JAVA BASED VIDEO CONFERENCING

WITH FACE TRACKING

WOON JIA JIE

ELECTRICAL & ELECTRONIC ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

SEPTEMBER 2015

Java Based Video Conferencing with Face Tracking

by

Woon Jia Jie

15391

Dissertation submitted in partial fulfillment of
The requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronic)

SEPTEMBER 2015

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Java Based Video Conferencing with Face Tracking

by

Woon Jia Jie

15391

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(Electrical & Electronic)

Approved by,

____
(Dr. Vijanth Sagayan A/L Asirvadam)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

September 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

WOON JIA JIE

ABSTRACT

This paper addresses the issue of communication tools among employees in a big company. Choices of communication tools are listed down and videoconferencing is selected in this project. However videoconferencing is not a rare technology, where examples can be easily found on the internet. This paper has differentiated ordinary videoconferencing and videoconferencing with face tracking. The main advantage of using face tracking allows user to constantly showing his face while moving around during his presentation. Java is used as the programming language since Java has wide library on the internet for free. This paper also addresses the unnecessary of using external circuit when the program can be run by the computer itself.

ACKNOWLEDGEMENT

This project would not be a great success without the help of a few people. They have guided me throughout the two semester, given me advices, motivated me and supported me during my down times. The experience I obtained during the project is valuable and cannot be obtained through normal academic study. I would like to express my highest gratitude to everyone that spent their precious time in completing this project.

The first person I would like to thank is my project supervisor, Dr. Vijanth Sagayan A/L Asirvadam. Without his guidance, this project would not be a great success. He has helped me for the full length period for this project. He has also given me the knowledge of using java language in programming since I have not learnt this language before. Special thanks to our Final Year Project Committees, and Coordinator. They have guided me and provided me all the information in order to complete this course.

I would also appreciate the help from my friend, willing to provide extra laptop, equipments during my project development. He was also willing to test my product and has given me valuable feedback.

Finally, I would like to apologize if any party was excluded from being mentioned above.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND NOMENCLATURES

FPGA - Field-Programmable Gate Array

QCIF - Quarter Common Intermediate Format

PC - Personal Computer

GPU - Graphics Processing Unit

ITU - International Telecommunication Union

OPNET - Optimized Network Engineering Tools

P2P - Peer-to-Peer

GUI - Graphical User Interface

LAN - Local Area Network

IDE - Integrated Development Environment

TCP/IP - Transmission Control Protocol/ Internet Protocol

CHAPTER 1

INTRODUCTION

1.1 Background Study

In today's globalized business environment, communication between employees is essential for the company. Companies such as PETRONAS, Shell, Schlumberger, have more than 100 branches all around the world. With possibly more than 100,000 of employees in a company, strong relationship between employees is more important than ever to maintain the company. Communication has considered as one of the key to globalize a business.

There are a lot of ways to have long distance communication, for example meeting in person, where managers from different branches, different countries will have to travel to a place and have a meeting there. However with this method, it might take up to 2 or 3 days for someone to travel from one part of the world to the other part. With the advancement of technology, this method has been considered as outdated.

Videoconferencing also known as video telephony is the tool to achieve effective communication between employees in the company. With low-cost and high-quality video, it is almost the same as meeting in person. However to produce a high quality video, it requires a very well developed videoconferencing software tool.

In 1879, videoconferencing started off as a communication tool by transmitting light and sound. It was first using a flat 2 meters wide television with telephone tube. At that time, this method of communication is not widely commercialized because of its huge size and expensive price. It was so inconvenient to have this kind of communication tool [1]. However as technology getting advance, videoconferencing has developed into an essential tool for all the globalized companies. In 1991, The first webcam was developed to only capture still photo. With a very low frame rate, the photos are captured and updated very slowly [2]. However webcams up to 30 frame rates per second are easily available nowadays.

1.2 Problem Statement

Videoconferencing has become very common in most of the globalized company. One can easily communicate with another across country with just a webcam and a computer. A proper designed software serve as a coding encoding platform can make people communicate effectively. The software will have to allow two way communication simultaneously for interactive video and audio.

There are a lot of videoconferencing software which can be downloaded from the internet, such as Skype, TeamViewer, and Microsoft Lync. These software have been widely used among companies for quite sometimes. However there are disadvantages even in these popular videoconferencing software. One of the disadvantages is these software do not control the webcam to move according to user's face. This problem makes user difficult to present if they are moving in front of the webcam.

The videoconferencing software must be able to have face recognition and able to move the webcam based on the user's face [3]. The user's face will thus appear at the screen all the time. This project, Java Based Video Conferencing with Face Tracking will allow the face tracking system activate on the client site.

1.3 Objective and Scope of Study

This project consists of several objectives.

- To design a small-scaled videoconferencing software which able to send text and multimedia file.
- Able to communicate both ways simultaneously with video and audio.
- Able to control and remote the webcam based on client site.

In order for all the objectives to become achievable, there are several topics need to be studied.

- Basic Java Programming
- Image Processing
- Prediction-verification Method
- Network Protocol
- Graphical User Interface Design

CHAPTER 2

LITERATURE REVIEW

Videoconferencing with face detection no longer consider as something new. Many people have invested and developed this kind of videoconferencing tool. However, the enhancement of conventional videoconferencing tool shall not increase the complication for users.

The idea of real-time face detection on videoconferencing tool consists of three main sections, such as subsampling, skin filtering and face detection [4]. All the sections are implemented into FPGA so that the data processing will not consume memory of the normal computer. Subsampling circuit allows the FPGA to process a QCIF 24 bit RGB images using a 16 x 16 mask. Original image will be replaced once it is subsampled and thus will reduce memory usage. The circuit basically consists of adder and register and continuously process each pixel byte every one clock cycle.

Since human skin and face have the same color, this requires the circuit to filter skin pixel which here consider as unwanted noise. Skin filtering circuit processes image from subsampling circuit and updates on the FPGA memory itself. As for face detection, the circuit will spatially filter the binary skin map from skin filtering stage. The circuit performs data analysis and statistic calculation.

The advantages of using such method will help to improve video quality. The transmission of only processed image other than raw data will consume less bandwidth.

However with the additional FPGA circuit, the cost of development will definitely be higher.

Aside from using external FPGA circuit, desktop videoconferencing is also a good choice as videoconferencing tool. As it only requires a PC or maybe an additional webcam and microphone, desktop videoconferencing is more convenient than other. Featuring sound cards and GPU, desktop videoconferencing will definitely offering a better quality of videoconferencing [5]. However with low transmission bandwidth, desktop videoconferencing can be a problem. In 1996, ITU introduced H.323 standard. This protocol allows real-time multimedia communications, in this case as videoconferencing. Research has been conducted in order to increase the performance of videoconferencing bandwidth. By using OPNET, a network design and simulation tool, the performance is believed to be enhanced [6]. Other than that, P2P architecture is also another approach on network protocol, however the focus on this architecture mainly on multipoint videoconferencing [7].

Every face tracking feature will require an algorithm to be executed. Viola-Jones algorithm was introduced as quite a simple face tracking algorithm than can be run on real time. Since this algorithm is simple, it runs faster than any other pixel-based system. The algorithm cascade multiple comparative selections known as the Haar Features [8].

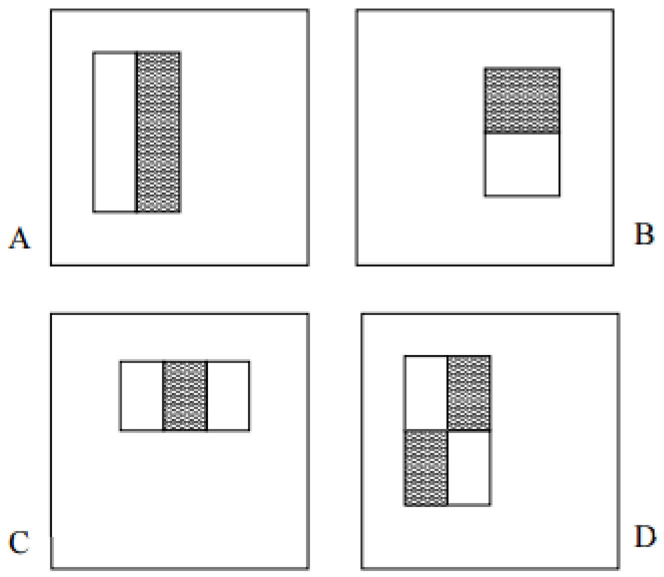


Figure 1: Haar Features

As shown in figure 1, human face actually has certain unique characteristics. For example, human nose bridge has the characteristics of block C, and human eyes has the characteristics of block B. So when an object is able to fulfill all the characteristics in all the Haar Features, it will be considered as a human face.

Many have listed down the effectiveness of videoconferencing is also limited by poor GUI design [5]. The interface designed is not user friendly enough as user might need to have better understanding on computer. Employees might not from IT department nor having good knowledge in computer having trouble using the interface of videoconferencing tool prefer to have a face-to-face conference. Binary or machine language used by a computer is not understandable by human. GUI serves the purpose of interacting human's input with the computer. However, usability of the GUI is more important than the creativity of the GUI design [9]. Instead of having unnecessary feature, the interface should keep it as simple as possible.

CHAPTER 3

METHODOLOGY

This project will mainly focus on creating a GUI software, using Java, allowing two computers to communicate with audio and video. Connection between two computers can be either WiFi or LAN. One computer is set to be master and the other computer is set to be slave. Both the computer will equip a webcam and microphone. When user on the slave site move, the webcam should able to detect the user's face and move the webcam according to the position of user's face.

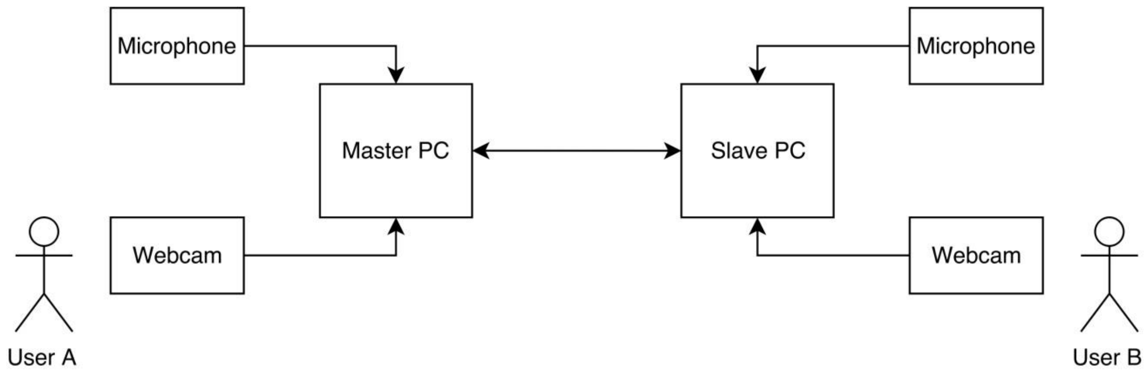


Figure 2: Simple blocks showing how the project will be presented

Based on Figure 2, when user B show face on the webcam, the webcam will be able to detect the face of user B and always keep the detected face on screen. User A will not be able to control the webcam of user B.

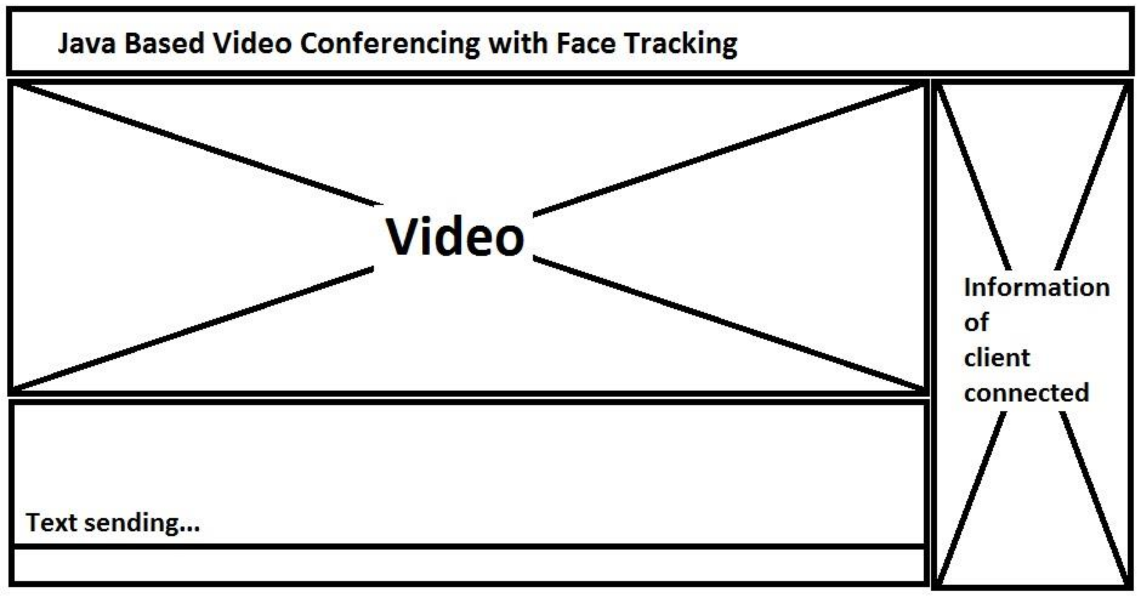

Java Based Video Conferencing with Face Tracking
Video
Information of client connected
Text sending...

Figure 3: GUI of the video conferencing software

Based on figure 3, the GUI will be programmed and designed using Java language. It basically allows user to input text and send to the connected client. At the right side, it also shows the information of client connected. The main video from connected client's webcam will be shown at the middle of the GUI.

Experiments will be conducted to test if the GUI works as expected and all the objectives of this project will be achieved. First, GUI will be tested on offline mode with only one PC involved.

- The video window should be able to show the capturing from the webcam.
- The PC should be able to produce sound captured from the microphone.
- The text window should be able to display exactly what the user typed.

When all the feature has been tested, the experiment will proceed to second stage with two PC involved.

- The video window should be able to show the capturing from the webcam from both master and slave PC.

- The master PC should be able to produce sound captured from the slave PC's microphone.

- The slave PC should be able to produce sound captured from the master PC's microphone.

- The text window should be able to display text typed by both users.

- Delay from transmitting and receiving is expected but should be minimized as much as possible.

When all the feature has been tested, the experiment will proceed to third stage. With addition of face tracking, the experiment will test if the webcam will be able to move and follow the user's face at the slave PC.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Results

At this phase of the project, GUI has been developed as the first step of the project. During this stage, no network protocol is needed and only Java programming knowledge is used to develop the GUI.

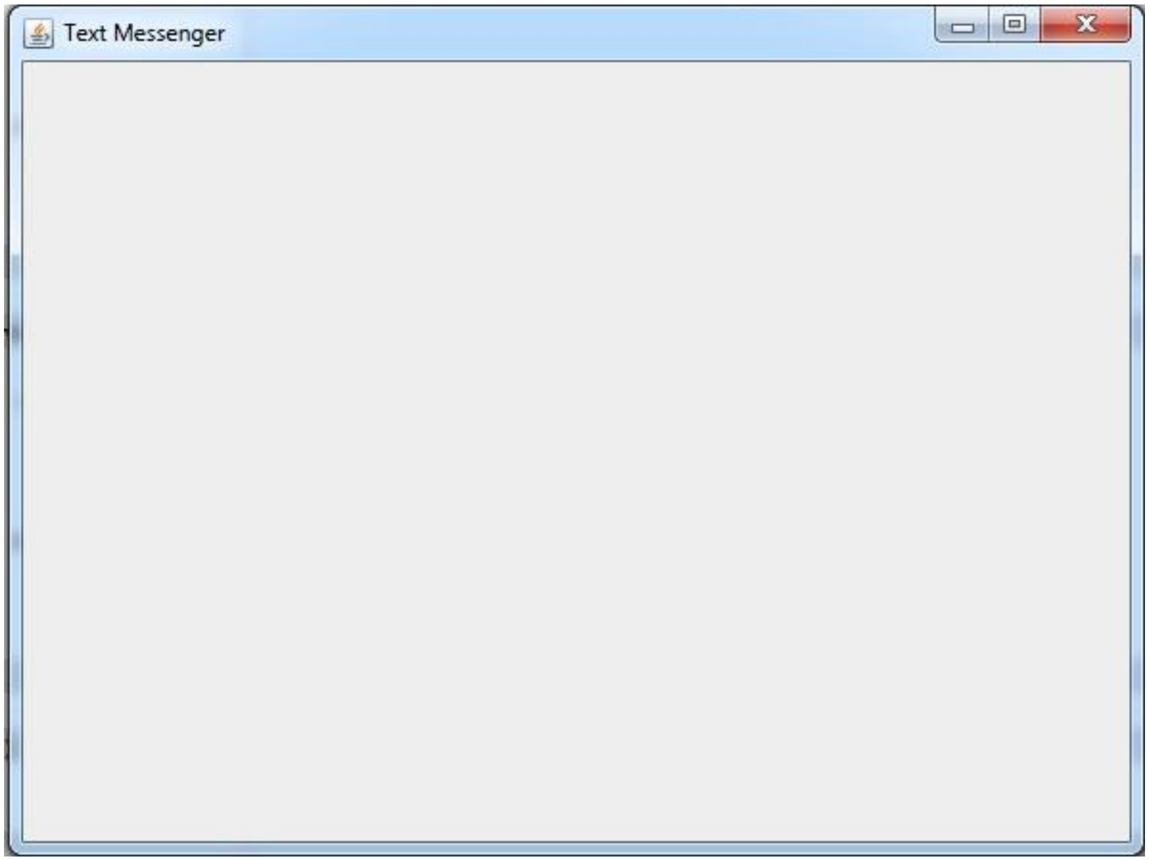


Figure 4: GUI for Text Messenger

A blank window is designed to have spaces for features of this project. The window is titled as "Text Messenger" and has an expandable space in the window.

Netbeans IDE 8.0.2 is used as the program to write Java language. Jframe is used since it is easier than the classical Java class.

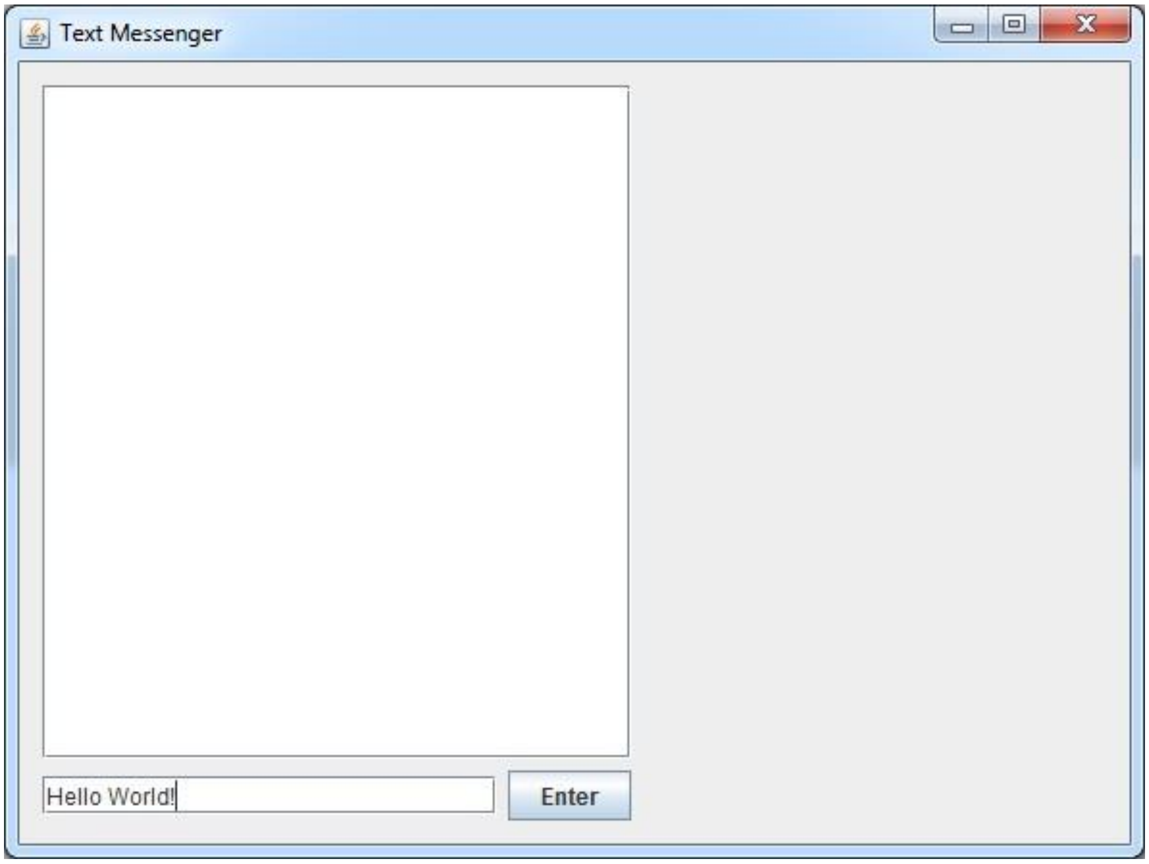


Figure 5: GUI for Text Messenger with textbox

As shown in figure 5, a textbox is added to the GUI. There is a textbox located at the bottom and a text area located above the textbox. A clickable button "Enter" is located next to the textbox. The textbox allows user to type any texts, numbers, or symbols. The maximum size is 2147483647 cases. After that user can either press enter on the keyboard or manually click the "Enter" button. The "Enter" button serves as an alternative way to press enter on the keyboard.

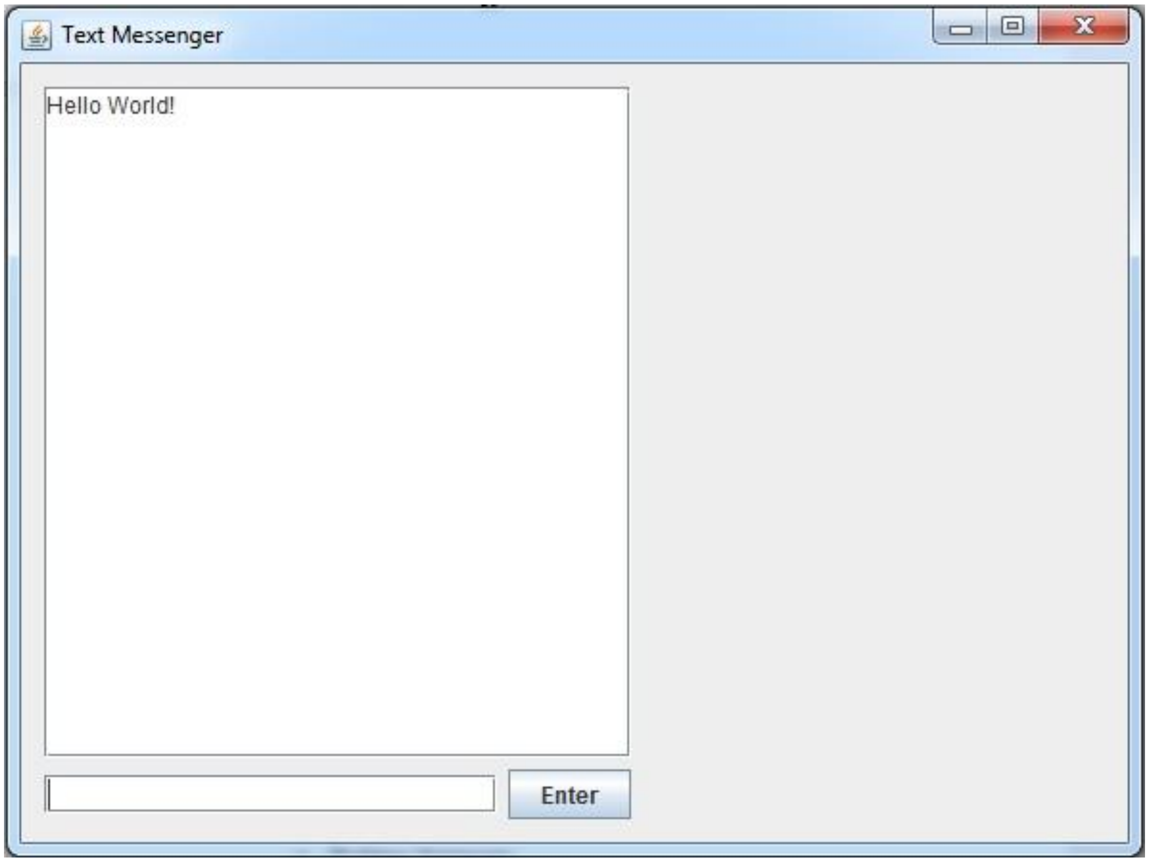
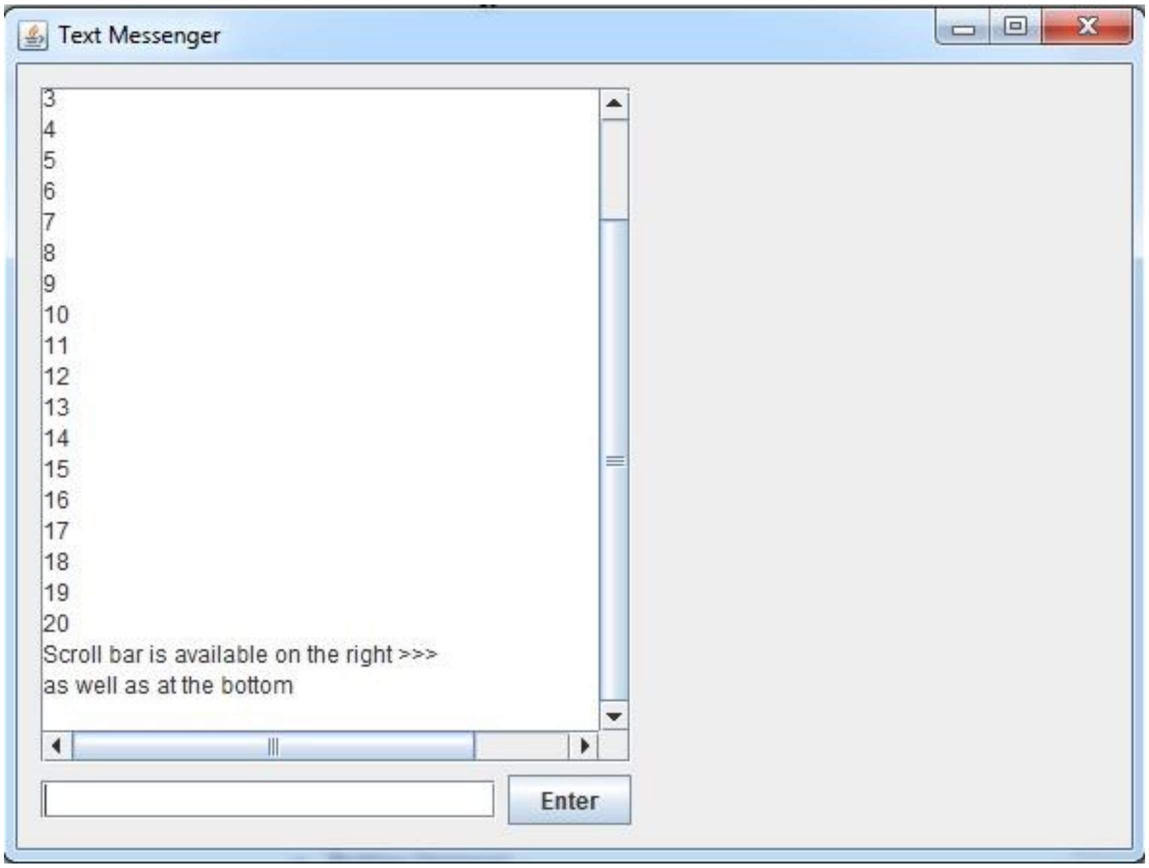


Figure 6: GUI for Text Messenger with text area

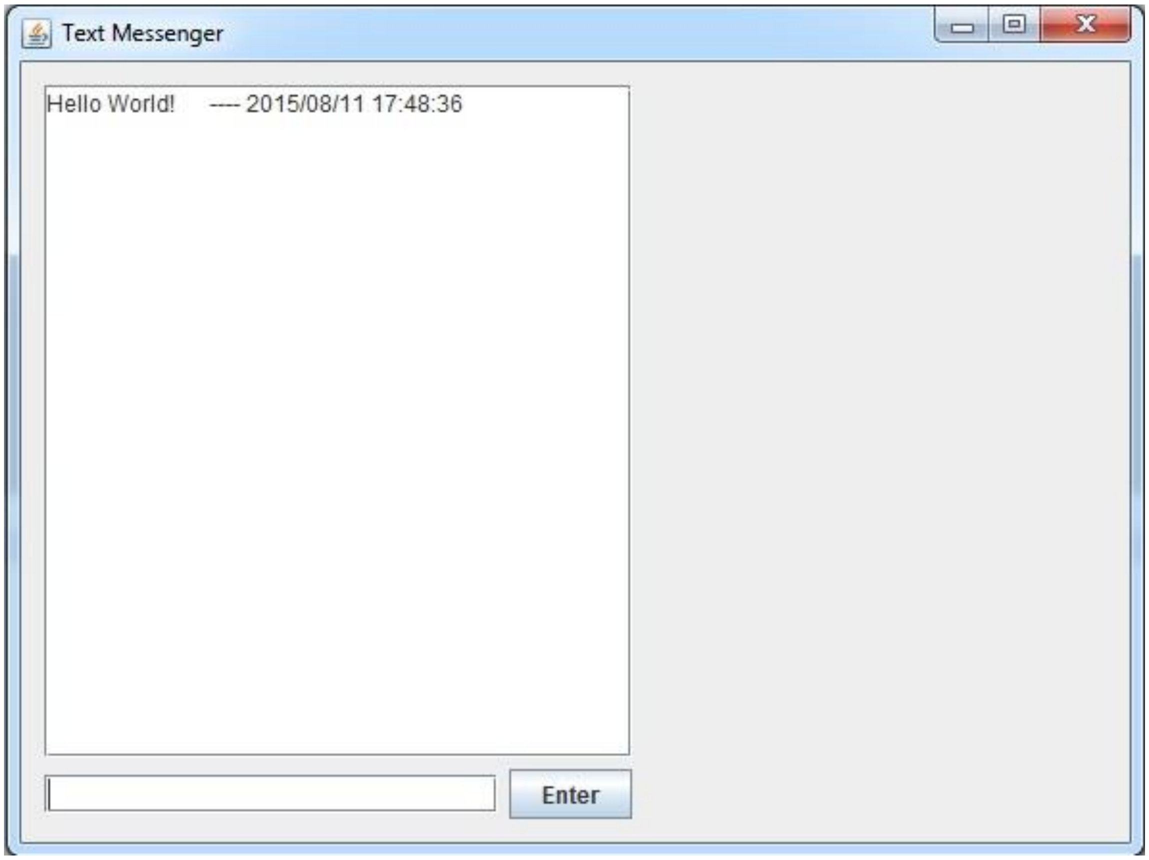
After "Enter" button is clicked, The text typed with be stored and displayed on the above text area as shown in figure 6. The text in the textbox will be cleared once "Enter" button is clicked.



```
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Scroll bar is available on the right >>>
as well as at the bottom
```

Figure 7: GUI for Text Messenger with scrollbar

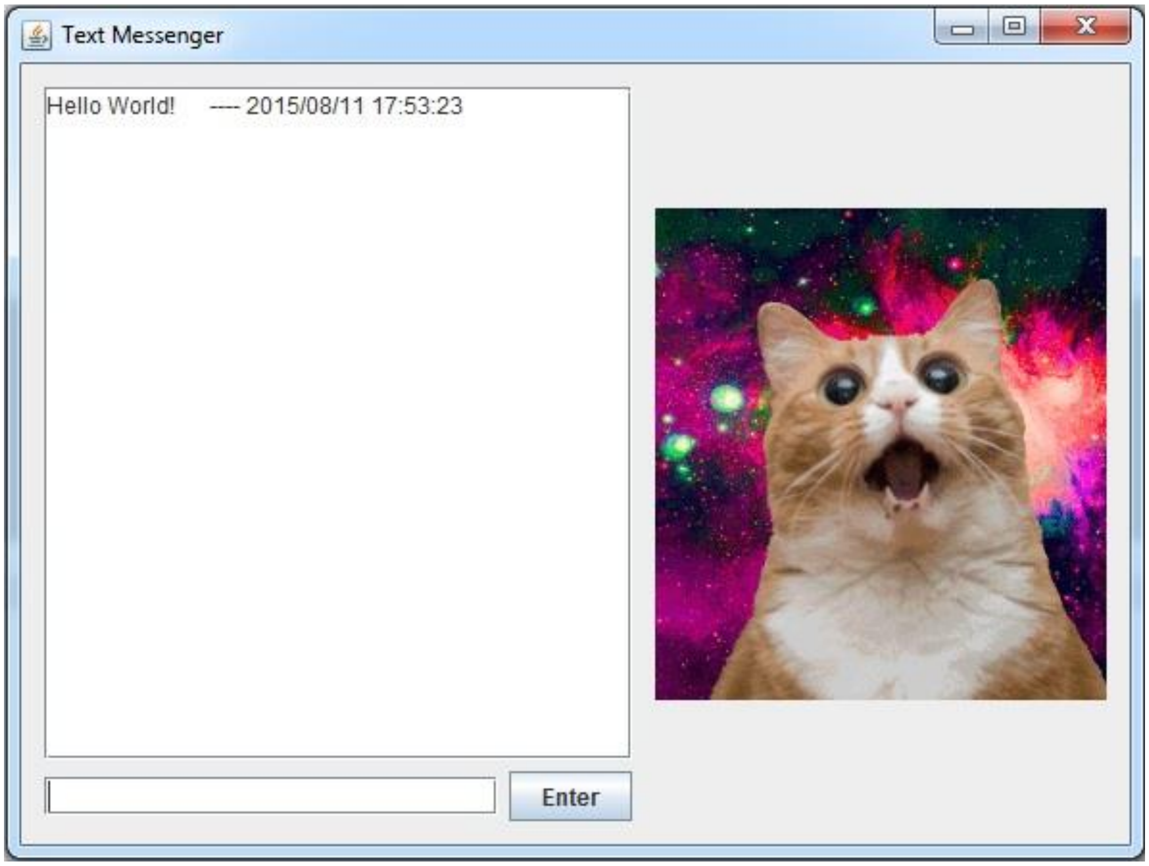
If another line of text is typed and entered into the textbox, the text area will display it in a new line. The message previously typed will still be stored in the text area. When the text area is fully occupied, scroll bar is available on the right and at the bottom so that user can scroll back to the previous message.



Hello World!   ---- 2015/08/11 17:48:36

Figure 8: GUI for Text Messenger with time display

In the stage of this project, an extra feature is added to the text area where the current date and time is displayed when text is entered. Figure 8 shows that "Hello World!" is typed at 11 August 2015, 5:48pm. This feature is added by using DateFormat available in Java.



Hello World!   ---- 2015/08/11 17:53:23

Figure 9: GUI for Text Messenger with profile picture

Figure 9 shows an addition of a profile picture on the right side of the window. The picture is added using JLabel which can be modified anytime.

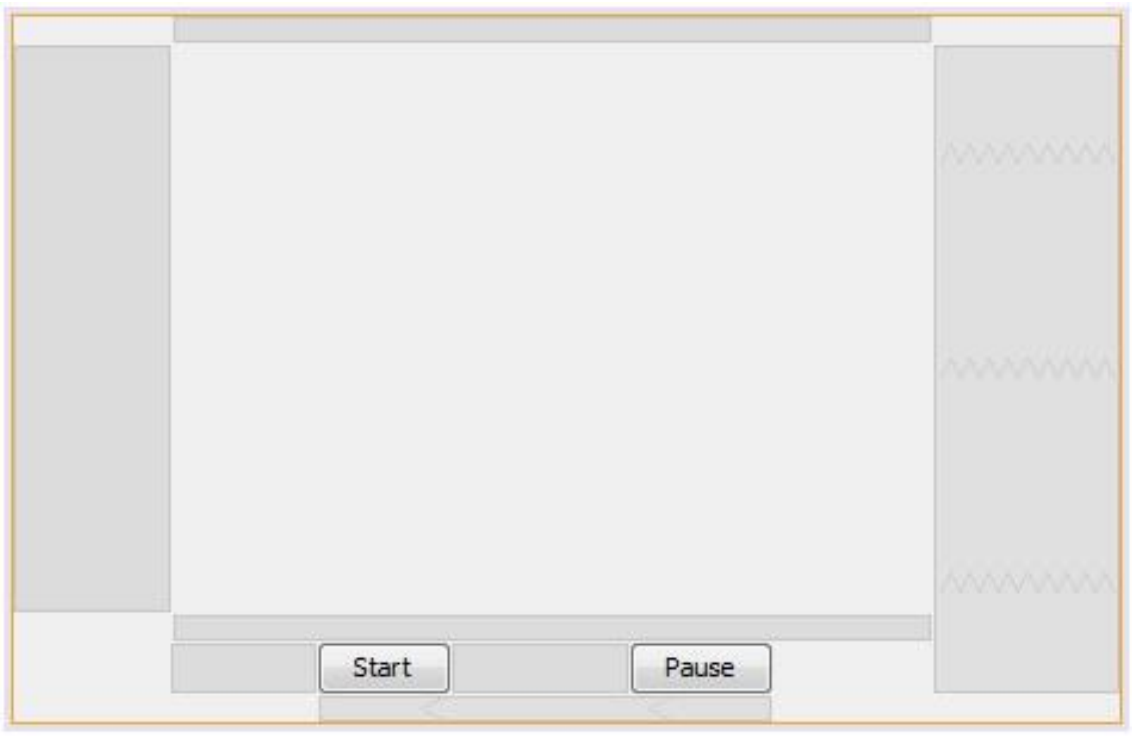


Figure 10: Webcam JForm Draft

After constructing a text messenger that able to send text messages and displaying picture. The project phase is moved to an addition of webcam utilization. The webcam is first drafted on a separate JForm. Figure 10 shows how the JForm looks like during the design stage. Start and Pause button is built using jButton, where start button will control when to start displaying webcam and pause button will control when to pause displaying webcam.

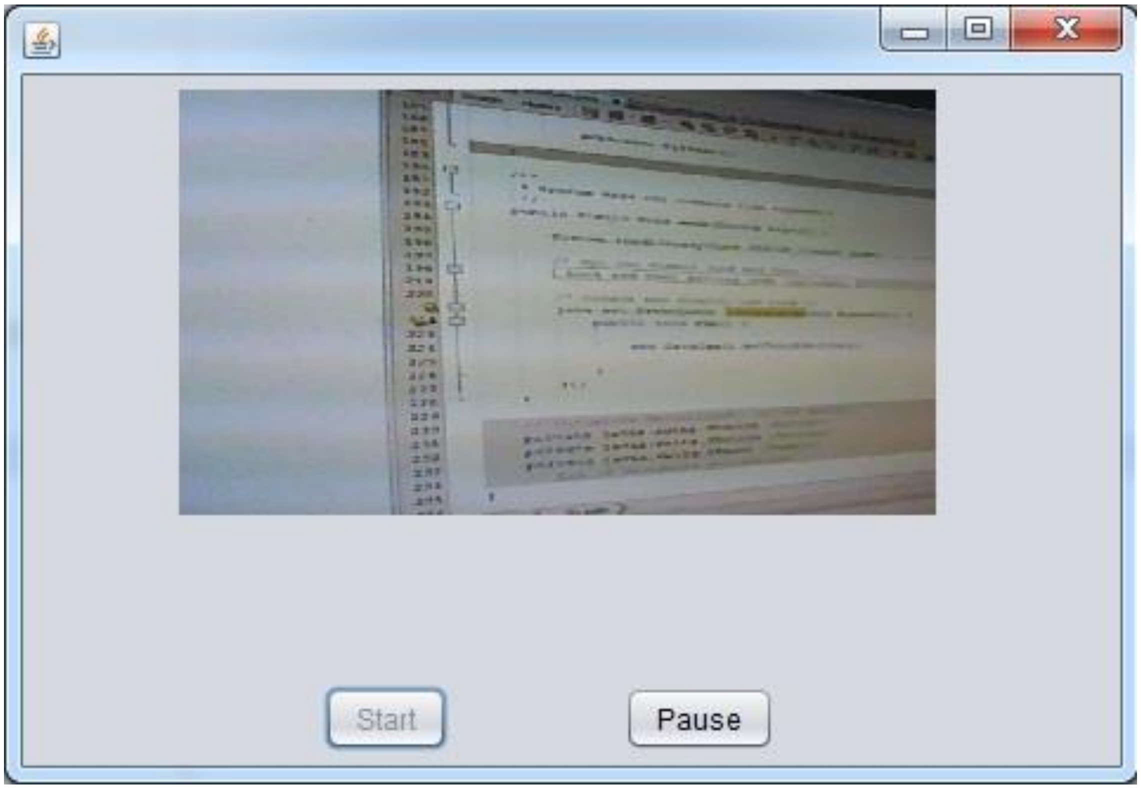


Figure 11: GUI for Webcam when Start Button is Pressed

Figure 11 shows when the start button is pressed, the window starts to display video captured by the webcam. The video changes in real-time with slight delay. However in this stage, the start button is disabled once pressed. User is able to press the pause button to pause displaying webcam.

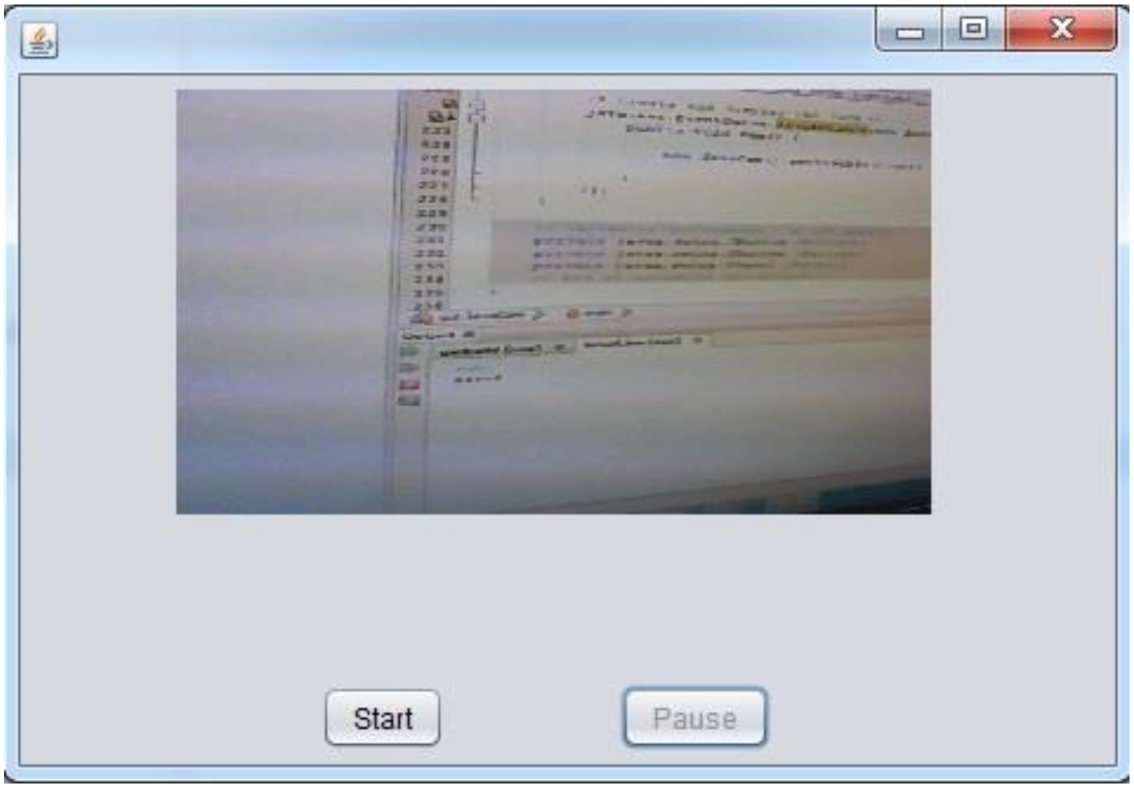


Figure 12: GUI for Webcam when Pause Button is Pressed

Figure 12 shows when the pause button is pressed, the windows stop displaying webcam. In this stage, the pause button is disabled and user have to press the start button in order to continue displaying the webcam.

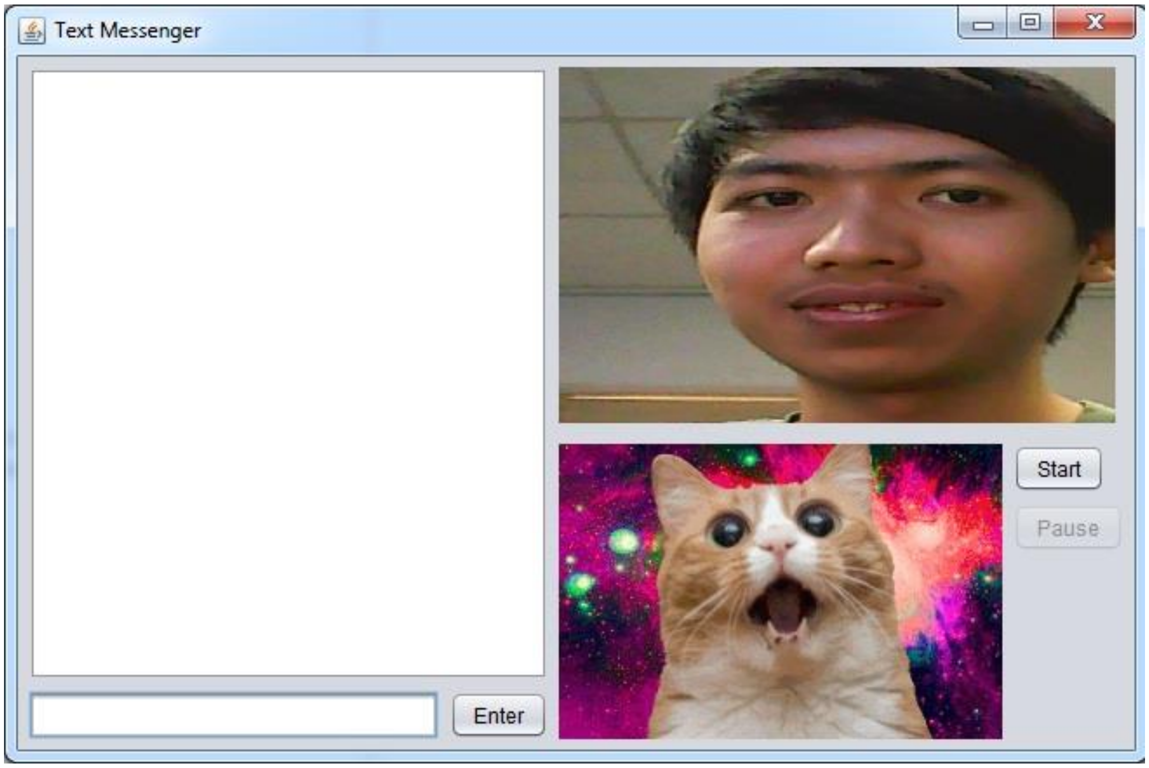
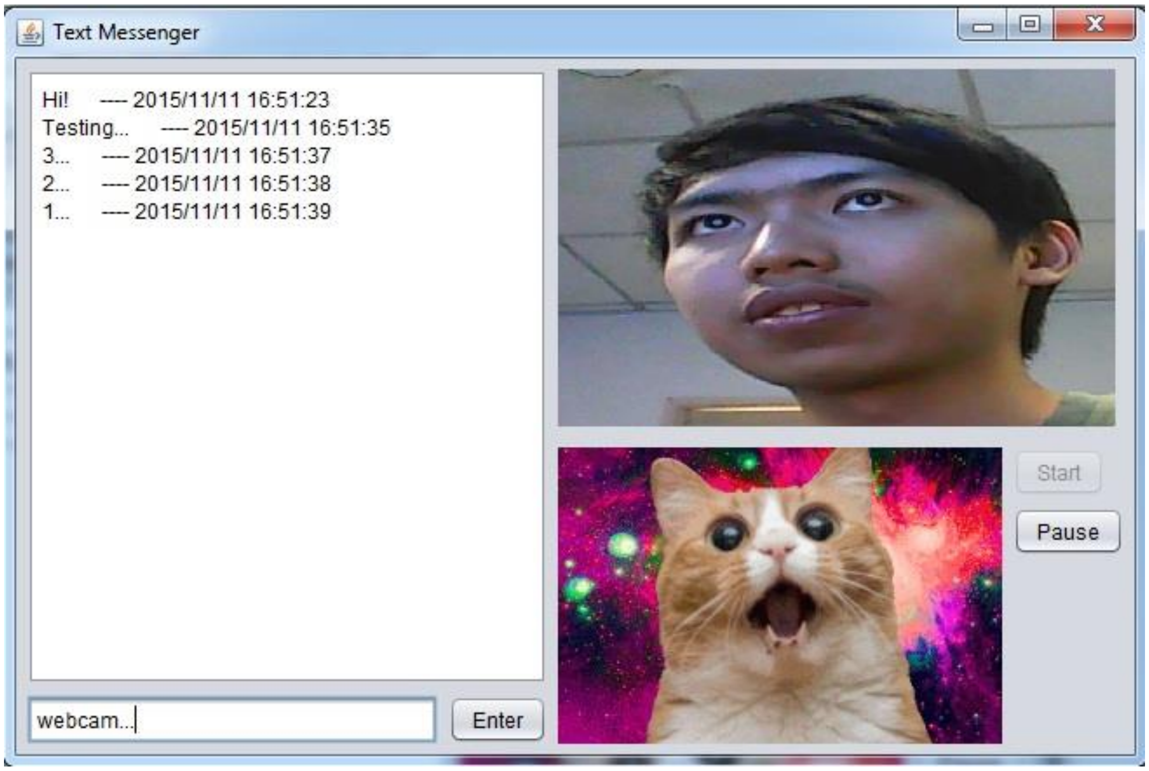


Figure 13: GUI for Text Messenger with Webcam

Figure 13 shows when the webcam structure is drafted successfully, the whole structure is being implemented into the original text messenger. The display picture is then moved to the bottom corner. the webcam display frame is set to 320 pixels x 240 pixels (width x height). Meanwhile the start button and pause button are moved to a side.



Hi! ---- 2015/11/11 16:51:23
Testing... ---- 2015/11/11 16:51:35
3... ---- 2015/11/11 16:51:37
2... ---- 2015/11/11 16:51:38
1... ---- 2015/11/11 16:51:39

Figure 14: GUI for Text Messenger with Webcam on test

The text messenger consists feature to display webcam, send text, display picture. Figure 14 shows the GUI is being tested and is ready to move on to the next stage.

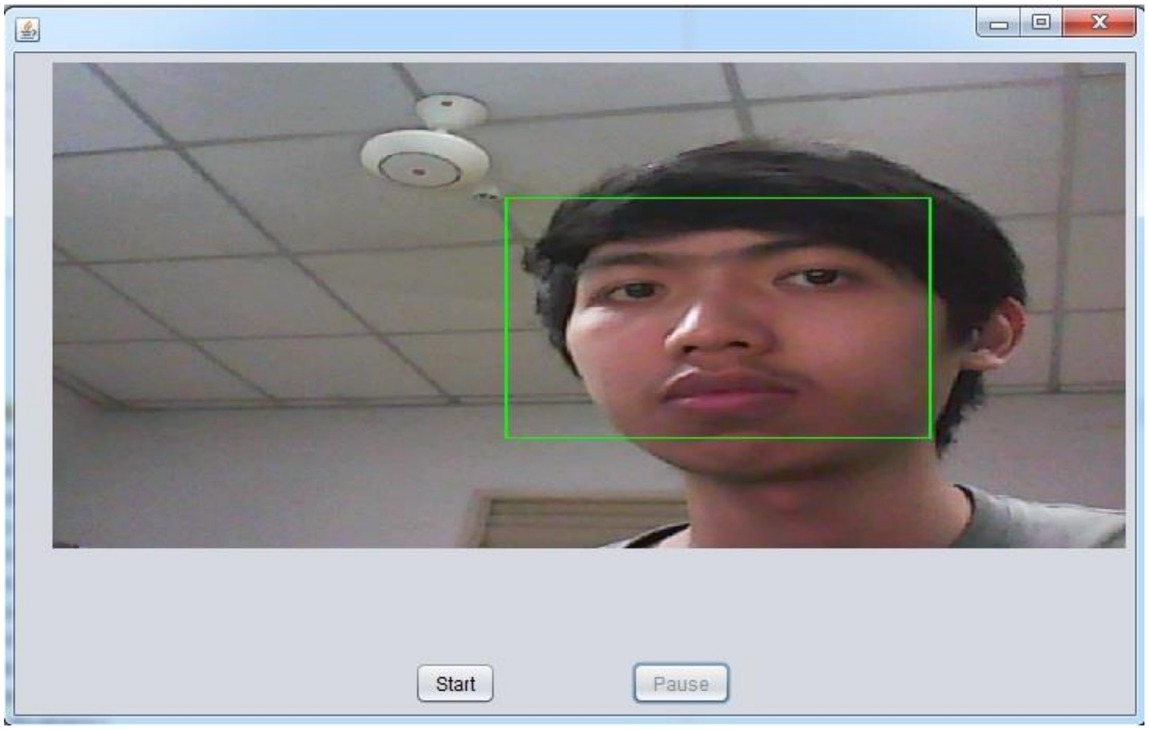


Figure 15: Webcam testing with Face Tracking

Figure 15 shows face tracking feature added into the webcam displaying. The program is able to form a green box surrounding all detected human face. The green box will follow the human face and react in real time.

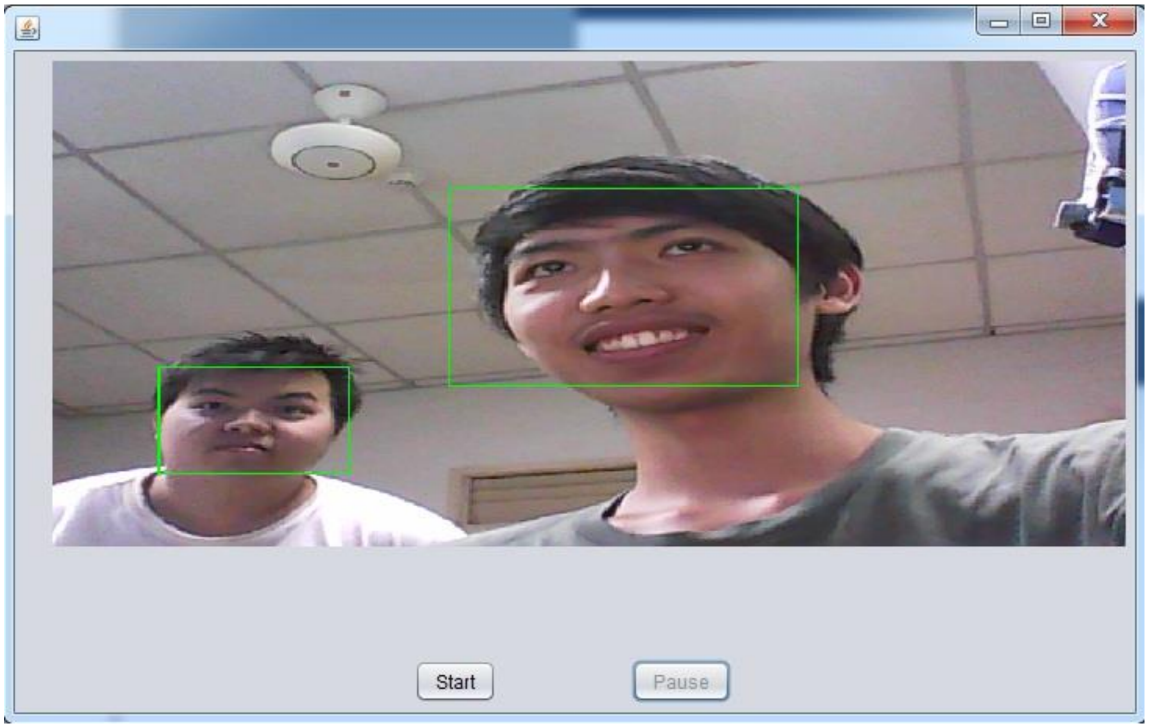


Figure 16: Face Tracking with two faces

The figure above shows when two human faces are in the boundary of the webcam, both faces will be surrounded by green box independently. However this will greatly affect the project when a moving webcam is implemented.

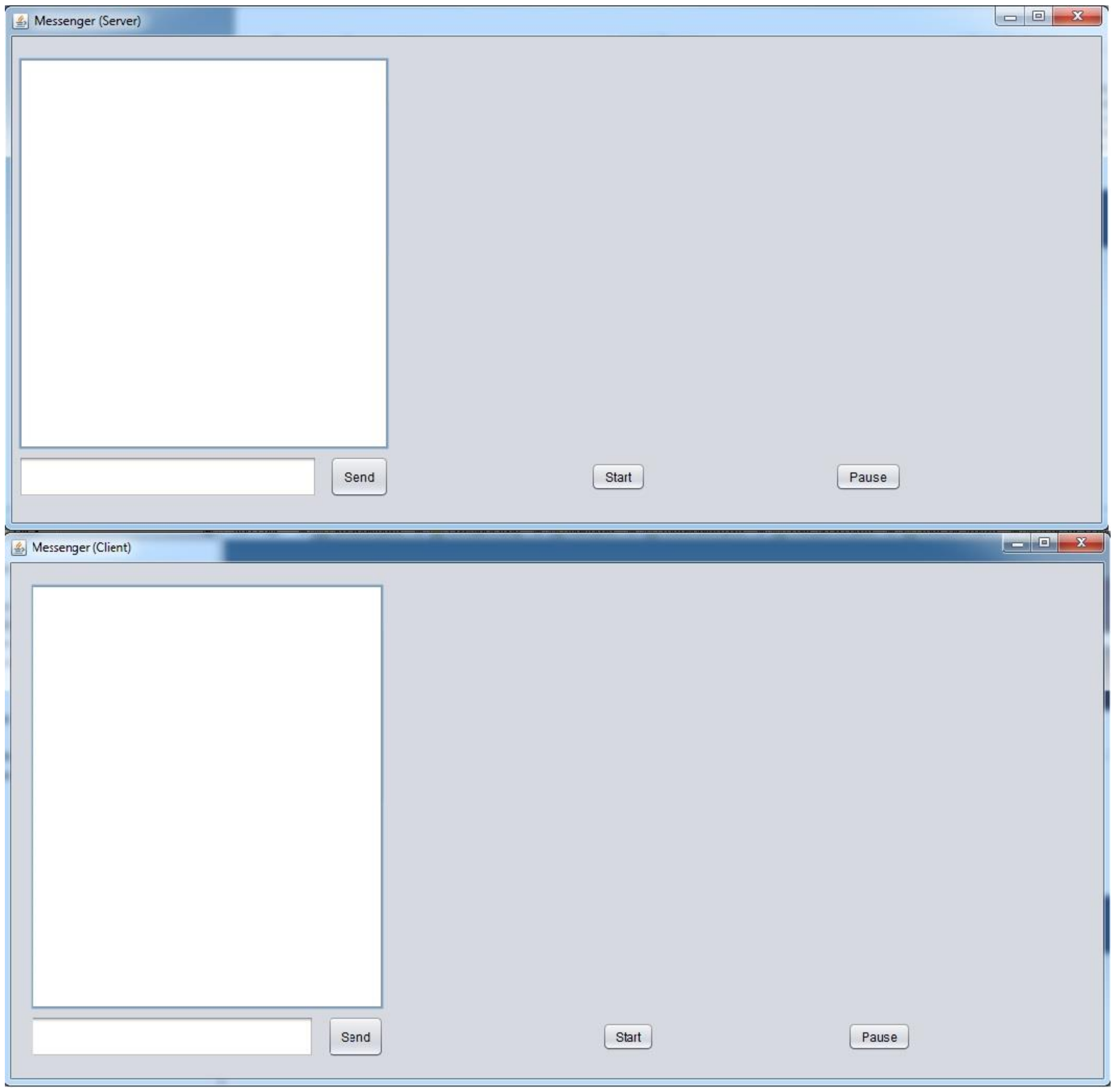
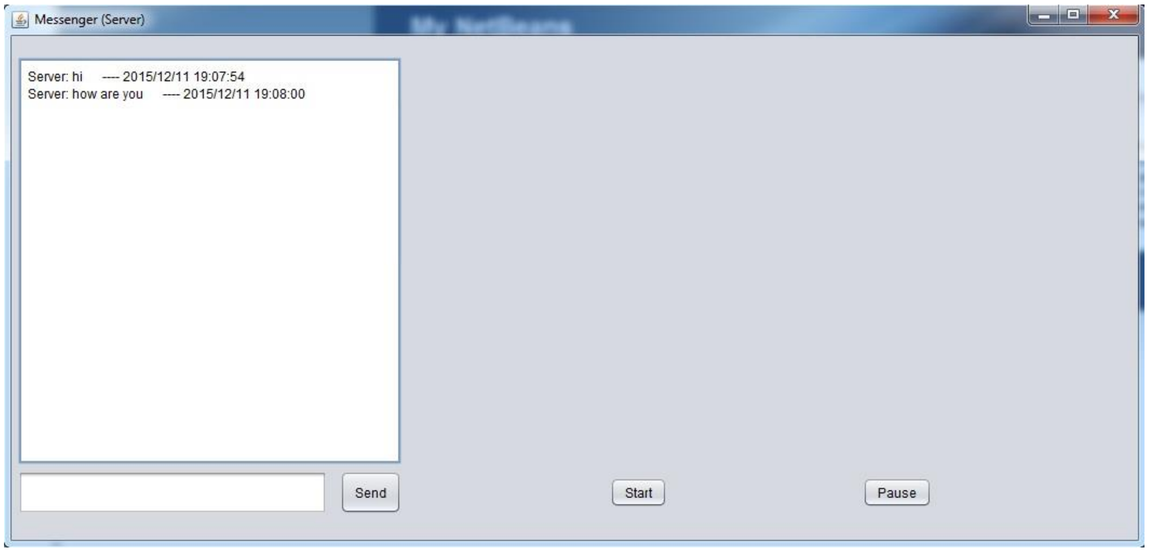


Figure 17: GUI of Messenger (Server and Client)

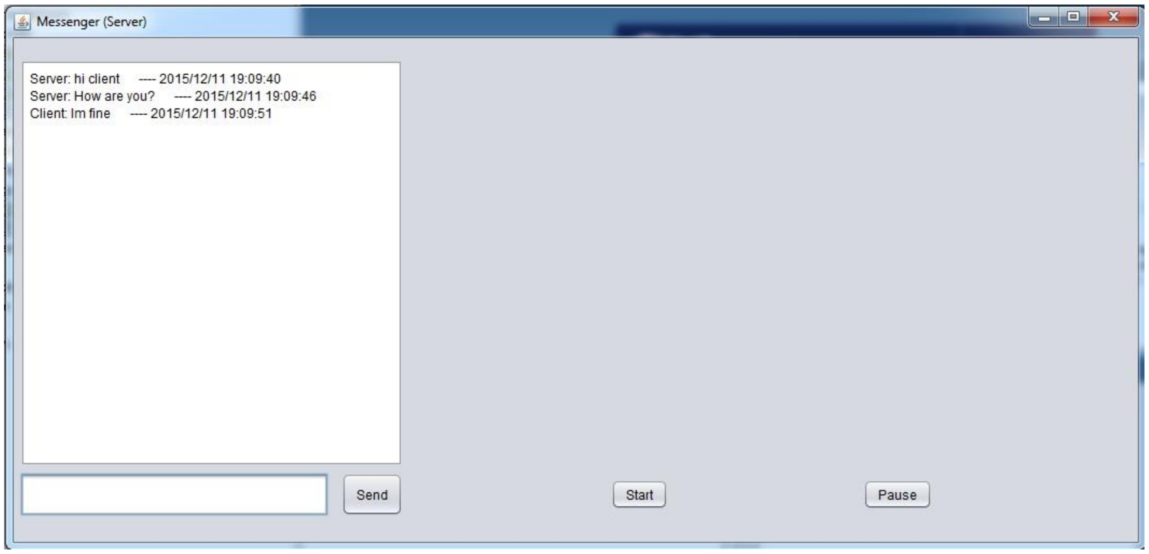
When majority of the feature are added into the GUI, The Messenger is split into server and client. Two messengers can be installed into different PC and connected to the same LAN. By using JAVA socket, they are able to send text to each other. The protocol used in this is TCP/IP.


Messenger (Server)

```
Server: hi      ---- 2015/12/11 19:07:54
Server: how are you      ---- 2015/12/11 19:08:00
```

Figure 18: GUI of Messenger with Server Header

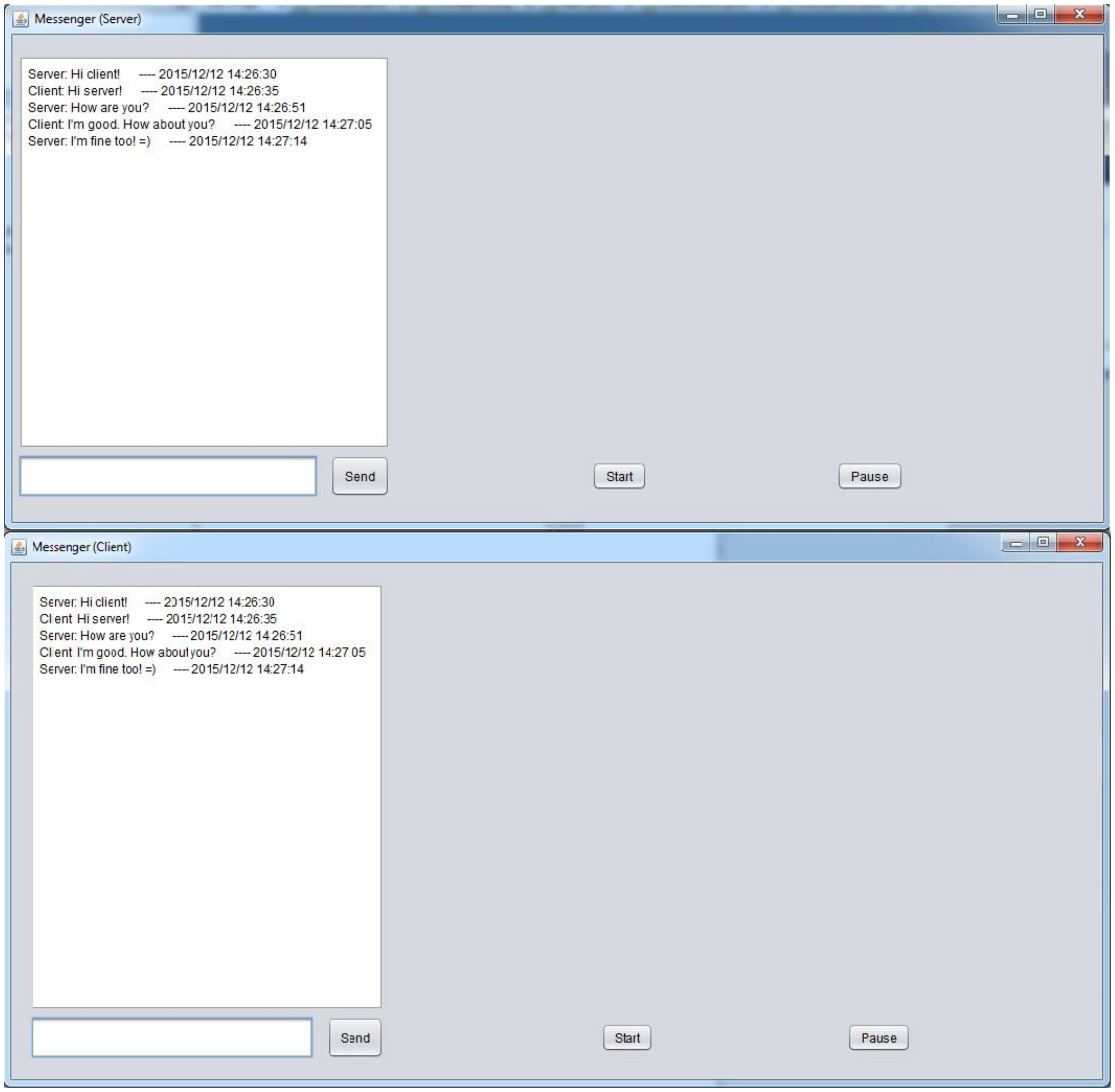
The GUI of Messenger is further developed to have more feature. When the user from the server site type a text, it will automatically add a header that display "Server: " into the text. This header will be displayed in both server and client site.


Messenger (Server)

```
Server: hi client      ---- 2015/12/11 19:09:40
Server: How are you?      ---- 2015/12/11 19:09:46
Client: Im fine      ---- 2015/12/11 19:09:51
```

Figure 19: GUI of Messenger with Server and Client Header

Figure 19 shows with the same as in Figure 18, when user from the client site send a text, it will appear in the server site with a client header. This will help to indicate which text are typed from which site.

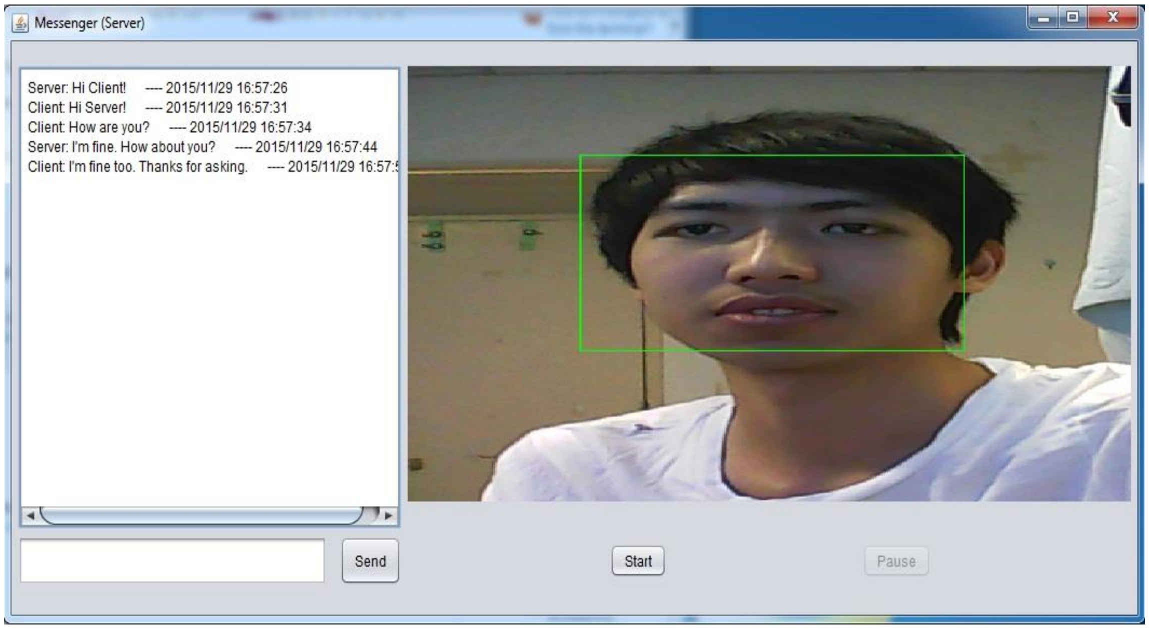


```
Server: Hi client!      ---- 2015/12/12 14:26:30
Client: Hi server!      ---- 2015/12/12 14:26:35
Server: How are you?      ---- 2015/12/12 14:26:51
Client: I'm good. How about you?      ---- 2015/12/12 14:27:05
Server: I'm fine too! =)      ---- 2015/12/12 14:27:14
```

```
Server: Hi client!      ---- 2015/12/12 14:26:30
Client: Hi server!      ---- 2015/12/12 14:26:35
Server: How are you?      ---- 2015/12/12 14:26:51
Client: I'm good. How about you?      ---- 2015/12/12 14:27:05
Server: I'm fine too! =)      ---- 2015/12/12 14:27:14
```

Figure 20: Messenger (Server and Client) with Header

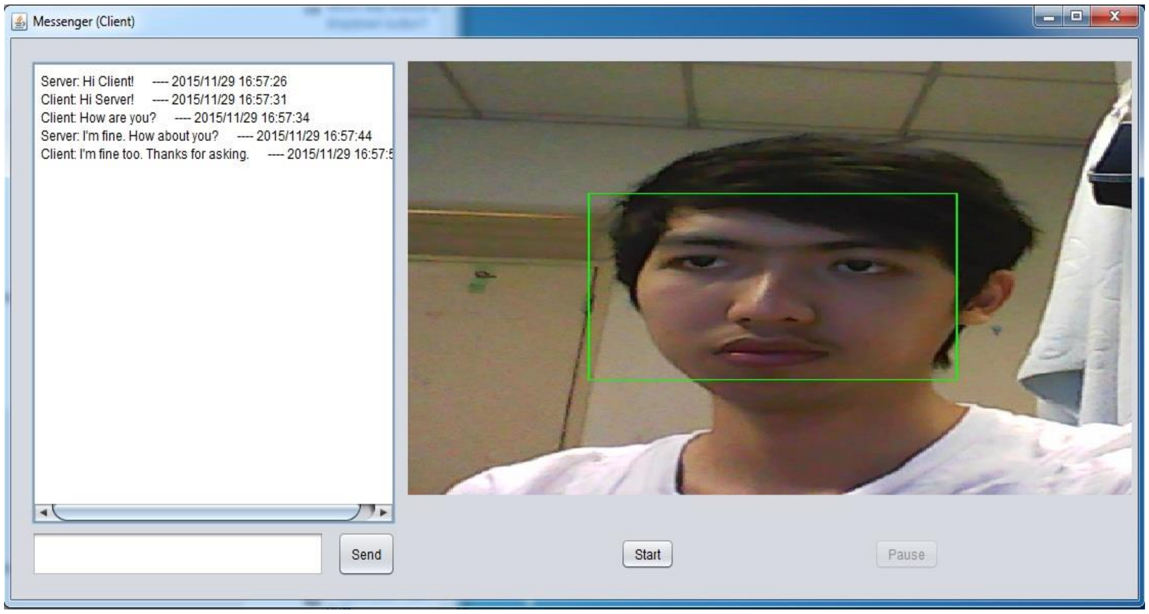
At the end of the header feature, figure 20 shows how both messengers works with the header. Server and client headers are able to differentiate all the texts on the chat area, whether which text are typed and sent from which site.



Server: Hi Client! ---- 2015/11/29 16:57:26
Client: Hi Server! ---- 2015/11/29 16:57:31
Client: How are you? ---- 2015/11/29 16:57:34
Server: I'm fine. How about you? ---- 2015/11/29 16:57:44
Client: I'm fine too. Thanks for asking. ---- 2015/11/29 16:57:5

Figure 21: GUI for Messenger (Server)

After finishing all the feature regarding texting, the final end product of this project, showing in figure 21, has all the features of sending text, displaying webcam and face tracking. This messenger is for the server.



Server: Hi Client! ---- 2015/11/29 16:57:26
Client: Hi Server! ---- 2015/11/29 16:57:31
Client: How are you? ---- 2015/11/29 16:57:34
Server: I'm fine. How about you? ---- 2015/11/29 16:57:44
Client: I'm fine too. Thanks for asking. ---- 2015/11/29 16:57:5

Figure 22: GUI of Messenger (Client)

Figure 22 shows the final end product of Messenger at the client site. This concludes all the feature tested and merged into one single messenger. It has the feature of sending text, displaying webcam , and face tracking.

4.2 Discussion

After going through the methodology of the project, the project has produced a java-based video conferencing tool with face tracking, named as Messenger. Since this tool is partially completed, it contains some features that fulfill the objectives of this project.

The project is done by using Netbeans IDE. All the codes are written in Java language. Codes are attached in Appendix. The project consists of two separate java codes which one is for server and another one is for client. The project also contain a Haar features lookup table for the face tracking feature.

This project uses a few libraries which are all available on the internet as open sources. For text sending, java socket is used. Java io and OpenCV are used for webcam video streaming and face tracking. OpenCV uses Viola-Jones algorithm in its library.

Up to the current phase of this project. It still does not involve any extra hardware except for a normal domestic PC with webcam. Any webcam will be sufficient for the program to work. Both PCs need to be connected under the same connection. With java socket, the program can be modified to connect through any port number and IP address.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Videoconferencing has been proven to be efficient in communicating between employees. Advancement in technology has allowed videoconferencing tool to have a lots of utility and user-friendly based GUI [10]. However current videoconferencing software only display video from the client without the capability to control the client's webcam. In order to have clear and accurate video display, the client is required to stay still on the webcam and the client's movement is restricted. This project will focus on designing a Java based small-scaled videoconferencing software which allow user to send text, display audio and video, with additional feature of controlling client's webcam using face detection.

5.2 Recommendation

This project does not focus on multipoint videoconferencing which limited the conference to have only two users. Further studies and development may be applied to the project to have multipoint videoconferencing, where multiple slave PCs are connected to one master PC.

This project will try to minimize the delay while sending data from PC to PC. However it does not focus on exact real-time communication with delay free. With more research on network protocol might help to achieve the delay free communication.

Reference

[1] D. Fisher. (2008, 22 June). *Edison's Telephonoscope*. Available: http://www.terramedia.co.uk/Chronomedia/years/Edison_Telephonoscope.htm

[2] Q. S.-. Fraser. (1995, 22 June). *The Trojan Room Coffee Pot*. Available: http://www.cl.cam.ac.uk/coffee/qsf/coffee.html

[3] N. Malasne, F. Yang, and M. Paindavoine, "Real-time face tracking and recognition for video conferencing," in *International Symposium on Optical Science and Technology*, 2001, pp. 357-365.

[4] S. Paschalakis and M. Bober, "Real-time face detection and tracking for mobile videoconferencing," *Real-Time Imaging,* vol. 10, pp. 81-94, 4// 2004.

[5] M. K. Littman, "Videoconferencing as a communications enhancement," *The Journal of Academic Librarianship,* vol. 21, pp. 359-364, 9// 1995.

[6] K. Salah, P. Calyam, and M. I. Buhari, "Assessing readiness of IP networks to support desktop videoconferencing using OPNET," *Journal of Network and Computer Applications,* vol. 31, pp. 921-943, 11// 2008.

[7] M. R. Civanlar, Ö. Özkasap, and T. Çelebi, "Peer-to-peer multipoint videoconferencing on the Internet," *Signal Processing: Image Communication,* vol. 20, pp. 743-754, 9// 2005.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-511-I-518 vol.1.

[9] D. Norman, *The Design of Everyday Things*. New York: Basic Books, 2002.

[10] J. M. Denstadli, M. Gripsrud, R. Hjorthol, and T. E. Julsrud, "Videoconferencing and business air travel: Do new technologies produce new interaction patterns?," *Transportation Research Part C: Emerging Technologies,* vol. 29, pp. 1-13, 4// 2013.

CHAPTER 6

APPENDICES

APPENDIX 1. SOURCE CODE FOR MESSENGER (SERVER)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package chats;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.imageio.ImageIO;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.objdetect.CascadeClassifier;

/**
 *
 * @author Jia Jie
 */
public class chat_server extends javax.swing.JFrame {

    /**
     * Creates new form chat_server
     */
    static ServerSocket ss;
    static Socket s;
    static DataInputStream din;
    static DataOutputStream dout;

    private DaemonThread myThread = null;
    int count = 0;
    VideoCapture webSource = null;
    Mat frame = new Mat();
    MatOfByte mem = new MatOfByte();
    CascadeClassifier faceDetector = new
```

```
CascadeClassifier(chat_server.class.getResource("haarcascade_frontalface_alt.xml").getPath().substring(1));
   MatOfRect faceDetections = new MatOfRect();
///

   class DaemonThread implements Runnable {

      protected volatile boolean runnable = false;

      @Override
      public void run() {
         synchronized (this) {
            while (runnable) {
               if (webSource.grab()) {
                  try {
                     webSource.retrieve(frame);
                     Graphics g = jPanel1.getGraphics();
                     faceDetector.detectMultiScale(frame, faceDetections);
                     for (Rect rect : faceDetections.toArray()) {
                        // System.out.println("ttt");
                        Core.rectangle(frame, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height),
                              new Scalar(0, 255, 0));
                     }
                     Highgui.imencode(".bmp", frame, mem);
                     Image im = ImageIO.read(new ByteArrayInputStream(mem.toArray()));
                     BufferedImage buff = (BufferedImage) im;
                     if (g.drawImage(buff, 0, 0, getWidth(), getHeight() - 150, 0, 0, buff.getWidth(), buff.getHeight(), null)) {
                        if (runnable == false) {
                           System.out.println("Paused ..... ");
                           this.wait();
                        }
                     }
                  } catch (Exception ex) {
                     System.out.println("Error");
                  }
               }
            }
         }
      }
   }

   public chat_server() {
      initComponents();
      System.out.println(chat_server.class.getResource("haarcascade_frontalface_alt.xml").getPath().substring(1));
   }

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

      jScrollPane1 = new javax.swing.JScrollPane();
      msg_area = new javax.swing.JTextArea();
      msg_text = new javax.swing.JTextField();
      msg_send = new javax.swing.JButton();
      jPanel1 = new javax.swing.JPanel();
      jButton1 = new javax.swing.JButton();
      jButton2 = new javax.swing.JButton();

      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
      setTitle("Messenger (Server)");

      msg_area.setEditable(false);
      msg_area.setColumns(20);
      msg_area.setRows(5);
      jScrollPane1.setViewportView(msg_area);
```

```
        msg_text.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                msg_textActionPerformed(evt);
            }
        });
        msg_text.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                msg_textKeyPressed(evt);
            }
        });

        msg_send.setText("Send");
        msg_send.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                msg_sendActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 670, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 376, Short.MAX_VALUE)
        );

        jButton1.setText("Start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Pause");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE, 286,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(msg_send))
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 355,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(193, 193, 193)
                        .addComponent(jButton1)
                        .addGap(181, 181, 181)
                        .addComponent(jButton2)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(20, 20, 20)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE)
                    .addComponent(jScrollPane1))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE, 40, 
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1)
                    .addComponent(jButton2)
                    .addComponent(msg_send, javax.swing.GroupLayout.PREFERRED_SIZE, 40, 
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(24, 24, 24))
        );

        pack();
    }// </editor-fold>

    private void msg_textActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void msg_sendActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            String msgout = "";
            String old_msgin = null;
            msgout = msg_text.getText().trim();
            msg_text.setText(null);
            old_msgin = msg_area.getText().trim();
            msg_area.setText(old_msgin + "\nServer: " + msgout);
            DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
            //get current date time with Date()
            Date date = new Date();
            msg_area.append("      ---- " + dateFormat.format(date));
            dout.writeUTF(msgout); //sending the sever message to the client.

        } catch (Exception e) {
            //handle the exception here
        }

    }

    private void msg_textKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        int key = evt.getKeyCode();
        if (key == KeyEvent.VK_ENTER) {
            try {
                String msgout = "";

                String old_msgin = null;
                msgout = msg_text.getText().trim();
                msg_text.setText(null);
                old_msgin = msg_area.getText().trim();
                msg_area.setText(old_msgin + "\nServer: " + msgout);
                DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                //get current date time with Date()
                Date date = new Date();
                msg_area.append("      ---- " + dateFormat.format(date));
                dout.writeUTF(msgout); //sending the sever message to the client.

            } catch (Exception e) {
                //handle the exception here
            }
        }
    }

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

        webSource = new VideoCapture(0); // video capture from default cam
        myThread = new DaemonThread(); //create object of threat class
        Thread t = new Thread(myThread);
        t.setDaemon(true);
        myThread.runnable = true;
        t.start();                  //start thrad
        jButton1.setEnabled(false);  // deactivate start button
        jButton2.setEnabled(true);  //  activate stop button

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        myThread.runnable = false;            // stop thread
        jButton2.setEnabled(false);   // activate start button
        jButton1.setEnabled(true);     // deactivate stop button

        webSource.release();  // stop caturing fron cam

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(chat_server.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(chat_server.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(chat_server.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(chat_server.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new chat_server().setVisible(true);
            }
        });
        String msgin = "";
        String old_msgin = null;
        //int first_line = 0;
        try {

            ss = new ServerSocket(1220); //server starts at 1201 port number
            s = ss.accept(); // now server will accepts the connections.

            din = new DataInputStream(s.getInputStream());
            dout = new DataOutputStream(s.getOutputStream());
            int first_line = 1;
            while (!msgin.equals("exit")) {
                msgin = din.readUTF();
                old_msgin = msg_area.getText().trim();
                msg_area.setText(null);
                if (first_line == 0) {
                    msg_area.setText(old_msgin + "Client: " + msgin); // displaying the messge.. from client.
                } else {
                    msg_area.setText(old_msgin + "\nClient: " + msgin); // displaying the messge.. from client.
                }
                first_line = first_line + 1;
                DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                //get current date time with Date()
                Date date = new Date();
                msg_area.append("      ---- " + dateFormat.format(date));
            }

        } catch (Exception e) {

        }

    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private static javax.swing.JTextArea msg_area;
    private javax.swing.JButton msg_send;
    private javax.swing.JTextField msg_text;
    // End of variables declaration
}
```

APPENDIX 2. SOURCE CODE FOR MESSENGER (CLIENT)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package chats;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.imageio.ImageIO;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.objdetect.CascadeClassifier;

/**
 *
 * @author Jia Jie
 */
public class chat_client extends javax.swing.JFrame {

    /**
     * Creates new form chat_client
     */
    static Socket s;
    static DataInputStream din;
    static DataOutputStream dout;

     private DaemonThread myThread = null;
    int count = 0;
    VideoCapture webSource = null;
    Mat frame = new Mat();
    MatOfByte mem = new MatOfByte();
    CascadeClassifier faceDetector = new
CascadeClassifier(chat_server.class.getResource("haarcascade_frontalface_alt.xml").getPath().substring(1));
    MatOfRect faceDetections = new MatOfRect();
///

    class DaemonThread implements Runnable {

        protected volatile boolean runnable = false;

        @Override
        public void run() {
            synchronized (this) {
                while (runnable) {
                    if (webSource.grab()) {
                        try {
                            webSource.retrieve(frame);
                            Graphics g = jPanel1.getGraphics();
                            faceDetector.detectMultiScale(frame, faceDetections);
                            for (Rect rect : faceDetections.toArray()) {
```

```
                    // System.out.println("ttt");
                    Core.rectangle(frame, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height),
                            new Scalar(0, 255, 0));
                }
                Highgui.imencode(".bmp", frame, mem);
                Image im = ImageIO.read(new ByteArrayInputStream(mem.toArray()));
                BufferedImage buff = (BufferedImage) im;
                if (g.drawImage(buff, 0, 0, getWidth(), getHeight() - 150, 0, 0, buff.getWidth(), buff.getHeight(), null)) {
                    if (runnable == false) {
                        System.out.println("Paused ..... ");
                        this.wait();
                    }
                }
            } catch (Exception ex) {
                System.out.println("Error");
            }
        }
      }
    }
  }
}

public chat_client() {
    initComponents();
    System.out.println(chat_server.class.getResource("haarcascade_frontalface_alt.xml").getPath().substring(1));
}

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

    jScrollPane1 = new javax.swing.JScrollPane();
    msg_area = new javax.swing.JTextArea();
    msg_text = new javax.swing.JTextField();
    msg_send = new javax.swing.JButton();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Messenger (Client)");

    msg_area.setEditable(false);
    msg_area.setColumns(20);
    msg_area.setRows(5);
    jScrollPane1.setViewportView(msg_area);

    msg_text.addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyPressed(java.awt.event.KeyEvent evt) {
            msg_textKeyPressed(evt);
        }
    });

    msg_send.setText("Send");
    msg_send.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            msg_sendActionPerformed(evt);
        }
    });

    jButton1.setText("Start");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
        });

        jButton2.setText("Pause");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 708, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 0, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(20, 20, 20)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jScrollPane1)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE, 286,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(13, 13, 13)
                        .addComponent(msg_send)))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton1)
                        .addGap(195, 195, 195)
                        .addComponent(jButton2)
                        .addGap(194, 194, 194))
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(20, 20, 20)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(msg_send, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap(20, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>
```

```
private void msg_sendActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String msgout = "";
        String old_msgin = null;
        msgout = msg_text.getText().trim();
        old_msgin = msg_area.getText().trim();
        msg_text.setText(null);
        msg_area.setText(old_msgin + "\nClient: " + msgout);
        DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        //get current date time with Date()
        Date date = new Date();
        msg_area.append("      ---- " + dateFormat.format(date));
        msg_text.setText(null);
        dout.writeUTF(msgout);


    } catch (Exception e) {
        //hadle exceptions here..
    }
}

private void msg_textKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    int key = evt.getKeyCode();
    if (key == KeyEvent.VK_ENTER) {
        try {
        String msgout = "";
        String old_msgin = null;
        msgout = msg_text.getText().trim();
        old_msgin = msg_area.getText().trim();
        msg_text.setText(null);
        msg_area.setText(old_msgin + "\nClient: " + msgout);
        DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        //get current date time with Date()
        Date date = new Date();
        msg_area.append("      ---- " + dateFormat.format(date));
        msg_text.setText(null);
        dout.writeUTF(msgout);


        } catch (Exception e) {
            //hadle exceptions here..
        }
    }
}

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

    webSource = new VideoCapture(0); // video capture from default cam
    myThread = new DaemonThread(); //create object of threat class
    Thread t = new Thread(myThread);
    t.setDaemon(true);
    myThread.runnable = true;
    t.start();                 //start thrad
    jButton1.setEnabled(false);  // deactivate start button
    jButton2.setEnabled(true);  //  activate stop button
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    myThread.runnable = false;            // stop thread
    jButton2.setEnabled(false);   // activate start button
    jButton1.setEnabled(true);     // deactivate stop button

    webSource.release();  // stop caturing fron cam
}

/**
 * @param args the command line arguments
 */
```

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

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

        try {
            s = new Socket("127.0.0.1", 1220); //here the ip address is local addr. because im running the client and server at same
computer.
            din = new DataInputStream(s.getInputStream());
            dout = new DataOutputStream(s.getOutputStream());
            String msgin = "";
            String old_msgin = null;

            int first_line = 1;
            while (!msgin.equals("exit")) {
                msgin = din.readUTF();
                old_msgin = msg_area.getText().trim();
                msg_area.setText(null);
                if (first_line == 0){
                    msg_area.setText(old_msgin + "Server: " + msgin);
                }
                else{
                    msg_area.setText(old_msgin + "\nServer: " + msgin);
                }
                first_line = first_line + 1;
                DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                //get current date time with Date()
                Date date = new Date();
                msg_area.append("      ---- " + dateFormat.format(date));

            }
        } catch (Exception e) {

        }
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private static javax.swing.JTextArea msg_area;
    private javax.swing.JButton msg_send;
    private javax.swing.JTextField msg_text;
    // End of variables declaration
}
```