CS 2310 Final Project Report

Shijia Liu (shl125@pitt.edu)

Dec 05 2016

Introduction

In this final project, I extended our previous work exercise 4 to implement the components. I used a gesture recognize sensor(Kinect) to recognize the patient's gesture as our system input. I also implemented the components "Emergency Manager" "Homecare staff" and the "uploader" by using the SIS system to accomplish the information exchange in order to help the patient have a faster response when they have a severe condition.

Background

■ SIS (Slow Intelligence System)

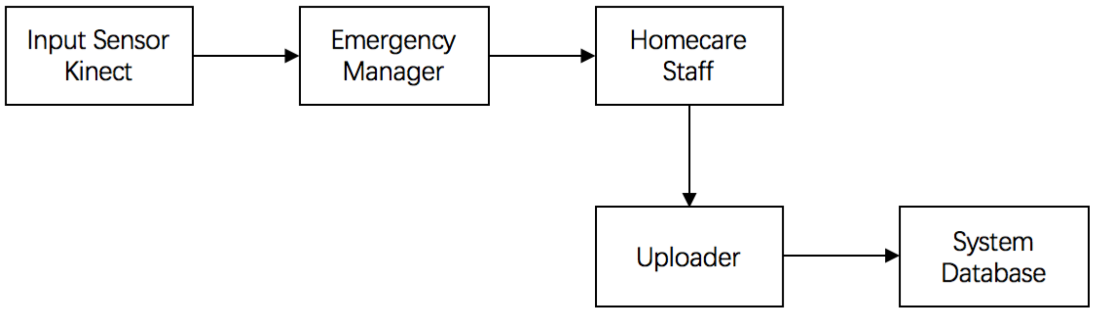
Our Slow Intelligence System is a system that solves problems by trying different solutions, it is context-aware to adapt to different situations and to propagate knowledge, and also it may not perform well in the short run but continuously learns to improve its performance over time.

■ Kinect (Input Sensor)

Kinect for Xbox 360, or simply Kinect, is a motion sensing input device by Microsoft for the Xbox 360 video game console. Based upon a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface using gestures and spoken commands.

Components

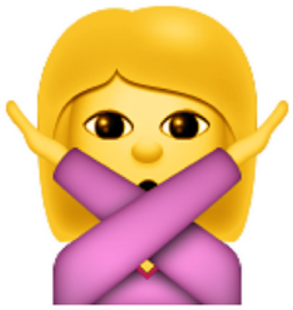
First, let me introduce the whole architecture in our system:



■ *Gesture Recognition*

In our exercise 4, we used the PrjRemote to imitate sending the input message to the Emergency Manager. In this final project, I used a gesture recognition sensor(Kinect) which I introduced above to replace the PrjRemote. Kinect will recognize a specific kind of gesture and transfer it into an emergency message then send it to next component. The Gesture Recognition component is a basic component in our system.

Since the gesture the patient perform should be a gesture which shouldn't be the gesture he or she usually perform during his or her daily time. I design the gesture just to cross the arms, which act as followed:



■ *Emergency Manager*
This component is the only super component in our system. It acts as a console in the whole progress. First, it listened to the Gesture Recognition component, when the Gesture Recognition receive an emergency gesture, then it sends an emergency message to this component. At this time, Emergency Manager should have a judgement mechanism to decide which message should send to the Homecare Staff: if this is the first time emergency message it received, then Emergency Manager should send an alert message "call the patient" to the Homecare Stuff; if this is the second or more than second time received the emergency message, then it directly tell the Homecare Stuff "visit the patient".

In this project, I implemented a timer and a counter to count during a set amount of time, the number of time the emergency message has been received from the Gesture Recognition component.

■ *Homecare Staff*

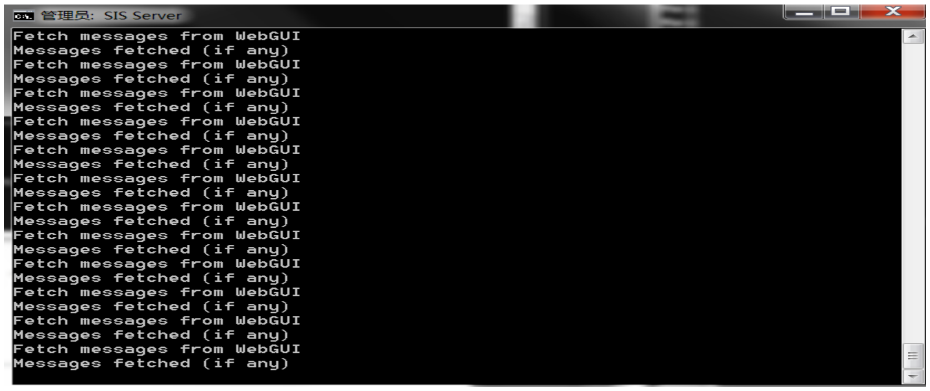
The Homecare Staff is a basic component in this system. What this component do is to receive the message from the Emergency Manager, then generate a GUI to show that should either to call or visit the patient. Furthermore, the Homecare Staff could also actively have an option to choose either call or visit the patient based on the patient answer the phone or not. If the patient answered the call, then the system ends; if the patient didn't answer the call, then this component will generate a GUI shows that "should visit the patient".
Another task of this component is to send an alert message to the uploader. I will introduce the uploader in the next.

■ *Uploader*

The Uploader is also a basic component in our system. It simply receives the alert message from the Homecare Staff, and upload the alert message to the database clarify by the data type, in my design, there only exits two data type: "call" and "visit". This progress could record that every time when the patient has sent an emergency message in order to make it easier for the future use or diagnosis.

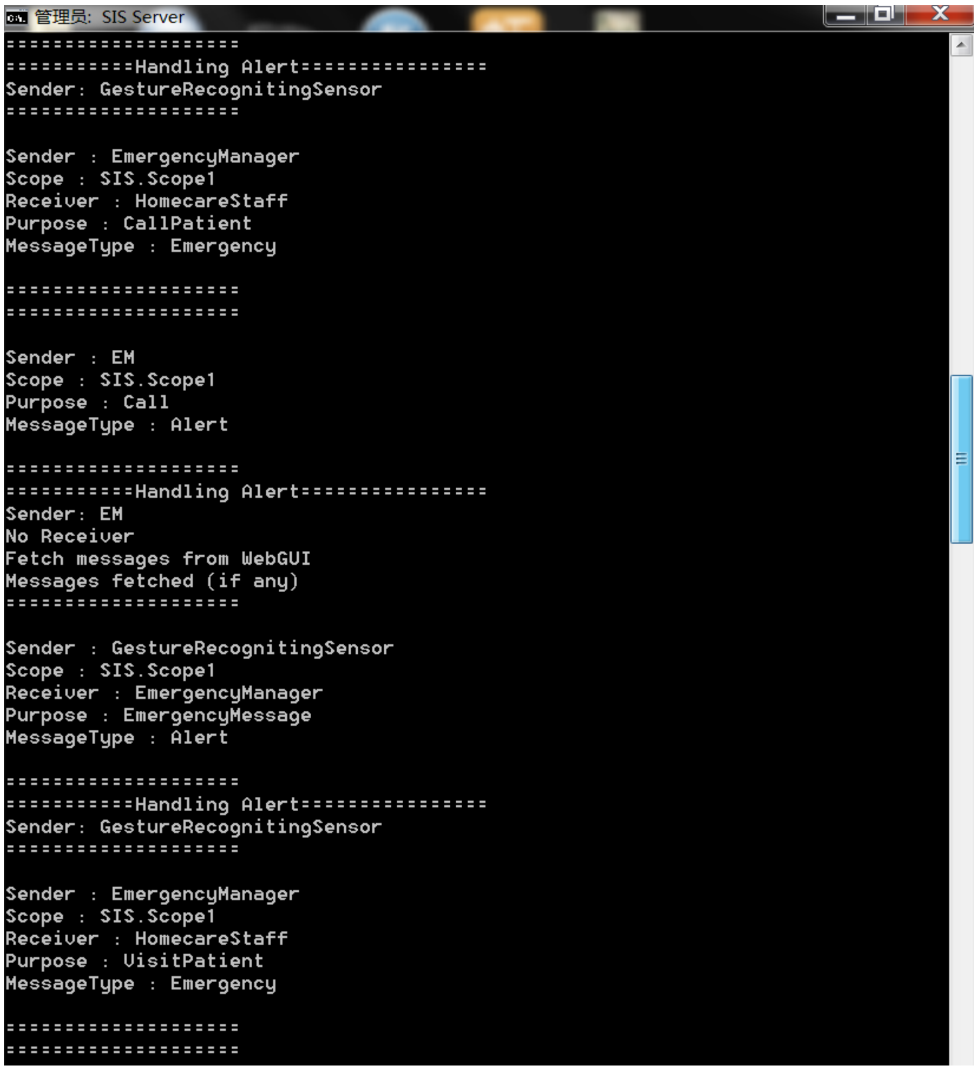
Scenario

Run the server



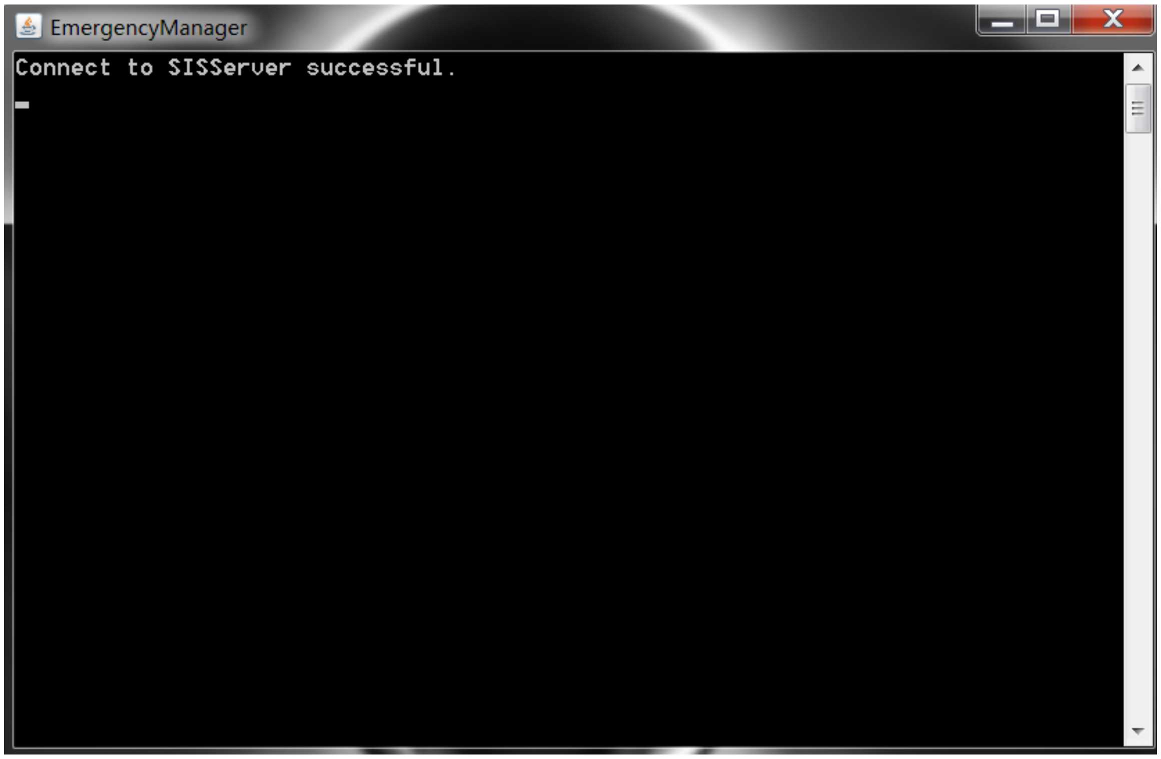
```
管理员: SIS Server
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
```

Initialize the server



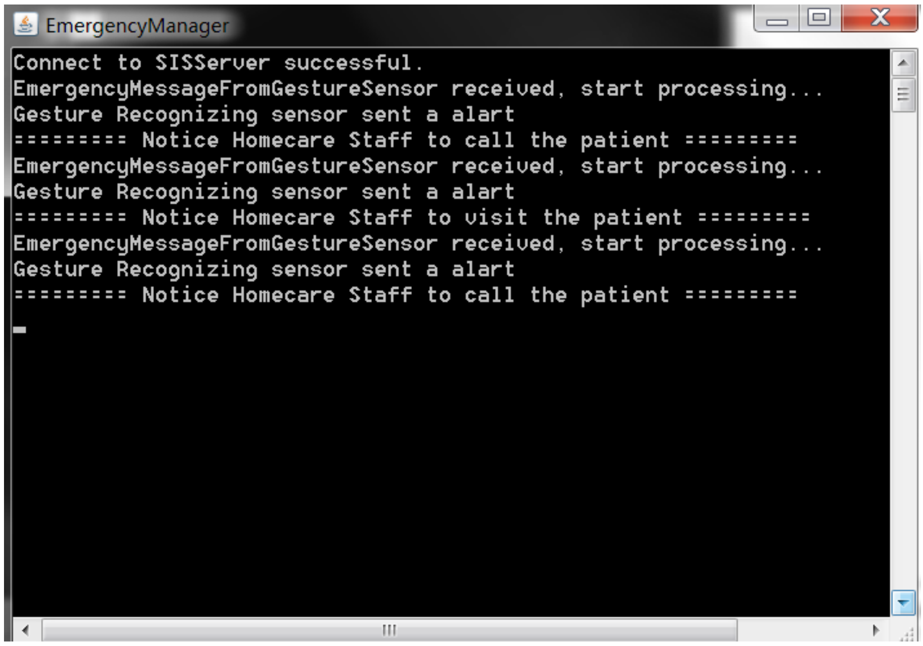
```
管理员: SIS Server
====================
==========Handling Alert===============
Sender: GestureRecognitingSensor
====================

Sender : EmergencyManager
Scope : SIS.Scope1
Receiver : HomecareStaff
Purpose : CallPatient
MessageType : Emergency

====================
====================

Sender : EM
Scope : SIS.Scope1
Purpose : Call
MessageType : Alert

====================
==========Handling Alert===============
Sender: EM
No Receiver
Fetch messages from WebGUI
Messages fetched (if any)
====================

Sender : GestureRecognitingSensor
Scope : SIS.Scope1
Receiver : EmergencyManager
Purpose : EmergencyMessage
MessageType : Alert

====================
==========Handling Alert===============
Sender: GestureRecognitingSensor
====================

Sender : EmergencyManager
Scope : SIS.Scope1
Receiver : HomecareStaff
Purpose : VisitPatient
MessageType : Emergency

====================
====================
```

Connect the Emergency Manager



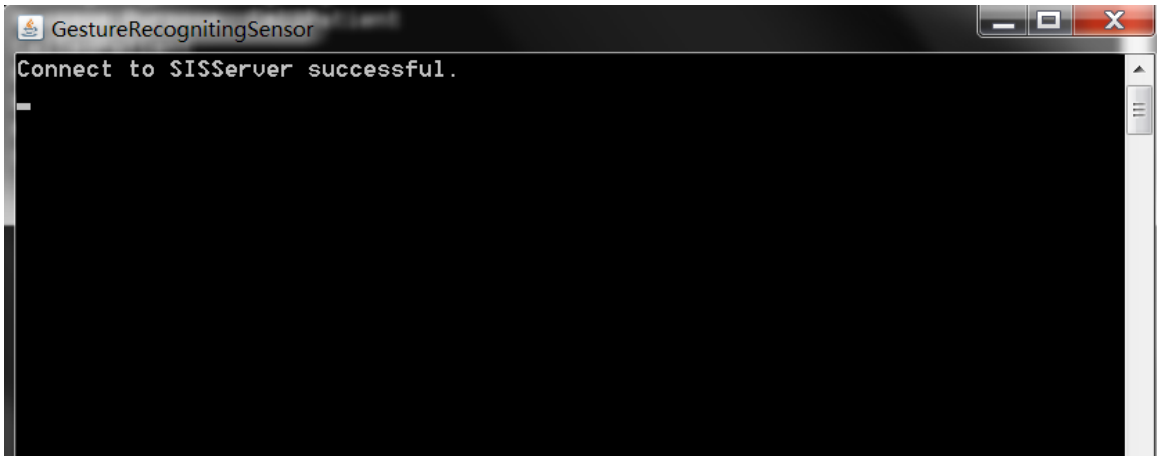
```
Connect to SISServer successful.
```

When the Emergency Manager received multiple times of the emergency messages:



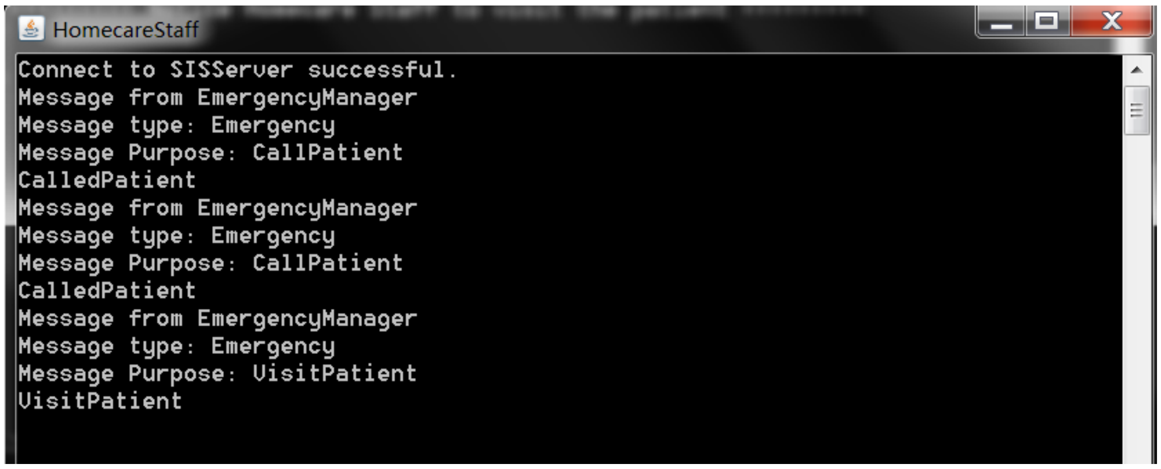
```
Connect to SISServer successful.
EmergencyMessageFromGestureSensor received, start processing...
Gesture Recognizing sensor sent a alart
========= Notice Homecare Staff to call the patient =========
EmergencyMessageFromGestureSensor received, start processing...
Gesture Recognizing sensor sent a alart
========= Notice Homecare Staff to visit the patient =========
EmergencyMessageFromGestureSensor received, start processing...
Gesture Recognizing sensor sent a alart
========= Notice Homecare Staff to call the patient =========
```

Create and connect the Gesture Recognition


GestureRecognitingSensor

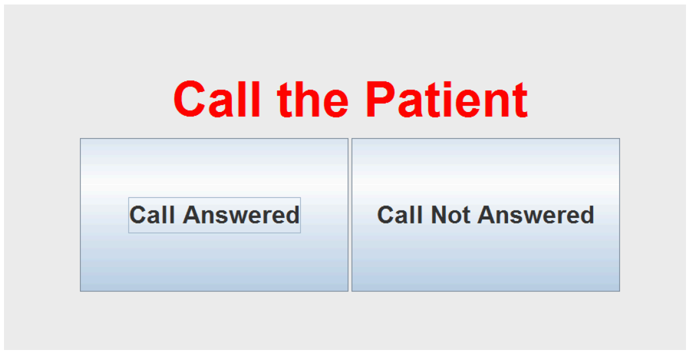
```
Connect to SISServer successful.
```

Create and connect the Homecare Staff, after received the emergency message, it looks like:

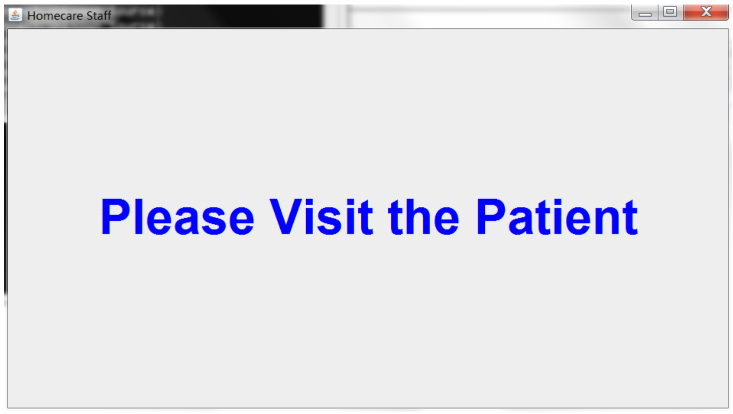

HomecareStaff

```
Connect to SISServer successful.
Message from EmergencyManager
Message type: Emergency
Message Purpose: CallPatient
CalledPatient
Message from EmergencyManager
Message type: Emergency
Message Purpose: CallPatient
CalledPatient
Message from EmergencyManager
Message type: Emergency
Message Purpose: VisitPatient
VisitPatient
```

The GUI which generated by the Homecare Staff:

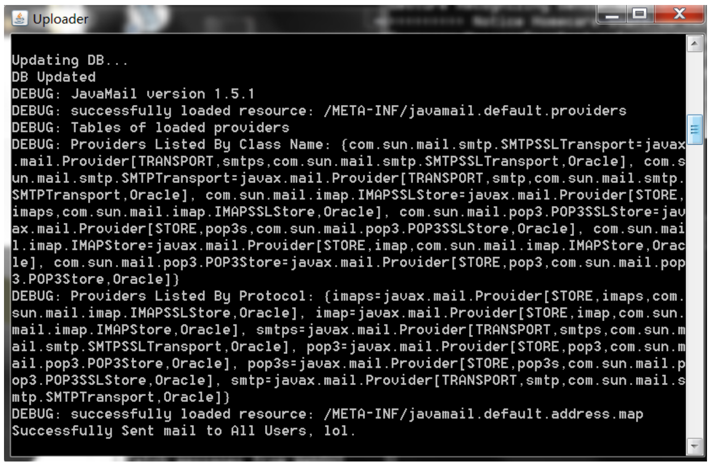


If the patient sends another emergency gesture or didn't answer the phone call, the GUI will be showed as below:



Please Visit the Patient

When the uploader works and upload the alert information to the database:



```
Updating DB...
DB Updated
DEBUG: JavaMail version 1.5.1
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name: {com.sun.mail.smtp.SMTPSSLTransport=javax
.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Oracle], com.s
un.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.
SMTPTransport,Oracle], com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,
imaps,com.sun.mail.imap.IMAPSSLStore,Oracle], com.sun.mail.pop3.POP3SSLStore=jav
ax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Oracle], com.sun.mai
l.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Orac
le], com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop
3.POP3Store,Oracle]}
DEBUG: Providers Listed By Protocol: {imaps=javax.mail.Provider[STORE,imaps,com.
sun.mail.imap.IMAPSSLStore,Oracle], imap=javax.mail.Provider[STORE,imap,com.sun.
mail.imap.IMAPStore,Oracle], smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.m
ail.smtp.SMTPSSLTransport,Oracle], pop3=javax.mail.Provider[STORE,pop3,com.sun.m
ail.pop3.POP3Store,Oracle], pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.p
op3.POP3SSLStore,Oracle], smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.s
mtp.SMTPTransport,Oracle]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
Successfully Sent mail to All Users, lol.
```

###

Possible GEMs:

1. Gesture Recognition: I used Kinect but not the original PrjRemote to be my input sensor to recognize the patient's gesture.

2. I implemented some GUI to visualize that what should do to the patient, either call or visit the patient.
3. Use the uploader to upload the emergency information to the database.

4. Designed an easy and an uncommon gesture in order to make it more clear and feasible to recognize the gesture.

5. Implement a selection mechanism, make it more real to act has a real homecare staff, for them to actively choose the button based on either the patient answered the phone call or not.

The implementation code for creating the Homecare Staff component and connect it with the system is the core part in my project, here is my codes:

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CreateHomecareStaff
{

    // socket for connection to SISServer
    static Socket universal;
    private static int port = 53217;
    // message writer
```

```
		static MsgEncoder encoder;
		// message reader
		static MsgDecoder decoder;
		// scope of this component
		private static final String SCOPE = "SIS.Scope1";
	// name of this component
		private static final String NAME = "HomecareStaff";
		// messages types that can be handled by this component
		private static final List<String> TYPES = new
ArrayList<String>(
				Arrays.asList(new String[] { "Emergency", "Confirm" }));

		private static final String incomingMessages =
"IN\tConfirm|Emergency:CallPatient|Emergency:VisitPatient\n";
		private static final String outgoingMessages =
"OUT\tConnect";

		private static Timer timer = new Timer();

		// shared by all kinds of records that can be generated by this
component
		private static KeyValueList record = new KeyValueList();
		// shared by all kinds of alerts that can be generated by this
component
		private static KeyValueList alert = new KeyValueList();

		public static void main(String[] args)
		{
			javax.swing.SwingUtilities.invokeLater(new Runnable() {
				public void run() {
```

```
                    showCall();
                }
            });
            while (true)
            {
                try
                {

                    // try to establish a connection to SISServer
                    universal = connect();

                    // bind the message reader to inputstream of the
socket
                    decoder = new
MsgDecoder(universal.getInputStream());
                    // bind the message writer to outputstream of the
socket
                    encoder = new
MsgEncoder(universal.getOutputStream());

                    /*
                     * construct a Connect message to establish the
connection
                     */
                    KeyValueList conn = new KeyValueList();
                    conn.putPair("Scope", SCOPE);
                    conn.putPair("MessageType", "Connect");
                    conn.putPair("IncomingMessages",
incomingMessages);
                    conn.putPair("OutgoingMessages",
outgoingMessages);
                        conn.putPair("Role", "Basic");
                    conn.putPair("Name", NAME);
                    encoder.sendMsg(conn);

                    // KeyValueList for inward messages, see
KeyValueList for
                    // details
                    KeyValueList kvList;

                    while (true)
                    {
                          // attempt to read and decode a message,
see MsgDecoder for
                          // details
                          kvList = decoder.getMsg();

                          // process that message
                          ProcessMsg(kvList);
                    }

                }
                catch (Exception e)
                {
                    // if anything goes wrong, try to re-establish the
connection
                    e.printStackTrace();
                    try
                    {
                          // wait for 1 second to retry
```

```
                    Thread.sleep(1000);
                }
                catch (InterruptedException e2)
                {
                }
                System.out.println("Try to reconnect");
                try
                {
                    universal = connect();
                }
                catch (IOException e1)
                {
                }
            }
        }
    }

    /*
     * used for connect(reconnect) to SISServer
     */
    static Socket connect() throws IOException
    {
        Socket socket = new Socket("127.0.0.1", port);
        return socket;
    }
    private static void ProcessMsg(KeyValueList kvList) throws
Exception
    {

        String scope = kvList.getValue("Scope");
```

```
            if (!SCOPE.startsWith(scope))
            {
                return;
            }

            String messageType = kvList.getValue("MessageType");
            if (!TYPES.contains(messageType))
            {
                return;
            }

            String sender = kvList.getValue("Sender");

            String receiver = kvList.getValue("Receiver");

            String purpose = kvList.getValue("Purpose");
            switch (messageType)
            {
            case "Confirm":
                System.out.println("Connect to SISServer
successful.");
                break;
            case "Emergency":
                if (receiver.equals(NAME))
                {
                    System.out.println("Message from " + sender);
                    System.out.println("Message type: " +
messageType);
                    System.out.println("Message Purpose: " +
purpose);
```

```
                    switch (purpose)
                    {
                        case "CallPatient":
                            System.out.println("CalledPatient");

javax.swing.SwingUtilities.invokeLater(new Runnable() {
                                public void run() {
                                    showCall();
                                }
                            });

                            KeyValueList new_alert = new
KeyValueList();
                        new_alert.putPair("Scope", SCOPE);
                        new_alert.putPair("MessageType",
"Alert");
                        new_alert.putPair("Sender", "EM");
                        new_alert.putPair("Purpose", "Call");
                        encoder.sendMsg(new_alert);

                            break;
                        case "VisitPatient":
                            System.out.println("VisitPatient");

javax.swing.SwingUtilities.invokeLater(new Runnable() {
                                public void run() {
                                    showVisit();
                                }
                            });
```

```
                                        KeyValueList new_alert_2 = new
KeyValueList();
                                   new_alert_2.putPair("Scope", SCOPE);
                                   new_alert_2.putPair("MessageType",
"Alert");
                                   new_alert_2.putPair("Sender", "EM");
                                   new_alert_2.putPair("Purpose", "Visit");
                                   encoder.sendMsg(new_alert_2);


                                        break;
                              }
                         }
                         break;

                    }
               }
               private static void showVisit() {
                    //Create and set up the window.
                    JFrame frame = new JFrame("Homecare Staff");

//frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                    // make the frame half the height and width
                    Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
                    int height = screenSize.height;
                    int width = screenSize.width;
                    frame.setSize(width/2, height/2);

                    // here's the part where i center the jframe on screen
```

```
			frame.setLocationRelativeTo(null);
			JLabel label = new JLabel("Please Visit the Patient",
SwingConstants.CENTER);
			label.setFont(label.getFont().deriveFont(64.0f));
			label.setForeground(Color.BLUE);
			// label.setPreferredSize(new Dimension(350, 200));
			frame.getContentPane().add(label,
BorderLayout.CENTER);
			//frame.setSize(1000, 600);
			//Display the window.
			//frame.pack();
			frame.setVisible(true);
		}
		public static void showCall() {
			//Create and set up the window.
			JFrame frame = new JFrame("Homecare Staff");
			Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
			int height = screenSize.height;
			int width = screenSize.width;
			frame.setSize(width/2, height/2);

			// here's the part where i center the jframe on screen
			frame.setLocationRelativeTo(null);

//frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			frame.setLayout(new BorderLayout());
			frame.add(new CallPane(frame),
BorderLayout.CENTER);
			//frame.setSize(1000, 600);
```

```
            //Display the window.
            //frame.pack();
            frame.setVisible(true);
        }
        public static class CallPane extends JPanel {

            public CallPane(JFrame myprogram) {
                setLayout(new GridBagLayout());
                GridBagConstraints constraint = new
GridBagConstraints();
                constraint.gridx = 0;
                constraint.gridy = 0;
                constraint.insets = new Insets(2, 2, 2, 2);
                JLabel label = new JLabel("Call the Patient");
                label.setFont(label.getFont().deriveFont(64.0f));
                label.setForeground(Color.RED);
                constraint.gridx=0;
                constraint.gridy++;
                constraint.fill = GridBagConstraints.NONE;
                // constraint.gridwidth = 2;
                JButton answeredButton = new JButton("Call
Answered");

answeredButton.setFont(answeredButton.getFont().deriveFont(32.0f));
                answeredButton.setPreferredSize(new
Dimension(350, 200));
                answeredButton.addActionListener(new
ActionListener() {

                    public void actionPerformed(ActionEvent e)
                    {
                        //Execute when notAnsweredButton is
pressed
                        myprogram.dispose();
                    }
                });
                JButton notAnsweredButton = new JButton("Call
Not Answered");

notAnsweredButton.setFont(notAnsweredButton.getFont().deriveFont(32
.0f));
                notAnsweredButton.setPreferredSize(new
Dimension(350, 200));
                //Add action listener to notAnsweredButton
                notAnsweredButton.addActionListener(new
ActionListener() {

                    public void actionPerformed(ActionEvent e)
                    {
                        //Execute when notAnsweredButton is
pressed
                        showVisit();
                        myprogram.dispose();
                    }
                });
                add(answeredButton, constraint);
                constraint.gridx++;
                add(notAnsweredButton, constraint);
                constraint.gridx = 0;
                constraint.gridy = 0;
                constraint.gridwidth = 2;
                add(label, constraint);

            }

        }
}
```