**AUTOMATED ATTENDANCE MANAGEMENT SYSTEM USING FACE RECOGNITION**

PROJECT SUBMITTED TO

**SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE**

**(Autonomous)**

AffiliatedTo **Bharathidasan University**

Nationally Re-Accredited with **“A”** Grade by **NAAC**

**ISO 9001:2015 CERTIFIED INSTITUTION**

**TIRUCHIRAPPALLI-620 005**

in partial fulfilment of the requirements for the award of the degree

**BACHELOR OF COMPUTER APPLICATION**

**Submitted by**

**M.G.SUDARSHAN**

**Reg.No.U16CA0060**

**Under the Guidance of**

**Ms.S.CHITRA,MCA.,M.Phil.,B.Ed.,(Ph.D.),**

****

**PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE**

**SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE**

**(Autonomous)**

**Tiruchirappalli-620 005**

**April-2019**

**SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE**

**(Autonomous)**

(AffiliatedTo **Bharathidasan University)**

**PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE**

**Tiruchirappalli-620 005**

**BACHELOR OF COMPUTER APPLICATION**

**CERTIFICATE**

This is to certify that the project work entitled “**AUTOMATED ATTENDANCE MANAGEMENT SYSTEM USING FACE RECOGNITION**” is a Bonafede work done by **M.G.SUDARSHAN** Register Number: **U16CA0060** submitted to **SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE,** Tiruchirappalli during the sixth semester in partial fulfilment of the requirement for the award of the degree **BACHELOR OF COMPUTER APPLICATION.**

**PROJECT GUIDE** **HEAD OF THE DEPARTMENT**

**VIVA-VOCE** Exam is held on

SIGNATURE OF EXAMINERS:

1)

2)

**SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE**

**(Autonomous)**

(AffiliatedTo **Bharathidasan University)**

**PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE**

**Tiruchirappalli-620 005**

**BACHELOR OF COMPUTER APPLICATION**

**DECLARATION**

I hereby declare that the project entitled “**AUTOMATED ATTENDANCE MANAGEMENT SYSTEM USING FACE RECOGNITION**”submitted towards the partial fulfilment of the requirement for the award of the degree **BACHELOR OF COMPUTER APPLICATION** is my original work and the project has not formed the basis for the award of any other degree.

**Signature of Student**

Place:

Date:

**ACKNOWLEDGEMENT**

First, I am very thankful to the Almighty for showering the Blessings throughout my life. I would like to express my heartfelt thanks to my beloved parents who have sacrificed a lot for my future. This support was the foundation stone for my efforts.

I acknowledge my sincere thanks to **Dr. N. RAMANUJAM, M.Sc., Ph.D**., Director Academic affairs, **Dr. J. RADHIKA, M.Sc., Ph.D**., Principal, **Dr. S. MADHU M.Sc., Ph,D**., Dean of Science, **Ms. R. UMADEVI, M.Sc., PGDCA., MCA., M.Phil,SET** Director of MCA., **Dr.M.SUGHASINY., M.Sc(IT)., M.Phil., MCP2007., Ph.D.,** Head of Department, **Dr. N. KARTHIKEYAN, M.Sc., M.Phil.,** **ADCSE., MCP2007., Ph.D**, Co-ordinator of Computer Science, Srimad Andavan Arts &Science College(Autonomous) for giving me the opportunity to do my Post Graduate Degree course in this renowned institution and providing facilities to carry out the project work.

I am grateful to **Ms.S.CHITRA,MCA.,M.Phil.,B.Ed.,(Ph.D.),**Asst. Professor the internal guide for timely suggestions and constant encouragement and support that led to the accomplishment of the project.

I extend my thanks to all the teaching, non-teaching staff members and all lab assistants of the Computer Science Department, who helped me either directly or indirectly to complete this project.

I wish to express my heartfelt thanks to my friends for their useful tips that helped me to complete this project successfully.

CONTENT

|  |  |  |
| --- | --- | --- |
| S.No | Title | Page Number |
| 1 | **Introduction** |  |
| 2 | **Requirement Specification**  2.1 System Requirement  2.2 Existing System  2.3 Proposed System  2.4 Software Description  2.5 Project Description |  |
| 3 | **Design**  3.1 Table Design  3.2 Use Case Diagram  3.3 Data Flow Diagram  3.4 UML Diagram |  |
| 4 | **Implementation**  4.1 Source Code  4.2 Screenshots |  |
| 5 | **Testing** |  |
| 6 | **Conclusion** |  |
| 7 | **Bibliography** |  |

**INTRODUCTION**

**1.INTRODUCTION**

Image Processing is a type of processing a signal for which the requirements are photograph, video frame or an image.

There are two types of Image processing: Analog and digital Processing.

Analogue image processing is an image processing technique which can be used for hard copies such as photographs and Printouts. While digital image processing involves manipulation of the digital images by using Pc’s. Now a days Student or attendance plays a significant role in many college, universities and schools.

There can be two types of attendance:

1.Attendance system (Manual)

2.attendance system (Automated)

Automated attendance system will excerpt the image when person comes in the classroom and will accordingly mark the attendance. On the other hand, manual attendance system will verify and manage each and every record of student in paper

which requires more time and effort of the faculty or staff and also chances of proxies are also more in manual attendance.

This system will be efficient and more user friendly as it can be run on devices which everyone has now a day. This study is the first attempt to provide an automated attendance system that identifies students using face recognition technology through an image or video stream for recording attendance in any classroom environment or and estimating the efficiency accordingly.

Through constantly detecting of facial info, this method will resolve less efficiency of technologies which are already existing, and advance the accurateness of recognition of faces. We studied and planned a technique or way that mark the presence or attendance using face recognition constructed on non-stop surveillance. In this proposed method or paper, our aim and purpose is to gain the images or video of the students face, their position and attendance which are beneficial info in the lecture or classroom environment.

**REQUIREMENT SPECIFICATION**

**2.1 SYSTEM REQURIEMNT**

Any software development process needs certain requirements to fulfil the software product. The following software and hardware requirement are necessary for the project. They are given below:

1.Windows :7 or higher

2. Processor : Intel i5 and AMD OPETARON

3.Storage : More than 2Gb

4.Ram : 1Gb

5.Software’s are:

a.Xampp

b.JDk(Java Developer Kit)

c.JVM(JAVA VIRTUAL MACHINE)

d.WEB camera

e.Netbeans IDE version(8.2)

**2.2 EXISTING SYSTEM**

1.Windows 10

2.Storage 1TB

3.Ram 8GB

4.Software’s are:

a.Xampp

b.JDk(Java Developer Kit)

c.JVM(JAVA VIRTUAL MACHINE)

d.WEB camera

e.Netbeans IDE version(8.2)

**2.3 Proposed System**

• User Friendly:- The proposed system is user friendly because the retrieval and storing of data is fast and data is maintained efficiently. Moreover the graphical user interface is provided in the proposed system, which provides user to deal with the system very easily.

• Reports are easily generated: reports can be easily generated in the proposed system so user can generate the report as per the requirement (monthly) or in the middle of the session. User can give the notice to the students so he/she become regular.

• Very less paper work: The proposed system requires very less paper work. All the data is feted into the computer immediately and reports can be generated through computers. Moreover work become very easy because there is no need to keep data on papers.

• Computer operator control: Computer operator control will be there so no chance of errors. Moreover storing and retrieving of information is easy. So work can be done speedily and in time.

**2.4Software Description**

**2.4.1. NETBEANS IDE:**

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5, and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

**2.4.2. XAMPP:**

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

XAMPP's ease of deployment means a WAMP or LAMP stack can be installed quickly and simply on an operating system by a developer, with the advantage a number of common add-in applications such as WordPress and Joomla! can also be installed with similar ease using Bitnami.

**2.5 Project Description**

The main aim of developing this java application is to provide a complete school or college-based attendance management system for students and profile system for staff and management.

Faculty information is to be maintained securely and should be accessible only to the respective faculty and the administrator when required. Student’s daily attendance should be entered by the faculty and should be managed perfectly. In the existing system manually, the corresponding faculty will enter attendance details in files. If the administrator or faculty want to know the student information, he must able to search the sheets, this consumes a lot of time. Here the project eliminates that problem and makes the process automation.

The Student Attendance and Staff Profile project mainly focus on maintaining of faculty profile, student daily attendance in which different levels of users are restricted to access the database. We can make the users access the data but they can’t modify or update the database. Only the authorized faculty and administrator can access and update the database. Each faculty and administrator should log in to get the details of the faculty and student attendance.

**2.5.1. Objectives of the project**

• Complete automation is possible in this sector, which is against the main disadvantage namely time-consuming.

• Can maintain company faculty details and student details.

• Any kind of information based on faculty profile or student attendance can be retrieved within less time.

• Only authorized faculty or administrator can access the database by providing necessary information.

**DESIGN**

**3.1Table Design**

**3.1.1. Teacher Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Field name** | **Data Type** | **Description** |
| 1. | User name | Text | Store user name for checking correct username |
| 2. | Password | Text | Store password corresponding to username |
| 3. | User Type | Text | User Type Administrator or User |

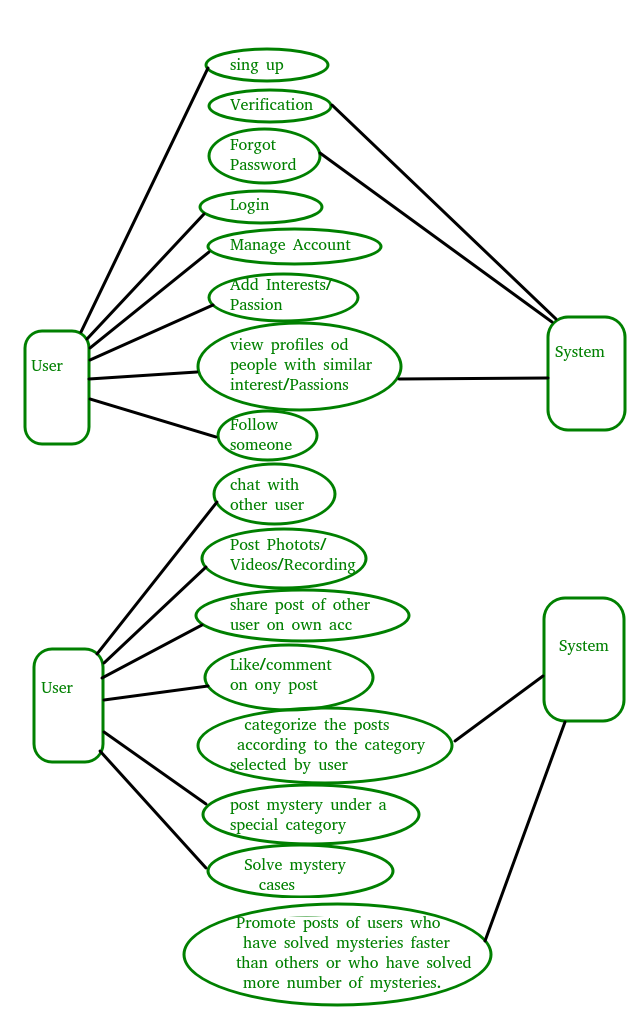
**3.1.2. Attendance Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Field name** | **Data Type** | **Description** |
| 1. | Student Name | Text | Name of Student |
| 2. | Status | Number | Total number classes attended by particular Student |
| 3. | Semester | Text | In which Semester Student is Studying |
| 4. | Subject | Text | The Subject Wise Attendance Is maintained |
| 5. | Month | Text | The Month Wise Attendance is maintained where total working days in month=20 |

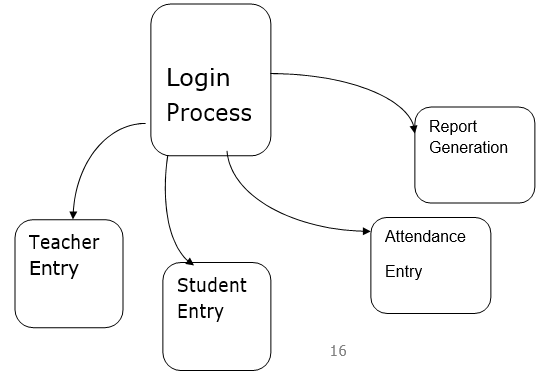
**3.1.3. Student Table:**

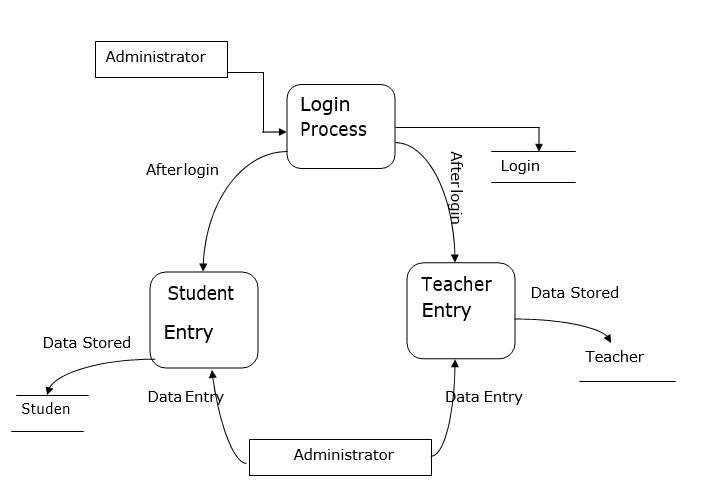
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Field name** | **Data Type** | **Description** |
| 1. | Student ID | Number | This is the roll no of the student |
| 2. | Student Name | Text | This is the name of student |
| 3. | Course | Text | This is the course in which the students are studying .By default course is MCA |
| 4. | Semester | Text | This is the semester in which Students are Studying. By default Semester is IV sem |

**3.2Use Case Diagram**

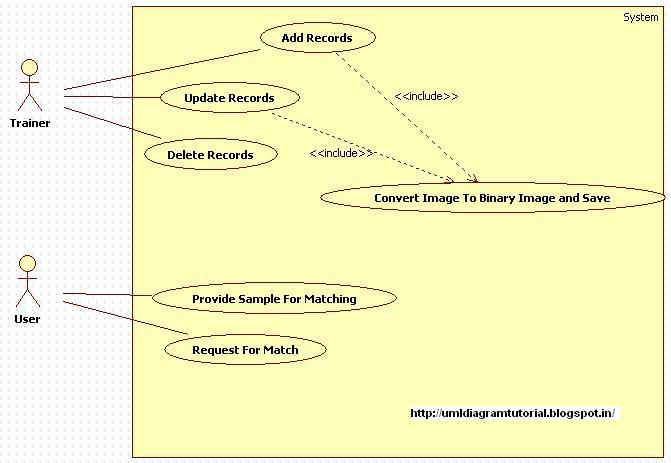


**3.3 DATA FLOW DIAGRAM:**



**3.3.2. Level DFD:**

**3.3. UML DIAGRAM:**



**IMPLEMENTATION**

**4. IMPLEMENTATION**

The process starts with training the system with face of students/employees for whom the attendance has to be marked in the near future. Different faces are assigned different names in the system.

The period decides the threshold time limit within which the student will be marked present. Once the threshold limit is crossed is crossed the student will not be marked absent in the system for that particular period Algorithms such as PCA, LDA and LBPH can be used in varying light scenarios, as light plays an important role in image processing.

Also, a webcam with high specifications should be used as it plays a key role in face detection and recognition. Hence, the better is the webcam used, the more is the efficiency in the system attained.

**4.1. SOURCE CODE**

**4.1.1. LOGIN PAGE:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Login;

import Attendence.MarkAttendence;

import Attendence.StudentsEntry;

import Database.DatabaseConnection;

import java.awt.HeadlessException;

import java.awt.event.KeyEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import javax.swing.UIManager;

/\*\*

\*

\* @author admin

\*/

public class LoginPage extends javax.swing.JFrame {

DatabaseConnection db;

public static String profname;

public static String stdbranch;

public static String stdusername;

public static String regno;

/\*\*

\* Creates new form LoginPage

\*/

public LoginPage() {

try {

initComponents();

db = new DatabaseConnection();

db.dbconnection();

setTitle("Please Enter Username and Password");

setResizable(false);

setLocationRelativeTo(null);

} catch (Exception e) {

}

}

public void login() {

try {

String query = "SELECT \* FROM teacher WHERE Username='" + txtUserName.getText() + "' AND password='" + txtUserPassword.getText() + "' ;";

ResultSet rs = db.getResultSet(query);

if (rs.next()) {

try {

this.dispose();

new StudentsEntry().setVisible(true);

} catch (Exception ae) {

JOptionPane.showMessageDialog(null, ae);

}

} else {

txtUserName.setText("");

txtUserPassword.setText("");

JOptionPane.showMessageDialog(this, "Wrong username or Password", "Error", JOptionPane.ERROR\_MESSAGE);

txtUserName.grabFocus();

}

} catch (SQLException | HeadlessException e) {

}

}

/\*\*

\* 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() {

buttonGroup1 = new javax.swing.ButtonGroup();

jCheckBox1 = new javax.swing.JCheckBox();

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

txtUserPassword = new javax.swing.JPasswordField();

txtUserName = new javax.swing.JTextField();

jPanel3 = new javax.swing.JPanel();

btnLogin = new javax.swing.JButton();

jButton1 = new javax.swing.JButton();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jCheckBox1.setText("jCheckBox1");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Login Form");

jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jPanel2.setBackground(new java.awt.Color(204, 204, 255));

jPanel2.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

jLabel1.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel1.setText("User name");

jLabel2.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel2.setText("Password");

txtUserPassword.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

txtUserPassword.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyReleased(java.awt.event.KeyEvent evt) {

txtUserPasswordKeyReleased(evt);

}

});

txtUserName.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1)

.addGap(18, 18, 18))

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)))

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(txtUserName)

.addComponent(txtUserPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 207, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addContainerGap(19, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(txtUserName, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(txtUserPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2))

.addGap(23, 23, 23))

);

jPanel3.setBackground(new java.awt.Color(204, 204, 255));

jPanel3.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

btnLogin.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

btnLogin.setMnemonic('L');

btnLogin.setText("Admin LOGIN");

btnLogin.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnLoginActionPerformed(evt);

}

});

jButton1.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jButton1.setText("Mark Attendence");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addContainerGap()

.addComponent(btnLogin)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, 154, Short.MAX\_VALUE)

.addContainerGap())

);

jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, 49, Short.MAX\_VALUE)

.addComponent(btnLogin, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap(48, Short.MAX\_VALUE))

);

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/login\_icon.png"))); // NOI18N

jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 36)); // NOI18N

jLabel4.setText("Attendance Using Face Recognition");

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGap(33, 33, 33)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(479, 479, 479))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(35, 35, 35)

.addComponent(jLabel4)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(18, Short.MAX\_VALUE)

.addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jLabel3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 267, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

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()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 620, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(24, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void txtUserPasswordKeyReleased(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

if (evt.getKeyChar() == KeyEvent.VK\_ENTER) {

try {

login();

} catch (Exception ex) {

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

}

}

}

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

try {

login();

} catch (Exception ex) {

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

}

}

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

// TODO add your handling code here:

try {

String query = "SELECT \* FROM teacher WHERE Username='" + txtUserName.getText() + "' AND password='" + txtUserPassword.getText() + "' ;";

ResultSet rs = db.getResultSet(query);

if (rs.next()) {

this.dispose();

Thread t = new Thread() {

public void run() {

try {

Thread.sleep(1000);

new MarkAttendence().recognizeFromCam();

} catch (Exception ae) {

JOptionPane.showMessageDialog(null, ae);

}

}

};

t.start();

} else {

txtUserName.setText("");

txtUserPassword.setText("");

JOptionPane.showMessageDialog(this, "Wrong username or Password", "Error", JOptionPane.ERROR\_MESSAGE);

txtUserName.grabFocus();

}

} catch (SQLException | HeadlessException e) {

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

UIManager.setLookAndFeel("com.jtattoo.plaf.aluminium.AluminiumLookAndFeel");

// UIManager.setLookAndFeel("com.jtattoo.plaf.bernstein.BernsteinLookAndFeel");

// UIManager.setLookAndFeel("com.jtattoo.plaf.fast.FastLookAndFeel");

// UIManager.setLookAndFeel("com.jtattoo.plaf.mint.MintLookAndFeel");

// UIManager.setLookAndFeel("com.jtattoo.plaf.hifi.HiFiLookAndFeel");

new LoginPage().setVisible(true);

} catch (Exception ex) {

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

}

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnLogin;

private javax.swing.ButtonGroup buttonGroup1;

private javax.swing.JButton jButton1;

private javax.swing.JCheckBox jCheckBox1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JTextField txtUserName;

private javax.swing.JPasswordField txtUserPassword;

// End of variables declaration

}

**4.1.2. STUDENT ENTRY**:

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Attendence;

import Database.DatabaseConnection;

import FaceRegognizer.FaceRecognizer1;

import java.awt.HeadlessException;

import java.awt.event.KeyEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

/\*\*

\*

\* @author test

\*/

public final class StudentsEntry extends javax.swing.JFrame {

DatabaseConnection dbcon;

boolean modifyflag = false;

boolean addflag = false;

public static String ID;

public static String NAME = "";

public static String STD = "";

public static String PHNO = "";

public static String EMAIL = "";

/\*\*

\* Creates new form StudentsEntry

\*/

public StudentsEntry() {

try {

initComponents();

jLabel6.setText("Srimad Andavan Arts and science college A+");

btnSave.setEnabled(false);

txtStudId.setEnabled(false);

cmbStd.setEditable(false);

setLocationRelativeTo(null);

dbcon = new DatabaseConnection();

dbcon.dbconnection();

loadListData("SELECT \* FROM student");

} catch (Exception ex) {

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

}

}

public void loadListData(String q) {

try {

Vector data = new Vector();

String Query = q;

//System.err.println(q);

ResultSet rs = dbcon.getResultSet(Query);

if (rs.next()) {

do {

data.add(rs.getString("Student\_Name"));

} while (rs.next());

}

rs.close();

lstStudentNames.removeAll();

lstStudentNames.setListData(data);

} catch (Exception e) {

}

}

/\*\*

\* 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() {

jPanel1 = new javax.swing.JPanel();

lblstudentImage = new javax.swing.JLabel();

jPanel2 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

txtSerchStudentName = new javax.swing.JTextField();

jScrollPane1 = new javax.swing.JScrollPane();

lstStudentNames = new javax.swing.JList();

jPanel3 = new javax.swing.JPanel();

jPanel4 = new javax.swing.JPanel();

btnClose = new javax.swing.JButton();

btnAdd = new javax.swing.JButton();

btnSave = new javax.swing.JButton();

btnUpdate = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jButton4 = new javax.swing.JButton();

jPanel5 = new javax.swing.JPanel();

txtParentPhoneNo = new javax.swing.JTextField();

jLabel5 = new javax.swing.JLabel();

txtStudId = new javax.swing.JTextField();

txtStudName = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel10 = new javax.swing.JLabel();

txtParentemail = new javax.swing.JTextField();

cmbStd = new javax.swing.JComboBox();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

jPanel2.setBackground(new java.awt.Color(255, 255, 255));

jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jPanel2.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel1.setText("ENTER STUDENT NAME");

txtSerchStudentName.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

txtSerchStudentName.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyPressed(java.awt.event.KeyEvent evt) {

txtSerchStudentNameKeyPressed(evt);

}

public void keyReleased(java.awt.event.KeyEvent evt) {

txtSerchStudentNameKeyReleased(evt);

}

});

lstStudentNames.setFont(new java.awt.Font("Lucida Bright", 1, 12)); // NOI18N

lstStudentNames.addListSelectionListener(new javax.swing.event.ListSelectionListener() {

public void valueChanged(javax.swing.event.ListSelectionEvent evt) {

lstStudentNamesValueChanged(evt);

}

});

jScrollPane1.setViewportView(lstStudentNames);

jPanel3.setBackground(new java.awt.Color(255, 255, 255));

jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jPanel3.setForeground(new java.awt.Color(255, 255, 255));

jPanel4.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

btnClose.setBackground(new java.awt.Color(255, 0, 153));

btnClose.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

btnClose.setText("CANCEL");

btnClose.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnCloseActionPerformed(evt);

}

});

btnAdd.setBackground(new java.awt.Color(255, 0, 153));

btnAdd.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

btnAdd.setText("ADD");

btnAdd.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnAddActionPerformed(evt);

}

});

btnSave.setBackground(new java.awt.Color(255, 0, 153));

btnSave.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

btnSave.setText("SAVE");

btnSave.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnSaveActionPerformed(evt);

}

});

btnUpdate.setBackground(new java.awt.Color(255, 0, 153));

btnUpdate.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

btnUpdate.setText("UPDATE");

btnUpdate.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnUpdateActionPerformed(evt);

}

});

jButton3.setBackground(new java.awt.Color(255, 0, 153));

jButton3.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jButton3.setText("Student List");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jButton4.setBackground(new java.awt.Color(255, 0, 153));

jButton4.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jButton4.setText("Log Out");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);

jPanel4.setLayout(jPanel4Layout);

jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(btnAdd, javax.swing.GroupLayout.PREFERRED\_SIZE, 97, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(btnUpdate, javax.swing.GroupLayout.PREFERRED\_SIZE, 97, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(btnClose, javax.swing.GroupLayout.DEFAULT\_SIZE, 179, Short.MAX\_VALUE)

.addComponent(btnSave, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton3))

.addGroup(jPanel4Layout.createSequentialGroup()

.addGap(37, 37, 37)

.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE, 113, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE))))

);

jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(btnSave, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(btnAdd, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)

.addComponent(btnClose, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(btnUpdate, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

jPanel5.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

txtParentPhoneNo.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel5.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel5.setText("Parent's Mobile No.");

txtStudId.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

txtStudId.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyReleased(java.awt.event.KeyEvent evt) {

txtStudIdKeyReleased(evt);

}

});

txtStudName.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

txtStudName.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyReleased(java.awt.event.KeyEvent evt) {

txtStudNameKeyReleased(evt);

}

});

jLabel3.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel3.setText("Student Std");

jLabel2.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel2.setText("Enter Student Name");

jLabel4.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel4.setText("Student Roll no.");

jLabel10.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

jLabel10.setText("Parent's Email ID");

txtParentemail.setFont(new java.awt.Font("Lucida Bright", 1, 14)); // NOI18N

cmbStd.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "B.C.A", "M.C.A", "B.sc C.S", "M.sc IT" }));

javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);

jPanel5.setLayout(jPanel5Layout);

jPanel5Layout.setHorizontalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel5Layout.createSequentialGroup()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED\_SIZE, 137, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(10, 10, 10)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(txtStudName, javax.swing.GroupLayout.DEFAULT\_SIZE, 258, Short.MAX\_VALUE)

.addComponent(txtParentemail)

.addComponent(txtStudId, javax.swing.GroupLayout.Alignment.TRAILING)))

.addGroup(jPanel5Layout.createSequentialGroup()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGap(11, 11, 11)

.addComponent(txtParentPhoneNo))

.addGroup(jPanel5Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(cmbStd, 0, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)))))

.addContainerGap())

);

jPanel5Layout.setVerticalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(txtStudId, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(txtStudName, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGap(14, 14, 14)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel5Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(cmbStd, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(11, 11, 11)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(txtParentPhoneNo)

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 23, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(11, 11, 11)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(txtParentemail)

.addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED\_SIZE, 23, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(18, Short.MAX\_VALUE))

);

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

.addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)))

);

jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

);

jLabel6.setFont(new java.awt.Font("Monotype Corsiva", 1, 24)); // NOI18N

jLabel6.setForeground(new java.awt.Color(255, 153, 51));

jLabel6.setText("jLabel6");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 199, Short.MAX\_VALUE)

.addComponent(txtSerchStudentName))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addGap(0, 14, Short.MAX\_VALUE)

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 176, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(txtSerchStudentName, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(16, 16, 16))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(jLabel6)

.addGap(18, 18, 18)))

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGap(94, 94, 94))

.addComponent(jScrollPane1))

.addGap(1, 1, 1))

);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(lblstudentImage)

.addGap(832, 832, 832)

.addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED\_SIZE, 99, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(42, 42, 42))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(28, 28, 28)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(281, 281, 281))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(lblstudentImage)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

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()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 710, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

);

setSize(new java.awt.Dimension(746, 509));

setLocationRelativeTo(null);

}// </editor-fold>

private void txtStudNameKeyReleased(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

String text = txtStudName.getText();

text = text.replaceAll("[\_]+", "-");

txtStudName.setText(text);

}

private void txtStudIdKeyReleased(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

JTextField tf = (JTextField) evt.getSource();

String text = tf.getText();

text = text.replaceAll("[^0-9]", "");

tf.setText(text);

}

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

// TODO add your handling code here:

modifyflag = true;

addflag = false;

btnAdd.setEnabled(false);

btnUpdate.setEnabled(false);

btnSave.setEnabled(true);

}

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

// TODO add your handling code here:

save();

}

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

// TODO add your handling code here:

modifyflag = false;

addflag = true;

btnAdd.setEnabled(false);

btnUpdate.setEnabled(false);

btnSave.setEnabled(true);

txtStudName.setText("");

txtStudId.setEnabled(true);

txtParentPhoneNo.setText("");

txtParentemail.setText("");

txtStudId.grabFocus();

txtStudId.setText("");

// txtStudId.setText(getStudId() + "");

}

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

// TODO add your handling code here:

txtSerchStudentName.grabFocus();

btnAdd.setEnabled(true);

btnUpdate.setEnabled(true);

btnSave.setEnabled(false);

loadListData("SELECT \* FROM Student");

if (lstStudentNames.getModel().getSize() >= 1) {

lstStudentNames.setSelectedIndex(0);

}

txtStudId.setText("");

txtStudName.setText("");

txtParentPhoneNo.setText("");

}

private void lstStudentNamesValueChanged(javax.swing.event.ListSelectionEvent evt) {

// TODO add your handling code here:

try {

if (lstStudentNames.getSelectedValue() != null) {

String name = lstStudentNames.getSelectedValue().toString();

String query = "SELECT \* FROM student WHERE Student\_name='" + name + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

txtStudId.setText(rs.getString(1));

txtStudName.setText(rs.getString(2));

//cmbStd.setText(rs.getString(3));

txtParentPhoneNo.setText(rs.getString(4));

txtParentemail.setText(rs.getString(5));

}

}

} catch (Exception e) {

}

}

private void txtSerchStudentNameKeyReleased(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

String text = txtSerchStudentName.getText().trim();

String query = "";

if (!text.equals("")) {

query = "select \* from student where Student\_name like'" + text + "%'";

} else {

query = "select \* from student";

}

loadListData(query);

}

private void txtSerchStudentNameKeyPressed(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

if ((evt.getKeyChar() == KeyEvent.VK\_ENTER) || (evt.getKeyCode() == 40)) {

lstStudentNames.grabFocus();

int row = lstStudentNames.getModel().getSize();

if (row >= 1) {

lstStudentNames.setSelectedIndex(0);

}

}

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

new Login.LoginPage().setVisible(true);

}

/\*\*

\* Check If Student Exists

\*

\*/

public boolean chkExistsStudent(String sid) {

boolean flag = false;

try {

String query = "SELECT \* FROM student WHERE sid='" + sid + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

flag = true;

}

} catch (Exception e) {

}

return flag;

}

/\*\*

\*

\*/

public void save() {

try {

ID = txtStudId.getText();

NAME = txtStudName.getText().trim();

STD = cmbStd.getSelectedItem().toString();

PHNO = txtParentPhoneNo.getText();

EMAIL = txtParentemail.getText();

if (chkExistsid(ID)) {

JOptionPane.showMessageDialog(this, "Student Id already in exists", "ERROR", JOptionPane.ERROR\_MESSAGE);

txtStudName.grabFocus();

return;

}

String studentStd = cmbStd.getSelectedItem().toString();

String student\_name = txtStudName.getText().trim();

if (student\_name.length() < 4 || student\_name.equals("")) {

JOptionPane.showMessageDialog(this, "Enter Proper Student Name !!", "ERROR", JOptionPane.ERROR\_MESSAGE);

txtStudName.grabFocus();

return;

}

String phoneNumber = txtParentPhoneNo.getText();

if (phoneNumber.trim().equals("")) {

JOptionPane.showMessageDialog(this, "Enter Phone Number !!!", "ERROR", JOptionPane.ERROR\_MESSAGE);

txtParentPhoneNo.grabFocus();

return;

}

String email = txtParentemail.getText();

if (!isEmailValid(email)) {

JOptionPane.showMessageDialog(this, "Enter proper EmailId", "Error", JOptionPane.ERROR\_MESSAGE);

txtParentemail.grabFocus();

return;

}

if (!isPhoneNumberValid(phoneNumber)) {

JOptionPane.showMessageDialog(this, "Enter Vaild Phone Number", "ERROR", JOptionPane.ERROR\_MESSAGE);

txtParentPhoneNo.grabFocus();

return;

}

if (addflag) {

if (!chkExists(student\_name, studentStd)) {

this.dispose();

Thread t = new Thread() {

public void run() {

try {

// Thread.sleep(1000);

new FaceRecognizer1().recognizeFromCam1();

} catch (Exception ae) {

JOptionPane.showMessageDialog(null, ae);

}

}

};

t.start();

} else {

JOptionPane.showMessageDialog(this, "Already Exists Entry!!!!");

}

}

if (modifyflag) {

String query = "UPDATE STUDENT SET "

+ " student\_name='" + student\_name + "',"

+ " student\_std='" + studentStd + "',"

+ " phoneno='" + phoneNumber + "'"

+ " WHERE sid='" + ID + "'";

int ans = dbcon.getUpdate(query);

if (ans == 1) {

JOptionPane.showMessageDialog(this, "Sucssefully Update Student Data!!!");

} else {

JOptionPane.showMessageDialog(this, "Failed To Update Data!!!");

}

}

txtStudId.setText("");

txtStudName.setText("");

txtParentPhoneNo.setText("");

txtParentemail.setText("");

btnAdd.setEnabled(true);

btnUpdate.setEnabled(true);

btnSave.setEnabled(false);

loadListData("SELECT \* from student");

} catch (NumberFormatException | HeadlessException e) {

}

}

public static boolean isPhoneNumberValid(String phoneNumber) {

boolean isValid = false;

String expression = "[789]{1}[0-9]{9}";//this to validate only 8 digit and 10 digit numbers

CharSequence inputStr = phoneNumber;

Pattern pattern = Pattern.compile(expression);

Matcher matcher = pattern.matcher(inputStr);

if (matcher.matches()) {

isValid = true;

}

return isValid;

}

public static boolean isEmailValid(String email) {

boolean isValid = false;

//Initialize reg ex for email.

String expression = "\\b[A-Za-z0-9.\_%+-]+@[A-Za-z.]+\\.[A-Za-z]{2,3}\\b";

CharSequence inputStr = email;

//Make the comparison case-insensitive.

Pattern pattern = Pattern.compile(expression, Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(inputStr);

if (matcher.matches()) {

isValid = true;

}

return isValid;

}

public boolean chkExists(String studname, String std) {

boolean flag = false;

try {

String query = "SELECT \* FROM student WHERE student\_Name='" + studname + "' AND student\_std='" + std + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

flag = true;

}

} catch (Exception e) {

}

return flag;

}

public boolean chkExistsid(String stdID) {

boolean flag = false;

try {

String query = "SELECT \* FROM student WHERE sid='" + stdID + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

flag = true;

}

} catch (Exception e) {

}

return flag;

}

public int getStudId() {

int id = 1;

try {

String query = "SELECT MAX(sid) FROM student";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

String retId = rs.getString("sid");

if (retId != null) {

id = Integer.parseInt(retId.trim()) + 1;

}

}

} catch (SQLException | NumberFormatException e) {

}

return id;

}

/\*\*

\* @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(StudentsEntry.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(StudentsEntry.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(StudentsEntry.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(StudentsEntry.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 StudentsEntry().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnAdd;

private javax.swing.JButton btnClose;

private javax.swing.JButton btnSave;

private javax.swing.JButton btnUpdate;

private javax.swing.JComboBox cmbStd;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JPanel jPanel4;

private javax.swing.JPanel jPanel5;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JLabel lblstudentImage;

private javax.swing.JList lstStudentNames;

private javax.swing.JTextField txtParentPhoneNo;

private javax.swing.JTextField txtParentemail;

private javax.swing.JTextField txtSerchStudentName;

private javax.swing.JTextField txtStudId;

private javax.swing.JTextField txtStudName;

// End of variables declaration

}

**4.1.3. REFRESH OR DELETING STUDENTS IMAGE FROM DATABASE:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package test;

import Database.DatabaseConnection;

import java.io.File;

import java.io.FileOutputStream;

/\*\*

\*

\* @author test

\*/

public class RefreshAll {

public void refresh() {

refreshTextFiles();

deleteDataFiles();

updateDB();

}

/\*\*

\* Deleting Data Files

\*/

public void deleteDataFiles() {

try {

File[] lstFiles = new File("data").listFiles();

for (File f : lstFiles) {

System.out.println("deleting: " + f.getName());

f.delete();

}

System.out.println("All Files Are Deleted");

} catch (Exception e) {

}

}

public void updateDB() {

try {

String query = "Delete from student";

DatabaseConnection dbcon = new DatabaseConnection();

dbcon.dbconnection();

dbcon.getUpdate(query);

} catch (Exception e) {

}

}

/\*\*

\* Refreshing Text Files ......

\*/

public void refreshTextFiles() {

try {

File file1 = new File("train.txt");

FileOutputStream fos1 = new FileOutputStream(file1);

fos1.write("".getBytes());

fos1.close();

} catch (Exception e) {

}

}

/\*\*

\*

\*/

public static void main(String[] args) {

RefreshAll rfrshall = new RefreshAll();

rfrshall.refresh();

}

}

**4.1.4. CONNECTION TO DATABASE*:***

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Database;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author kp

\*/

public class DatabaseConnection {

String db = "vidattendence";

String username = "root";

String password = "";

Connection con = null;

Statement stmt;

ResultSet rs = null;

public Connection dbconnection() {

try {

String url = "jdbc:mysql://localhost/" + db;

Class.forName("com.mysql.jdbc.Driver").newInstance();

con = DriverManager.getConnection(url, username, password);

}

catch (Exception ex) {

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

}

return con;

}

public ResultSet getResultSet(String query) {

try {

stmt = con.createStatement();

rs = stmt.executeQuery(query);

} catch (SQLException ex) {

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

}

return rs;

}

public int getUpdate(String query) {

int i = 0;

try {

stmt = con.createStatement();

i = stmt.executeUpdate(query);

} catch (Exception e) {

}

return i;

}

}

**4.1.5. DATE GATHERER:**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Attendence;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

/\*\*

\*

\* @author kp

\*/

public class Dategetter

{

public static String getCurrentDate()

{

String date = "";

Calendar cal = Calendar.getInstance();

int day = cal.get(Calendar.DATE);

int month = cal.get(Calendar.MONTH) + 1;

int year = cal.get(Calendar.YEAR);

date = year + "-" + month + "-" + day;

return date;

}

public static String getCurrentTime()

{

String time = "";

SimpleDateFormat sdf=new SimpleDateFormat("HH.mm");

time=sdf.format(new Date());

return time;

}

public static String getDayName()

{

String day = "";

SimpleDateFormat dtformat = new SimpleDateFormat("EEEEEEEEEE");

day = dtformat.format(new Date());

return day;

}

public static void main(String[] args)

{

System.out.println(getDayName());

System.out.println(getCurrentTime());

System.out.println(getCurrentDate());

}

}

**4.1.6. FACE RECOGNIZATION SCREEN**:

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package FaceRegognizer;

import Attendence.StudentsEntry;

import Database.DatabaseConnection;

import com.googlecode.javacpp.FloatPointer;

import com.googlecode.javacpp.Pointer;

import com.googlecode.javacpp.PointerPointer;

import com.googlecode.javacv.CanvasFrame;

import com.googlecode.javacv.FrameGrabber;

import com.googlecode.javacv.OpenCVFrameGrabber;

import static com.googlecode.javacv.cpp.opencv\_core.\*;

import com.googlecode.javacv.cpp.opencv\_core.CvFileStorage;

import com.googlecode.javacv.cpp.opencv\_core.CvMat;

import com.googlecode.javacv.cpp.opencv\_core.CvMemStorage;

import com.googlecode.javacv.cpp.opencv\_core.CvRect;

import com.googlecode.javacv.cpp.opencv\_core.CvSeq;

import com.googlecode.javacv.cpp.opencv\_core.CvSize;

import com.googlecode.javacv.cpp.opencv\_core.CvTermCriteria;

import com.googlecode.javacv.cpp.opencv\_core.IplImage;

import static com.googlecode.javacv.cpp.opencv\_highgui.\*;

import static com.googlecode.javacv.cpp.opencv\_imgproc.\*;

import static com.googlecode.javacv.cpp.opencv\_legacy.\*;

import static com.googlecode.javacv.cpp.opencv\_objdetect.\*;

import com.googlecode.javacv.cpp.opencv\_objdetect.CvHaarClassifierCascade;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JButton;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author test

\*/

public class FaceRecognizer1 {

int numberOfTrainingFaces = 30;

private static final Logger LOGGER = Logger.getLogger(FaceRecognizer1.class.getName());

private int nTrainFaces = 0;

/\*\*

\* the training face image array

\*/

IplImage[] trainingFaceImgArr;

/\*\*

\* the test face image array

\*/

IplImage[] FaceImgArr;

/\*\*

\* the person number array

\*/

IplImage[] testFaceImgArr;

CvMat personNumTruthMat;

/\*\*

\* the number of persons

\*/

int nPersons;

/\*\*

\* the person names

\*/

final List<String> personNames = new ArrayList<String>();

/\*\*

\* the number of eigenvalues

\*/

int nEigens = 0;

/\*\*

\* eigenvectors

\*/

IplImage[] eigenVectArr;

/\*\*

\* eigenvalues

\*/

CvMat eigenValMat;

/\*\*

\* the average image

\*/

IplImage pAvgTrainImg;

/\*\*

\* the projected training faces

\*/

CvMat projectedTrainFaceMat;

CvMat trainPersonNumMat;

//Cascade File Name

String faceCascadeFilename = "./HarrClassiifator/haarcascade\_frontalface\_alt2.xml";

//face width faceHeight, faceWidth

int faceWidth = 120; // Default dimensions for faces in the face recognition database. Added by Shervin.

int faceHeight = 90;

//===========

FrameGrabber grabber;

CvMemStorage storage;

public static CanvasFrame input;

boolean stopflag = false;

char actionChar = '0';

char action = '0';

//waitScreen waitscreen;

public boolean isstart = false;

public FaceRecognizer1() {

try {

storage = cvCreateMemStorage(0);

} catch (Exception e) {

}

}

@SuppressWarnings("static-access")

public void recognizeFromCam1() throws InterruptedException {

try {

int i;

// waitscreen = new waitScreen(null, false);

CvHaarClassifierCascade faceCascade;

String cstr;

boolean saveNextFaces = false;

String newPersonName = "";

int newPersonFaces;

final FrameGrabber grabber = new OpenCVFrameGrabber(0);

grabber.setImageWidth(210);

grabber.setImageHeight(210);

grabber.start();

saveNextFaces = false;

newPersonFaces = 0;

if (loadTrainingData1() == 1) {

faceWidth = pAvgTrainImg.width();

faceHeight = pAvgTrainImg.height();

} else {

}

final CanvasFrame input = new CanvasFrame("Face Training");

input.setLayout(new FlowLayout());

JButton save = new JButton("SAVE");

JButton refresh = new JButton("CANCEL");

MyOwnListener listener = new MyOwnListener(this);

save.addActionListener(listener);

input.add(save);

input.add(refresh);

refresh.setEnabled(false);

input.setLocation(277, 165);

input.setAlwaysOnTop(true);

input.setDefaultCloseOperation(CanvasFrame.DO\_NOTHING\_ON\_CLOSE);

refresh.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (!isstart) {

isstart = true;

input.dispose();

// waitscreen.setVisible(false);

try {

grabber.stop();

JOptionPane.showMessageDialog(null, "User will not Register" + "\n" + "Try again");

} catch (FrameGrabber.Exception ex) {

Logger.getLogger(FaceRecognizer1.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

});

faceCascade = new CvHaarClassifierCascade(cvLoad(faceCascadeFilename));

while (!isstart) {

IplImage camImg;

IplImage greyImg;

IplImage faceImg;

IplImage sizedImg;

IplImage equalizedImg;

IplImage processedFaceImg;

CvRect faceRect;

IplImage shownImg;

char ch = actionChar;

// System.out.println(ch);

if (ch == KeyEvent.VK\_ESCAPE) {

break;

}

switch (ch) {

case 'n':

newPersonName = StudentsEntry.NAME+"&"+StudentsEntry.STD;

if (newPersonName.equals("")) {

setActionChar('0');

continue;

}

newPersonName = modifyName(newPersonName);

if (newPersonName.length() > 0) {

System.out.println("Collecting all images until you hit 't', to start Training the images as " + newPersonName);

// waitscreen.setVisible(true);

newPersonFaces = 0; // restart training a new person

saveNextFaces = true;

setActionChar('0');

//===================== Disable Buttons ================

save.setEnabled(false);

refresh.setEnabled(false);

} else {

System.out.println("Did not get a valid name from you, so will ignore it. Hit 'n' to retry.\n");

}

break;

case 't':

//Strat training

refresh.setEnabled(true);

saveNextFaces = false;

System.out.println("Storing the training data for new person " + newPersonName);

for (i = 0; i < newPersonFaces; i++) {

cstr = "data/" + (nPersons + 1) + "\_" + newPersonName + "\_" + (i + 1) + ".jpg";

String svData = (nPersons + 1) + " " + cstr + " " + newPersonName + "\n";

saveFile(svData);

}

saveNextFaces = false;

newPersonFaces = 0;

if (!(trainPersonNumMat == null)) {

cvFree(trainPersonNumMat);// Free the previous data before getting new data

}

trainPersonNumMat = retrainOnline();

setActionChar('0');

//waitscreen.setVisible(false);

DatabaseConnection dbcon = new DatabaseConnection();

dbcon.dbconnection();

String studentId = StudentsEntry.ID;

String student\_name = StudentsEntry.NAME;

String studentStd = StudentsEntry.STD;

String phoneNumber = StudentsEntry.PHNO;

String email = StudentsEntry.EMAIL;

String query = "INSERT INTO student VALUES(" + studentId + ",'" + student\_name + "','" + studentStd + "','" + phoneNumber + "','"+email+"')";

dbcon.getUpdate(query);

isstart = true;

continue;

}

// Get the camera frame

try {

camImg = grabber.grab();

} catch (Exception e) {

continue;

}

// Make sure the image is greyscale, since the Eigenfaces is only done on greyscale image.

greyImg = convertImageToGreyscale(camImg);

// Perform face detection on the input image, using the given Haar cascade classifier.

faceRect = detectFaceInImage(greyImg, faceCascade);

// Make sure a valid face was detected.

if (faceRect.width() > 0) {

faceImg = cropImage(greyImg, faceRect);

// Make sure the image is the same dimensions as the training images.

sizedImg = resizeImage(faceImg, faceWidth, faceHeight);

// Give the image a standard brightness and contrast, in case it was too dark or low contrast.

equalizedImg = IplImage.create(new CvSize(sizedImg.width(), sizedImg.height()), 8, 1); // Create an empty greyscale image

cvEqualizeHist(sizedImg, equalizedImg);

processedFaceImg = equalizedImg;

if (saveNextFaces) {

if (newPersonFaces < (numberOfTrainingFaces)) {

cstr = "data/" + (nPersons + 1) + "\_" + newPersonName + "\_" + (newPersonFaces + 1) + ".jpg";

cvSaveImage(cstr, processedFaceImg);

newPersonFaces++;

} else {

saveNextFaces = false;

setActionChar('t');

}

}

}

// Show the data on the screen.

shownImg = cvCloneImage(camImg);

if (faceRect.width() > 60)// Check if a face was detected.

{

// Show the detected face region.

cvRectangle(shownImg, cvPoint(faceRect.x(), faceRect.y()), cvPoint(faceRect.x() + faceRect.width() - 1, faceRect.y() + faceRect.height() - 1), CV\_RGB(0, 255, 0), 1, 8, 0);

}

// Display the image.

input.showImage(shownImg);

}

cvReleaseMemStorage(storage);

grabber.stop();

cvReleaseHaarClassifierCascade(faceCascade);

input.dispose();

JOptionPane.showMessageDialog(null, "Student is successfully Register");

new StudentsEntry().setVisible(true);

} catch (FrameGrabber.Exception e) {

}

}

public IplImage resizeImage(IplImage origImg, int newWidth, int newHeight) {

try {

IplImage outImage;

int origWidth = 0;

int origHeight = 0;

if (!(origImg == null)) {

origWidth = origImg.width();

origHeight = origImg.height();

}

if (newWidth <= 0 || newHeight <= 0 || origWidth <= 0 || origHeight <= 0) {

return origImg;

}

// Scale the image to the new dimensions, even if the aspect ratio will be changed.

outImage = IplImage.create(cvSize(newWidth, newHeight), origImg.depth(), origImg.nChannels());

if (newWidth > origImg.width() && newHeight > origImg.height()) {

// Make the image larger

cvResetImageROI(origImg);

cvResize(origImg, outImage, CV\_INTER\_LINEAR); // CV\_INTER\_CUBIC or CV\_INTER\_LINEAR is good for enlarging

} else {

// Make the image smaller

cvResetImageROI(origImg);

cvResize(origImg, outImage, CV\_INTER\_AREA); // CV\_INTER\_AREA is good for shrinking / decimation, but bad at enlarging.

}

return outImage;

} catch (Exception e) {

}

return null;

}

//=====================================================================================

public IplImage cropImage(IplImage img, CvRect region) {

try {

IplImage imageTmp;

IplImage imageRGB;

CvSize size = new CvSize(img.width(), img.height());

if (img.depth() != IPL\_DEPTH\_8U) {

return img;

}

// First create a new (color or greyscale) IPL Image and copy contents of img into it.

imageTmp = IplImage.create(size, IPL\_DEPTH\_8U, img.nChannels());

cvCopy(img, imageTmp);

// Create a new image of the detected region

// Set region of interest to that surrounding the face

cvSetImageROI(imageTmp, region);

// Copy region of interest (i.e. face) into a new iplImage (imageRGB) and return it

size = size.width(region.width());

size = size.height(region.height());

imageRGB = IplImage.create(size, IPL\_DEPTH\_8U, img.nChannels());

cvCopy(imageTmp, imageRGB);

return imageRGB;

} catch (Exception e) {

}

return null;

}

public CvRect detectFaceInImage(IplImage inputImg, CvHaarClassifierCascade cascade) {

try {

IplImage detectImg;

IplImage greyImg = null;

CvRect rc;

double t;

CvSeq rects;

int i;

if (storage == null) {

storage = cvCreateMemStorage(0);

}

cvClearMemStorage(storage);

detectImg = inputImg;

if (inputImg.nChannels() > 1) {

greyImg = IplImage.create(cvGetSize(inputImg), IPL\_DEPTH\_8U, 1);

cvCvtColor(inputImg, greyImg, CV\_BGR2GRAY);

detectImg = greyImg; // Use the greyscale version as the input.

}

// Detect all the faces.

rects = cvHaarDetectObjects(detectImg, cascade, storage, 1.1, 1, 0);

if (rects.total() > 0) {

rc = new CvRect(cvGetSeqElem(rects, 0));

} else {

rc = new CvRect(-1, -1, -1, -1);

}

if (!(greyImg == null)) {

cvReleaseImage(greyImg);

}

return rc;

} catch (Exception e) {

}

return null;

}

//======================================================================================

public IplImage convertImageToGreyscale(IplImage imageSrc) {

try {

IplImage imageGrey;

// Either convert the image to greyscale, or make a copy of the existing greyscale image.

// This is to make sure that the user can always call cvReleaseImage() on the output, whether it was greyscale or not.

if (imageSrc.nChannels() == 3) {

imageGrey = cvCreateImage(cvGetSize(imageSrc), IPL\_DEPTH\_8U, 1);

cvCvtColor(imageSrc, imageGrey, CV\_BGR2GRAY);

} else {

imageGrey = cvCloneImage(imageSrc);

}

return imageGrey;

} catch (Exception e) {

}

return null;

}

//==================================================================================================================

public CvMat retrainOnline() {

try {

CvMat trainPersonNumMat;

int i = 0;

if (FaceImgArr != null) {

if (FaceImgArr.length > 0) {

for (i = 0; i < nTrainFaces; i++) {

if (!FaceImgArr[i].isNull()) {

cvReleaseImage(FaceImgArr[i]);

}

}

}

}

// cvFree(FaceImgArr); // array of face images

if (personNumTruthMat != null) {

cvFree(personNumTruthMat);

} // array of person numbers

personNames.clear();

nPersons = 0; // the number of people in the training set. .

nTrainFaces = 0; // the number of training images

nEigens = 0; // the number of eigenvalues

if (pAvgTrainImg != null) {

cvReleaseImage(pAvgTrainImg); // the average image

}

for (i = 0; i < nTrainFaces; i++) {

if (!eigenVectArr[i].isNull()) {

cvReleaseImage(eigenVectArr[i]);

}

}

if (eigenValMat != null) {

cvFree(eigenValMat);

}

if (projectedTrainFaceMat != null)// eigenvalues

{

cvFree(projectedTrainFaceMat); // projected training faces

}

// Retrain from the data in the files

System.out.println("Retraining with the new person ...\n");

String path = new File(".").getCanonicalPath();

learn(path + "/train.txt");

System.out.println("Done retraining.\n");

} catch (IOException ex) {

Logger.getLogger(FaceRecognizer1.class.getName()).log(Level.SEVERE, null, ex);

}

return null;

}

//==================================================================================================================

private int loadTrainingData1() {

LOGGER.info("loading training data");

trainPersonNumMat = null; // the person numbers during training

CvFileStorage fileStorage;

int i;

// create a file-storage interface

fileStorage = cvOpenFileStorage(

"data/facedata.xml", // filename

null, // memstorage

CV\_STORAGE\_READ, // flags

null); // encoding

if (fileStorage == null) {

LOGGER.severe("Can't open training database file 'data/facedata.xml'.");

return 0;

}

// Load the person names.

personNames.clear(); // Make sure it starts as empty.

nPersons = cvReadIntByName(

fileStorage, // fs

null, // map

"nPersons", // name

0); // default\_value

if (nPersons == 0) {

LOGGER.severe("No people found in the training database 'data/facedata.xml'.");

return 0;

} else {

LOGGER.info(nPersons + " persons read from the training database");

}

// Load each person's name.

for (i = 0; i < nPersons; i++) {

String sPersonName;

String varname = "personName\_" + (i + 1);

sPersonName = cvReadStringByName(

fileStorage, // fs

null, // map

varname,

"");

personNames.add(sPersonName);

}

LOGGER.info("person names: " + personNames);

// Load the data

nEigens = cvReadIntByName(

fileStorage, // fs

null, // map

"nEigens",

0); // default\_value

nTrainFaces = cvReadIntByName(

fileStorage,

null, // map

"nTrainFaces",

0); // default\_value

Pointer pointer = cvReadByName(

fileStorage, // fs

null, // map

"trainPersonNumMat"); // name

trainPersonNumMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage, // fs

null, // map

"eigenValMat"); // name

eigenValMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage, // fs

null, // map

"projectedTrainFaceMat"); // name

projectedTrainFaceMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage,

null, // map

"avgTrainImg");

pAvgTrainImg = new IplImage(pointer);

eigenVectArr = new IplImage[nTrainFaces];

for (i = 0; i < nEigens; i++) {

String varname = "eigenVect\_" + i;

pointer = cvReadByName(

fileStorage,

null, // map

varname);

eigenVectArr[i] = new IplImage(pointer);

}

// release the file-storage interface

cvReleaseFileStorage(fileStorage);

LOGGER.log(Level.INFO, "Training data loaded ({0} training images of {1} people)", new Object[]{nTrainFaces, nPersons});

final StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("People: ");

if (nPersons > 0) {

stringBuilder.append("<").append(personNames.get(0)).append(">");

}

for (i = 1; i < nPersons; i++) {

stringBuilder.append(", <").append(personNames.get(i)).append(">");

}

LOGGER.info(stringBuilder.toString());

return 1;

}

//=========================================================================

public static void saveFile(String text) {

try {

String path = new File(".").getCanonicalPath();

File file = new File(path + "/train.txt");

if (!file.exists()) {

file.createNewFile();

}

String trainData;

try (FileInputStream fis = new FileInputStream(file)) {

byte[] buff = new byte[fis.available()];

fis.read(buff);

trainData = new String(buff);

trainData = trainData + text;

}

try (FileOutputStream fos = new FileOutputStream(file)) {

fos.write(trainData.getBytes());

}

} catch (Exception e) {

}

}

//==============================================================================

public void learn(final String trainingFileName) {

int i;

// load training data

LOGGER.info("===========================================");

LOGGER.info("Loading the training images in " + trainingFileName);

trainingFaceImgArr = loadFaceImgArray(trainingFileName);

nTrainFaces = trainingFaceImgArr.length;

LOGGER.info("Got " + nTrainFaces + " training images");

if (nTrainFaces < 3) {

LOGGER.severe("Need 3 or more training faces\n"

+ "Input file contains only " + nTrainFaces);

return;

}

// do Principal Component Analysis on the training faces

doPCA();

LOGGER.info("projecting the training images onto the PCA subspace");

// project the training images onto the PCA subspace

projectedTrainFaceMat = cvCreateMat(

nTrainFaces, // rows

nEigens, // cols

CV\_32FC1); // type, 32-bit float, 1 channel

// initialize the training face matrix - for ease of debugging

for (int i1 = 0; i1 < nTrainFaces; i1++) {

for (int j1 = 0; j1 < nEigens; j1++) {

projectedTrainFaceMat.put(i1, j1, 0.0);

}

}

LOGGER.info("created projectedTrainFaceMat with " + nTrainFaces + " (nTrainFaces) rows and " + nEigens + " (nEigens) columns");

final FloatPointer floatPointer = new FloatPointer(nEigens);

for (i = 0; i < nTrainFaces; i++) {

cvEigenDecomposite(

trainingFaceImgArr[i], // obj

nEigens, // nEigObjs

new PointerPointer(eigenVectArr), // eigInput (Pointer)

0, // ioFlags

null, // userData (Pointer)

pAvgTrainImg, // avg

floatPointer); // coeffs (FloatPointer)

for (int j1 = 0; j1 < nEigens; j1++) {

projectedTrainFaceMat.put(i, j1, floatPointer.get(j1));

}

}

if (nTrainFaces < 5) {

LOGGER.log(Level.INFO, "projectedTrainFaceMat after cvEigenDecomposite:\n{0}", projectedTrainFaceMat);

}

// store the recognition data as an xml file

storeTrainingData();

}

private IplImage[] loadFaceImgArray(final String filename) {

IplImage[] faceImgArr;

BufferedReader imgListFile;

String imgFilename;

int iFace = 0;

int nFaces = 0;

int i;

try {

// open the input file

imgListFile = new BufferedReader(new FileReader(filename));

// count the number of faces

while (true) {

final String line = imgListFile.readLine();

if (line == null || line.isEmpty()) {

break;

}

nFaces++;

}

LOGGER.info("nFaces: " + nFaces);

imgListFile = new BufferedReader(new FileReader(filename));

// allocate the face-image array and person number matrix

faceImgArr = new IplImage[nFaces];

personNumTruthMat = cvCreateMat(

1, // rows

nFaces, // cols

CV\_32SC1); // type, 32-bit unsigned, one channel

// initialize the person number matrix - for ease of debugging

for (int j1 = 0; j1 < nFaces; j1++) {

personNumTruthMat.put(0, j1, 0);

}

personNames.clear(); // Make sure it starts as empty.

nPersons = 0;

// store the face images in an array

for (iFace = 0; iFace < nFaces; iFace++) {

String personName;

String sPersonName;

int personNumber;

// read person number (beginning with 1), their name and the image filename.

final String line = imgListFile.readLine();

// JOptionPane.showMessageDialog(null,"Line:"+line);

if (line.isEmpty()) {

break;

}

final String[] tokens = line.split(" ");

// JOptionPane.showMessageDialog(null, tokens[0]+" and "+tokens[1]);

personNumber = Integer.parseInt(tokens[0]);

personName = tokens[2];

imgFilename = tokens[1];

sPersonName = personName;

System.out.println(tokens[0] + "=> " + tokens[1] + " =>" + tokens[2]);

LOGGER.info("Got " + iFace + " " + personNumber + " " + personName + " " + imgFilename);

// Check if a new person is being loaded.

if (personNumber > nPersons) {

// Allocate memory for the extra person (or possibly multiple), using this new person's name.

personNames.add(sPersonName);

nPersons = personNumber;

LOGGER.info("Got new person " + sPersonName + " -> nPersons = " + nPersons + " [" + personNames.size() + "]");

}

// Keep the data

personNumTruthMat.put(

0, // i

iFace, // j

personNumber); // v

// load the face image

faceImgArr[iFace] = cvLoadImage(

imgFilename, // filename

CV\_LOAD\_IMAGE\_GRAYSCALE); // isColor

if (faceImgArr[iFace] == null) {

throw new RuntimeException("Can't load image from " + imgFilename);

}

}

imgListFile.close();

} catch (IOException ex) {

throw new RuntimeException(ex);

}

LOGGER.info("Data loaded from '" + filename + "': (" + nFaces + " images of " + nPersons + " people).");

final StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("People: ");

if (nPersons > 0) {

stringBuilder.append("<").append(personNames.get(0)).append(">");

}

for (i = 1; i < nPersons && i < personNames.size(); i++) {

stringBuilder.append(", <").append(personNames.get(i)).append(">");

}

LOGGER.info(stringBuilder.toString());

return faceImgArr;

}

/\*\*

\* Does the Principal Component Analysis, finding the average image and the

\* eigenfaces that represent any image in the given dataset.

\*/

private void doPCA() {

int i;

CvTermCriteria calcLimit;

CvSize faceImgSize = new CvSize();

// set the number of eigenvalues to use

nEigens = nTrainFaces - 1;

LOGGER.info("allocating images for principal component analysis, using " + nEigens + (nEigens == 1 ? " eigenvalue" : " eigenvalues"));

// allocate the eigenvector images

faceImgSize.width(trainingFaceImgArr[0].width());

faceImgSize.height(trainingFaceImgArr[0].height());

eigenVectArr = new IplImage[nEigens];

for (i = 0; i < nEigens; i++) {

eigenVectArr[i] = cvCreateImage(

faceImgSize, // size

IPL\_DEPTH\_32F, // depth

1); // channels

}

// allocate the eigenvalue array

eigenValMat = cvCreateMat(

1, // rows

nEigens, // cols

CV\_32FC1); // type, 32-bit float, 1 channel

// allocate the averaged image

pAvgTrainImg = cvCreateImage(

faceImgSize, // size

IPL\_DEPTH\_32F, // depth

1); // channels

// set the PCA termination criterion

calcLimit = cvTermCriteria(

CV\_TERMCRIT\_ITER, // type

nEigens, // max\_iter

1); // epsilon

LOGGER.info("computing average image, eigenvalues and eigenvectors");

// compute average image, eigenvalues, and eigenvectors

cvCalcEigenObjects(

nTrainFaces, // nObjects

new PointerPointer(trainingFaceImgArr), // input

new PointerPointer(eigenVectArr), // output

CV\_EIGOBJ\_NO\_CALLBACK, // ioFlags

0, // ioBufSize

null, // userData

calcLimit,

pAvgTrainImg, // avg

eigenValMat.data\_fl()); // eigVals

LOGGER.info("normalizing the eigenvectors");

cvNormalize(

eigenValMat, // src (CvArr)

eigenValMat, // dst (CvArr)

1, // a

0, // b

CV\_L1, // norm\_type

null); // mask

}

/\*\*

\* Stores the training data to the file 'data/facedata.xml'.

\*/

private void storeTrainingData() {

CvFileStorage fileStorage;

int i;

LOGGER.info("writing data/facedata.xml");

// create a file-storage interface

fileStorage = cvOpenFileStorage(

"data/facedata.xml", // filename

null, // memstorage

CV\_STORAGE\_WRITE, // flags

null); // encoding

// Store the person names. Added by hemant.

cvWriteInt(

fileStorage, // fs

"nPersons", // name

nPersons); // value

for (i = 0; i < nPersons; i++) {

String varname = "personName\_" + (i + 1);

cvWriteString(

fileStorage, // fs

varname, // name

personNames.get(i), // string

0); // quote

}

// store all the data

cvWriteInt(

fileStorage, // fs

"nEigens", // name

nEigens); // value

cvWriteInt(

fileStorage, // fs

"nTrainFaces", // name

nTrainFaces); // value

cvWrite(

fileStorage, // fs

"trainPersonNumMat", // name

personNumTruthMat); // value

cvWrite(

fileStorage, // fs

"eigenValMat", // name

eigenValMat); // value

cvWrite(

fileStorage, // fs

"projectedTrainFaceMat", // name

projectedTrainFaceMat);

cvWrite(fileStorage, // fs

"avgTrainImg", // name

pAvgTrainImg); // value

for (i = 0; i < nEigens; i++) {

String varname = "eigenVect\_" + i;

cvWrite(

fileStorage, // fs

varname, // name

eigenVectArr[i]); // value

}

cvReleaseFileStorage(fileStorage);

}

public void setActionChar(char ch) {

this.actionChar = ch;

}

public String modifyName(String name) {

try {

name = name.trim();

name = name.replaceAll("[ ]+", "\_");

return name;

} catch (Exception e) {

}

return name;

}

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

new FaceRecognizer1().recognizeFromCam1();

}

}

/\*\*

\*

\*/

class MyOwnListener implements ActionListener {

FaceRecognizer1 recognizer1;

public MyOwnListener(FaceRecognizer1 recognizer1) {

this.recognizer1 = recognizer1;

}

@Override

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

if (command.equals("SAVE")) {

recognizer1.setActionChar('n');

}

}

}

**4.1.7.ATTENDENCE JAVA**:

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Attendence;

import Database.DatabaseConnection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author test

\*/

public class Attendence {

DatabaseConnection dbcon;

String date;

String time;

public List<String> studentAtt;

int thresholdFactor = 20;

public static String from = "";

public static String to = "";

public Attendence() {

dbcon = new DatabaseConnection();

dbcon.dbconnection();

date = Dategetter.getCurrentDate();

time = Dategetter.getCurrentTime();

studentAtt = new ArrayList<>();

}

public void MarkAttendence(String sid) {

try {

studentAtt.add(sid);

if (studentAtt.size() >= thresholdFactor) {

/\*\*

\* check weather last 10 images are of same sid

\*

\*/

int cnter = 0;

for (String sidnum : studentAtt) {

if (sidnum.equals(sid)) {

cnter++;

}

}

if (cnter >= thresholdFactor - 5) {

String Date = this.date;

this.time = Dategetter.getCurrentTime();

String subject = getSubject(time);

if (!subject.equals("")) {

String fromtime = from;

String totime = to;

if ((!chkAttendenceExists(sid, Date, subject, fromtime, totime))) {

String query = "INSERT INTO attendence(sid,present,attDate,entryTime,subject,fromtime,totime) VALUES(" + sid + ",'p','" + Date + "','" + this.time + "','" + subject + "','" + fromtime + "','" + totime + "')";

dbcon.getUpdate(query);

JOptionPane.showMessageDialog(null, "ATTENDENCE MARKED FOR STUDENT " + getStudentName(sid) + " At Time " + this.time);

}

}

}

System.out.println("before: " + studentAtt);

studentAtt.removeAll(studentAtt);

System.out.println("after: " + studentAtt);

}

} catch (Exception e) {

}

}

public boolean chkAttendenceExists(String sid, String Date, String subject, String fromtime, String totime) {

boolean flag = false;

try {

String query = "SELECT \* FROM attendence WHERE sid='" + sid + "' AND attDate='" + Date + "' AND subject='" + subject + "' AND fromtime='" + fromtime + "' AND totime='" + totime + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

flag = true;

}

} catch (Exception e) {

}

return flag;

}

public String getSubject(String currtime) {

String subject = "";

System.out.println("CurrTime IS: " + currtime);

try {

String day = Dategetter.getDayName();

float currtimef = Float.parseFloat(currtime);

String query = "SELECT \* FROM timetable WHERE day='" + day + "'";

ResultSet rs = dbcon.getResultSet(query);

while (rs.next()) {

float stTime = Float.parseFloat(rs.getString("Fromtime"));

float enTime = Float.parseFloat(rs.getString("Totime"));

if (currtimef >= stTime && currtimef <= enTime) {

subject = rs.getString("Subject");

from = rs.getString("fromtime");

to = rs.getString("totime");

break;

}

}

} catch (NumberFormatException | SQLException e) {

}

return subject;

}

public String getSid(String StudentName, String std) {

String sid = "";

try {

String query = "SELECT sid from student WHERE STUDENT\_NAME='" + StudentName + "' AND student\_std='" + std + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

sid = rs.getString(1);

}

} catch (SQLException | NumberFormatException e) {

}

return sid;

}

public String getStudentName(String sid) {

String name = "";

try {

String query = "SELECT STUDENT\_NAME from student WHERE sid='" + sid + "'";

ResultSet rs = dbcon.getResultSet(query);

if (rs.next()) {

name = rs.getString(1);

}

} catch (Exception e) {

}

return name;

}

}

**4.1.9. MARK ATTENDENCE**:

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package Attendence;

import static Attendence.Attendence.from;

import static Attendence.Attendence.to;

import Database.DatabaseConnection;

import com.googlecode.javacv.OpenCVFrameGrabber;

import com.googlecode.javacpp.FloatPointer;

import com.googlecode.javacpp.Pointer;

import com.googlecode.javacpp.PointerPointer;

import com.googlecode.javacv.CanvasFrame;

import com.googlecode.javacv.FrameGrabber;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import static com.googlecode.javacv.cpp.opencv\_core.\*;

import static com.googlecode.javacv.cpp.opencv\_legacy.\*;

import static com.googlecode.javacv.cpp.opencv\_imgproc.\*;

import static com.googlecode.javacv.cpp.opencv\_objdetect.\*;

import java.awt.FlowLayout;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Random;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author test

\*/

public class MarkAttendence {

DatabaseConnection dbcon;

String date;

String time;

//==================================================

float MaxConfidence = 0.800f;

//==================================================

private static final Logger LOGGER = Logger.getLogger(MarkAttendence.class.getName());

private int nTrainFaces = 0;

/\*\*

\* the training face image array

\*/

IplImage[] trainingFaceImgArr;

/\*\*

\* the test face image array

\*/

IplImage[] FaceImgArr;

/\*\*

\* the person number array

\*/

IplImage[] testFaceImgArr;

CvMat personNumTruthMat;

/\*\*

\* the number of persons

\*/

int nPersons;

/\*\*

\* the person names

\*/

final List<String> personNames = new ArrayList<String>();

/\*\*

\* the number of eigenvalues

\*/

int nEigens = 0;

/\*\*

\* eigenvectors

\*/

IplImage[] eigenVectArr;

/\*\*

\* eigenvalues

\*/

CvMat eigenValMat;

/\*\*

\* the average image

\*/

IplImage pAvgTrainImg;

/\*\*

\* the projected training faces

\*/

CvMat projectedTrainFaceMat;

CvMat trainPersonNumMat;

//Cascade File Name

String faceCascadeFilename = "./HarrClassiifator/haarcascade\_frontalface\_alt2.xml";

//face width faceHeight, faceWidth

int faceWidth = 120; // Default dimensions for faces in the face recognition database. Added by Shervin.

int faceHeight = 90;

//===========

FrameGrabber grabber;

CvMemStorage storage;

String login = null;

//====== Additional classes==========================

Attendence attendence;

public MarkAttendence() {

try {

storage = cvCreateMemStorage(0);

attendence = new Attendence();

} catch (Exception e) {

}

}

public void recognizeFromCam() {

try {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

float projectedTestFace[];

CvHaarClassifierCascade faceCascade;

FrameGrabber grabber = new OpenCVFrameGrabber(0);

grabber.setImageWidth(210);

grabber.setImageHeight(210);

grabber.start();

if (loadTrainingData1() == 1) {

faceWidth = pAvgTrainImg.width();

faceHeight = pAvgTrainImg.height();

} else {

}

projectedTestFace = new float[nEigens];

CanvasFrame input = new CanvasFrame("Mark Attendence frame");

input.setLayout(new FlowLayout());

input.setLocation(580, 150);

input.setAlwaysOnTop(true);

input.setDefaultCloseOperation(CanvasFrame.DO\_NOTHING\_ON\_CLOSE);

faceCascade = new CvHaarClassifierCascade(cvLoad(faceCascadeFilename));

while (true) {

int iNearest, nearest = 0;

IplImage camImg;

IplImage greyImg;

IplImage faceImg;

IplImage sizedImg;

IplImage equalizedImg;

IplImage processedFaceImg;

CvRect faceRect;

IplImage shownImg;

float confidence = 0;

try {

camImg = grabber.grab();

} catch (Exception e) {

continue;

}

// Make sure the image is greyscale, since the Eigenfaces is only done on greyscale image.

greyImg = convertImageToGreyscale(camImg);

// Perform face detection on the input image, using the given Haar cascade classifier.

faceRect = detectFaceInImage(greyImg, faceCascade);

// faceRect.width();

// Make sure a valid face was detected.

if (faceRect.width() > 0) {

faceImg = cropImage(greyImg, faceRect);

// Make sure the image is the same dimensions as the training images.

sizedImg = resizeImage(faceImg, faceWidth, faceHeight);

// Give the image a standard brightness and contrast, in case it was too dark or low contrast.

equalizedImg = IplImage.create(new CvSize(sizedImg.width(), sizedImg.height()), 8, 1); // Create an empty greyscale image

cvEqualizeHist(sizedImg, equalizedImg);

processedFaceImg = equalizedImg;

if (processedFaceImg.isNull()) {

continue;

}

// If the face rec database has been loaded, then try to recognize the person currently detected.

if (nEigens > 0) {

// project the test image onto the PCA subspace

cvEigenDecomposite(

processedFaceImg,

nEigens,

new PointerPointer(eigenVectArr),

0, null,

pAvgTrainImg,

projectedTestFace);

// Check which person it is most likely to be.

final FloatPointer pConfidence = new FloatPointer(confidence);

iNearest = findNearestNeighbor(projectedTestFace, new FloatPointer(pConfidence));

confidence = pConfidence.get();

nearest = trainPersonNumMat.data\_i().get(iNearest);

}

}

// Show the data on the screen.

shownImg = cvCloneImage(camImg);

if (faceRect.width() > 0)// Check if a face was detected.

{

// Show the detected face region.

cvRectangle(shownImg, cvPoint(faceRect.x(), faceRect.y()), cvPoint(faceRect.x() + faceRect.width() - 1, faceRect.y() + faceRect.height() - 1), CV\_RGB(0, 255, 0), 1, 8, 0);

if (nEigens > 0 && confidence >= MaxConfidence) // Check if the face recognition database is loaded and a person was recognized.

{

// Show the name of the recognized person, overlayed on the image below their face.

CvFont font = new CvFont();

cvInitFont(font, CV\_FONT\_HERSHEY\_PLAIN, 1.0, 1.0, 0, 1, CV\_AA);

CvScalar textColor = CV\_RGB(255, 0, 0); // light blue text

String text = "";

String arr[] = personNames.get(nearest - 1).toString().split("&");

String name = "" + arr[0];

text = name;

cvPutText(shownImg, text, cvPoint(faceRect.x(), faceRect.y() + faceRect.height() + 15), font, textColor);

String pname = getOrgnlName(name);

String std = arr[1];

System.out.println("hello----" + text + "" + std);

String id = null, emailid = null;

// DB.Connect.openConnection();

// try {

// String str = "select \* from student where Student\_Name='" + text + "'";

// System.out.println(str);

// DB.Connect.rs = DB.Connect.stat.executeQuery(str);

// if (DB.Connect.rs.next()) {

// // id = DB.Connect.rs.getString("sid");

// emailid = DB.Connect.rs.getString("email");

//

// }

// DB.Connect.closeConnection();

//

// } catch (Exception e) {

//

DB.Connect.openConnection();

// this.time = Dategetter.getCurrentTime();

String sy = "p";

String query = "INSERT INTO att values('" + text + "','" + DB.Connect.getDate() + "','" + sy + "')";

DB.Connect.stat.executeUpdate(query);

DB.Connect.closeConnection();

login = emailid;

String[] to = {

login

};

// System.out.println(a);

// sendMail.sendFromGMail("apdtc2013@gmail.com", "apdtc@123", to, "OTP", "Your Pending Amount Contact with Admin");

//attendence.MarkAttendence(attendence.getSid(pname, std));

}

// Display the image.

input.showImage(shownImg);

JOptionPane.showMessageDialog(null, "ATTENDENCE MARKED Successfully ");

break;

// KeyEvent key = input.waitKey(10);

}

}

} catch (Exception e) {

}

}

public String getSubject(String currtime) {

String subject = "";

System.out.println("CurrTime IS: " + currtime);

try {

String day = Dategetter.getDayName();

float currtimef = Float.parseFloat(currtime);

String query = "SELECT \* FROM timetable WHERE day='" + day + "'";

ResultSet rs = dbcon.getResultSet(query);

while (rs.next()) {

float stTime = Float.parseFloat(rs.getString("Fromtime"));

float enTime = Float.parseFloat(rs.getString("Totime"));

if (currtimef >= stTime && currtimef <= enTime) {

subject = rs.getString("Subject");

from = rs.getString("fromtime");

to = rs.getString("totime");

break;

}

}

} catch (NumberFormatException | SQLException e) {

}

return subject;

}

public IplImage resizeImage(IplImage origImg, int newWidth, int newHeight) {

try {

IplImage outImage;

int origWidth = 0;

int origHeight = 0;

if (!(origImg == null)) {

origWidth = origImg.width();

origHeight = origImg.height();

}

if (newWidth <= 0 || newHeight <= 0 || origWidth <= 0 || origHeight <= 0) {

return origImg;

}

// Scale the image to the new dimensions, even if the aspect ratio will be changed.

outImage = IplImage.create(cvSize(newWidth, newHeight), origImg.depth(), origImg.nChannels());

if (newWidth > origImg.width() && newHeight > origImg.height()) {

// Make the image larger

cvResetImageROI(origImg);

cvResize(origImg, outImage, CV\_INTER\_LINEAR); // CV\_INTER\_CUBIC or CV\_INTER\_LINEAR is good for enlarging

} else {

// Make the image smaller

cvResetImageROI(origImg);

cvResize(origImg, outImage, CV\_INTER\_AREA); // CV\_INTER\_AREA is good for shrinking / decimation, but bad at enlarging.

}

return outImage;

} catch (Exception e) {

}

return null;

}

//=====================================================================================

public IplImage cropImage(IplImage img, CvRect region) {

try {

IplImage imageTmp;

IplImage imageRGB;

CvSize size = new CvSize(img.width(), img.height());

// size=size.height(img.height());

// size=size.width(img.width());

if (img.depth() != IPL\_DEPTH\_8U) {

return img;

}

// First create a new (color or greyscale) IPL Image and copy contents of img into it.

imageTmp = IplImage.create(size, IPL\_DEPTH\_8U, img.nChannels());

cvCopy(img, imageTmp);

// Create a new image of the detected region

// Set region of interest to that surrounding the face

cvSetImageROI(imageTmp, region);

// Copy region of interest (i.e. face) into a new iplImage (imageRGB) and return it

size = size.width(region.width());

size = size.height(region.height());

imageRGB = IplImage.create(size, IPL\_DEPTH\_8U, img.nChannels());

cvCopy(imageTmp, imageRGB);

// cvReleaseImage(imageTmp);

return imageRGB;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

public CvRect detectFaceInImage(IplImage inputImg, CvHaarClassifierCascade cascade) {

try {

IplImage detectImg;

IplImage greyImg = null;

// CvMemStorage storage = null;

CvRect rc;

double t;

CvSeq rects;

int i;

if (storage == null) {

storage = cvCreateMemStorage(0);

}

cvClearMemStorage(storage);

detectImg = inputImg;

if (inputImg.nChannels() > 1) {

greyImg = IplImage.create(cvGetSize(inputImg), IPL\_DEPTH\_8U, 1);

cvCvtColor(inputImg, greyImg, CV\_BGR2GRAY);

detectImg = greyImg; // Use the greyscale version as the input.

}

// Detect all the faces.

rects = cvHaarDetectObjects(detectImg, cascade, storage, 1.1, 1, 0);

if (rects.total() > 0) {

rc = new CvRect(cvGetSeqElem(rects, 0));

// System.out.println(rc.width());

} else {

rc = new CvRect(-1, -1, -1, -1);

}

if (!(greyImg == null)) {

cvReleaseImage(greyImg);

}

// System.out.println(rc.width());

// cvReleaseMemStorage(storage);

return rc;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

//======================================================================================

public IplImage convertImageToGreyscale(IplImage imageSrc) {

try {

IplImage imageGrey;

// Either convert the image to greyscale, or make a copy of the existing greyscale image.

// This is to make sure that the user can always call cvReleaseImage() on the output, whether it was greyscale or not.

if (imageSrc.nChannels() == 3) {

imageGrey = cvCreateImage(cvGetSize(imageSrc), IPL\_DEPTH\_8U, 1);

cvCvtColor(imageSrc, imageGrey, CV\_BGR2GRAY);

} else {

imageGrey = cvCloneImage(imageSrc);

}

return imageGrey;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

//==================================================================================================================

private int loadTrainingData1() {

LOGGER.info("loading training data");

trainPersonNumMat = null; // the person numbers during training

CvFileStorage fileStorage;

int i;

// create a file-storage interface

fileStorage = cvOpenFileStorage(

"data/facedata.xml", // filename

null, // memstorage

CV\_STORAGE\_READ, // flags

null); // encoding

if (fileStorage == null) {

LOGGER.severe("Can't open training database file 'data/facedata.xml'.");

return 0;

}

// Load the person names.

personNames.clear(); // Make sure it starts as empty.

nPersons = cvReadIntByName(

fileStorage, // fs

null, // map

"nPersons", // name

0); // default\_value

if (nPersons == 0) {

LOGGER.severe("No people found in the training database 'data/facedata.xml'.");

return 0;

} else {

LOGGER.info(nPersons + " persons read from the training database");

}

// Load each person's name.

for (i = 0; i < nPersons; i++) {

String sPersonName;

String varname = "personName\_" + (i + 1);

sPersonName = cvReadStringByName(

fileStorage, // fs

null, // map

varname,

"");

personNames.add(sPersonName);

}

LOGGER.info("person names: " + personNames);

// Load the data

nEigens = cvReadIntByName(

fileStorage, // fs

null, // map

"nEigens",

0); // default\_value

nTrainFaces = cvReadIntByName(

fileStorage,

null, // map

"nTrainFaces",

0); // default\_value

Pointer pointer = cvReadByName(

fileStorage, // fs

null, // map

"trainPersonNumMat", null); // name

trainPersonNumMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage, // fs

null, // map

"eigenValMat", null); // name

eigenValMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage, // fs

null, // map

"projectedTrainFaceMat", null); // name

projectedTrainFaceMat = new CvMat(pointer);

pointer = cvReadByName(

fileStorage,

null, // map

"avgTrainImg", null);

pAvgTrainImg = new IplImage(pointer);

eigenVectArr = new IplImage[nTrainFaces];

for (i = 0; i < nEigens; i++) {

String varname = "eigenVect\_" + i;

pointer = cvReadByName(

fileStorage,

null, // map

varname, null);

eigenVectArr[i] = new IplImage(pointer);

}

// release the file-storage interface

cvReleaseFileStorage(fileStorage);

LOGGER.log(Level.INFO, "Training data loaded ({0} training images of {1} people)", new Object[]{nTrainFaces, nPersons});

final StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("People: ");

if (nPersons > 0) {

stringBuilder.append("<").append(personNames.get(0)).append(">");

}

for (i = 1; i < nPersons; i++) {

stringBuilder.append(", <").append(personNames.get(i)).append(">");

}

LOGGER.info(stringBuilder.toString());

return 1;

}

/\*\*

\* Find the most likely person based on a detection. Returns the index, and

\* stores the confidence value into pConfidence.

\*

\* @param projectedTestFace the projected test face

\* @param pConfidencePointer a pointer containing the confidence value

\* @param iTestFace the test face index

\* @return the index

\*/

private int findNearestNeighbor(float projectedTestFace[], FloatPointer pConfidencePointer) {

double leastDistSq = Double.MAX\_VALUE;

int i = 0;

int iTrain = 0;

int iNearest = 0;

LOGGER.info("................");

LOGGER.info("find nearest neighbor from " + nTrainFaces + " training faces");

for (iTrain = 0; iTrain < nTrainFaces; iTrain++) {

//LOGGER.info("considering training face " + (iTrain + 1));

double distSq = 0;

for (i = 0; i < nEigens; i++) {

//LOGGER.debug(" projected test face distance from eigenface " + (i + 1) + " is " + projectedTestFace[i]);

float projectedTrainFaceDistance = (float) projectedTrainFaceMat.get(iTrain, i);

float d\_i = projectedTestFace[i] - projectedTrainFaceDistance;

distSq += d\_i \* d\_i; // / eigenValMat.data\_fl().get(i); // Mahalanobis distance (might give better results than Eucalidean distance)

// if (iTrain < 5) {

// LOGGER.info(" \*\* projected training face " + (iTrain + 1) + " distance from eigenface " + (i + 1) + " is " + projectedTrainFaceDistance);

// LOGGER.info(" distance between them " + d\_i);

// LOGGER.info(" distance squared " + distSq);

// }

}

if (distSq < leastDistSq) {

leastDistSq = distSq;

iNearest = iTrain;

LOGGER.info(" training face " + (iTrain + 1) + " is the new best match, least squared distance: " + leastDistSq);

}

}

// Return the confidence level based on the Euclidean distance,

// so that similar images should give a confidence between 0.5 to 1.0,

// and very different images should give a confidence between 0.0 to 0.5.

float pConfidence = (float) (1.0f - Math.sqrt(leastDistSq / (float) (nTrainFaces \* nEigens)) / 255.0f);

pConfidencePointer.put(pConfidence);

LOGGER.info("training face " + (iNearest + 1) + " is the final best match, confidence " + pConfidence);

return iNearest;

}

public String modifyName(String name) {

try {

name = name.trim();

name = name.replaceAll("[ ]+", "\_");

return name;

} catch (Exception e) {

}

return name;

}

public String getOrgnlName(String name) {

try {

name = name.trim();

name = name.replaceAll("[\_]+", " ");

return name;

} catch (Exception e) {

}

return name;

}

// public static void main(String[] args) {

// new MarkAttendence().recognizeFromCam();

// }

}

**4.1.10. FUNCTION PACKAGE:**

package aevent;

import java.awt.Button;

import java.awt.TextField;

import javax.swing.\*;

import javax.swing.event.\*;

class Jevent extends JFrame implements AncestorListener{

TextField o;

JEvent(){

o=new TextField();

o.setBounds(60, 50, 170, 20);

Button b= new Button("click me");

b.setBounds(100,120, 80, 30);

b.addActionListener(this);

add(b);

add(o);

setSize(300,300);

setLayout(null);

setVisible(true);

}

}

public class Aevent {

public static void main(String[] args) {

}

@overload

function overload call over

do{

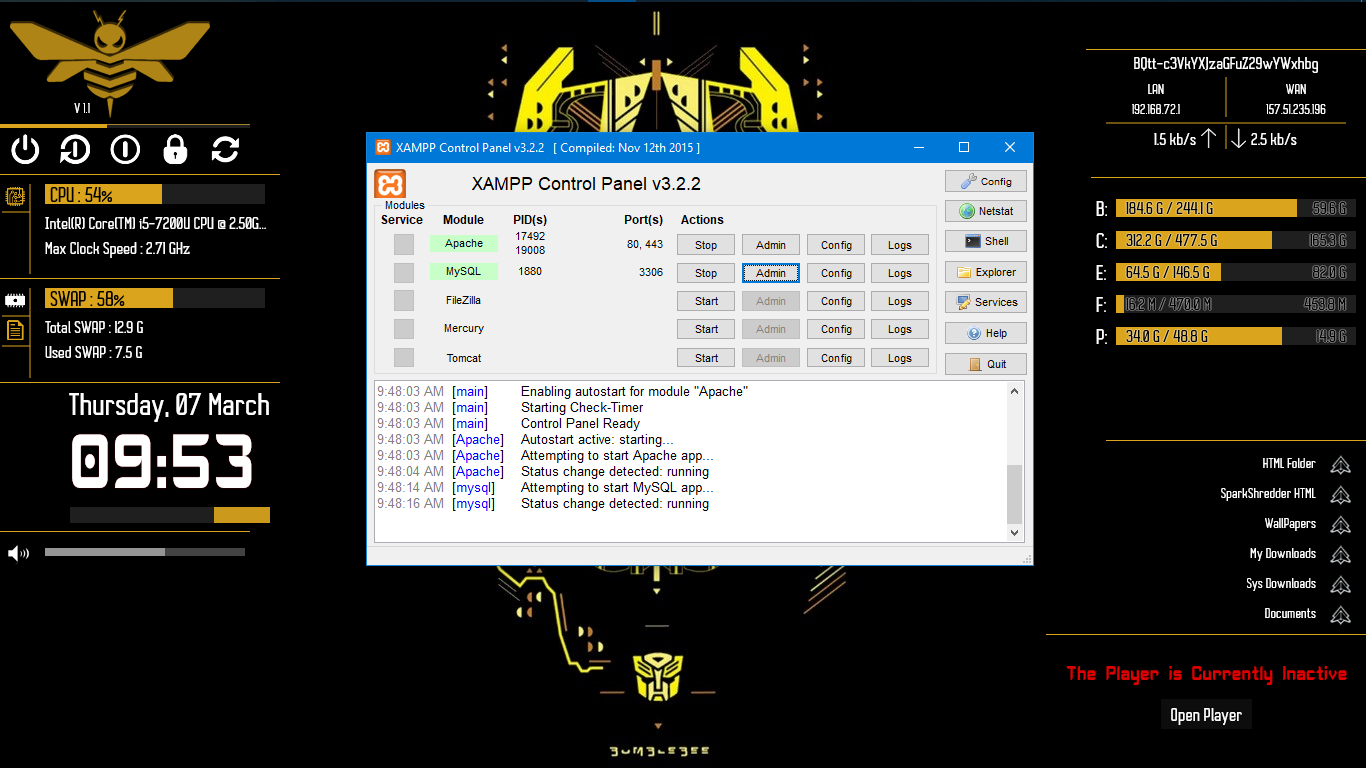
Exit(0)

}

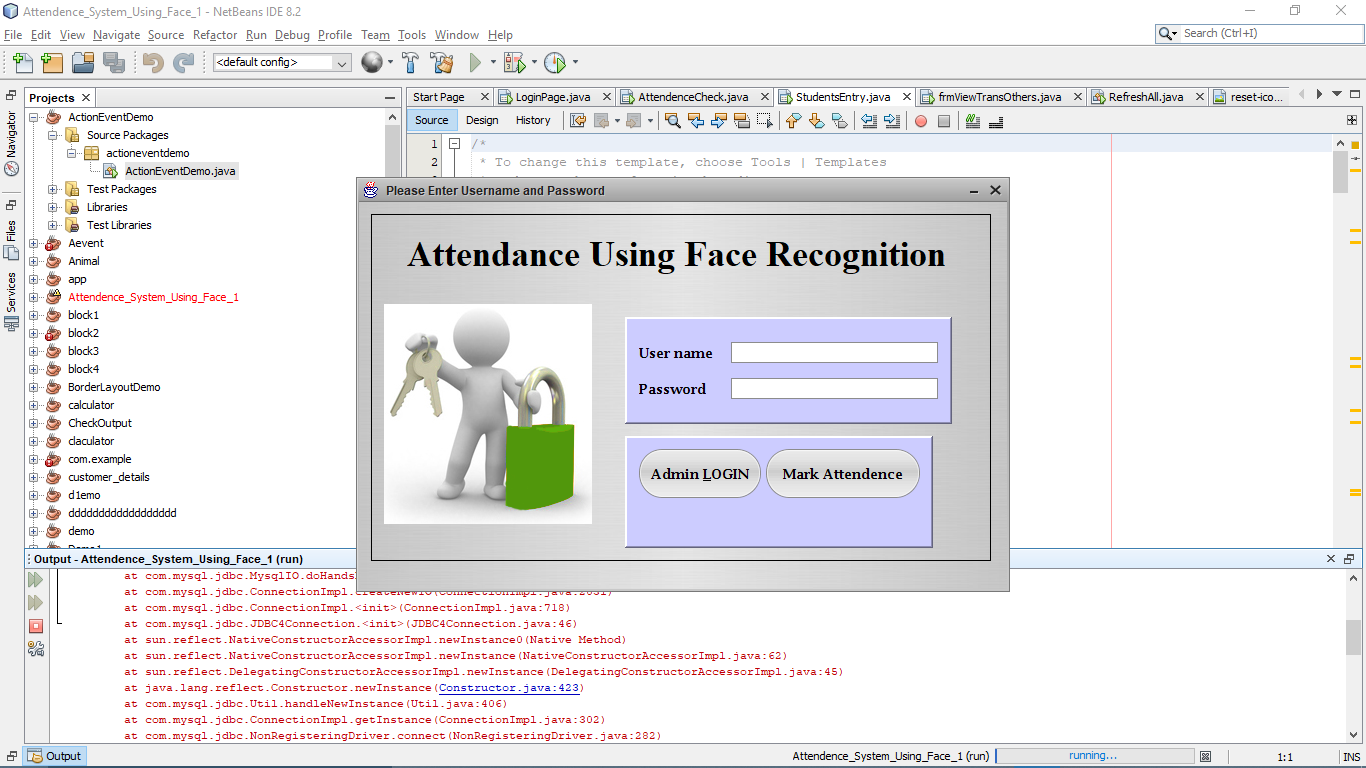
}function=00;

**4.2. SCREEN SHOTS**

4.2.1. XAMPP CONTROL PANNEL



4.2.2. FRAMES GET OPEN EVEN MINIMIZING THE NETBEANS IDE

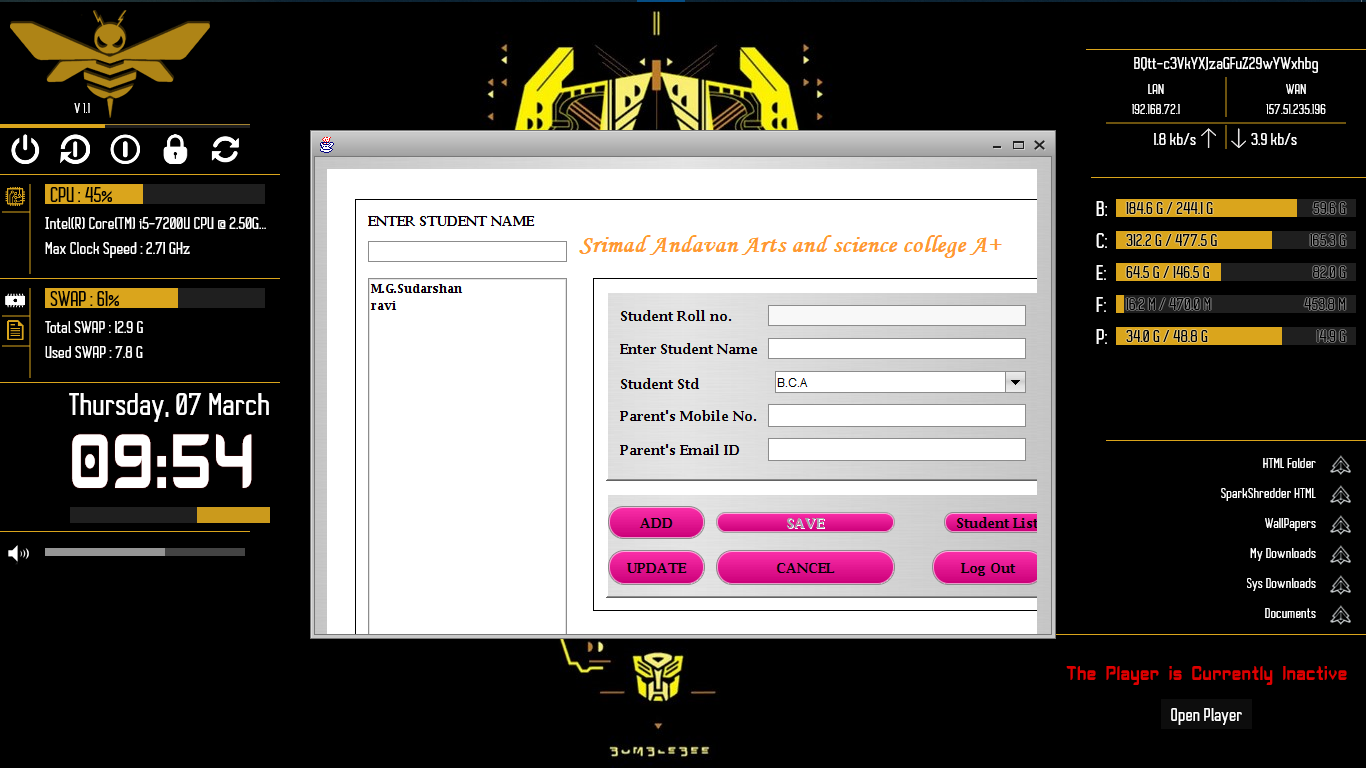




4.2.3. ENTERING USERNAME AND PASSWORD



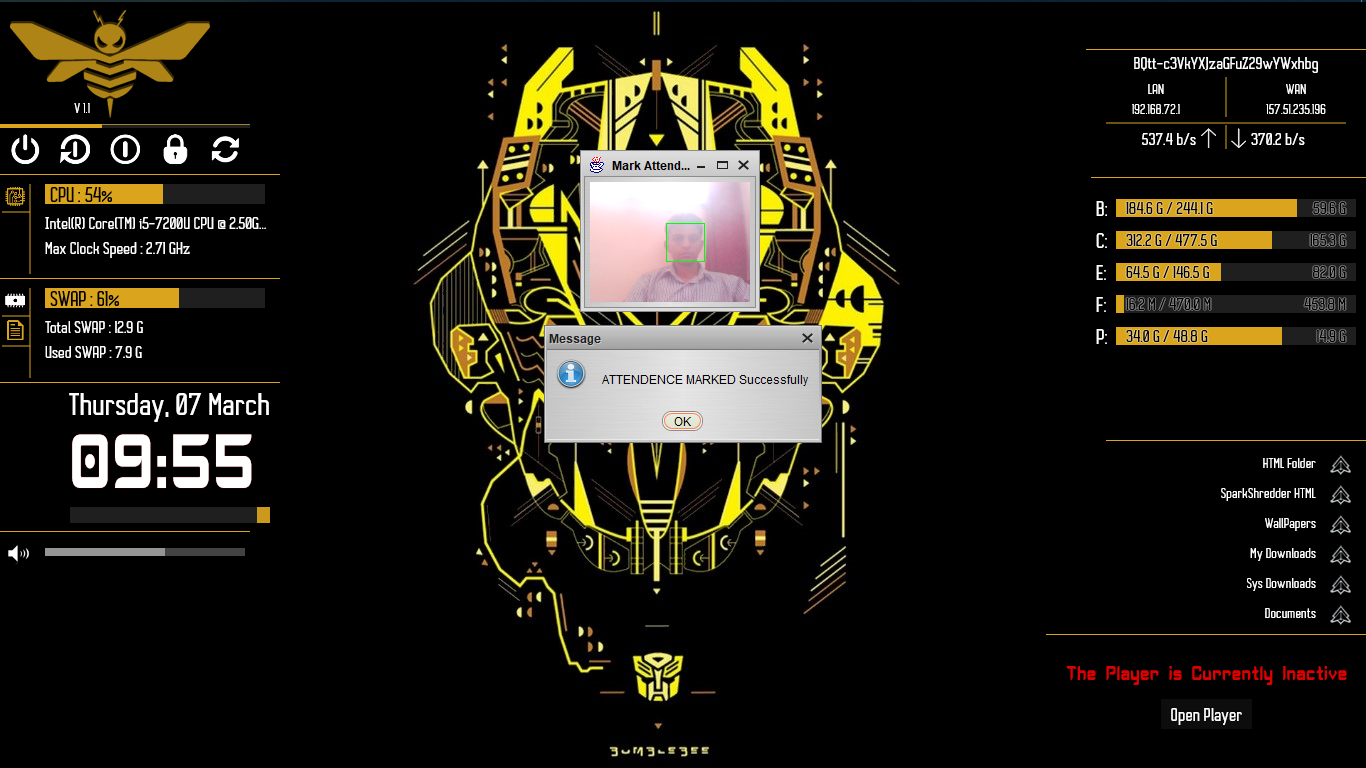
4.2.4. REGISTERING STUDENTS DETAIL



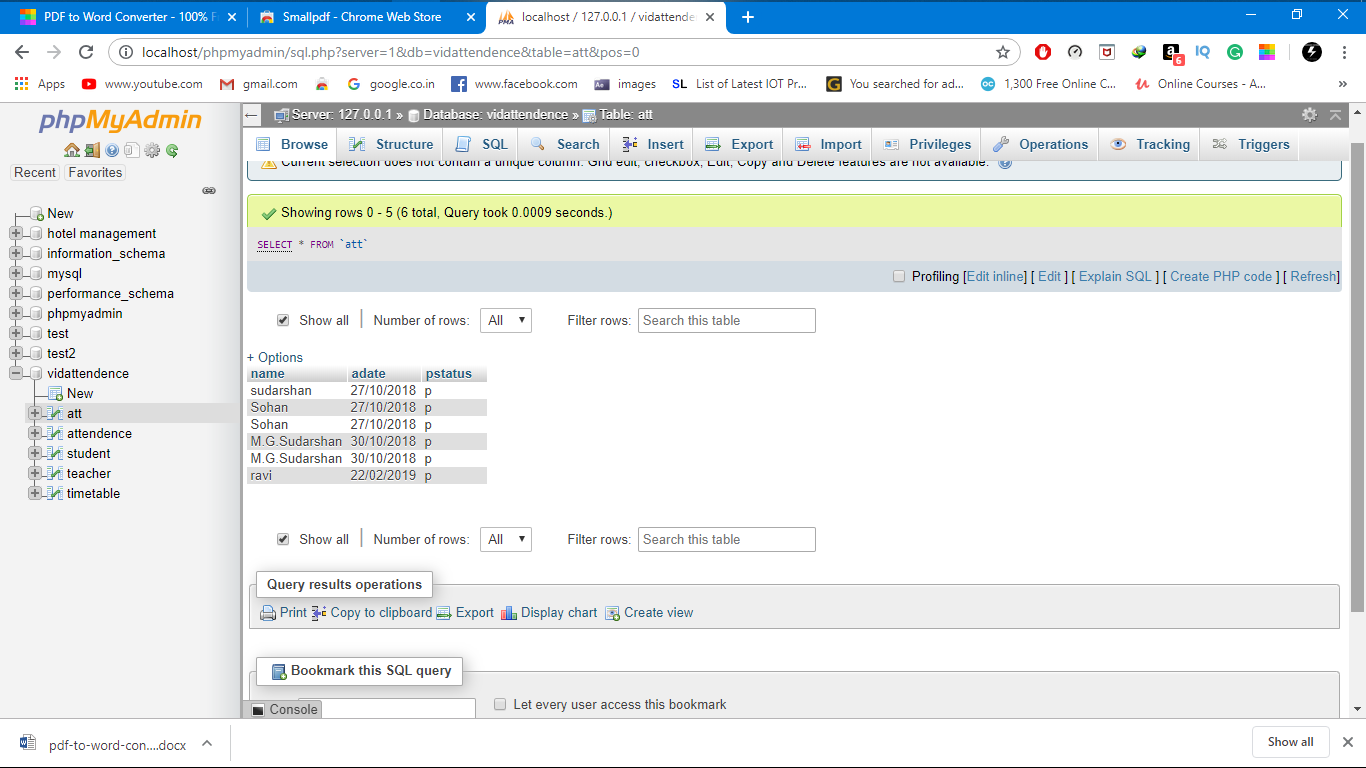
4.2.5. AGAIN, LOGIN PAGE FOR MARKING ATTENDENCE

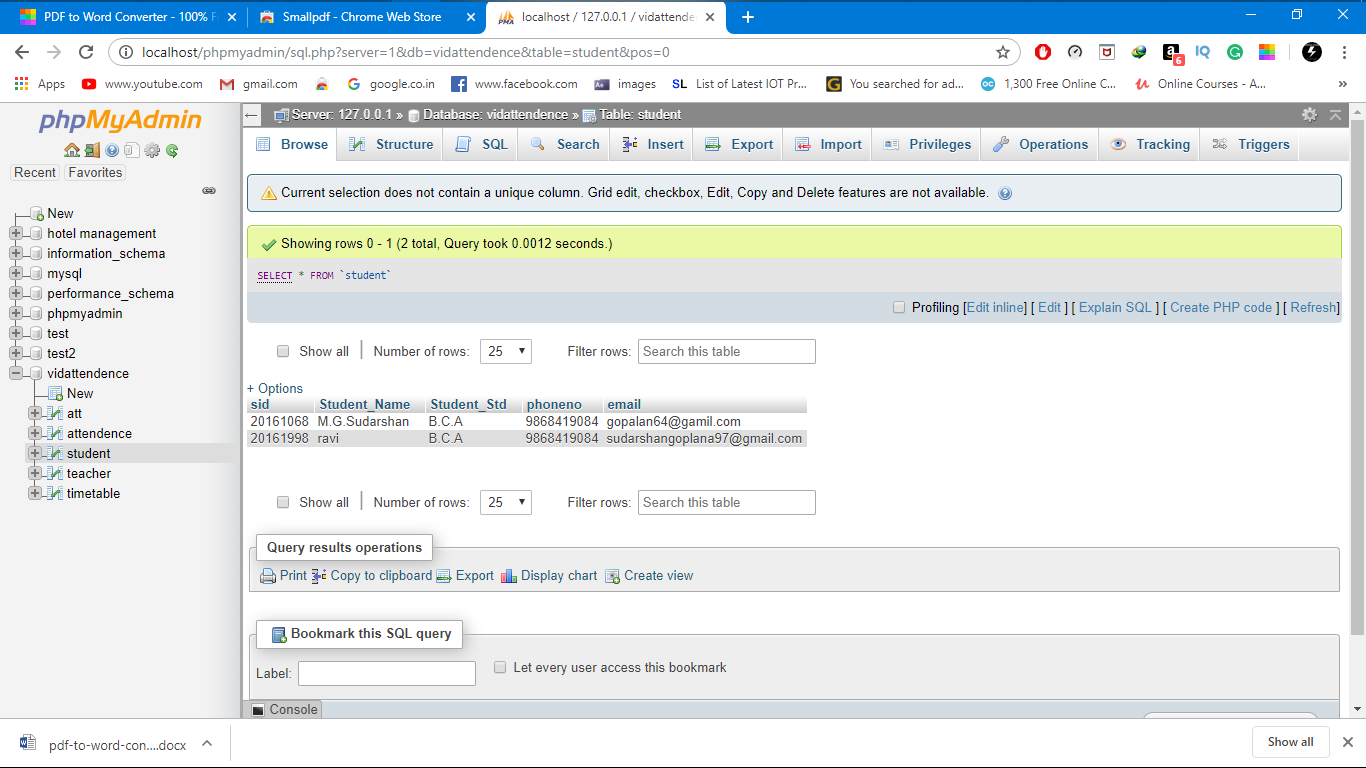


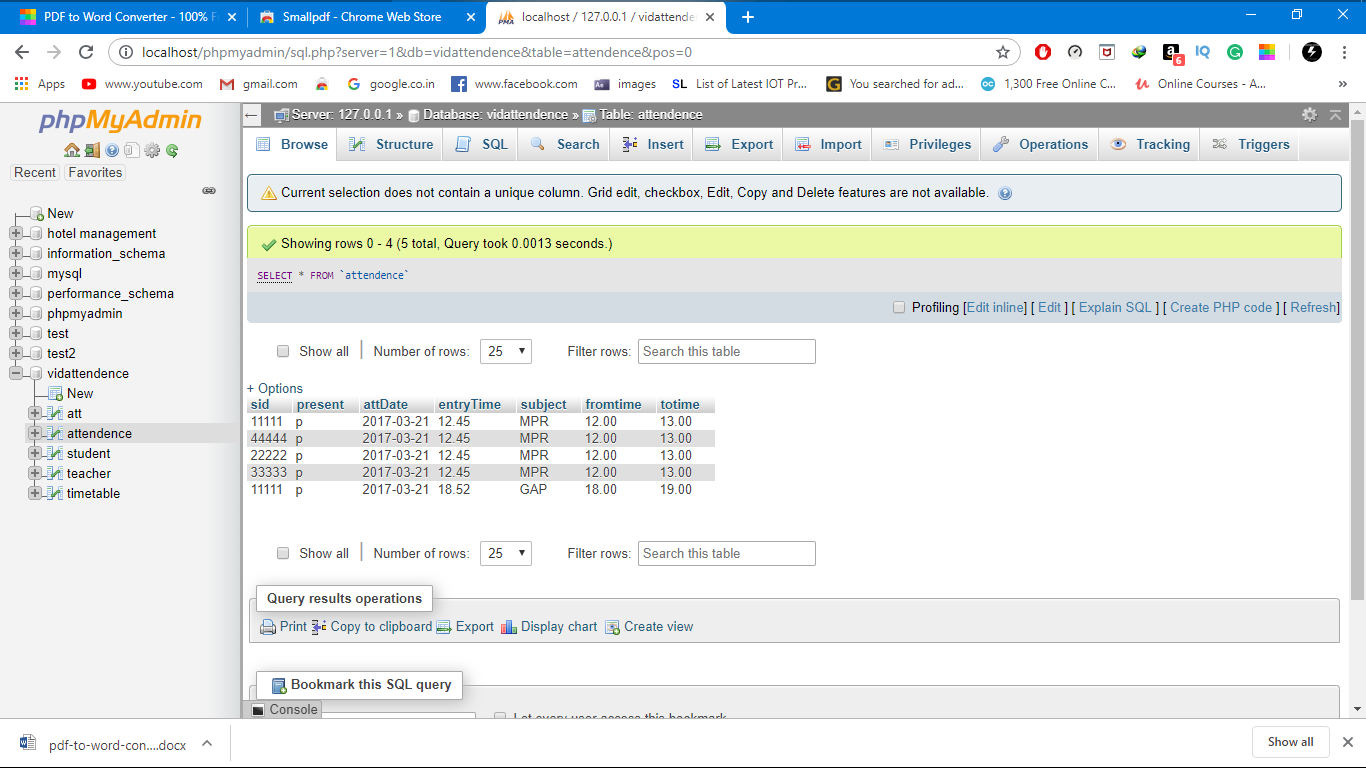
4.2.6. CAMERA OPENS AND MARKS ATTENDENCE



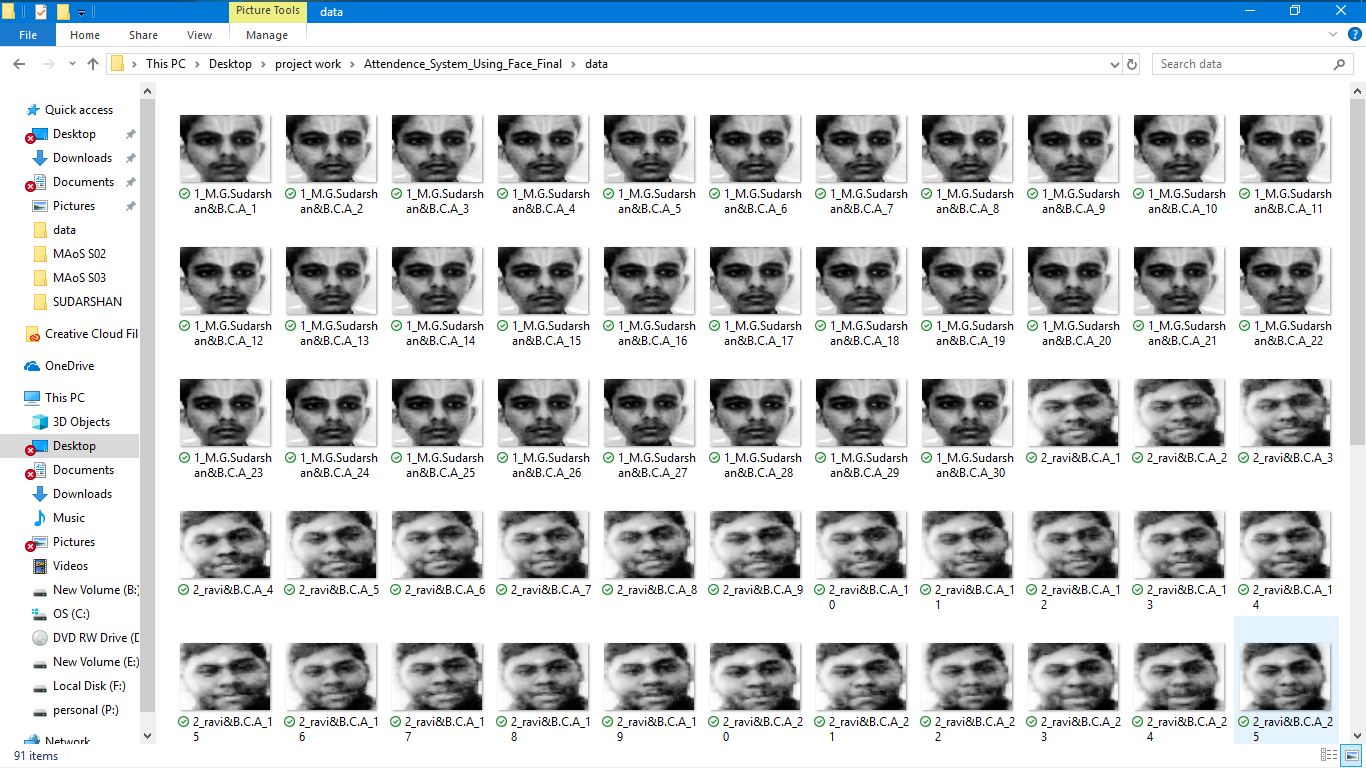
4.2.7. DATA BASE







4.2.8. IMAGE SAVED IN CLOUD AS JPG. FILE



**TESTING**

**5. TESTING**

**5.1. Camera**

Challenge: The biggest challenge is to capture quality images of all the people in a moving vehicle. Environmental conditions such as changing lighting, wide ranging light levels, windshield reflection, varied weather conditions, haze, motion blur caused by moving vehicle, etc. may present challenge in capturing a usable image.

Solution: Selecting the right camera is the most important task.

Some of the critical features that we are looking for in a camera are:

Wide dynamic range (WDR): For this project, we expect surveillance scenes with very bright and dark areas hence a camera with wide dynamic range may provide the best solution. WDR cameras often incorporate an image sensor that takes different exposures of a scene (e.g., short exposure for very bright areas and long exposure for dark areas) and combine them into one image, enabling objects in both bright and dark areas of a scene to be visible. We will be using cameras with built-in WDR.

Infrared (IR) illuminators: Built-in IR LEDs send out near infrared light that allows cameras to capture good quality black and white images in darkness. This near Infrared light is not visible to the human eye. We will be using cameras with IR of 50 feet or more.

Resolution: Cameras with Megapixel sensors offer great detail in the image which would give us the best option to capture faces; most face detection algorithms need minimum 60 pixels between eyes. We will be testing cameras with resolution as high as 20MP. While high resolution cameras give a more detailed image, they also increase the bandwidth and storage expense.

Lens’ light gathering ability (f-number): Camera lens’ with the small f-number can gather more light and perform better in low-light settings. Lens’ f-number, exposure control settings, shutter speed, focal length, image sensor and image processing all play an important role in getting the right image and compensate for low lighting and motion blur. We will be optimizing these settings that get us an ideal image.

Shutter speed Lower shutter speed will give us a sharper image and compensate for motion blur.

Autofocus: We will be using cameras that use a laser beam for range measurement and auto focus. This helps in getting the right focus on the object fast and without distorting the image.

Auto-tracking: We will be using the auto-tracking feature of the cameras. This feature automatically detects, zooms in on and follows moving objects. Once we track a vehicle, we can programmatically zoom into this object to see if we can find a face.

**CONCLUSION**

**6. CONCLUSION**

This system has been proposed for maintaining the attendance record. The main motive behind developing this system is to eliminate all the drawbacks which were associated with manual attendance system.

The drawbacks ranging from wastage of time and paper,till the proxy issues arising in a class, will completely be eliminated.

Hence, desired results with user friendly interface is expected in the future, from the system. The efficiency of the system could also be increased by integrating various steps and techniques in the future developing stages of the system

**BIBLIOGRAPHY**

**7. BIBLIOGRAPHY**

1.Adelson, E. H., and Bergen, J. R. (1986) The Extraction of Spatio-Temporal Energy in

2.Human and Machine Vision, Proceedings of Workshop on Motion: Representation and

3.Analysis (pp. 151-155) Charleston, SC; May 7-9 • AAFPRS(1997). A newsletter from the American Academy of Facial Plastic and Reconstructive Surgery. Third Quarter 1997, Vol. 11

4.Baron, R. J. (1981). Mechanisms of human facial recognition. International Journal of Man Machine Studies, 15:137-178

5.Beymer, D. and Poggio, T. (1995) Face Recognition From One Example View, A.I. Memo No. 1536, C.B.C.L. Paper No. 121. MIT

6.Bichsel, M. (1991). Strategies of Robust Objects Recognition for Automatic Identification of Human Faces. PhD thesis, , Eidgenossischen Technischen Hochschule, Zurich.

7.Brennan, S. E. (1982) The caricature generator. M.S. Thesis. MIT.

8.Brunelli, R. and Poggio, T. (1993), Face Recognition: Features versus Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052

9.Craw, I., Ellis, H., and Lishman, J.R. (1987). Automatic extraction of face features. Pattern Recognition Letters, 5:183-187, February.