**NETWORKING USING SOCKETS:**

**YOGESHWARAN R**

**1ST IMPLEMENTATION:**

**REGISTER USER USING SOCKETS**

**CLIENT SIDE:**

package WebHost;

import TOOLS1.HintAreaField;

import TOOLS1.HintTextField;

import TOOLS1.PasswordPanel;

import TOOLS1.TemplateClass;

import TOOLS1.validateOps;

import com.github.lgooddatepicker.components.DatePicker;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.font.TextAttribute;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.io.Serializable;

import java.net.InetAddress;

import java.net.Socket;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

import javax.swing.\*;

public class Register extends JFrame implements ActionListener, Serializable {

String userType, formType;

static JLabel Signup, fName, lName, Age, Num, Mail, occupation, DateOfBirth, Address, Gender, pass, cPass, healthLabel, dynamicLabel;

static JTextField fNameField, lNameField, AgeField, NumberField, MailField, healthField, dynamicField;

PasswordPanel passwordField = new PasswordPanel();

PasswordPanel cPasswordField = new PasswordPanel();

static JComboBox occupationcombo, genderComboBox;

static HintAreaField addressArea;

static JButton signUpButton;

static JScrollPane scrollPane;

JPanel innerBody = new JPanel(null);

static Object values[] = new Object[12];

static DatePicker date;

public Register(String userType, String formType) throws IOException {

this.userType = userType;

this.formType = formType;

Container cc = getContentPane();

JPanel header = TemplateClass.getHeader();

JPanel body = TemplateClass.getBody();

JPanel footer = TemplateClass.getFooter();

cc.add(header, BorderLayout.NORTH);

cc.add(body, BorderLayout.WEST);

cc.add(footer, BorderLayout.SOUTH);

Signup = new JLabel(" " + formType + " ");

fName = new JLabel("Firstname");

lName = new JLabel("Lastname");

Age = new JLabel("Age");

Num = new JLabel("Number");

Mail = new JLabel("Mail");

Address = new JLabel("Address");

Gender = new JLabel("Gender");

occupation = new JLabel("Occupation");

DateOfBirth = new JLabel("DateOfBirth");

pass = new JLabel("Password");

cPass = new JLabel("Confirm Password");

healthLabel = new JLabel("Any Health conditions ?");

dynamicLabel = new JLabel((userType.equals(Login.USER) ? "Goal" : "Experience"));

date = new DatePicker();

date.setDateToToday();

Gender = new JLabel("Gender");

fNameField = new HintTextField("Enter your firstname");

lNameField = new HintTextField("Enter your lastname");

AgeField = new HintTextField("Enter the Age");

NumberField = new HintTextField("Enter the Phone Number");

MailField = new HintTextField("Enter the Mail-id");

healthField = new HintTextField("Enter if any");

dynamicField = new HintTextField(userType.equals(Login.USER) ? "Goal" : "Experience:");

scrollPane = new JScrollPane(innerBody, JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

scrollPane.setBounds(100, 80, 600, 470);

scrollPane.setViewportView(innerBody);

body.add(scrollPane);

occupationcombo = new JComboBox(new String[]{"Self-Employed", "Student", "Private Sector", "Government Employee"});

genderComboBox = new JComboBox<>(new String[]{"Male", "Female", "Others"});

addressArea = new HintAreaField("Enter the Address");

signUpButton = new JButton(formType);

Signup.setFont(new Font("Arial", Font.BOLD, 24));

Signup.setBounds(310, 30, 200, 30);

Signup = (JLabel) TemplateClass.formatFont(Signup, TextAttribute.UNDERLINE\_ON, 28);

body.add(Signup);

innerBody.add(fName);

innerBody.add(fNameField);

innerBody.add(lName);

innerBody.add(lNameField);

innerBody.add(Age);

innerBody.add(AgeField);

innerBody.add(Num);

innerBody.add(NumberField);

innerBody.add(Mail);

innerBody.add(MailField);

innerBody.add(occupation);

innerBody.add(occupationcombo);

innerBody.add(Gender);

innerBody.add(genderComboBox);

innerBody.add(Address);

innerBody.add(addressArea);

innerBody.add(DateOfBirth);

innerBody.add(date);

innerBody.add(pass);

innerBody.add(passwordField);

innerBody.add(cPass);

innerBody.add(cPasswordField);

innerBody.add(signUpButton);

innerBody.add(healthLabel);

innerBody.add(healthField);

innerBody.add(dynamicLabel);

innerBody.add(dynamicField);

innerBody.setPreferredSize(new Dimension(600, 800));

innerBody.setBackground(Color.LIGHT\_GRAY);

fName.setBounds(70, 50, 200, 50);

fNameField.setBounds(70, 90, 200, 50);

lName.setBounds(300, 50, 200, 50);

lNameField.setBounds(300, 90, 200, 50);

Age.setBounds(70, 140, 200, 50);

AgeField.setBounds(70, 180, 200, 50);

DateOfBirth.setBounds(300, 140, 200, 50);

date.setBounds(300, 180, 200, 50);

Num.setBounds(300, 230, 200, 50);

NumberField.setBounds(300, 270, 200, 50);

Mail.setBounds(70, 230, 200, 50);

MailField.setBounds(70, 270, 200, 50);

occupation.setBounds(300, 320, 200, 50);

occupationcombo.setBounds(300, 360, 200, 50);

Gender.setBounds(70, 320, 200, 50);

genderComboBox.setBounds(70, 360, 200, 50);

Address.setBounds(70, 410, 200, 50);

addressArea.setBounds(70, 450, 430, 80);

healthLabel.setBounds(70, 530, 200, 50);

healthField.setBounds(70, 570, 200, 50);

dynamicLabel.setBounds(300, 530, 200, 50);

dynamicField.setBounds(300, 570, 200, 50);

pass.setBounds(70, 620, 200, 50);

passwordField.setBounds(70, 660, 200, 50);

cPass.setBounds(300, 620, 200, 50);

cPasswordField.setBounds(300, 660, 200, 50);

signUpButton.setBounds(180, 730, 200, 50);

setSize(800, 800);

setLocationRelativeTo(null);

setIconImage(ImageIO.read(new File("resource/yLogo.png")));

setLayout(null);

setResizable(false);

setVisible(true);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

signUpButton.addActionListener(this);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == signUpButton) {

try {

values = Register.getValues(passwordField.getpassword());

if (checkValues(values)) {

if (passwordField.getpassword().equals(cPasswordField.getpassword())) {

if (validateOps.validatePassword(passwordField.getpassword())) {

if (validateOps.validateForm(values)) {

if (userType.equals(Login.USER)) {

startSerialize(new UserData(values));

writeToServer();

dispose();

}

} else {

JOptionPane.showMessageDialog(innerBody, "Enter valid data to proceed");

}

} else {

JOptionPane.showMessageDialog(innerBody, "Password should contain an uppercase,a lowercase,a number and a special character and 8 characters long");

}

} else {

JOptionPane.showMessageDialog(innerBody, "Passwords do not match");

}

} else {

JOptionPane.showMessageDialog(innerBody, "Please complete the form before trying again");

}

} catch (HeadlessException ex) {

System.out.println(ex);

}

}

}

static void writeToServer() {

try {

Socket client = new Socket(InetAddress.getLocalHost(), 7000);

File file = new File("Udata.blah");

byte[] bytes = new byte[16 \* 1024];

InputStream in = new FileInputStream(file);

OutputStream out = client.getOutputStream();

int count;

while ((count = in.read(bytes)) > 0) {

out.write(bytes, 0, count);

}

out.close();

in.close();

client.close();

} catch (Exception ex) {

Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);

}

}

static void startSerialize(UserData usd) {

try {

FileOutputStream fout = new FileOutputStream("Udata.blah");

ObjectOutputStream obout = new ObjectOutputStream(fout);

obout.writeObject(obout);

fout.close();

obout.close();

} catch (FileNotFoundException ex) {

Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);

}

}

static Object[] getValues(String pass) {

Object values[] = new Object[12];

values[0] = fNameField.getText();

values[1] = lNameField.getText();

values[2] = AgeField.getText();

values[3] = date.getDate().toString();

values[4] = MailField.getText();

values[5] = NumberField.getText();

values[6] = genderComboBox.getSelectedItem();

values[7] = occupationcombo.getSelectedItem();

values[8] = addressArea.getText();

values[9] = healthField.getText();

values[10] = dynamicField.getText();

values[11] = pass;

return values;

}

static boolean checkValues(Object values[]) {

for (Object value : values) {

String str = (String) value;

if (str.isEmpty()) {

return false;

}

}

return true;

}

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

new Register(Login.USER, "Register");

}

}

**SERVER SIDE:**

package TOOLS1;

import static TOOLS1.DbTools.getUserFromUserPool;

import WebHost.UserData;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author rajay

\*/

public class RegisterServer {

static private Connection connection;

static public Connection getConnect() {

try {

connection = DriverManager.getConnection("jdbc:mysql://localhost/gym31", "yogesh", "java");

} catch (SQLException ex) {

System.out.println(ex);

}

return connection;

}

public static void ReadFromStream() {

try {

ServerSocket serverSocket = null;

try {

serverSocket = new ServerSocket(4444);

} catch (IOException ex) {

System.out.println("Can't setup server on this port number. ");

}

Socket socket = null;

InputStream in = null;

OutputStream out = null;

try {

socket = serverSocket.accept();

} catch (IOException ex) {

System.out.println("Can't accept client connection. ");

}

try {

in = socket.getInputStream();

} catch (IOException ex) {

System.out.println("Can't get socket input stream. ");

}

try {

out = new FileOutputStream("Udata2.blah");

} catch (FileNotFoundException ex) {

System.out.println("File not found. ");

}

byte[] bytes = new byte[16 \* 1024];

int count;

while ((count = in.read(bytes)) > 0) {

out.write(bytes, 0, count);

}

out.close();

in.close();

socket.close();

serverSocket.close();

} catch (IOException ex) {

Logger.getLogger(RegisterServer.class.getName()).log(Level.SEVERE, null, ex);

}

}

public static boolean register\_user(Object values[]) {

try {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost/gym31", "yogesh", "java");

Object poolValues[] = getUserFromUserPool(values);

String query = ("insert into usertable values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

PreparedStatement ps = con.prepareStatement(query);

ps.setString(1, (String) poolValues[0]);

ps.setString(2, (String) values[0]);

ps.setString(3, (String) values[1]);

ps.setInt(4, Integer.parseInt((String) values[2]));

ps.setString(5, (String) values[3]);

ps.setString(6, (String) values[4]);

ps.setString(7, (String) values[5]);

ps.setString(8, (String) values[6]);

ps.setString(9, (String) values[7]);

ps.setString(10, (String) values[8]);

ps.setString(11, (String) values[9]);

ps.setString(12, (String) values[10]);

ps.setString(13, CryptUtility.encryptString((String) values[11]));

ps.setString(14, (String) poolValues[1]);

ps.setInt(15, (Integer) poolValues[2]);

ps.executeUpdate();

Statement st = connection.createStatement();

st.execute("Delete from userpool where id='" + poolValues[0] + "'");

return true;

} catch (Exception e) {

System.out.println(e);

}

return false;

}

static Object[] getValues(UserData object) {

Object values[] = new Object[12];

values[0] = object.fname;

values[1] = object.lname;

values[2] = object.age;

values[3] = object.date;

values[4] = object.mail;

values[5] = object.number;

values[6] = object.gender;

values[7] = object.occupation;

values[8] = object.address;

values[9] = object.health;

values[10] = object.dynamicfield;

values[11] = object.pass;

return values;

}

public static void main(String[] args) {

ReadFromStream();

UserData object = null;

try {

// Reading the object from a file

FileInputStream file = new FileInputStream("Udata2.blah");

ObjectInputStream in = new ObjectInputStream(file);

// Method for deserialization of object

object = (UserData) in.readObject();

in.close();

file.close();

System.out.println("Object has been deserialized\n"

+ "Data after Deserialization.");

} catch (Exception ex) {

System.out.println("IOException is caught");

}

Object values[] = getValues(object);

register\_user(values);

}

}

**2ND IMPLEMENTATION**

**STREAM VIDEO FROM SERVER TO CLIENT USING SOCKETS**

**CLIENT SIDE:**

package WebHost;

import java.awt.BorderLayout;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.InetAddress;

import java.net.Socket;

import java.net.UnknownHostException;

import java.nio.ByteBuffer;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

public class ClientSide extends Thread {

JFrame frame = new JFrame("Client Side");

JLabel label = new JLabel();

@Override

public void run() {

synchronized (this) {

try {

System.out.println("client waiting for ack from server");

this.wait(3000);

System.out.println("Resumed");

} catch (InterruptedException ex) {

Logger.getLogger(ClientSide.class.getName()).log(Level.SEVERE, null, ex);

}

try {

Socket client = new Socket(InetAddress.getLocalHost(), 7000);

InputStream is = client.getInputStream();

while (true) {

byte[] size = new byte[10000];

is.read(size);

int arSize = ByteBuffer.wrap(size).asIntBuffer().get();

byte[] image = new byte[arSize];

is.read(image);

BufferedImage bimage = ImageIO.read(new ByteArrayInputStream(image));

ImageIcon ico = new ImageIcon(bimage);

label.setIcon(ico);

}

} catch (UnknownHostException ex) {

Logger.getLogger(ClientSide.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(ClientSide.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public ClientSide() throws Exception {

System.out.println("Creating Client and waiting");

frame.setLayout(new BorderLayout());

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.add(label, BorderLayout.CENTER);

frame.setLocationRelativeTo(null);

frame.pack();

frame.setResizable(true);

frame.setSize(700, 600);

frame.setVisible(true);

}

}

**SERVER SIDE**

package WebHost;

import com.github.sarxos.webcam.Webcam;

import com.github.sarxos.webcam.WebcamResolution;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.nio.ByteBuffer;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

public class Serverside extends Thread {

OutputStream os = null;

ServerSocket server = null;

public Serverside() throws Exception {

System.out.println("Creating server and waiting");

}

public void startCam() throws IOException {

synchronized (this) {

this.server = new ServerSocket(7000);

System.out.println("Created and Notifying client");

this.notifyAll();

System.out.println("Notified waiting for client");

Socket client = server.accept();

System.out.println("Connected....");

os = client.getOutputStream();

Webcam webcam = Webcam.getDefault();

webcam.setViewSize(WebcamResolution.VGA.getSize());

webcam.open();

while (true) {

BufferedImage bImg = webcam.getImage();

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ImageIO.write(bImg, "jpg", bos);

byte[] imageSize = ByteBuffer.allocate(10000).putInt(bos.size()).array();

this.os.write(imageSize);

this.os.write(bos.toByteArray());

this.os.flush();

}

}

}

@Override

public void run() {

try {

this.startCam();

} catch (IOException ex) {

Logger.getLogger(ClientSide.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

**WRAPPER CLASS**

package WebHost;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author rajay

\*/

public class Wrapper {

public static void main(String[] args) {

try {

Thread t1 = new Thread(new Serverside());

Thread t2 = new Thread(new ClientSide());

t1.start();

t2.start();

} catch (Exception ex) {

Logger.getLogger(Wrapper.class.getName()).log(Level.SEVERE, null, ex);

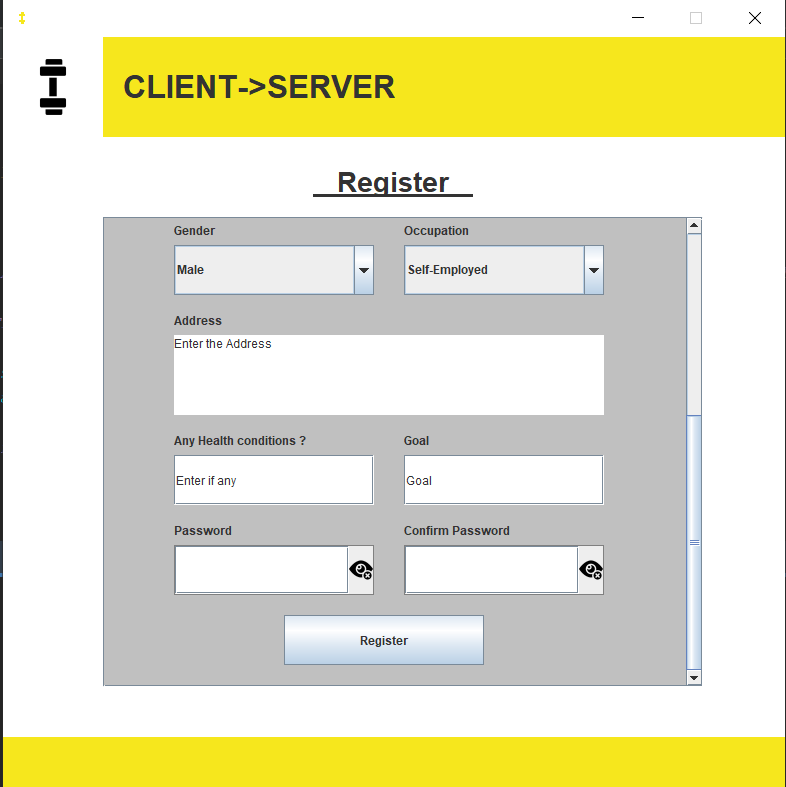
}

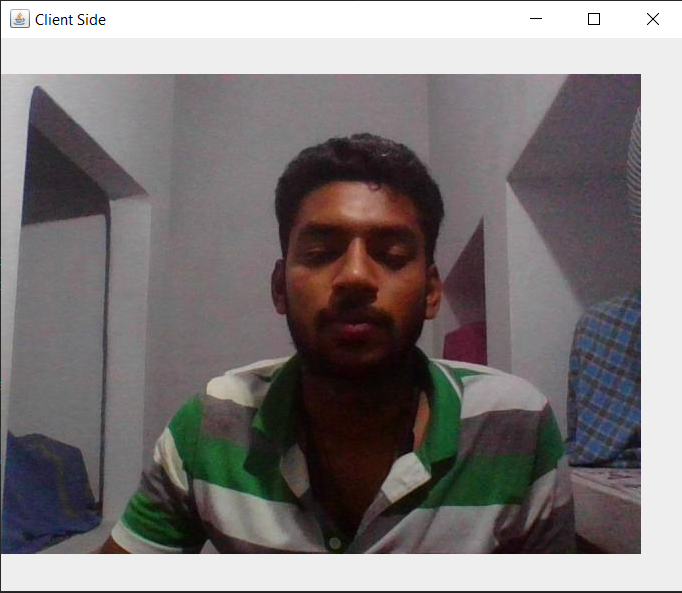
}

}

**SCREENSHOTS**

****

****

****