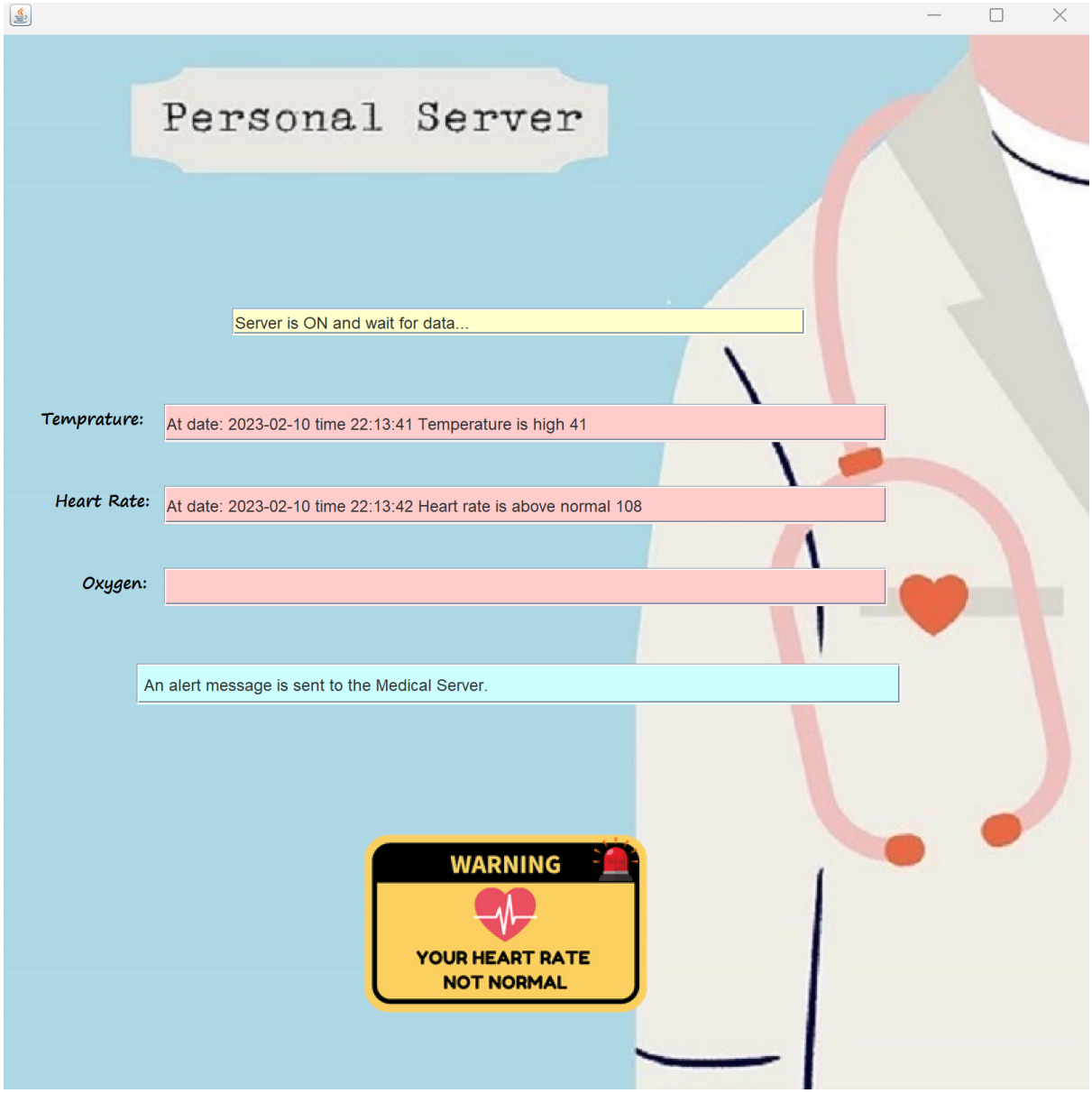
The data will be processed on this server. If there are any serious vital signs, this server will connect to the medical server.

The GUI of the personal server will show a massage and graphs based on the vital signs, such as temperature is abnormal if its high or low.

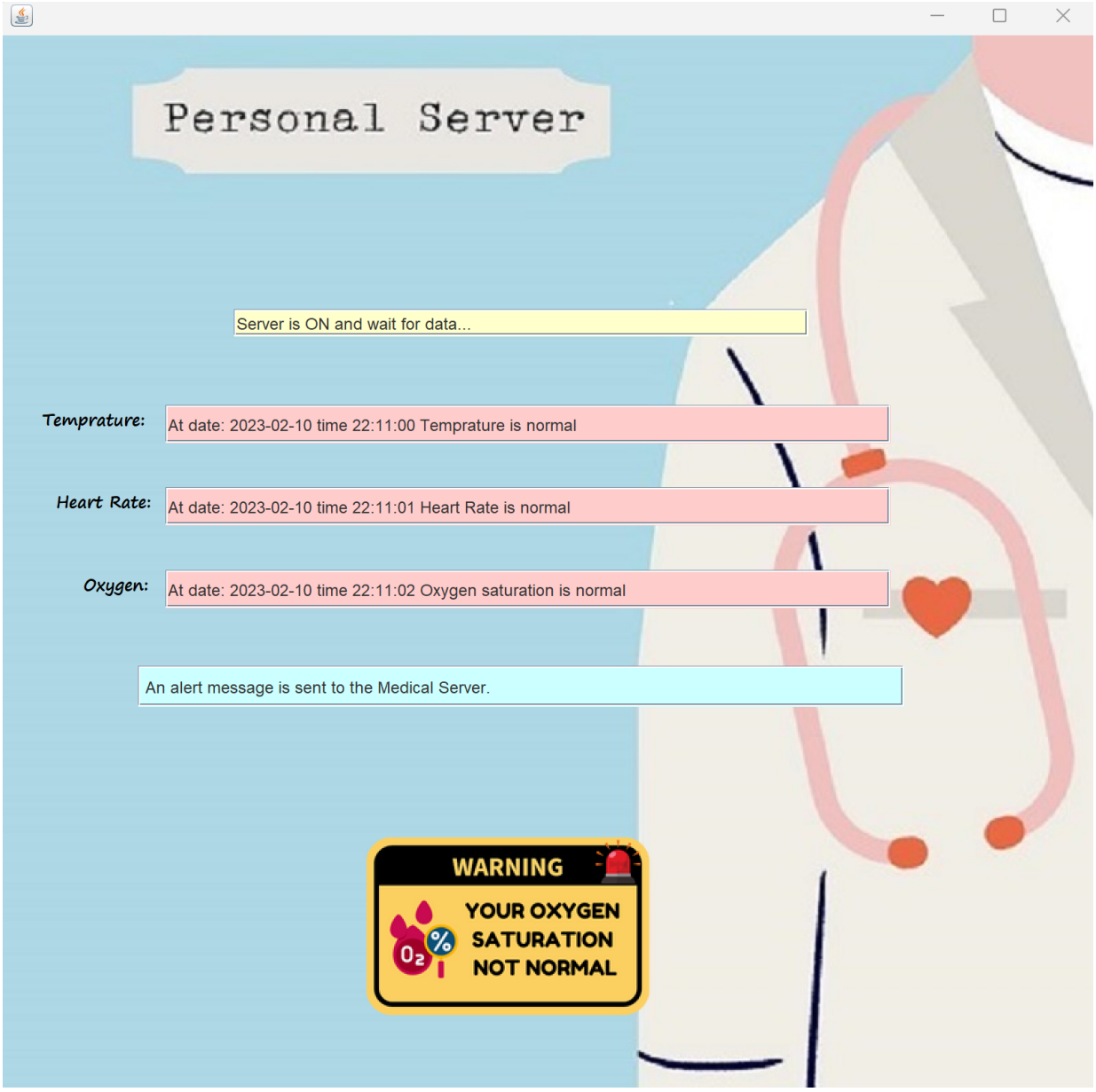
Output abnormal temperature:



output abnormal heartrate:



Output abnormal oxygen saturation level:

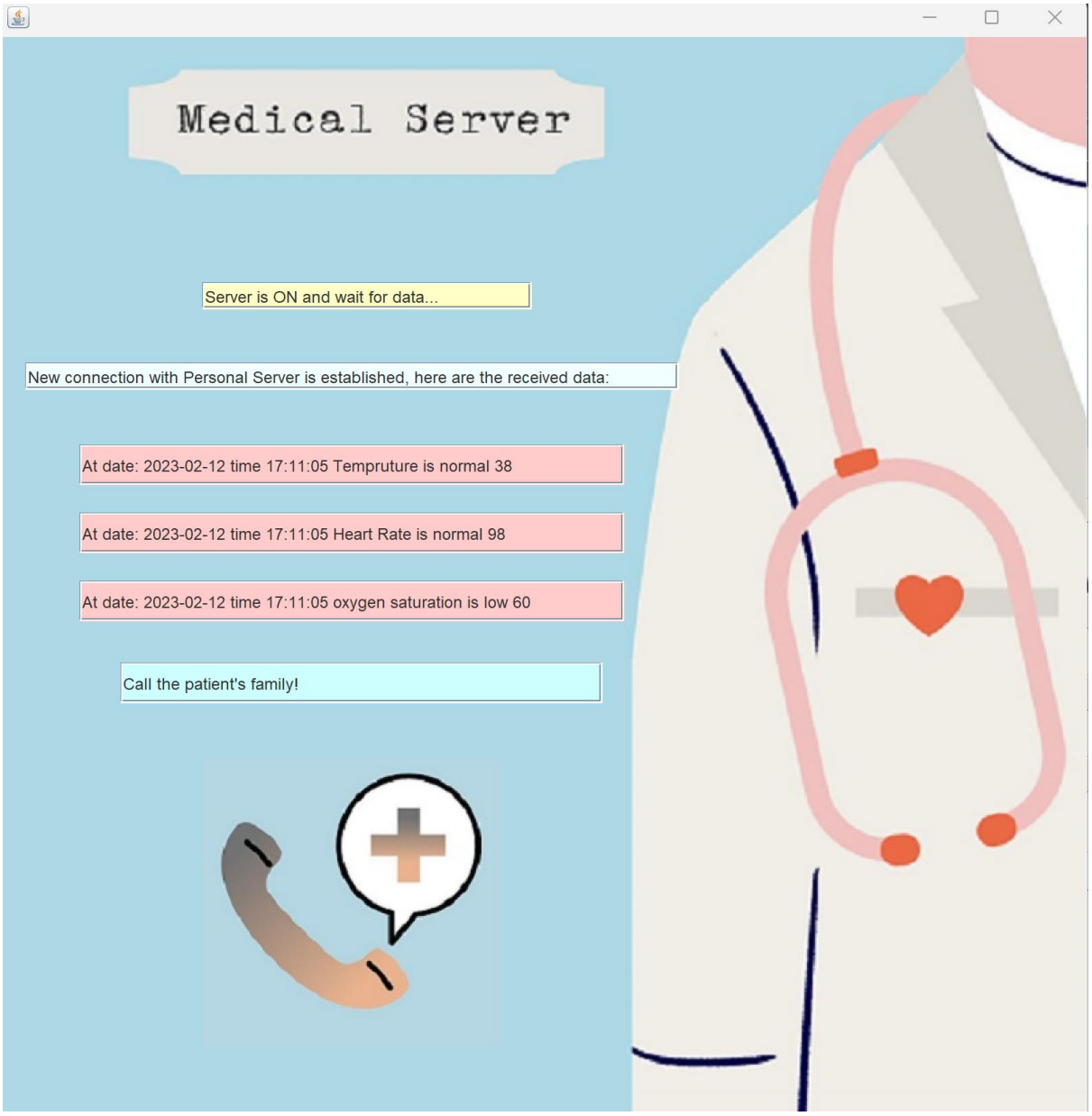


- Sample run of the Medical Server:

  Case 1:

  The medical server will send warning massage to the patient's caregiver to call the patient's family if the following has happened.
  - The temperature is between (38 and 38.9)
  - The heart rate is between (95 and 98)
  - The oxygen saturation level is below 80

  Output massage to call the patient's family:

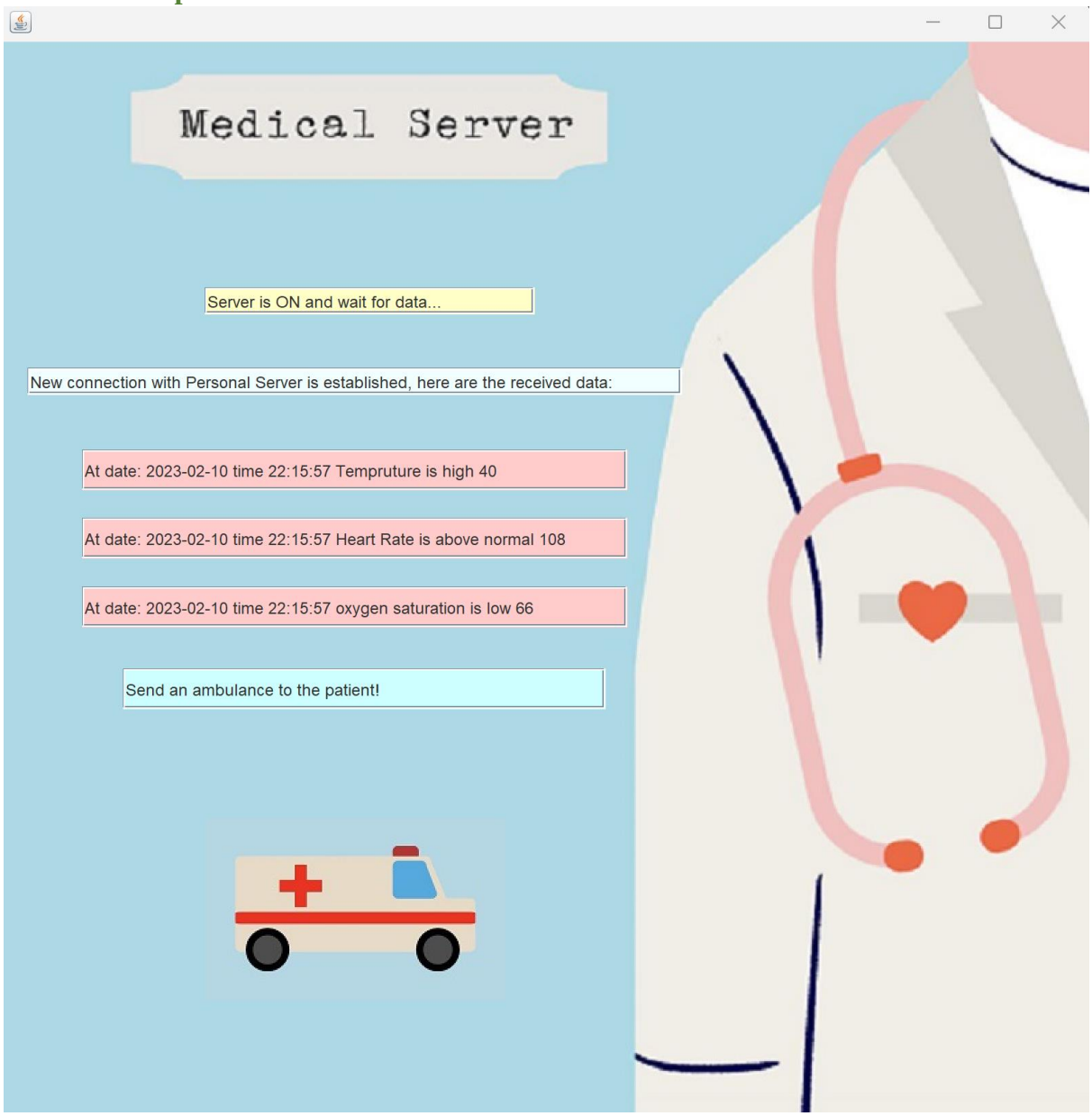


Case 2:

The medical server will send a massage to the patient's caregiver to send an ambulance if the following has happened :

- The temperature is above 39
- The heart rate is above 100
- The oxygen saturation level is below 95

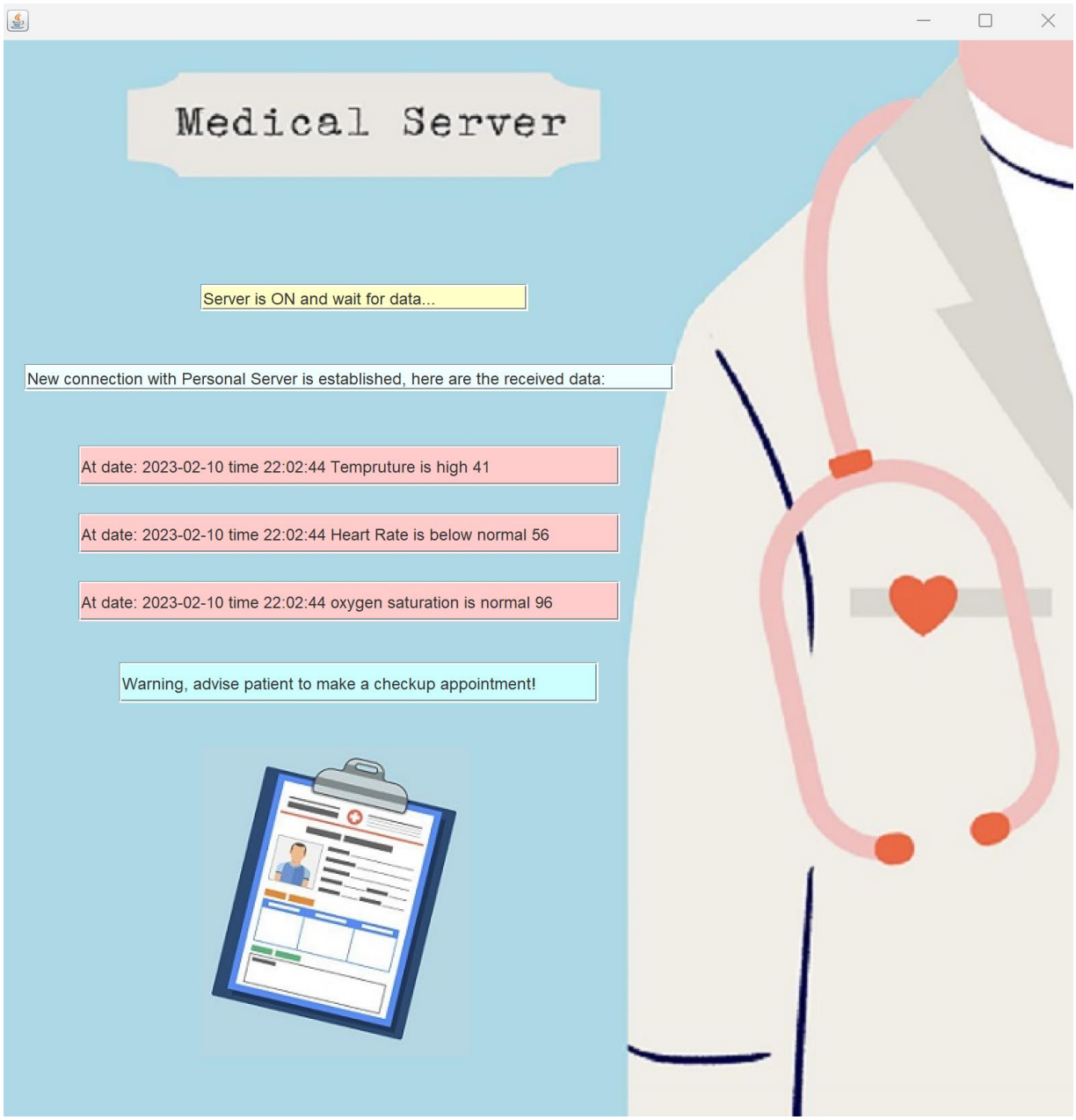
output send an ambulance:



Case 3:

Otherwise, based on the processing of the sensed data, the medical server will advise the caregiver to set an appointment to check-up on the patient.

Output massage make an appointment:



5. Teamwork

5.1 work distribution

Assigned tasks	Members					
Create the sensor client application. And the personal server side	Areej Suleman	Ebtihaj Alnaqeeb	Mai Khalil			
Create the Personal server client side and the medical server	Sarah Abukhammas	Shahad Bin Kulaib	Raghad Alghamdi			
Write the report	Areej Suleman	Ebtihaj Alnaqeeb	Mai Khalil	Sarah Abukhammas	Shahad Bin Kulaib	Raghad Alghamdi
Create the GUI for sensor client application	Sarah Abukhammas	Raghad Alghamdi				
Create GUI for the personal server	Ebtihaj Alnaqeeb	Shahad Bin Kulaib				
Create GUI for the medical server	Areej Suleman	Mai Khalil				

Table 1:Work Distribution among group members

5.2 work coordination

We first divided the work between us, and we set deadlines to finish each task. We also set up a meeting at the end of the deadline of each task to discuss the outcome of the task with the rest of the group members who did not participate in the task. Also, the members that were assigned to do a certain task together had to do meetings whether online or in person to do the task.

5.3 Encountered difficulties

We had difficulty in restricting the connection time between the client and the server according to the time given by the user, as well as scheduling the process of sending the sensed data to the personal server.

We overcame these difficulties by watching YouTube tutorials and discussing the course instructors.

5.4 Lessons learned

By completing this Project, we as computer networking students gained a lot of experience in this field and others, practically applying what we learned in the lectures. Such as creating and closing TCP connection, connecting two devices in a LAN network, dealing with the client and server, and creating a GUI.

6. Conclusion

To conclude, the remote health monitor system (RHMS) is an example of a client-server application that uses 3 types of sensors. It uses a TCP connection to process health information for an individual, process the sensed data and show the caregiver what action should they take to care for their patient based on the conclusions of the processed sensed data.

7. References

1- *Client-server Application - OOSE*. (n.d.). Madooei.github.io.

https://madooei.github.io/cs421_sp20_homepage/client-server-app/

2- Kurose, J. F., & Ross, K. W. (2013). Computer networking a top-down approach. Pearson.

3- *JFrame in Java | How to Create Java JFrames*. (2019, September 9). Edureka. https://www.edureka.co/blog/java-jframe/