



ENEL3SF H2 - Software Engineering 2

Computer vision based class attendance monitoring

214502393 Devashnee Bhagawandin	
214551467 Nivaan Krishundutt	
213508238 Keshav Jeewanlall	
214540248 Adin Arumugam	

Abstract

A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the way is to do this is by comparing selected facial features from the image and a facial database. Our goal was to develop a system to function in a classroom in which a camera is placed at a position where faces of all students attending the lecture can be captured at any time during the lecture. The camera captures the class room at each defined time instant; the photo taken will then be processed to record the presence of students. The technique we used in order to achieve this goal was Linear Discriminant Analysis (LDA), which can be derived from an idea suggested by R.A. Fisher in 1936. When LDA is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces.

Table of Contents

Abstract	i
Table of Figures	iv
Introduction	1
1. Requirements.....	2
1.1. Systems Requirements.....	2
1.2. User Requirements	2
1.3. Functional Requirements	2
1.4. Non-functional Requirements	2
2. Use case diagram and analysis.....	3
2.1 Precondition	3
2.2 Application.....	3
2.3 Use Case Description.....	3
2.3.1 The lecturer.....	3
2.3.2 The main class.....	3
3. Sequence Diagram.....	4
4. Class Diagram	5
4.1. AttendanceStats() class:	5
4.1.1 Methods in this class:.....	5
4.1.2 Attributes in this class:	5
4.2. Main class():.....	6
4.2.1 Methods in this class:.....	6
4.2.2 Attributes in class:.....	6
5. List of files	7
6. Requirement for Usage of the Software.....	8
6.1 Installation of OpenCV	8
6.2 Using the class attendance software	8
7. Implementation.....	9
7.1) Testing of OpenCV libraries.....	9
7.2) Calibration of webcam	9
7.3) Face Detection.....	9
7.4) Facial Recognition.....	10
7.5) Creation of data base.....	11

7.6) Connection to Database.....	11
7.7) Update Database	12
7.8) Displaying Data from Database	12
7.9) Setting Attendance	12
8. Testing.....	13
8.1) Software segments and methods required to be testing:.....	13
8.2) Identify interesting input values	13
8.3) Identify expected results (functional) and execution characteristics (non-functional)	13
8.4) Run the software on the input values	14
8.5) Compare results and execution characteristics to expectations	14
9. Bugs	15
10. Conclusion.....	16
11. References.....	17
12. Appendices.....	A
Appendix A (UML Diagrams)	A
Appendix B (Testing).....	B
Appendix C (Source Code)	H
Main Class Source Code.....	H
Attendance Stats Source Code.....	O

Table of Figures

Figure 1: Code for Face Detection	9
Figure 2: Example of CSV file	10
Figure 3: Code to train the Face Recognizer	10
Figure 4: Code for Face Recognition.....	11
Figure 5: Sequence Diagram	A
Figure 6: Use Case Diagram	A
Figure 7: Testing schematic	B
Figure 8: Integration testing top-down method	B
Figure 9: Control Flow Graph of Setting Attendance	C
Figure 10: Calibration of camera	C
Figure 11: Face Detection	C
Figure 12: Face Recognition	D
Figure 13: Face Recognition with different expressions.....	D
Figure 14: Main Screen	D
Figure 15: Choosing Lecture.....	E
Figure 16: Error Message	E
Figure 17: Taking out pictures.....	E
Figure 18: Data Base connected.....	E
Figure 19: working program.....	F
Figure 20: input data	F
Figure 21: Statistics Screen	G

Introduction

This system is designed to mark the attendance of a class during a lecture period using facial recognition software. The system is able to determine whether a student is either present for the lecture, late for the lecture, left the lecture early or absent for the lecture. The attendance statistics for the entire semester can also be viewed for each student.

The system captures an image of the class four times with a 10 minute interval between each image captured. From the captured image, faces are detected, recognised and processed. The results are stored in a database. After the final capture, the results from the four captures are processed and the respective presence is allocated to the student for that lecture day. For example; if a student was recognized in all four captures, that student is marked present for that lecture day.

Source control was used during the development of this software. This was advantageous as this software included many developers. The software source code can be found on github ^[1].

1. Requirements

1.1. Systems Requirements

- i. Frequently captures an image of the class
- ii. Extract faces from the image
- iii. Compare with database and if match found, mark student present
- iv. Record attendance stats

1.2. User Requirements

- i. Take attendance of a classroom/lecture venue
- ii. Periodically assess the attendance

1.3. Functional Requirements

- i. When an image is captured, it is processed to extract faces.
- ii. When a face does not match any of the faces in the database, a message must be sent.
- iii. When a face is recognised, marks present.
- iv. When a face from the database is not in the image, the person is marked absent.
- v. System must be able to detect if a student leaves a classroom.

1.4. Non-functional Requirements

- i. Timing
- ii. Picture quality
- iii. Processing time
- iv. Accuracy

2. Use case diagram and analysis

2.1 Precondition

The lecturer is required to provide the information of the students registered for the module together with identity photos of the students to be recorded in the database.

The actors associated with the use case are:

- i. the lecturer of the module
- ii. the main class

2.2 Application

The use case describes the interactions between different actors associated with a facial recognition software programmed to display the attendance pattern of the class over a period of a semester.

2.3 Use Case Description

2.3.1 The lecturer

The lecturer of the system is required to start the program .The lecturer also is able to view the attendance pattern of the students.

2.3.2 The main class

The main class is programed to take an image of the class periodically and detect faces present in the image. The detected faces are run through facial recognition software implemented using OpenCV and C++. The facial recognition software extracts images from an external database and does its comparison.

When facial recognition software detects a face present in the image captured during a lecture period, the system should record the face as present and upload it to the database for later analysis. The default setting of the program will be students marked absent if their face is not detected. An error setting is established when an unrecognized face is detected in the image.

The main class should perform analysis of the information and display this information in the form of a table showing the attendance pattern of the students.

See Appendices for the Use Case Diagram.

3. Sequence Diagram

- i. The lecturer starts the program and chooses the lecture number.
- ii. The main class implements the *captureImage()* method which captures an image of the class. This method is also responsible for detecting and recognizing faces in the captured image.
- iii. The *setAttendance()* method then matches the detected faces with previously stored images of each student. If match is found, the main class then sends data to the database.
- iv. The lecturer provides a student number to the Attendance Stats class. This class uses *getData()* method in which it reads data from the database, processes it and then displays the stats.

See Appendices for the Sequence Diagram.

4. Class Diagram

4.1. AttendanceStats() class:

The responsibility of this class is to acquire and display, using the program's database, the statistical information of the students. The information that this class will give the user is the student's name, surname, identity number, number of days present, number of days absent, number of days the student arrived late and number of days the student left early. This class will access the program's database, using SQL code, for the required information and then store this information in the respective variables.

4.1.1 Methods in this class:

Method	Parameters	Return Type	Function
getData()	None	void	Method getData() will obtain the required information from the database.
displayData()	None	void	Method displayData() will display the acquired information from getData() on the user's screen.

4.1.2 Attributes in this class:

Attribute name	Data type	Function
studentName	string	Stores the name of the student
studentSurname	string	Stores the surname of the student
studentID	int	Stores the identity/student number of the student
NoOfDaysPresent	int	Stores the number of days the student attended the class
NoOfDaysAbsent	int	Stores the number of days the student did not attend the class
NoOfDaysCameLate	int	Stores the number of days the student was late for class
NoOfDaysLeftEarly	int	Stores the number of days the student left the class early

4.2.Main class():

This class is the core of the program. It is responsible for the capturing of the image and storing the image in the database, detection of faces and setting the attendance of the student and storing this information in the database. This class will have the main method that will call all other classes and run the program. The code used in the method captureImage() to capture an image will make use of the computer's webcam in order to capture an image. The method detectFaces() will then use this captured image in order to detect the faces of students and compare the detected faces to the ones already stored in the database. The main() method will call all other methods in this program and run the program.

4.2.1 Methods in this class:

Method	Parameters	Return type	Function
captureImage()	None	void	Captures an image of the students in the venue. Detects and recognizes faces from the captured image.
setAttendance()	vector	void	Sets student attendance as either "Absent", "Present", "Arrived late", "Left early" for a specific lecture number in the database.
main()	none	void	calls all other methods and runs the software

4.2.2 Attributes in class:

Attribute name	Data type	Function
lecture_Number	int	Stores the current lecture number
faces	Vector <Mat>	Stores the images of the faces in the database
Ids	Vector <Int>	Stores the IDs of the faces in the database
quarter	int	Controls the number of times to capture an image

5. List of files

- i) *opencv_calib3d2411d.lib*
- ii) *opencv_contrib2411d.lib*
- iii) *opencv_core2411d.lib*
- iv) *opencv_features2d2411d.lib*
- v) *opencv_flann2411d.lib*
- vi) *opencv_gpu2411d.lib*
- vii) *opencv_highgui2411d.lib*
- viii) *opencv_imgproc2411d.lib*
- ix) *opencv_legacy2411d.lib*
- x) *opencv_ml2411d.lib*
- xi) *opencv_nonfree2411d.lib*
- xii) *opencv_objdetect2411d.lib*
- xiii) *opencv_photo2411d.lib*
- xiv) *opencv_stitching2411d.lib*
- xv) *opencv_ts2411d.lib*
- xvi) *opencv_video2411d.lib*
- xvii) *opencv_videostab2411d.lib*
- xviii) *csv_file.csv*
- xix) *studentattendancedb.mwb*

6. Requirement for Usage of the Software

6.1 Installation of OpenCV

The OpenCV source files were downloaded from the official website ^[2]. This file is self-extracting; the user will have to double click the extract button to start the extraction process. Once the process is completed the extracted file should be pasted I the C drive of the computer.

6.2 Using the class attendance software

When the system is started, the Main Screen is displayed and the user is required to choose a lecture number from a combo box containing value ranging from 1 to 25, as this is the number of lectures per semester. If attendance for that lecture number has already been captured, an error message is given and the user has to choose another lecture number.

Once the lecture number is chosen, the first of four images is captured. The image is processed and faces are detected. These faces are then recognised and the recognised students are marked present for the first quarter of the lecture.

A timer is then started to run for 10 minutes. When the 10 minutes are up, the second image is captured and the same routine is done as in the first quarter. This process is repeated for the third and final quarter.

After the final quarter, the data is analysed and the attendance is set for the specified lecture number.

The user can access the Attendance Statistics at anytime while the application is running. This is achieved by pressing the “View Attendance Statistics Screen” button on the Main Screen. A new screen will appear where the user can enter a student ID. By pressing the search button, the attendance statistics of the specified student ID will be displayed.

7. Implementation

Open source computer vision (OpenCV) currently has the following available algorithms to perform facial recognition; Eigenfaces, Fisherfaces and Local Binary Patterns Histograms (LBPH). The algorithm chosen for this project was Fisherfaces.

7.1) Testing of OpenCV libraries

The OpenCV source files were downloaded from the official website ^[2] and installed following the necessary procedures ^[3]. The library files needed to be linked to Visual Studio as this was the IDE used to design and develop this software.

A simple test of capturing an image and displaying it using OpenCV was conducted thus proving that the linking of the OpenCV library files to Visual Studio was a success.

7.2) Calibration of webcam

When capturing an image using OpenCV, a camera ID number is sent as a parameter in order for that camera to be used. This works well if multiple cameras are on a computer system. This software was designed to use a system's default camera therefore by sending a 0 in place of the camera's ID, the default camera will be used to capture an image.

7.3) Face Detection

In OpenCV, Haar-Cascades are used to detect faces in an image. In this software, the "haarcascade_frontalface_default.xml" file was used to detect faces in an image.

```
vector<Rect_<int>> facePositions;
haar_cascade.detectMultiScale(greyImage, facePositions);
```

Figure 1: Code for Face Detection

Figure 1 shows the function that is used to detect faces in an image. The image is passed as a parameter in the variable *greyImage*. The function detects the faces in the image using the haar-cascade defined earlier and then stores the positions of the detected faces in a vector of integers called *facePositions*. Vectors are used instead of arrays due to the fact that the number of faces detected in an image can not be previously determined.

7.4) Facial Recognition

The fisherfaces method requires multiple images of a person as these images are compared to a face from the image captured and a prediction is made. A larger sample of images results in greater accuracy of the prediction.

A CSV file is created in order to group together all the images of one person. Each image of a person is assigned the unique ID of that person.

```
C:/Class_Attendance_System_Files/image1.jpg;214551467
C:/Class_Attendance_System_Files/image2.jpg;214551467
C:/Class_Attendance_System_Files/image3.jpg;214551467

C:/Class_Attendance_System_Files/image4.jpg;214502393
C:/Class_Attendance_System_Files/image5.jpg;214502393
C:/Class_Attendance_System_Files/image6.jpg;214502393
```

Figure 2: Example of CSV file

Figure 2 shows an example of the CSV file that is created. Each line contains an image path then followed by a delimiter; in this case a semicolon is used; which is then followed by the ID of the person in that image.

When reading the CSV file, for each line that is read, the image is stored in a vector of images and the ID of that image is stored in a vector of integers. These vectors are going to be used to train the Face Recognizer.

In order for the fisherfaces prediction to be accurate, the images used for comparisons and the captured image both have to be in greyscale. OpenCV contains a function `cvtColor(Mat, Mat, int)` which can convert an image to greyscale.

```
Ptr<FaceRecognizer> model = createFisherFaceRecognizer();
model->train(faces, ids);
```

Figure 3: Code to train the Face Recognizer

The first line of code in Figure 3 creates a Fisherface recognizer that is used to perform Face Recognition. The second line of code utilizes a function called `train` which accepts two parameters. The first parameter is a vector of images. The second parameter is a vector of integers. These vectors were both created when the CSV file was read therefore these vectors contain the images that are used for comparisons and their respective IDs. By sending these parameters, the Face Recogniser is being 'trained' to recognise the faces that were in the images of the vector.

```

Rect face_i = facePositions[i];
Mat getFace = greyImage(face_i);
Mat face_resized;
cv::resize(getFace, face_resized, cv::Size(im_width, im_height), 1.0, 1.0, INTER_CUBIC);
int foundID = model->predict(face_resized);

```

Figure 4: Code for Face Recognition

Refer to figure 4, since the positions of the detected faces in the captured image are known, a face is extracted from the image, turned to greyscale and resized to match the size of the images used for comparisons. This is then stored in the variable `face_resized`.

Using the trained Fisherface Recognizer, the image `face_resized` is then sent as a parameter to the `predict` function. This function compares `face_resized` with the images that it was trained with. If a match is found, it returns the ID of the matching image thus completing the face recognition process.

7.5) Creation of data base

The database was created using MySQL Workbench which is a graphical tool for working with MySQL Servers and databases. The database created for this system was called 'studentattendancedb'. This database contains one table called 'studentattendancetbl' which stores the details students and their attendance statistics in the module for the system. The table has 32 columns which stores the student's ID, name, surname, and lectures 1 – 25 for 25 lectures per module per semester. StudentID column is of datatype INT(11), is set to not null and is our primary key because each student has a unique student number. All other columns are of type VARCHAR(45) and is set to not null. The database used in this software has a data source as localhost with the username as 'root', the port as 33063 and the password as 'password@0105'.

7.6) Connection to Database

Connection to the database was done by connecting to MySQL using .NET. This component adds .NET drivers for MySQL to Visual Studio which allows us to write .NET code that accesses MySQL databases. The 'Connection', 'Command', 'DataReader', 'DataSet', and 'DataProvider' are the core elements of the .NET model. The 'Connection' creates a connection to a specific data source. To do this, we added a reference to the MySQL .NET library in our project and specified the correct parameters in a database connection string.

7.7) Update Database

The MySQL update statement was used to update the table in the database 'studentattendancedb' called 'studentattendancetbl'. The update method identifies the database to be updated. The SET clause specifies which column that is wants to modify. The WHERE clause identifies the primary key to be updated. If the where clause is omitted, all the rows in the table will be updated. The primary key in the database called studentID. When updating the database remember to connect the database to the method allowing it access.

7.8) Displaying Data from Database

To read data from the database we used the MySQLDataReader, which is an object used to retrieve data from the database. It provides fast, forward-only, read-only access to query results. To create a MySQLDataReader, we called the ExecuteReader() method of the MySqlCommand object. The Read() method was used to advance the data reader to the next record. It returns true if there are more rows; otherwise false. The required data was then displayed in their respective text fields.

7.9) Setting Attendance

Since an image is captured four times during a lecture, four string variables are created i.e. quarter1, quater2, quater3 and quater4. These variables contain whether a student was absent or present during each quarter of the lecture.

For each face detected in quater1, the students are marked present for quater1. For the faces that weren't detected in that quarter, the student is marked absent for that quarter. This process is repeated for the remaining three quarters. Once all four images are captured, the system then analyses all four quarters for each student.

If a student was present for all four quarters, the student is present for that day. If a student was present for the first three quarters but absent in the last quarter, the student is marked 'left early'. If a student is absent for the first quarter but present for the remaining three, the student is marked 'came late'. If a student is absent for all four quarters, the student is marked absent for the day. These results are stored in a database for future references.

8. Testing

8.1) Software segments and methods required to be testing:

- i) Testing of OpenCV libraries
- ii) Calibration of webcam
- iii) facial detection
- iv) facial recognition
- v) connection to database
- vi) update database
- vii) displaying data from database
- viii) setting attendance method

8.2) Identify interesting input values

- i) Software is to be tested using students who are already stored in the CSV file
- ii) Software is to be tested using students who are Not stored in the CSV file
- iii) Pictures of the students stored in the CSV file
- iv) Pictures of the students who are not stored in the CSV file

8.3) Identify expected results (functional) and execution characteristics (non-functional)

- i. OpenCV libraries links up visual C++ to open cv
- ii. Webcam captures images to be analysed later
- iii. Facial detection correctly identifies students faces from images
- iv. Facial recognition method predicts faces in the CSV file
- v. Database must open from visual C++
- vi. visual C++ is updates the database
- vii. visual accesses database and correctly displays data
- viii. Setting attendance method then shows whether student was present or absent in the lecture

8.4) Run the software on the input values

- i. A student was to be present during the whole lecture
- ii. A student was to be absent throughout the entire lecture
- iii. A student was to arrive late for a lecture
- iv. Student not a part of database was present for the lecture

8.5) Compare results and execution characteristics to expectations

- i. OpenCV was linked to visual C++ through the OpenCV library
- ii. Webcam successfully captured images in quarters
- iii. Facial detection method correctly identified faces from other objects in images
- iv. Facial recognition correctly identified students and predicted files in CSV file
- v. The connection to the database was successful and correctly opened database from visual C++
- vi. Database was updated by visual C++
- vii. Data was correctly displayed from database when accessed by visual C++
- viii. Attendance method then showed whether students were present or absent during lecture

9. Bugs

The software however does not perform 100% of its duties as there are scenarios that can cause the software to output incorrect data.

- i. If a student were to use a cap to class the software would not recognise him/her as they would not be using a cap in the image in the database.
- ii. Any change made to the students face such as glasses being worn could also result in a failure due to a 100% match not being made.
- iii. Bad room lighting can also cause the system to not perform correctly as the images would not be clear enough to identify the students.
- iv. If the student were to put his/her head down during image capturing they will be marked absent as a result of their face not being detected.
- v. Because facial recognition uses 2D images, they are not invariant to facial poses, facial accessories and expressions which makes results in poor recognition.

10. Conclusion

In the past facial recognition software have been usually associated with very costly applications and hardware. Today however due to the ever evolving world and modern technology, the cost of developing and even buying such software has decreased, leading to more affordable ways to obtain deploy facial recognition for purposes such as taking attendance records. Our software was inexpensive to develop and has a high accuracy rate. It not only fulfils the purpose of detecting students in the class and marking them present but does so efficiently.

FURTHER DEVELOPMENT

- i. The attendance stats captured for the day could be emailed to the lecturer signed in instead of him logging in to see attendance stats all the time.
- ii. Alert lecturer if a student that doesn't belong to the class is present.
- iii. Notify lecturer if a student is frequently missing class.
- iv. Takes picture of lecturer and logs in for him to save lecturer time taken to login to system.
- v. System can generate a report at the end of a semester about a student's attendance stats.
- vi. System can be available in many languages to make interaction with lecturer more convenient.
- vii. System should go into sleep mode after acquiring image in order to save power and restart once next image is to be captured.

11. References

- 1) Github (2016) *Software Engineering 2 Project* [Online] Available from:
https://github.com/Nivaan214551467/Software_Engineering_2_Project.
- 2) OpenCV (2016) *DOWNLOADS* [Online] Available from:
<http://opencv.org/downloads.html>
[Accessed: 1st October 2016]
- 3) OpenCV (2014) *How to build applications inside the Microsoft Visual Studio* [Online]
Available from:
http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html
[Accessed: 1st October 2016]

12. Appendices

Appendix A (UML Diagrams)

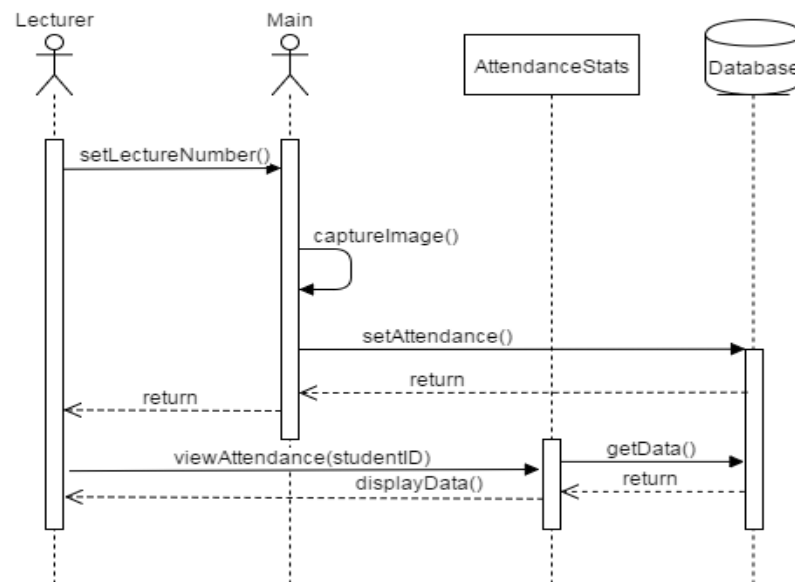


Figure 5: Sequence Diagram

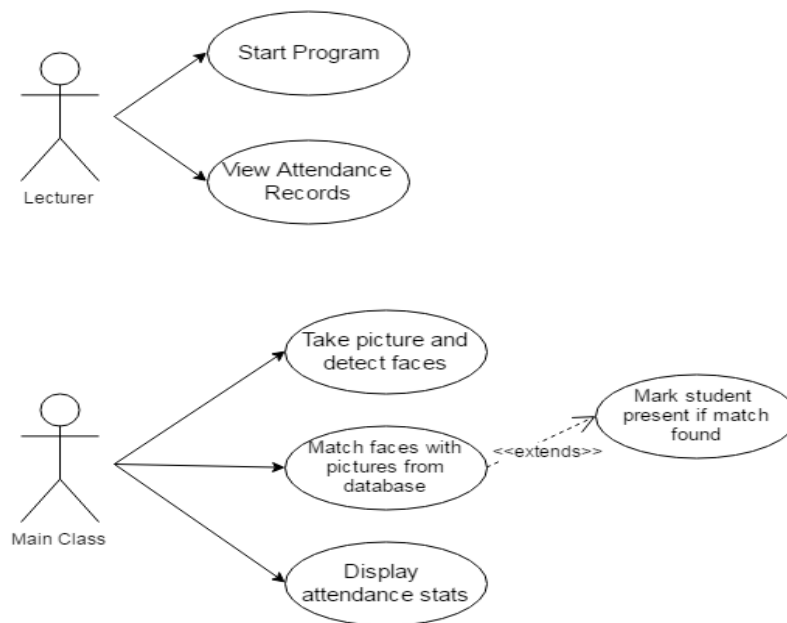


Figure 6: Use Case Diagram

Appendix B (Testing)

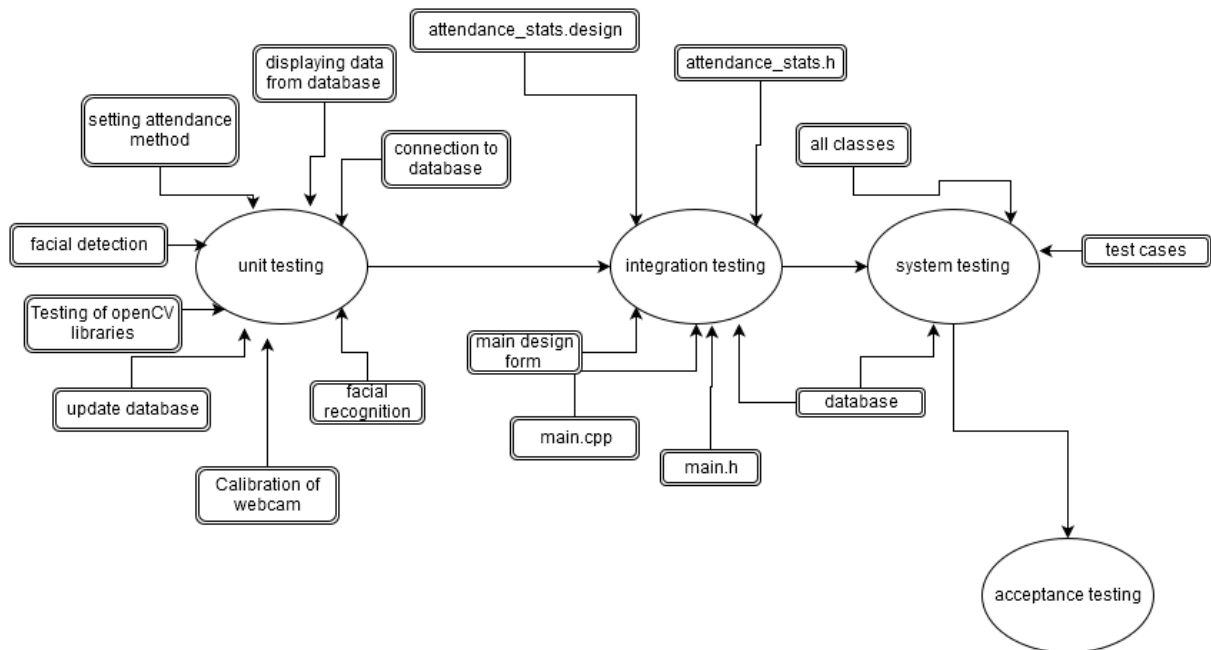


Figure 7: Testing schematic

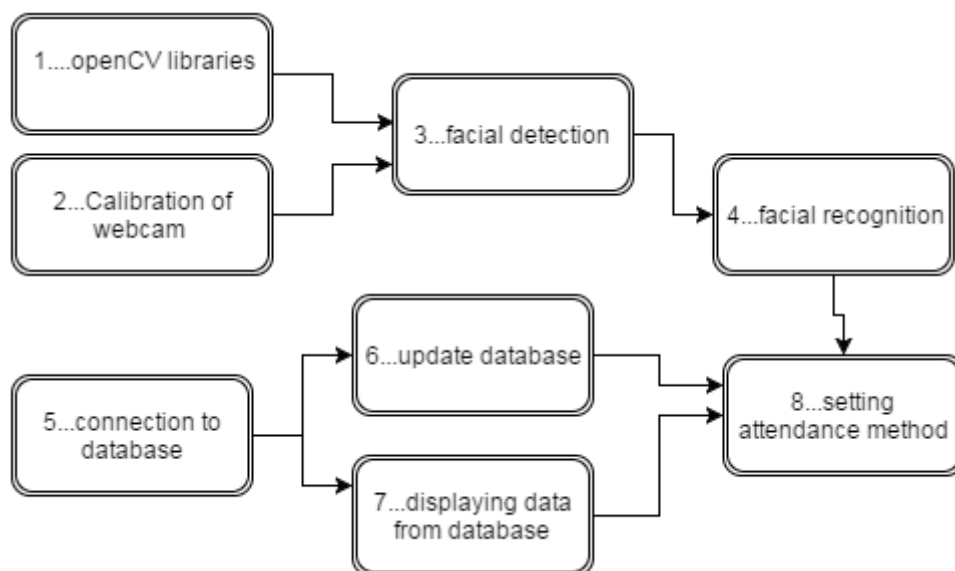


Figure 8: Integration testing top-down method

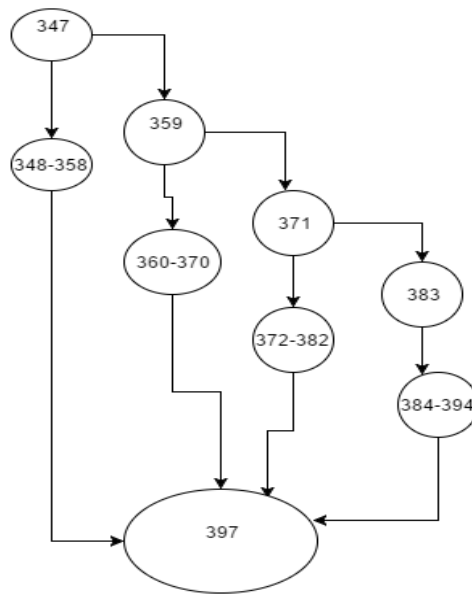


Figure 9: Control Flow Graph of Setting Attendance

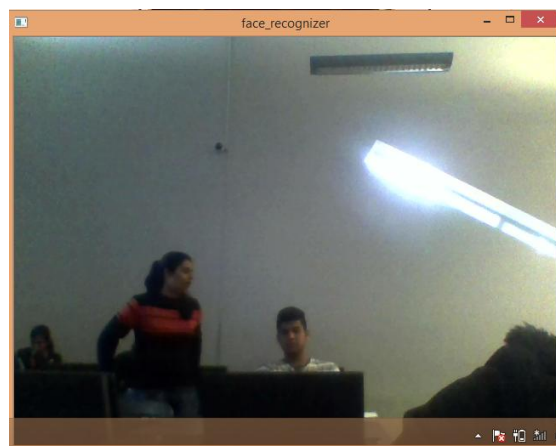


Figure 10: Calibration of camera

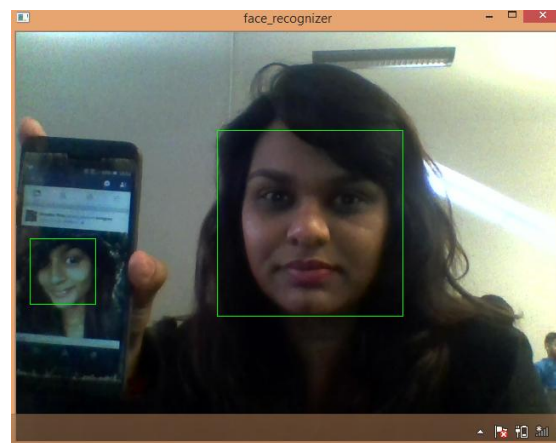


Figure 11: Face Detection

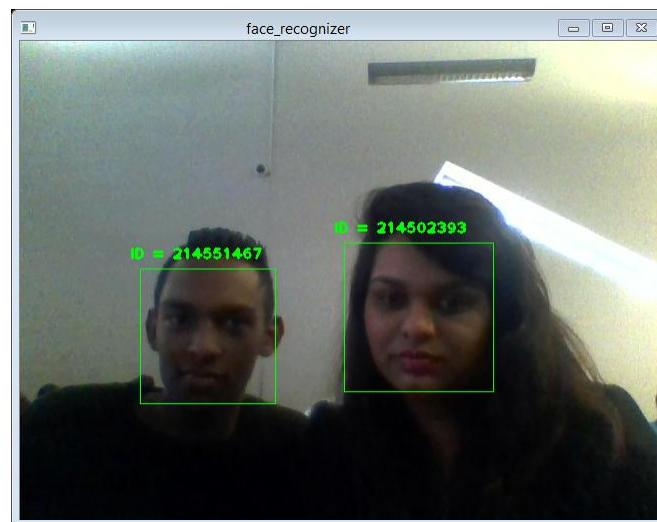


Figure 12: Face Recognition

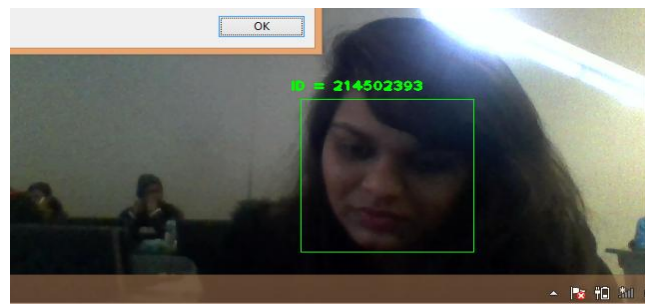


Figure 13: Face Recognition with different expressions

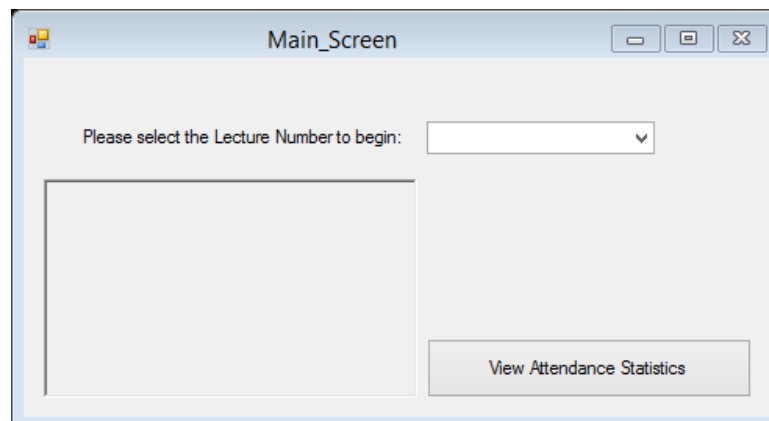


Figure 14: Main Screen

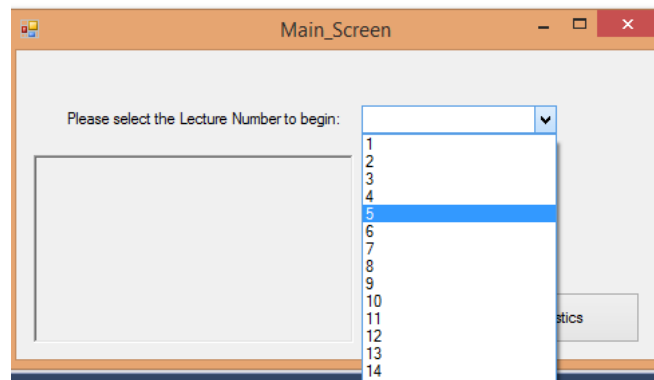


Figure 15: Choosing Lecture

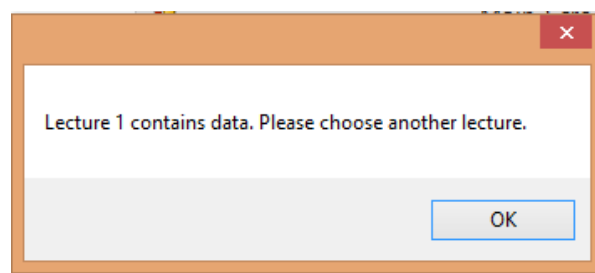


Figure 16: Error Message

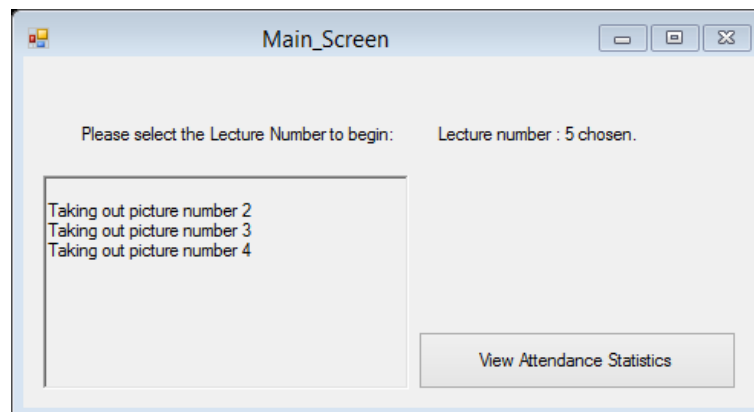


Figure 17: Taking out pictures

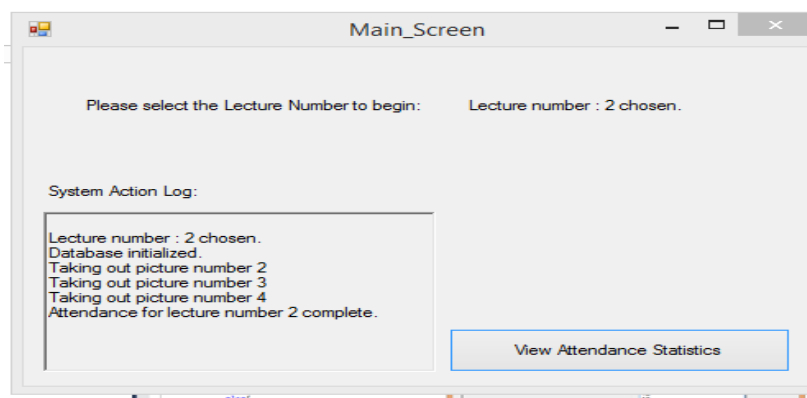


Figure 18: Data Base connected

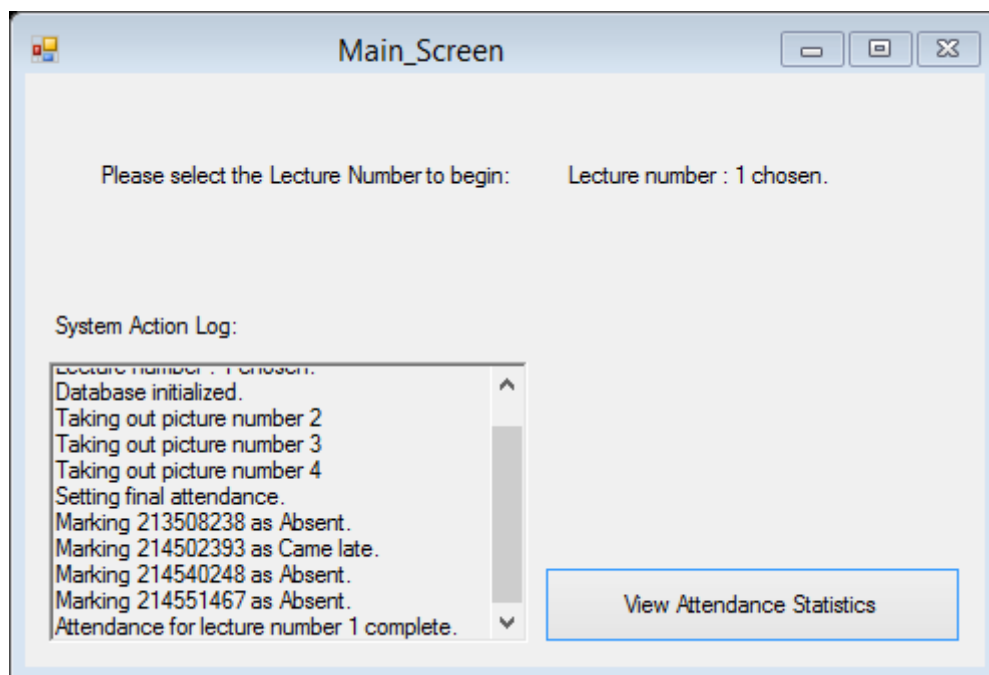
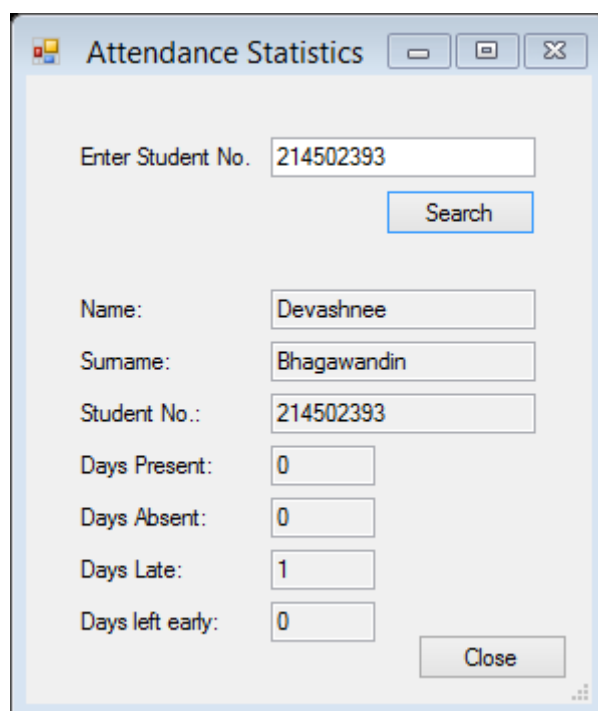


Figure 19: working program

The screenshot shows a window titled "Attendance Statistics" with standard Windows window controls (minimize, maximize, close). The main content area contains a form for entering student information. At the top, there is a label "Enter Student No." followed by a text input field containing the value "214502393". To the right of this field is a "Search" button. Below this, there are several labels followed by text input fields: "Name:", "Surname:", "Student No.:", "Days Present:", "Days Absent:", "Days Late:", and "Days left early:". At the bottom right of the form is a "Close" button.

Figure 20: input data



The image shows a software window titled "Attendance Statistics". It contains several input fields and buttons. At the top, there is a label "Enter Student No." followed by a text box containing "214502393" and a "Search" button. Below this, there are labels for "Name:", "Surname:", "Student No.:", "Days Present:", "Days Absent:", "Days Late:", and "Days left early:". Each label is followed by a text box. The "Name:" box contains "Devashnee", "Surname:" contains "Bhagawandin", "Student No.:" contains "214502393", "Days Present:" contains "0", "Days Absent:" contains "0", "Days Late:" contains "1", and "Days left early:" contains "0". At the bottom right of the window is a "Close" button. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Field	Value
Enter Student No.	214502393
Name:	Devashnee
Surname:	Bhagawandin
Student No.:	214502393
Days Present:	0
Days Absent:	0
Days Late:	1
Days left early:	0

Figure 21: Statistics Screen

Appendix C (Source Code)

Main Class Source Code

```
#pragma once
#include <opencv\cv.h>
#include <opencv\highgui.h>
#include <opencv2\core.hpp>
#include <opencv2\contrib\contrib.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\imgproc.hpp>
#include <opencv2\objdetect.hpp>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include "Attendance_Stats_Screen.h"

using namespace cv;
using namespace std;

namespace Group4_Class_Attendance_System {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace MySql::Data::MySqlClient;

    public ref class Main_Screen : public System::Windows::Forms::Form
    {
    public:

        Main_Screen(void)
        {
            InitializeComponent();
        }

    protected:
        ~Main_Screen()
        {
            if (components)
            {
                delete components;
            }
        }

    public:
        static System::Windows::Forms::Timer^ myTimer = gcnew
        System::Windows::Forms::Timer;

        //Timer used to control capturing time
        static int lectureNumber = 0; //Stores
        the lecture number
        static int quarter = 1;
        //The current quarter of the lecture
        static System::String^ constring; //Used for
        database communication
        static MySqlConnection^ conDataBase;
        static MySqlConnection^ conDataBase1;
        static MySqlConnection^ conDataBase2;
        static MySqlDataReader^ myReader;
        static MySqlDataReader^ myReader1;
        static MySqlDataReader^ myReader2;
        static System::String^ quarter1; //Stores
        the presence of a student in quater1
        static System::String^ quarter2; //Stores
        the presence of a student in quater2
        static System::String^ quarter3; //Stores
        the presence of a student in quater3
    };
}
```

```

        static System::String^ quarter4; //Stores
the presence of a student in quater4

private:
    System::Windows::Forms::ComboBox^ lectrueNumComboBox;
    System::Windows::Forms::Label^ lectureNumberlbl;
    System::Windows::Forms::Label^ chosenLecturelbl;
    System::Windows::Forms::Button^ button1;
    System::ComponentModel::Container ^components;
    static System::Windows::Forms::RichTextBox^ richTextBox1;
    System::Windows::Forms::Label^ labell1;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)
{
    this->lectrueNumComboBox = (gcnew System::Windows::Forms::ComboBox());
    this->lectureNumberlbl = (gcnew System::Windows::Forms::Label());
    this->chosenLecturelbl = (gcnew System::Windows::Forms::Label());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());
    this->labell1 = (gcnew System::Windows::Forms::Label());
    this->SuspendLayout();
    //
    // lectrueNumComboBox
    //
    this->lectrueNumComboBox->FormattingEnabled = true;
    this->lectrueNumComboBox->Items->AddRange(gcnew cli::array< System::Object^
>(25) {
        L"1", L"2", L"3", L"4", L"5", L"6", L"7",
        L"8", L"9", L"10", L"11", L"12", L"13", L"14", L"15", L"16",
        L"17", L"18", L"19", L"20", L"21", L"22", L"23", L"24", L"25"
    });
    this->lectrueNumComboBox->Location = System::Drawing::Point(258, 41);
    this->lectrueNumComboBox->Name = L"lectrueNumComboBox";
    this->lectrueNumComboBox->Size = System::Drawing::Size(146, 21);
    this->lectrueNumComboBox->TabIndex = 0;
    this->lectrueNumComboBox->SelectedIndexChanged += gcnew
System::EventHandler(this, &Main_Screen::lectrueNumComboBox_SelectedIndexChanged);
    //
    // lectureNumberlbl
    //
    this->lectureNumberlbl->AutoSize = true;
    this->lectureNumberlbl->Location = System::Drawing::Point(35, 44);
    this->lectureNumberlbl->Name = L"lectureNumberlbl";
    this->lectureNumberlbl->Size = System::Drawing::Size(217, 13);
    this->lectureNumberlbl->TabIndex = 1;
    this->lectureNumberlbl->Text = L>Please select the Lecture Number to begin: ";
    //
    // chosenLecturelbl
    //
    this->chosenLecturelbl->AutoSize = true;
    this->chosenLecturelbl->Location = System::Drawing::Point(268, 44);
    this->chosenLecturelbl->Name = L"chosenLecturelbl";
    this->chosenLecturelbl->Size = System::Drawing::Size(0, 13);
    this->chosenLecturelbl->TabIndex = 2;
    //
    // button1
    //
    this->button1->Location = System::Drawing::Point(259, 246);
    this->button1->Name = L"button1";
    this->button1->Size = System::Drawing::Size(208, 38);
    this->button1->TabIndex = 3;
    this->button1->Text = L"View Attendance Statistics";
    this->button1->UseVisualStyleBackColor = true;
    this->button1->Click += gcnew System::EventHandler(this,
&Main_Screen::button1_Click);
    //
    // richTextBox1
    //
    this->richTextBox1->Location = System::Drawing::Point(12, 144);
    this->richTextBox1->Name = L"richTextBox1";
    this->richTextBox1->ReadOnly = true;
    this->richTextBox1->Size = System::Drawing::Size(239, 140);
    this->richTextBox1->TabIndex = 4;
    this->richTextBox1->Text = L"";
}

```

```

        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location = System::Drawing::Point(12, 119);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(98, 13);
        this->label1->TabIndex = 5;
        this->label1->Text = L"System Action Log: ";
        //
        // Main_Screen
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(479, 296);
        this->Controls->Add(this->label1);
        this->Controls->Add(this->richTextBox1);
        this->Controls->Add(this->button1);
        this->Controls->Add(this->chosenLecture1b1);
        this->Controls->Add(this->lectureNumber1b1);
        this->Controls->Add(this->lectrueNumComboBox);
        this->Name = L"Main_Screen";
        this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Main_Screen";
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion

private: void static TimerEventProcessor(Object^ myObject, EventArgs^ myEventArgs)
{
    //This method executes when timer has reached 10 minutes
    myTimer->Stop(); //Stop the timer
    richTextBox1->Text += "\nTaking out picture number " + quarter;
    captureImage(); //Call the
captureImage function

    if (quarter > 4){
        richTextBox1->Text += "\nAttendance for lecture number "+lectureNumber+"
complete.";
        myTimer->Stop();
        myTimer->Enabled = false; //If all four quaters are
captured, do not restart the timer
    }
    else{
        myTimer->Enabled = true; //Restart the timer for another 10
minutes
    }
}

private: System::Void lectrueNumComboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
    //this method runs when the lecture number is chosen from the combo box

    constring = L"datasource=localhost; port=3306; username=root; password=password@0105";
    lectureNumber = Convert::ToInt32(this->lectrueNumComboBox->SelectedItem::get());
    //Stores the lecture number
    conDataBasel = gcnew MySqlConnection(constring);
    MySqlCommand^ cmdDataBasel = gcnew MySqlCommand("select lecture"+lectureNumber+" from
studentattendancedb.studentattendancetbl;", conDataBasel);

    try{
        conDataBasel->Open();
        myReader1 = cmdDataBasel->ExecuteReader();
        myReader1->Read();

        if (myReader1->GetString("Lecture" + lectureNumber) != "unknown")
        {
            MessageBox::Show("Lecture "+lectureNumber+" contains data. Please
choose another lecture.");
            //If the attendance for this lecture has already been taken, display
error message and wait for another slelection
        }
        else{
            richTextBox1->Text += "\nLecture number : " + lectureNumber + "
chosen.";

```



```

        chosenLecturelbl->Text = "Lecture number : " + lectureNumber + "
chosen.";

        lecttrueNumComboBox->Visible = false; //Disable
the combo box so the lecture number cannot be altered
        lecttrueNumComboBox->Enabled = false;

        conDataBase = gcnew MySqlConnection(constring);
        std::string sql1 = "UPDATE `studentattendancedb`.`studentattendancetbl`
SET `quarter1`='absent' , `quarter2`='absent', `quarter3`='absent', `quarter4`='absent' WHERE
`studentID` > '0' ";
        System::String^ sql2 = gcnew System::String(sql1.c_str());
        MySqlCommand^ cmdDataBase = gcnew MySqlCommand(sql2, conDataBase);
        try{
            conDataBase->Open();
            myReader = cmdDataBase->ExecuteReader();
            //Marks all the quarters as absent for every student
            richTextBox1->Text += "\nDatabase initialized.";
        }
        catch (System::Exception^ ex){
            MessageBox::Show(ex->Message);
        }
        conDataBase->Close();

        captureImage();
            //Captures the image for quarter 1
        myTimer->Tick += gcnew EventHandler(TimerEventProcessor);
        myTimer->Interval = 2000;
        // Sets the timer interval to 10 minutes.
        myTimer->Start();
            //Starts the timer
    }
}
catch (System::Exception^ ex){
    MessageBox::Show(ex->Message);
}
conDataBase1->Close();
}

void static setAttendance(int stid, int y){

    std::string qtr; //Used to alter the SQL
    if (y == 1)
        qtr = "quarter1";
    else if (y == 2)
        qtr = "quarter2";
    else if (y == 3)
        qtr = "quarter3";
    else
        qtr = "quarter4";

    //Marks student present for quarter defined by y

    constring = L"datasource=localhost; port=3306; username=root;
password=password@0105";
    conDataBase = gcnew MySqlConnection(constring);
    std::string sql1 = "UPDATE `studentattendancedb`.`studentattendancetbl` SET `" + qtr
+ "`='present' WHERE `studentID`=' " + std::to_string(stid) + " ";
    System::String^ sql2 = gcnew System::String(sql1.c_str());
    MySqlCommand^ cmdDataBase = gcnew MySqlCommand(sql2, conDataBase);

    try{
        conDataBase->Open();
        myReader = cmdDataBase->ExecuteReader();
        while (myReader->Read()){
        }
    }
    catch (System::Exception^ ex){
        MessageBox::Show(ex->Message);
    }
    conDataBase->Close();
}

static void captureImage()
{
    Mat capturedImage; //Stores the captured image
    vector<Mat> faces; //Stores the images used for
comparisons

```

```

vector<int> ids; //Stores the IDs of the images

read_csv("C:/Class_Attendance_System_Files/csv_file.csv", faces, ids);
//Fills the faces and ids vectors

int im_width = faces[0].cols; //Gets width and height of the images
used for comparisons
int im_height = faces[0].rows;

Ptr<FaceRecognizer> model = createFisherFaceRecognizer();
model->train(faces, ids); //Creates a face recognizer, trains it
with the images used for comparisons

CascadeClassifier haar_cascade; //Chooses haar-cascade that is used for
face detection

haar_cascade.load("C:/Class_Attendance_System_Files/haarcascade_frontalface_default.xml");

VideoCapture cap; //Opens a video feed
cap.open(0);
if (!cap.isOpened())
{
    MessageBox::Show("ERROR: Could not open camera.");
}

cap >> capturedImage; //Captures an image and
stores in the captruedImage

Mat original = capturedImage.clone(); //Keeps a copy of the original
image
Mat greyImage;
cvtColor(capturedImage, greyImage, CV_BGR2GRAY); //Converts the image to
greyscale

vector<Rect_<int>> facePositions;
haar_cascade.detectMultiScale(greyImage, facePositions); //Gets the
positions of the detected faces from the image

for (int i = 0; i < facePositions.size(); i++)
{
    Rect face_i = facePositions[i]; //Position
of the ith face
    Mat getFace = greyImage(face_i); //Converts
to greyscale
    Mat face_resized;
    //Resizes the detected face
    cv::resize(getFace, face_resized, cv::Size(im_width, im_height), 1.0, 1.0,
INTER_CUBIC);
    int foundID = model->predict(face_resized); //Gets prediction from
face recognizer
    setAttendance(foundID, quarter); //Calls the
setAttendance method. Sends the recognized ID as a parameter

    rectangle(original, face_i, CV_RGB(0, 255, 0), 1); //Places rectangle over
detected face and ID of the recognized faces
    string box_text = format("ID = %d", foundID);
    int pos_x = std::max(face_i.tl().x - 10, 0);
    int pos_y = std::max(face_i.tl().y - 10, 0);
    putText(original, box_text, cv::Point(pos_x, pos_y), FONT_HERSHEY_PLAIN, 1.0,
CV_RGB(0, 255, 0), 2.0);////////
}

imshow("face_recognizer", original); //Shows
original image with recognized faces and their IDs

if (quarter == 4) //This is
executed after all four quarters have been captured
{
    std::string sql1 = "SELECT * FROM `studentattendancedb`.`studentattendancetbl`";
    //Reads details of every student
    System::String^ sql2 = gcnew System::String(sql1.c_str());
    MySqlCommand^ cmdDataBase = gcnew MySqlCommand(sql2, conDataBase);

    try{
        conDataBase->Open();
        myReader = cmdDataBase->ExecuteReader();
        richTextBox1->Text += "\nSetting final attendance.";
    }
}

```

```

        while (myReader->Read())
        //Loop runs for every student
        {
            quarter1 = myReader->GetString("quarter1");           //Gets the required
data
            quarter2 = myReader->GetString("quarter2");
            quarter3 = myReader->GetString("quarter3");
            quarter4 = myReader->GetString("quarter4");
            int stID = myReader->GetInt32("studentID");
            int i = 0;

            if (quarter1 == "present")
                i = i + 1;
            if (quarter2 == "present")
                i = i + 1;
            if (quarter3 == "present")
                i = i + 1;
            if (quarter4 == "present")
                i = i + 1;

            if (i == 4 || (quarter1 == "present" && quarter4 == "present"))
            //Conditions for present
            {
                constraining = L"datasource=localhost; port=3306; username=root;
password=password@0105";
                conDataBase2 = gcnew MySqlConnection(constring);
                std::string sql1 = "UPDATE
`studentattendancedb`.`studentattendancetbl` SET `lecture` + std::to_string(lectureNumber) +
"= 'present' WHERE `studentID` = '" + std::to_string(stID) + "';";
                System::String^ sql2 = gcnew System::String(sql1.c_str());
                MySqlCommand^ cmdDataBase1 = gcnew MySqlCommand(sql2, conDataBase2);
                conDataBase2->Open();
                myReader2 = cmdDataBase1->ExecuteReader();
                richTextBox1->Text += "\nMarking " + stID + " as Present.";
                conDataBase2->Close();
            }
            else if ((i == 2 || i == 3) && quarter1 != "present")
            //Conditions for 'Came late'
            {
                constraining = L"datasource=localhost; port=3306; username=root;
password=password@0105";
                conDataBase2 = gcnew MySqlConnection(constring);
                std::string sql1 = "UPDATE
`studentattendancedb`.`studentattendancetbl` SET `lecture` + std::to_string(lectureNumber) +
"= 'came late' WHERE `studentID` = '" + std::to_string(stID) + "';";
                System::String^ sql2 = gcnew System::String(sql1.c_str());
                MySqlCommand^ cmdDataBase1 = gcnew MySqlCommand(sql2, conDataBase2);
                conDataBase2->Open();
                myReader2 = cmdDataBase1->ExecuteReader();
                richTextBox1->Text += "\nMarking " + stID + " as Came late.";
                conDataBase2->Close();
            }
            else if (i == 2 || i == 3 && quarter4 != "present")
            //Conditions for 'Left early'
            {
                constraining = L"datasource=localhost; port=3306; username=root;
password=password@0105";
                conDataBase2 = gcnew MySqlConnection(constring);
                std::string sql1 = "UPDATE
`studentattendancedb`.`studentattendancetbl` SET `lecture` + std::to_string(lectureNumber) +
"= 'left early' WHERE `studentID` = '" + std::to_string(stID) + "';";
                System::String^ sql2 = gcnew System::String(sql1.c_str());
                MySqlCommand^ cmdDataBase1 = gcnew MySqlCommand(sql2, conDataBase2);
                conDataBase2->Open();
                myReader2 = cmdDataBase1->ExecuteReader();
                richTextBox1->Text += "\nMarking " + stID + " as Left early.";
                conDataBase2->Close();
            }
            else
            {
                //mark student absent
                constraining = L"datasource=localhost; port=3306; username=root;
password=password@0105";
                conDataBase2 = gcnew MySqlConnection(constring);
                std::string sql1 = "UPDATE
`studentattendancedb`.`studentattendancetbl` SET `lecture` + std::to_string(lectureNumber) +
"= 'absent' WHERE `studentID` = '" + std::to_string(stID) + "';";

```

```

        System::String^ sql2 = gcnew System::String(sql1.c_str());
        MySqlCommand^ cmdDataBase1 = gcnew MySqlCommand(sql2, conDataBase2);
        conDataBase2->Open();
        myReader2 = cmdDataBase1->ExecuteReader();
        richTextBox1->Text += "\nMarking " + stID + " as Absent.";
        conDataBase2->Close();
    }
}
catch (System::Exception^ ex){
    MessageBox::Show(ex->Message);
}
conDataBase->Close();
}
quarter++; //move to next quarter
}

static void read_csv(const string& filepath, vector<Mat>& images, vector<int>& labels)
{
    std::ifstream file(filepath.c_str(), ifstream::in);
    if (!file)
    {
        MessageBox::Show("ERROR: Could not read CSV file.");
    }
    string line, path, classlabel;

    while (getline(file, line)) //Loop runs for every line
of the CSV file
    {
        stringstream liness(line);
        getline(liness, path, ';'); //Delimiter seperates data
        getline(liness, classlabel);
        if (!path.empty() && !classlabel.empty())
        {
            images.push_back(imread(path, 0)); //Stores image in
vector
            labels.push_back(atoi(classlabel.c_str())); //Stores ID in vector
        }
    }
}

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    Attendance_Stats_Screen^ ass = gcnew Attendance_Stats_Screen();
    //Displays the attendance stats screen
    ass->ShowDialog();
}
};
}

```

Attendance Stats Source Code

```
#pragma once
#include <opencv\cv.h>
#include <opencv\highgui.h>
#include <opencv2\core.hpp>
#include <opencv2\contrib\contrib.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\imgproc.hpp> // including all required libraries and header
files
#include <opencv2\objdetect.hpp>
#include <iostream>
#include <fstream>
#include <sstream>
#include <iostream>
#include <string>
#include <iostream>

using namespace cv;
using namespace std;

namespace Group4_Class_Attendance_System {

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace MySql::Data::MySqlClient;

/// <summary>
/// Summary for Attendance_Stats_Screen
/// </summary>
public ref class Attendance_Stats_Screen : public System::Windows::Forms::Form
{
public:
    Attendance_Stats_Screen(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Attendance_Stats_Screen()
    {
        if (components)
        {
            delete components;
        }
    }

public:
    System::String^ constring; // connection string variable to connect to database
    MySqlConnection^ conDataBase; // connection variable to create database connection

    MySqlDataReader^ myReader; // reader variable to read from database
    static System::String^ studentName; // stores student's name
    static System::String^ studentSurname; // stores student's surname
    static int studentID = 0; // stores student number
    static int NoOfDaysPresent = 0; // stores number of days present
    static int NoOfDaysAbsent = 0; // stores number of days absent
    static int NoOfDaysCameLate = 0; // stores number of days late
    static int NoOfDaysLeftEarly = 0; // stores number of days left early

private: System::Windows::Forms::TextBox^ srchtextBox;
protected:
```

```

private: System::Windows::Forms::Button^ searchBtn;
protected:

private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::Label^ label8;
private: System::Windows::Forms::TextBox^ nametextBox;
private: System::Windows::Forms::TextBox^ snametextBox;

private: System::Windows::Forms::TextBox^ textBox3;
private: System::Windows::Forms::TextBox^ prtextBox;
private: System::Windows::Forms::TextBox^ latetextBox;
private: System::Windows::Forms::TextBox^ abtextBox;
private: System::Windows::Forms::TextBox^ earlytextBox;
private: System::Windows::Forms::Button^ button1;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->srchtextBox = (gcnew System::Windows::Forms::TextBox());
        this->searchBtn = (gcnew System::Windows::Forms::Button());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->label5 = (gcnew System::Windows::Forms::Label());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->label7 = (gcnew System::Windows::Forms::Label());
        this->label8 = (gcnew System::Windows::Forms::Label());
        this->nametextBox = (gcnew System::Windows::Forms::TextBox());
        this->snametextBox = (gcnew System::Windows::Forms::TextBox());
        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
        this->prtextBox = (gcnew System::Windows::Forms::TextBox());
        this->latetextBox = (gcnew System::Windows::Forms::TextBox());
        this->abtextBox = (gcnew System::Windows::Forms::TextBox());
        this->earlytextBox = (gcnew System::Windows::Forms::TextBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->SuspendLayout();
        //
        // srchtextBox
        //
        this->srchtextBox->Location = System::Drawing::Point(122, 31);
        this->srchtextBox->Name = L"srchtextBox";
        this->srchtextBox->Size = System::Drawing::Size(132, 20);
        this->srchtextBox->TabIndex = 0;
        //
        // searchBtn
        //
        this->searchBtn->Location = System::Drawing::Point(179, 57);
        this->searchBtn->Name = L"searchBtn";
        this->searchBtn->Size = System::Drawing::Size(75, 23);
        this->searchBtn->TabIndex = 1;
        this->searchBtn->Text = L"Search";
        this->searchBtn->UseVisualStyleBackColor = true;
        this->searchBtn->Click += gcnew System::EventHandler(this,
&Attendance_Stats_Screen::searchBtn_Click);
        //
        // label1
        //

```

```

this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(24, 34);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(92, 13);
this->label1->TabIndex = 2;
this->label1->Text = L"Enter Student No.";
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(24, 110);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(38, 13);
this->label2->TabIndex = 3;
this->label2->Text = L"Name:";
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(24, 136);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(52, 13);
this->label3->TabIndex = 4;
this->label3->Text = L"Surname:";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(24, 162);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(67, 13);
this->label4->TabIndex = 5;
this->label4->Text = L"Student No.:";
//
// label5
//
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(24, 188);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(73, 13);
this->label5->TabIndex = 6;
this->label5->Text = L"Days Present:";
//
// label6
//
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(24, 214);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(70, 13);
this->label6->TabIndex = 7;
this->label6->Text = L"Days Absent:";
//
// label7
//
this->label7->AutoSize = true;
this->label7->Location = System::Drawing::Point(24, 240);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(58, 13);
this->label7->TabIndex = 8;
this->label7->Text = L"Days Late:";
//
// label8
//
this->label8->AutoSize = true;
this->label8->Location = System::Drawing::Point(24, 266);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(76, 13);
this->label8->TabIndex = 9;
this->label8->Text = L"Days left early:";
//
// nametextBox
//
this->nametextBox->Location = System::Drawing::Point(122, 107);
this->nametextBox->Name = L"nametextBox";
this->nametextBox->ReadOnly = true;
this->nametextBox->Size = System::Drawing::Size(132, 20);
this->nametextBox->TabIndex = 10;

```

```

//
// snameTextBox
//
this->snameTextBox->Location = System::Drawing::Point(122, 133);
this->snameTextBox->Name = L"snameTextBox";
this->snameTextBox->ReadOnly = true;
this->snameTextBox->Size = System::Drawing::Size(132, 20);
this->snameTextBox->TabIndex = 11;
//
// textBox3
//
this->textBox3->Location = System::Drawing::Point(122, 159);
this->textBox3->Name = L"textBox3";
this->textBox3->ReadOnly = true;
this->textBox3->Size = System::Drawing::Size(132, 20);
this->textBox3->TabIndex = 12;
//
// prtextBox
//
this->prtextBox->Location = System::Drawing::Point(122, 185);
this->prtextBox->Name = L"prtextBox";
this->prtextBox->ReadOnly = true;
this->prtextBox->Size = System::Drawing::Size(52, 20);
this->prtextBox->TabIndex = 13;
//
// latetextBox
//
this->latetextBox->Location = System::Drawing::Point(122, 237);
this->latetextBox->Name = L"latetextBox";
this->latetextBox->ReadOnly = true;
this->latetextBox->Size = System::Drawing::Size(52, 20);
this->latetextBox->TabIndex = 14;
//
// abtextBox
//
this->abtextBox->Location = System::Drawing::Point(122, 211);
this->abtextBox->Name = L"abtextBox";
this->abtextBox->ReadOnly = true;
this->abtextBox->Size = System::Drawing::Size(52, 20);
this->abtextBox->TabIndex = 15;
//
// earlytextBox
//
this->earlytextBox->Location = System::Drawing::Point(122, 263);
this->earlytextBox->Name = L"earlytextBox";
this->earlytextBox->ReadOnly = true;
this->earlytextBox->Size = System::Drawing::Size(52, 20);
this->earlytextBox->TabIndex = 16;
//
// button1
//
this->button1->Location = System::Drawing::Point(195, 279);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(75, 23);
this->button1->TabIndex = 17;
this->button1->Text = L"Close";
this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnew System::EventHandler(this,
&Attendance_Stats_Screen::button1_Click);
//
// Attendance_Stats_Screen
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(282, 314);
this->Controls->Add(this->button1);
this->Controls->Add(this->earlytextBox);
this->Controls->Add(this->abtextBox);
this->Controls->Add(this->latetextBox);
this->Controls->Add(this->prtextBox);
this->Controls->Add(this->textBox3);
this->Controls->Add(this->snameTextBox);
this->Controls->Add(this->nametextBox);
this->Controls->Add(this->label8);
this->Controls->Add(this->label7);
this->Controls->Add(this->label6);
this->Controls->Add(this->label5);

```



```

        this->Controls->Add(this->label4);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->label2);
        this->Controls->Add(this->label1);
        this->Controls->Add(this->searchBtn);
        this->Controls->Add(this->srchtextBox);
        this->Name = L"Attendance_Stats_Screen";
        this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Attendance Statistics";
        this->Load += gcnew System::EventHandler(this,
&Attendance_Stats_Screen::Attendance_Stats_Screen_Load);
        this->ResumeLayout(false);
        this->PerformLayout();
    }

#pragma endregion

private: System::Void Attendance_Stats_Screen_Load(System::Object^ sender, System::EventArgs^ e)
{
    constring = L"datasource=localhost; port=3306; username=root; password=password@0105";
    // connection string to connect to database
    conDataBase = gcnew MySqlConnection(constring); // creates new connection
}

private: System::Void searchBtn_Click(System::Object^ sender, System::EventArgs^ e)
{
    studentID = Int32::Parse(srchtextBox->Text); // gets the student number entered by the
user in the text box and converts it to an integer
    getData(studentID); // function to get data
    displayData(); // function to display data
}

void getData(int stdNo)
{
    MySqlCommand^ cmdDataBase = gcnew MySqlCommand("select * from
studentattendancedb.studentattendancetbl where studentID = '" + stdNo + "';", conDataBase); //
SQL statement to obtain information from database
    try{
        conDataBase->Close(); // closes connection to database
        conDataBase->Open(); // opens connection to database
        myReader = cmdDataBase->ExecuteReader(); // runs the reader to read from
database

        while (myReader->Read()) // reads data from database while there is still information
to be read
        {
            for (int i = 1; i <= 25; i++)
            {
                if (myReader->GetString("Lecture" + i) == "absent")
                {
                    NoOfDaysAbsent++; // counts the number of days absent
                }
                else if (myReader->GetString("Lecture" + i) == "present")
                {
                    NoOfDaysPresent++; // counts the number of days present
                }
                else if (myReader->GetString("Lecture" + i) == "came late")
                {
                    NoOfDaysCameLate++; // counts the number of days arrived late
                }
                else if (myReader->GetString("Lecture" + i) == "left early")
                {
                    NoOfDaysLeftEarly++; // counts the number of days left early
                }
            }
        }
        studentID = myReader->GetInt32("StudentID"); // reads the student number from database and
converts it to an integer

        studentName = myReader->GetString("Name"); // reads student's name from the database

        studentSurname = myReader->GetString("Surname"); //reads student's surname from the database
    }
    catch (System::Exception^ ex)
    {

```

```

        MessageBox::Show(ex->Message);
    }

}

void displayData()
{
    textBox3->Text = Convert::ToString(studentID); // converts student number from integer
to string and displays it in the text box
    nametextBox->Text = studentName; // displays student's name in textbox
    snameTextBox->Text = studentSurname; // displays student's surname in textbox
    abtextBox->Text = Convert::ToString(NoOfDaysAbsent); // converts number of days absent
from integer to string and displays it in the text box
    prtextBox->Text = Convert::ToString(NoOfDaysPresent); // converts number of days
present from integer to string and displays it in the text box
    latetextBox->Text = Convert::ToString(NoOfDaysCameLate); // converts number of days
came late from integer to string and displays it in the text box
    earlytextBox->Text = Convert::ToString(NoOfDaysLeftEarly); // converts number of days
left early from integer to string and displays it in the text box

    NoOfDaysPresent = 0;
    NoOfDaysAbsent = 0; // resets all counts for next display
    NoOfDaysCameLate = 0;
    NoOfDaysLeftEarly = 0;
}

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    this->Close(); // closes attendance stats screen
}

};
}

```