BINUS UNIVERSITY INTERNATIONAL

**Assignment Cover Letter**

**(Individual Work)**

| **Student Information**: | **Surname** | **Given Names** | **Student ID Number** |
|---|---|---|---|
| 1. | **Senjaya** | **Monique** | **2440061285** |

**Course Code** : **COMP 6502** **Course Name** : **Program Design Methods**

**Class** : **L1BC** **Name of Lecturer(s)** : **Ida Bagus Kerthyayana Manuaba**

**Major** : **Computer Science**

**Title of Assignment**: **Facial Recognition Attendance System**
(if any)

**Type of Assignment** : **Final Project**

**Submission Pattern**

**Due Date** : **10/01/2021** **Submission Date** : **10/01/2021**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.

4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Monique Senjaya

## Table of Contents

**"Facial Recognition Attendance System"**

**Name: Monique Senjaya**

**ID: 2440061285**

## A. Description

### I. Introduction

When the final project was first introduced, I started to brainstorm and search on what I wanted to make. I first got the idea of doing a visual novel game using the pygame library, however, after more research, I came across a face recognition library in python made by Adam Geitgey. This got me really engaged and I thought it would be a really good idea if this external library is implemented to a sort of attendance system. As this is my first time doing a proper python project, I plan to do more research on how the library works and also other libraries to be used for the frontend. I am really looking forward to this project.

I begin this project on the 16th of November 2020. I chose Visual Studio Code as my IDE for this particular project and I will commit and push all progress to my GitHub account. The link to this specific project is **https://github.com/moniquesenjaya/PDM-Project**.

### II. The function of the program

The purpose of this program is to allow an easier and a more efficient way of taking attendance for classes by using facial recognition. Not only that, this program can also be used as a tool to keep student's data and check their eligibility when it comes to their exams (assuming that they only have a specific amount of times that they are allowed to be absent for).

The teachers or admins who have access to this program will constantly update the student details (Student ID, Name, Gender, and Picture). Each of this data can be inserted, updated, and deleted by the admin/teacher in charge (sqlite3 will be used for this). To do the attendance, the image of the students who attended the class should be uploaded to the application. Then, the faces on that image will be compared to all the images of the students stored in the database. When the faces are recognized, the number of present classes will increase. If there's no match, then the absent count will increase by one. This is way better as it might save so much time rather than having to call the students one by one during roll calls.

Specifications regarding the project includes:

- Input of the program:
  - Student's details (Student ID, Name, Gender, and Picture)
  - Class Picture
  - Keywords for search filter
- Output of the program:
  - List of student's details based on the search filter and attendance calculations
- External libraries used:
  - Tkinter
    - Used for designing the GUI window for the project
  - PIL (Python Imaging Library)
    - Used for opening, manipulating, and saving image files
  - OS
    - Used for interacting with the operating system (deleting picture files in case of deletion in data of a student)
  - Sqlite3
    - Used for creating a database, defining tables, inserting and changing rows, run queries and manage an SQLite database file. (without needing a separate server)
  - Face-recognition v1.2.2
    - Used for locating, encoding, and comparing faces in an image

## B. Solution Design Plan

### I. Parts of the program

In this project (aiming for simplicity), the GUI menu will only consist of one window with five main parts/sections that is important to help the user navigate the workings of the application. The list of the sections is:

- Data Entry Form (top left of the window, width: 50%, height: 80%)
- SQL Buttons (bottom left of the window, width: 50%, height: 20%)
- Search Bar (top right of the window, width: 45%, height: 10%)
- Attendance Button (top right of the window, width: 5%, height: 10%)
- List of Data (top right of the window, width: 50%, height: 90%)

Note: The width and height are an approximate plan of how much each part will cover the window. The figure below will further visualize this.

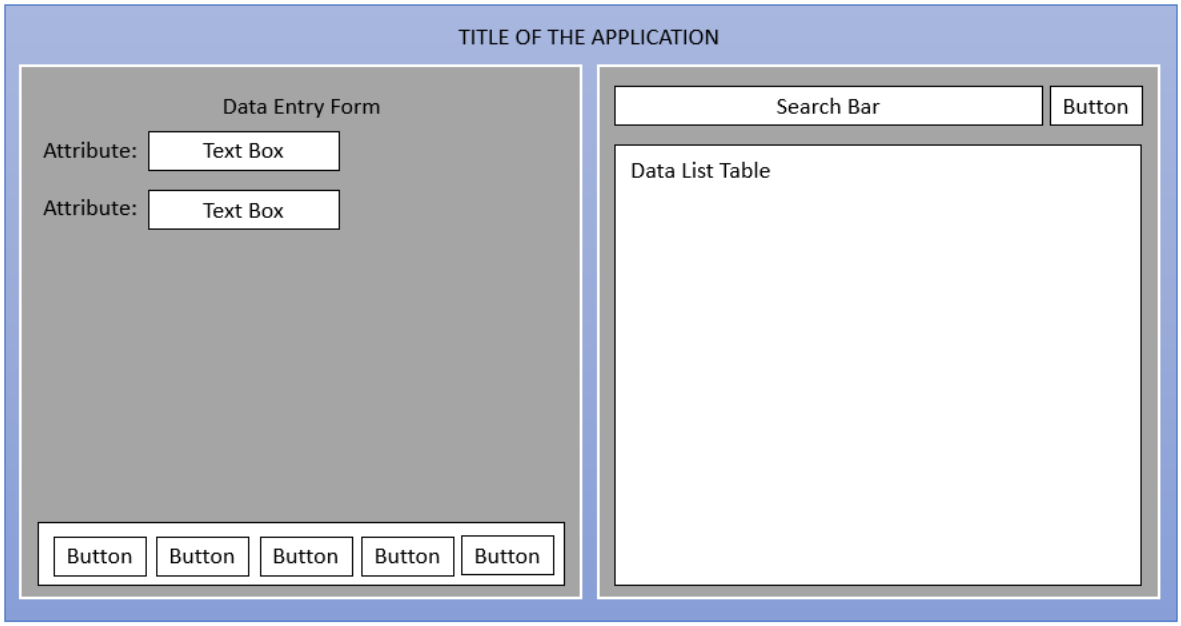


**Figure 1**: Design sketch of the project GUI

## II. Function of each parts of the program

*Data Entry Form*

This part of the program functions as a tool for the admin/teachers to input the details of the students. This is where the student id, names, gender, and picture of the students are being typed in and stored to the database. It allows the user to easily manipulate the data directly from the GUI without having to access the data. In this form, I will make use of various entry tools such as textboxes, dropdown buttons, and file uploader in attempts to make the user experience better.

*SQL Buttons*

This part of the program is where all the necessary SQL buttons are put. The buttons that will be available here includes:

- Add
  - This button will contain the insert query so that the user can add new student details into the database.
- Delete

    - This button will contain the delete query so that the user can delete an entry from the database using the student id.
- Update
    - This button allows the user to change the data of the fields in the database using the update query.
- Clear
    - This button empties the entry tools. This is needed because at times, users will need to clear it after updating the data.

*Search Bar*

This search bar contains three elements: a drop-down list, a textbox, and two buttons. This is where users can filter the data they view. The drop-down list contains categories to allow users to "search by" the attribute that is chosen in the drop-down list. The textbox is where the user type in what they are searching for, according to the category picked. The two buttons are "search" and "cancel". The search button contains the select query and the cancel button removes the filter from the select query and show all the data in the database.

*Attendance Button*

This button is a file uploader so that the user can upload the class picture to be further analyzed by the program. This will then update the data in the database and show who are present and absent in a class.

*List of Data*

This part of the program is where a table is used to show the data whether it is filtered or not filtered. This is also where the user can click to select the data to be shown in the entry tools so that it is easier to visualize and update the data if it is needed.

## C. Implementation

### I. Data dictionary (Cool_School.db)

students table

| Field Name | Data Type | Description |
|---|---|---|
| studentid | text | The format will be "SXXXX" where X are integers and studentid has to be unique |
| firstname | text | Shows the first name of the student |
| lastname | text | Shows the last name of the student |
| gender | text | Can only accept three values "Male", "Female" and "Others", so drop down list is used |
| picturepath | text | Stores uploaded pictures file paths |
| present | text | Initial value will be 0 and it will be automatically increased according to attendance |
| absent | text | Initial value will be 0 and it will be automatically increased according to attendance |

### II. Classes Diagram

| **Student** |
|---|
| -__studentid<br>-__firstname<br>-__lastname<br>-__gender<br>-__picturepath<br>-__present<br>-__absent |
| - __init__(studentid, firstname, lastname, gender, picturepath, present, absent)<br><br>+ get_full_name()<br>+ validate_empty(list_of_textbox)<br>+ validate_studentid()<br>+ validate_name()<br>+ validate_new_number()<br>+ validate_number()<br>+ picture_saving() |

**Figure 2**: UML of the student class

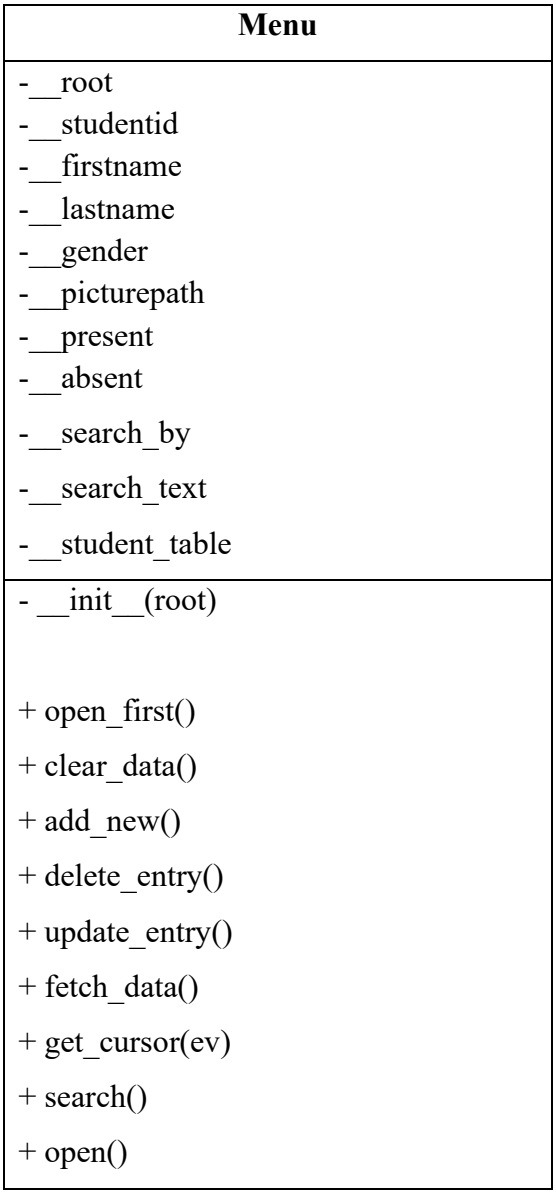


**Figure 3**: UML of the menu class

## III. Flowchart

Adding New Student Records

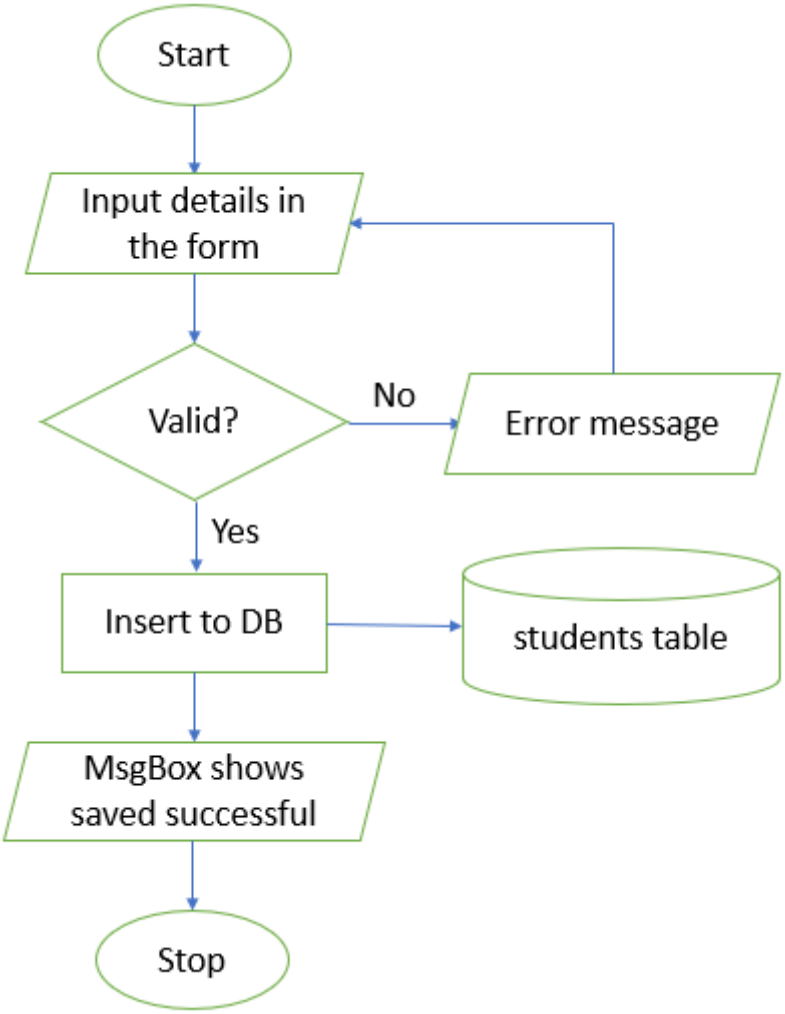


**Figure 4**: Add new record flowchart

Update Student Record

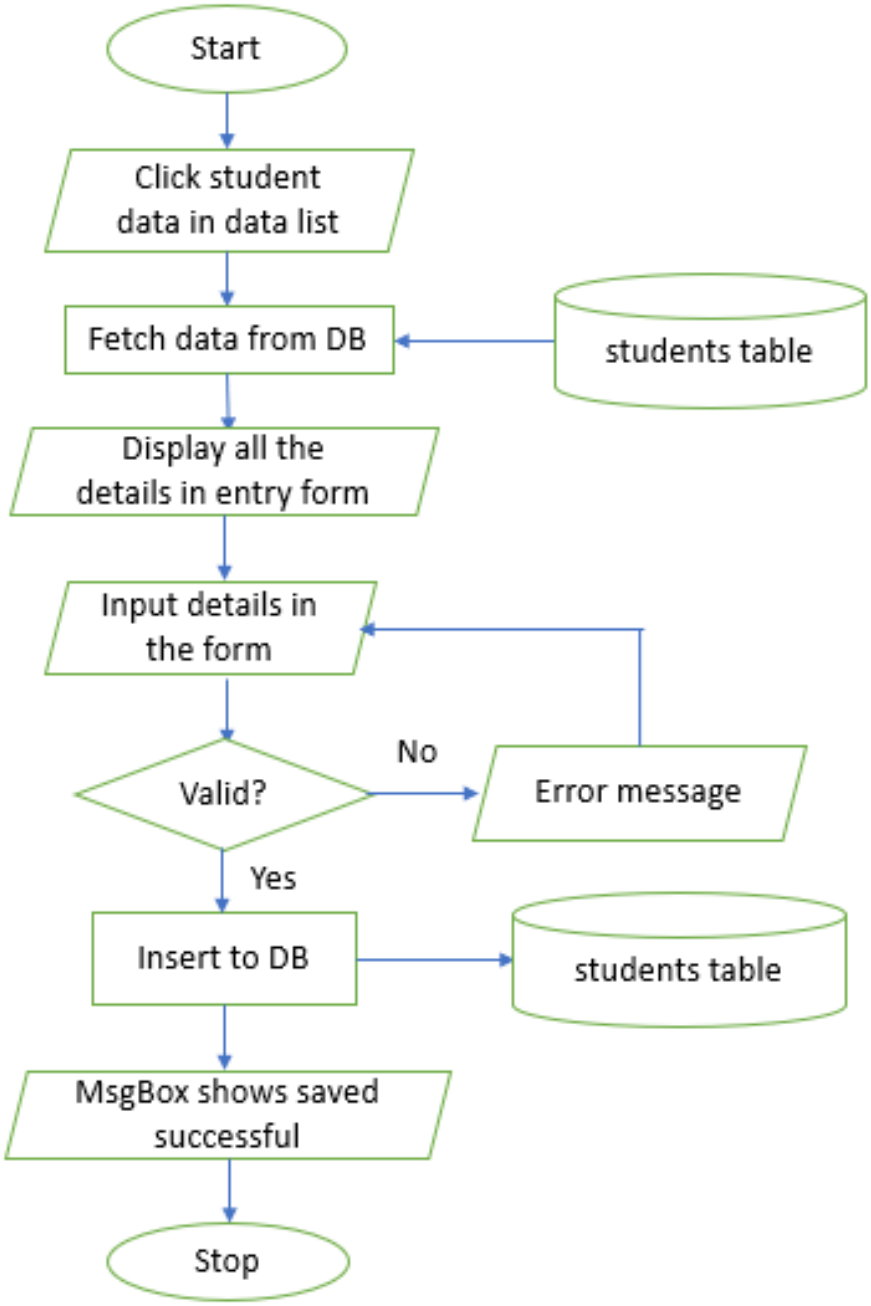


**Figure 5**: Update existing record flowchart

## Delete an Existing Record

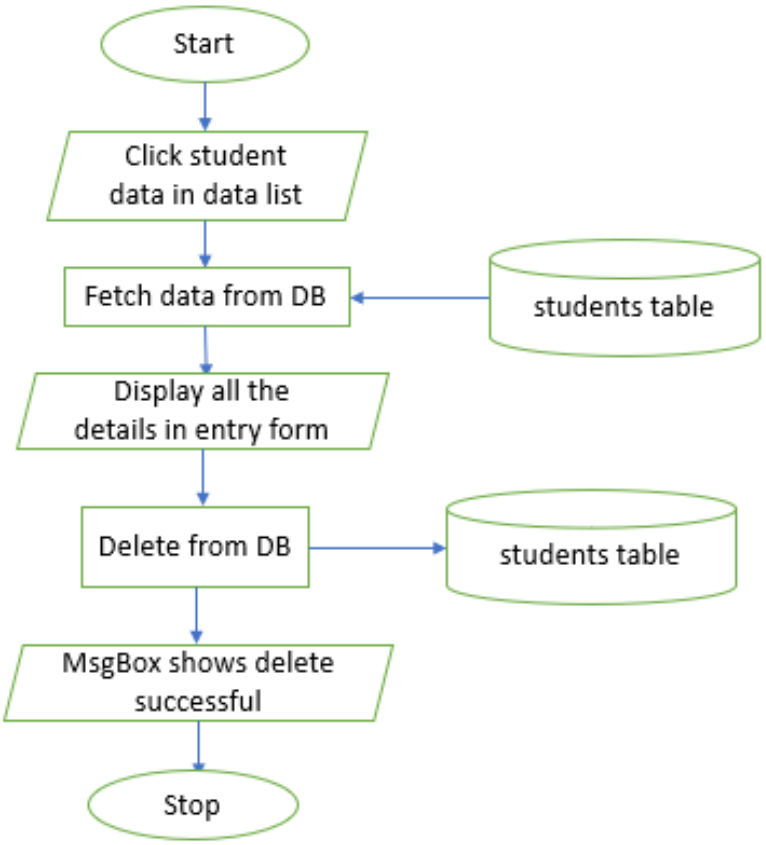


**Figure 6**: Delete an existing record flowchart

## Searching for Specific Data

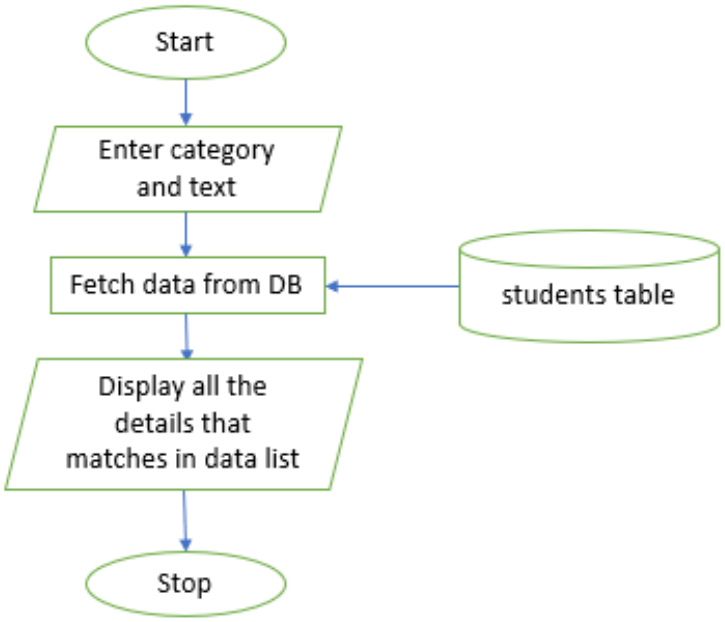


**Figure 7**: Searching for specific data flowchart

Saving Picture Paths

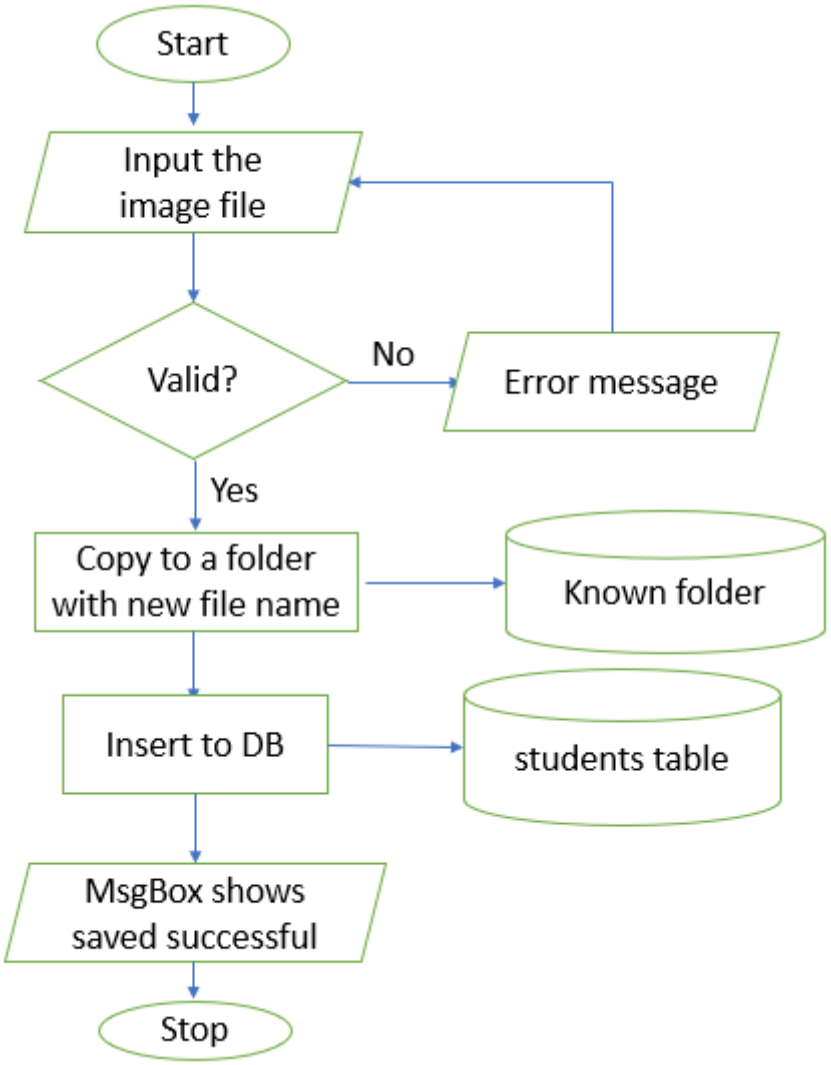


**Figure 8**: Saving the uploaded picture flowchart

Facial Recognition Update

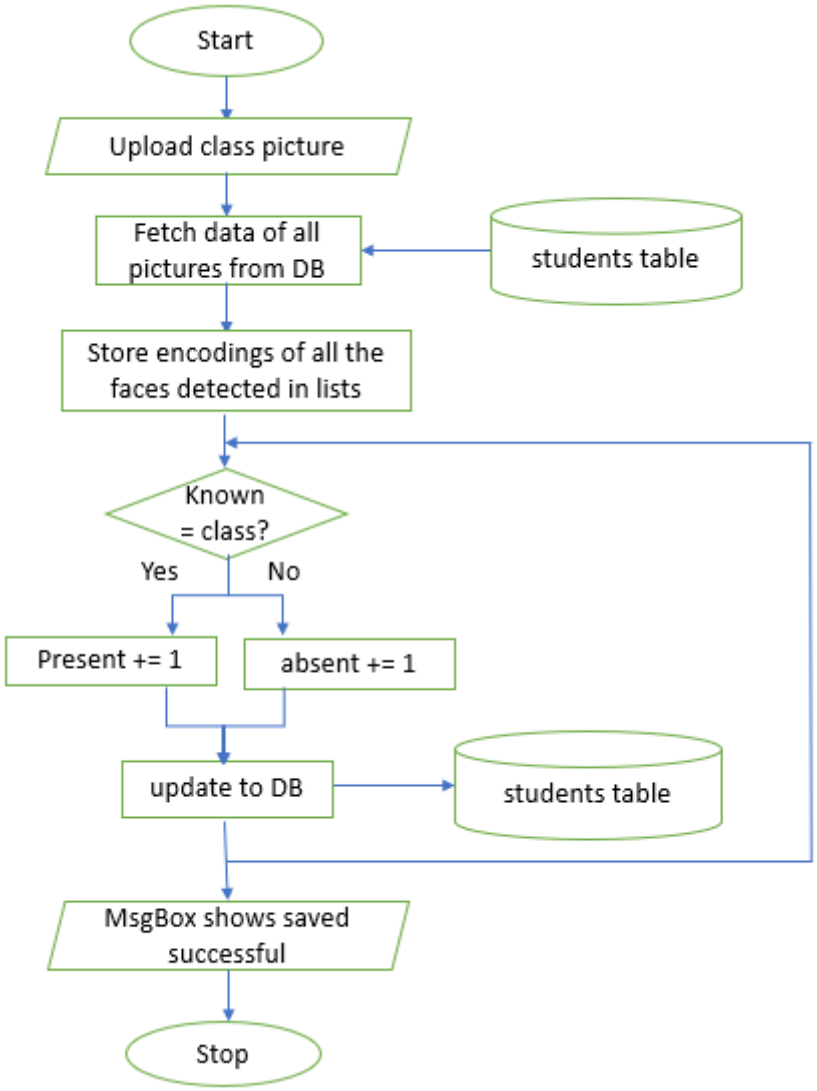


**Figure 9**: Facial Recognition updating flowchart

## IV. Extensibility

1. Comment lines are used to explain the functions of certain codes to let the user understand the working system of the specific codes to help future programmers.
2. Meaningful identifiers for variables which allows user to understand the use of certain variables.
3. Dividing the problems by using meaningful sections. This will allow a more organized planning of development and a more thorough testing of each section. This also allow future developers understand the various parts of the programs.

4. Flowchart is used to help future developer understand the technical weaknesses in the program and how it works. This will allow them to easily recognize the logic towards the solutions of problems.
5. Data structure and class structure is given to allow users/future developers easily identify the content of each entities that is recorded in the database.
6. Data is made to provide descriptions of the fieldname (how it is entered, what It means etc.)

## V. Explanation of all the functions made and used

*student.py*

- __init__(self, studentid, firstname, lastname, gender, picturepath, present, absent):
  - Creates variables for storing each of the attributes of a student
- studentid(self):
  - Creates setter and getter for student id variable using property decorator
- firstname(self):
  - Creates setter and getter for first name variable using property decorator
- lastname(self):
  - Creates setter and getter for last name variable using property decorator
- gender(self):
  - Creates setter and getter for gender variable using property decorator
- picturepath(self):
  - Creates setter and getter for picture path variable using property decorator
- present(self):
  - Creates setter and getter for present variable using property decorator
- absent(self):
  - Creates setter and getter for absent variable using property decorator
- get_full_name(self):
  - Returns a string of first name and last name together by using the getter

- validate_empty(self, list_of_textbox):
  - Checks if any of the entry fields are empty
  - Returns error string if there's an empty string in the list

- validate_studentid(self):
  - Create a connection to the database and select the student id, append it to a list
  - Checks if the student id format is "SXXXX" where X is an integer
  - Checks if the student id is unique by comparing with the list of student id
  - Returns error string if any of the condition is not satisfied

- validate_name(self):
  - Checks if the names are string
  - Returns error string if any of the condition is not satisfied

- validate_new_number(self):
  - Checks if the initial value of the numbers is 0
  - Returns error string if any of the condition is not satisfied

- validate_number(self):
  - Checks if the initial value of the numbers is an integer
  - Returns error string if any of the condition is not satisfied

- picture_saving(self):
  - Use the picturepath variable and PIL to open the image file
  - Save the picture in a folder named known with the full name as the file name

*query_functions.py*

- add_new_data(studentid, firstname, lastname, gender, picturepath, present, absent):
  - Creates connection with the database and executes an insert query to insert the values to the database as a new record

- delete_data(studentid):
  - Creates connection with the database and executes a delete query to delete the record in a database where the student id matches the one given as a parameter

- check_valid_id(studentid):
  - Creates connection with the database and append all the student id in a list
  - Checks if the newly entered student id is empty or is not unique

  - Returns an error string if any of the conditions is True
- view_all_data():
  - Creates connection with the database and selects all the data from the database
  - Returns the data as list
- update_data(studentid, firstname, lastname, gender, picturepath, present, absent):
  - Creates connection with the database and executes an update query to update the values to the database where the student id is the same
- search_data_by(searchby, searchtxt):
  - Creates connection with the database and selects all the data from the database where searchby (student id/first name/absent) matches the searchtxt (string in the textbox in the search bar)
  - Returns the data as list
- add_attendance(filename):
  - Create a database connection and fetch all the data, puts the student id into one list and the encodings (uses the functions from face recognition library) of the picture in another list
  - Load the image of filename that is uploaded to the system and face locations and face encodings are used to find all the encodings of the faces detected in the picture
  - Check every face detected from the image file uploaded to the system if it matches the faces in the picture path lists from the database
  - If there's a match, the id will be stored in a list of "faces matched"
  - All id will be stored in the "all faces" list after being compared (if none of the faces matches the one in the group, the id will be "unknown person"
  - A loop is used to update all the present value to present += 1 to all the ids in the faces matched list
  - If the id is not in the "faces matched", the absent value will be updated to absent += 1
  - Lastly, another loop is used to check how many people are present but the face is not matched with the database.
  - Returns the number of students undetected, id of students present and id of students absent

*app.py*

- __init__(self, root):
  - Creates the variables needed to be used in the Tkinter.
  - Creates the title of the app using label
  - Creates entry form frame and uses labels and text boxes to create the form itself
  - Creates button frame that contain several SQL buttons that has its own commands (this will be coded in other functions)
  - Search bar frame is created to hold the search by drop-down, text box, and buttons, this is also where the attendance button is. (the buttons will have its own codes in a different function according to its "command")
  - Creates a list frame that contains a tree view of the attributes to the students and get the cursor of the row after BM-1 is released.

- clear_data(self):
  - Sets all the textboxes and dropdown lists to "" and clears the data for a better user experience

- add_new(self):
  - Create a new object using the student class
  - Does all the validation that was in the student class, if there is a return value, an error message box will be shown then the function will be stopped. If the checking using the validation function returns None, it goes to another checking until the last validation
  - If all the validation is passed, the picture_saving function will be implemented to the object.
  - Then it will be added to the database by using the add_new_data in the query_functions module
  - Creates a message box to show that the save is successful and clears the data in the data entry tools using the clear_data function

- delete_entry(self):
  - Checks if the id in the textbox has any matches to the database, if not, error message is shown
  - If it's a match, a message box will be shown to confirm the deletion of the record

  - If it is confirmed, picture will be removed from the known folder and then delete_data function from the query_functions module will be implemented

- update_entry(self):
  - Create a new object using the student class
  - Does all the validation that was in the student class, if there is a return value, an error message box will be shown then the function will be stopped. If the checking using the validation function returns None, it goes to another checking until the last validation
  - If all the validation is passed, data will be updated to the database by using the update_data in the query_functions module
  - Creates a message box to show that the save is successful and clears the data in the data entry tools using the clear_data function

- fetch_data(self):
  - Check if the view_all_data function from the query_functions module doesn't return 0
  - If it isn't 0, the contents of the tree view will be deleted and replaced with the contents from the return of view_all_data function

- get_cursor(self, ev):
  - Gets the focus of the item in the tree view
  - If the focus is gotten, the contents of the row is stored in a dictionary by using the .item() function
  - A list containing the values of that dictionary is made and the contents of the list are assigned to the variables created where the contents of the entry boxes are stored. (this will make sure that the content appears in the data entry form)

- search(self):
  - search_data_by function from the query_functions module is used to get the all the entries that matches the filter of the search determined by the user.
  - If the function from the query_functions module doesn't return 0, the contents of the tree view will be deleted and replaced with the contents from the return of search_data_by function

- open(self):
  - Create a file dialog for the image file to be uploaded.
  - add_attendance function is implemented the image file and a message box will be shown if attendance is successfully noted. (using the return string of the function)
  - fetch_data function is used to update the tree view contents

## D. Lessons that have been learnt

### I. Learning the ins and outs of Tkinter

Since this extended library is not discussed in the class, I learnt how to create GUI window with the use of Tkinter. I have learnt how to make use of frames to make sure the positioning of each section is clear and neat. Other than that, I learnt how to make a data entry form through the use of labels, text boxes, file dialogs, and drop-down lists.

```
entry_frame = Frame(self.__root, bd=3, relief=RIDGE, bg="light gray")
entry_frame.place(x=20, y=100, width=450, height=560)

entry_title = Label(entry_frame, text="Student Information",bg="light gray", font=("Helvetica", 20, "bold"))
entry_title.grid(row=0, columnspan=2, pady=20, padx=10, sticky="w")

studentid_label = Label(entry_frame, text = "Student ID:",bg="light gray", font=("Helvetica", 16, "bold"))
studentid_label.grid(row=1, column=0, pady=5, padx=20, sticky="w")
studentid_text = Entry(entry_frame, font=("Helvetica", 16), bd=3, relief=GROOVE, textvariable = self.__studentid)
studentid_text.grid(row=1, column=1, padx=8, pady=5, sticky = "w",columnspan=2)
```

**Figure 10**: Code for the making of data entry form using labels and text boxes

```
picturepath_label = Label(entry_frame, text = "Picture:",bg="light gray", font=("Helvetica", 16, "bold"))
picturepath_label.grid(row=5, column=0, pady=5, padx=20, sticky="w")
picturepath_text = Entry(entry_frame, font=("Helvetica", 12), bd=3, relief=GROOVE, textvariable = self.__picturepath)
picturepath_text.grid(row=5, column=1, padx=8, pady=5, sticky = "w")
picturepath_button = Button(entry_frame, text="Choose", command=open_first)
picturepath_button.grid(row=5, column=2, pady=5)
```

**Figure 11**: Code for file dialog button

```
def open_first():
    root.filename = filedialog.askopenfilename(initialdir="/desktop", title="Select the picture", filetypes=(
        ("jpeg files", "*.jpeg"),("jpg files", "*.jpg"),("png files", ".png")))
    picturepath_text.delete(0,END)
    picturepath_text.insert(END, root.filename)
```

**Figure 12**: Code for the making of file dialog for picture upload

```
gender_label = Label(entry_frame, text = "Gender:",bg="light gray", font=("Helvetica", 16, "bold"))
gender_label.grid(row=4, column=0, pady=5, padx=20, sticky="w")
gender_combobox = ttk.Combobox(entry_frame, font=("Helvetica", 15), state="readonly", textvariable = self.__gender)
gender_combobox['values'] = ("Male", "Female", "Others")
gender_combobox.grid(row=4, column=1, pady=5,columnspan=2)
```

**Figure 13**: Code for the making of combo box for the gender

Other than this, I also learnt how to make a tree view to show data from the database as a table data list using ttk.Treeview.

## II. Learning the ins and outs of sqlite3

For this project, I understood that I needed to use database for the program to work. I chose to use sqlite3 as it can be done without the use of a separate server. As SQL isn't discussed in class yet, I had to do research to learn how to use it to manipulate data entries. Here, I learnt how to create a database, create a connection to it, how to create a table with specific field names and data types. I also learnt the basics to the insert, update, select, delete queries that is used widely for this project.

```
# Adding data to the database after validation
def add_new_data(studentid, firstname, lastname, gender, picturepath, present, absent):
    conn = sqlite3.connect("Cool_School.db")
    c = conn.cursor()
    c.execute('''INSERT INTO students (studentid, firstname, lastname, gender, picturepath, present, absent)
    VALUES (?, ?,?,?,?,?,?)''', (studentid, firstname, lastname, gender, picturepath, present, absent))
    conn.commit()
    conn.close()

# Deleting data in the database
def delete_data(studentid):
    conn = sqlite3.connect("Cool_School.db")
    c = conn.cursor()
    c.execute("DELETE FROM students where studentid=?", (studentid,))
    conn.commit()
    conn.close()
```

**Figure 14**: Code for insert and delete query

```
# Updating data in the database after validation
def update_data(studentid, firstname, lastname, gender, picturepath, present, absent):
    conn = sqlite3.connect("Cool_School.db")
    c = conn.cursor()
    c.execute("UPDATE students set firstname=?, lastname=?, gender=?, picturepath=?, present=?, absent=? WHERE studentid=?"
    (firstname, lastname, gender, picturepath, present, absent, studentid))
    conn.commit()
    conn.close()

# Searching function that uses category filtering
def search_data_by(searchby, searchtxt):
    print(searchtxt)
    conn = sqlite3.connect("Cool_School.db")
    c = conn.cursor()
    c.execute("SELECT * from students WHERE " + str(searchby) + " LIKE '%" + str(searchtxt) + "%' ORDER BY studentid")
    rows = c.fetchall()
    conn.close()
    return rows
```

**Figure 15**: Code for select and update query

## III. Use of PIL

For this project, because the image processing needed is not that advanced, I decided to learn how to use PIL to open and read an image file. This can be done by easily importing

Image for PIL and use the open method. I also make use of the save method to save the file to a specific folder in the project with a specific name.

```
# Saving the picture in the known file whenever it is uploaded as student data
def picture_saving(self):
    path = self.__picturepath
    img1 = Image.open(path)
    name = self.get_full_name()
    newpath = r".\known\{}.jpg".format(name)
    img1 = img1.save(newpath)
    self.__picturepath = newpath
```

**Figure 16**: Code for reading image files and saving them

## IV. Use of OS

For this project, I make use of the functions in the OS module for interacting with the operating system. In this case, I want to delete the file of the picture if the student's data is being deleted from the database, aiming to reduce the storage and making the searching more efficient (especially when comparing the face encodings of the image file in the folder). Therefore, I make use of the remove function in the OS module to remove the file whenever a delete query is being implemented to the database.

```
#Function to delete the data
def delete_entry(self):
    check = query_functions.check_valid_id(self.__studentid.get())
    if check != None:
        messagebox.showerror(title="Error", message=check)
        return
    else:
        # Warning message
        answer = messagebox.askyesno("Deleting data", f"Are you sure you want to del
        if answer == True:
            os.remove(self.__picturepath.get())
            query_functions.delete_data(self.__studentid.get())
            messagebox.showinfo(title="Successfully deleted!", message=f"Data of {se
            self.fetch_data()
            self.clear_data()
        else:
            return
```

**Figure 17**: Code for removing the file from the folder using the file path

## V. Facial recognition machine learning

The main part of the project is using the facial recognition machine learning to identify faces. I used an external library for this instead of training my own machine, however, I learnt the basics of how this library works through reading and researching various articles. In this section I will be explaining the basics of how the functions in the extended library, made by Adam Geitgey works.

The first step that needs to be taken when wanting to recognize faces are detecting the location of human faces through the image. This library used Histogram of Oriented Gradients (HOG) to find faces in an image. The image is made to black and white as colour is not needed to detect faces. The aim of this image processing is to make the image show the gradient. This is done by comparing every single pixel of the image and compare it with the one around it. Then, an arrow will be drawn to point to where the image is getting darker.

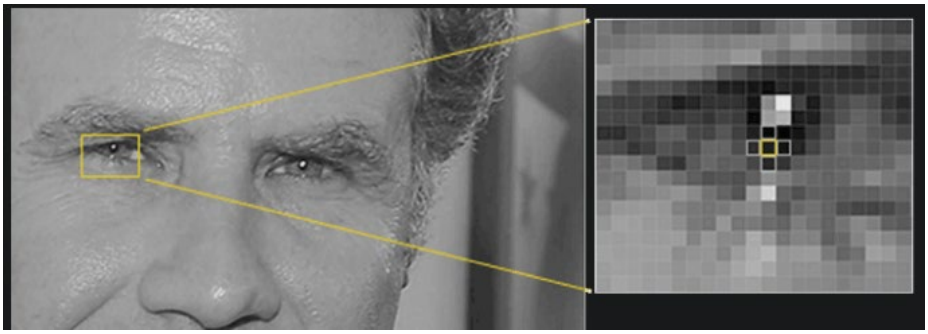

**Figure 18**: Analyzing each pixel and

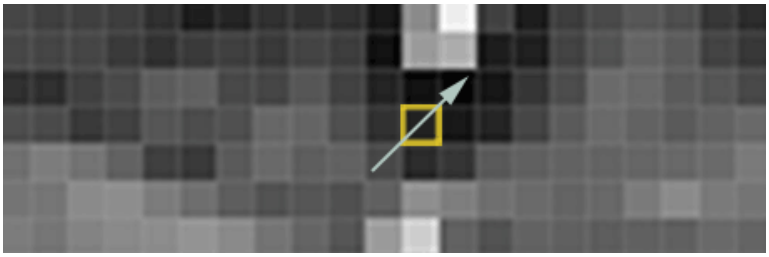

**Figure 19**: Replacing the pixel with an arrow to point to the darker side

When this process is repeated to all the pixel, it will create an image that shows the flow from light to dark across the entire image. This is done to make the images to have the same exact representation (using arrows) instead of pixels that will result in different pixel values depending on how dark or how light the pixel is. To further simplify this, the library breaks up the image into small squares of 16x16 pixels each. In each square, the gradient point is each direction is calculated. The square is then replaced with arrow directions that were the

strongest. This will produce a very simple representation that captures the basic structure of a face.

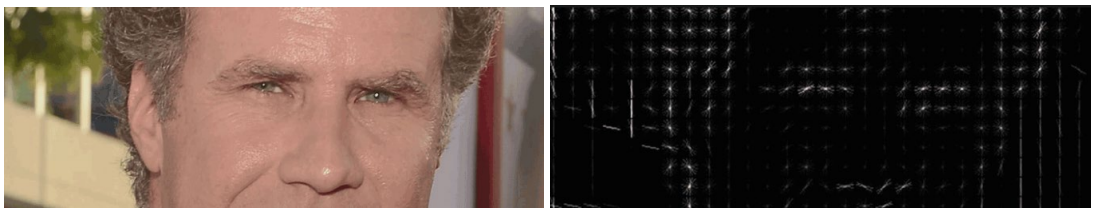

**Figure 20**: Before and after an image is turned into a HOG representation that captures the major features of the image regardless of image brightness.

This is used to detect the faces. A bunch of face images are being fed to the machine and from there, universal face pattern in being generated through a lot of training. This will be the "base" that is used to compare the faces in other images. When the pattern of the image is similar to the "base" it will be detected as a face.

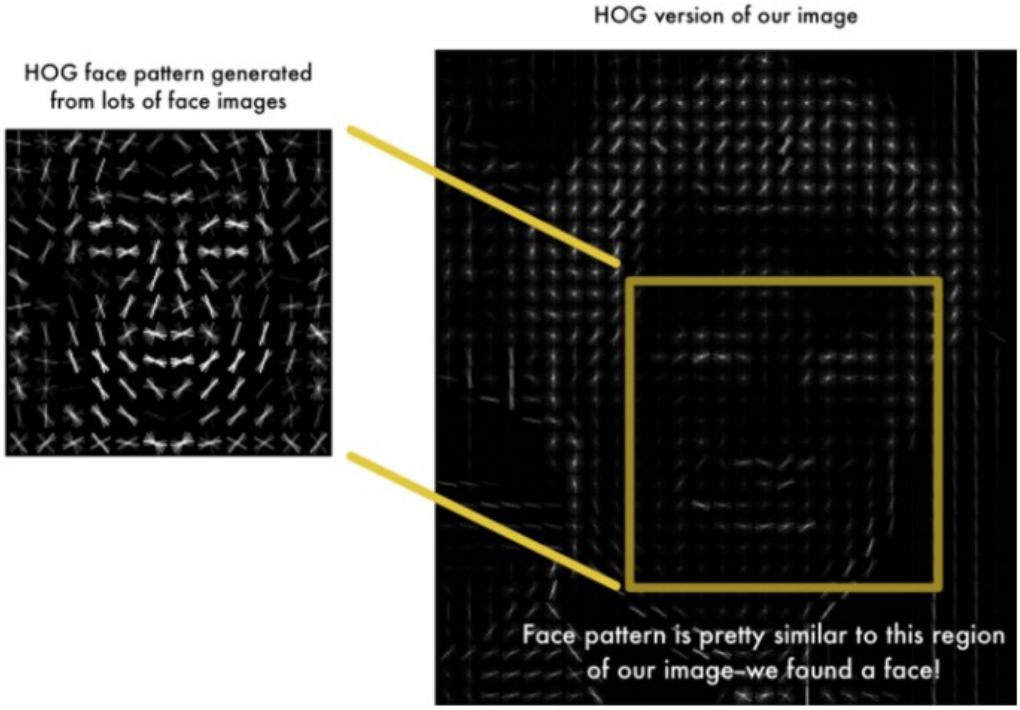


**Figure 21**: The use of the "base" face pattern as face identifier in the program

To improve the accuracy of face comparison, an algorithm called face landmark estimation is used in this library. This is where 68 specific landmarks points needs to be marked. Then, the machine is trained to find all 68 specific points on any face. The aim of this is to try to modify each picture so that the eyes and lips are always in the image.

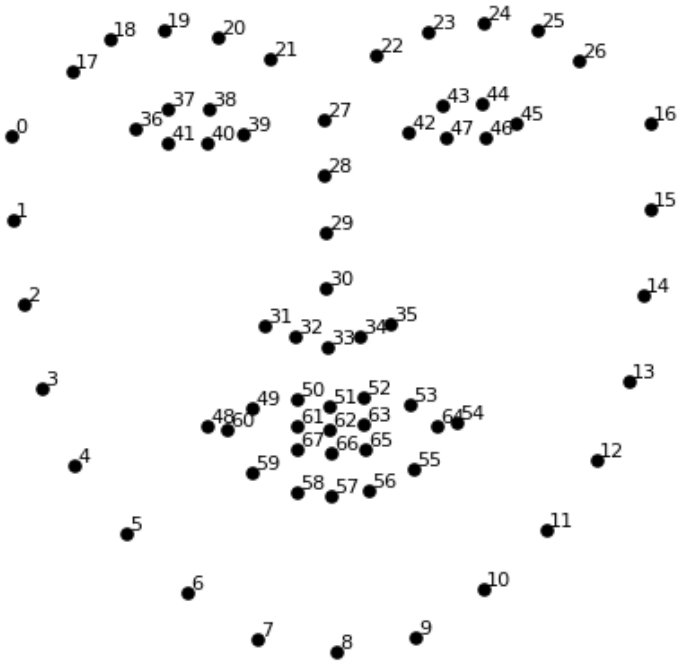


**Figure 22**: The 68 points that needs to be identified in a face

After knowing the position of the eyes and mouth, the machine will rotate, scale and shear the image so that it is centered. It is said that the transformations done are rotation and scale that preserve parallel lines. This aims to make sure that the eyes and mouth are in the same position no matter how the face is turned so that the encoding will be much easier and more accurate.

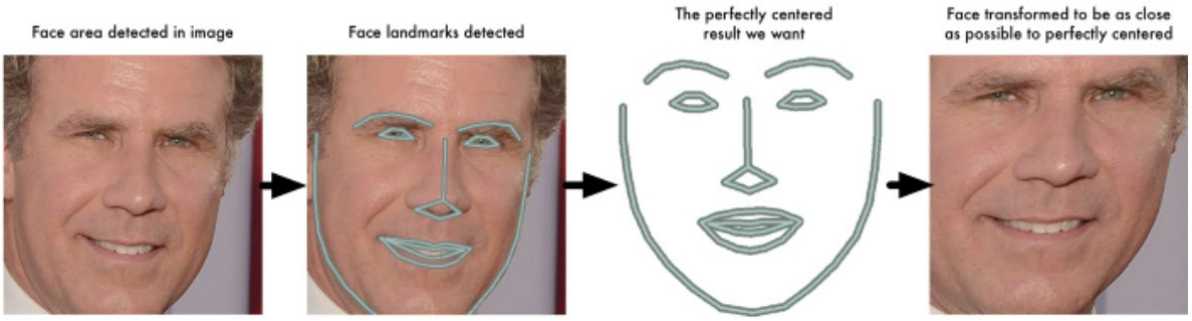


**Figure 23**: The 68 points that needs to be identified in a face

After this step, the most important step follows, this is used in the comparing of faces to see which one matches. In a lot of cases, we humans compare faces for a physical feature such as colour of eyes, shape of face, shape of nose, length or ears, etc. However, it is difficult for a machine to use these as comparisons.

Therefore, deep convolutional neural network is used. This trains the machine to generate 128 measurements of each face and figure out what needs to be measured to represent the features of a human face. This training will need three images to work. Load two pictures of the same person and one of a completely different people.

The algorithm then generates measurements for the three images and tweaks the neural network so that the measurements that are generated for the two same people are closer than the one different person. This way, it can easily tell people apart. This is repeated over and over again until the neural network can consistently generate 128 measurements that is supposedly unique for each one but has a close distance to one another if it's the same person.

These 128 measurements of each face are called an embedding. This concept of processing a complicated raw data like an image to a simplified computer-generated code is really common in the machine learning industry. Once the machine is trained enough, it will be able to easily perform the task with a high accuracy. However, it is unclear what the 128 measurements that is generated means. It is proven that deep is way better than humans when figuring out which parts are needed to be measured in a human face.

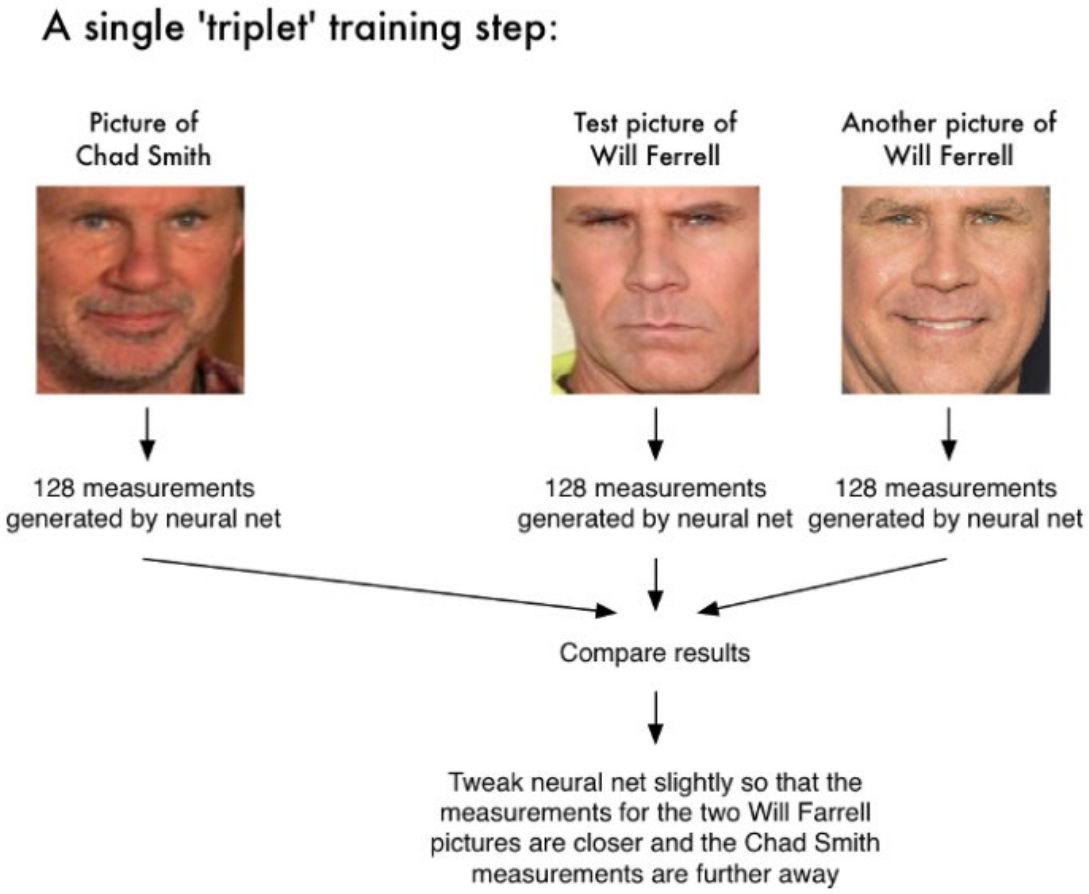


**Figure 24**: How the neural network is trained

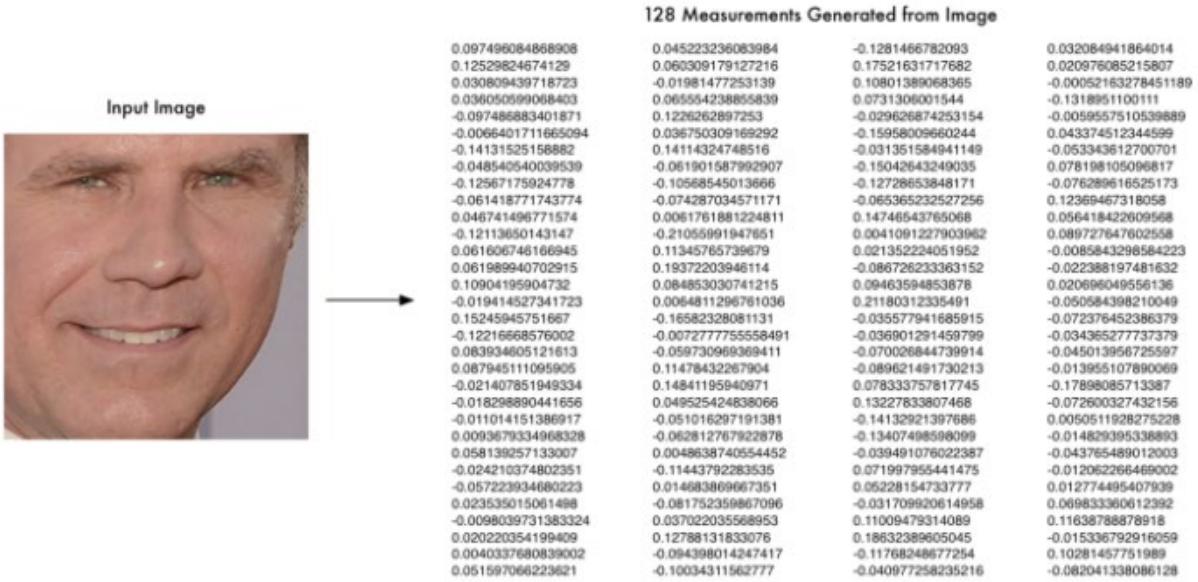


**Figure 25**: The 128 measurements that is generated by a neural network

## E. Evaluation

### I. How effective is this program?

The final version of this program works just fine when managing the database, such as insert delete, update and search. As for the attendance algorithm, it is accurate for some test cases, it can easily detect the faces of the same person accurately. However, it seems to be less accurate when it is trying to find the locations of the faces itself and detecting how many people are in the image. This may be related to the quality of the image, but it seems like if the image contains too many people, not every human face can be detected.

In terms of the efficiency, it does take some time to compare the faces during the attendance, as the algorithm loops through the whole folder of known faces to check the attendance (this is assuming that the program is only used for one class only). However, I tried to make the program more efficient in terms of storage and computing power needed. Therefore, although there are still flaws to this program, it mostly works according to the initial purpose of this project.

### II. Future improvements that can be done

As there are still a lot of flaws, the list below are the things that can be improved in the future to make the program better and more efficient:

- Make a proper Relational Database Management System (RDBMS) where there are multiple entities such as students, class, courses, etc. With all the data being in the third normalized form to make the data less redundant and accurate. This will allow the system to have better clearance and it can be used in an actual organization instead of just one class.
- Adding a login page for different level of access and security. As the data can be sensitive, it is important for the program to be protected using authentication of username and password with different level of access given for different people.
- Having multiple modules and pages to navigate through instead of just having one big window to navigate everything. It would be much neater if there are specific pages for different functionalities of the program.
- Another way to reduce the hassle for the teachers can also be by letting the students have access to the program and taking their own attendance using their webcams (in case of online classes).

**III. Reflection**

Overall, I'm satisfied with the final product of my project. Although I found a lot of room for improvements in the program, it still serves the initial purpose of the program. Throughout the development of this program, I have improved in my research skills and most definitely in my problem-solving skills too.

One of the hardest problems I had to solve was how to store the images in the database. I initially plan to use BLOB (Binary Large OBject). However, I found out that to retrieve or read the data, I will need to save it as a file again and again. I realized that this will be a problem as I will need to loop through the image again and again. This will not be efficient in terms of the storage. I then tried to store just the encodings as a text in the database, however that didn't work either. I finally decided to save the picture in a fixed folder and store its file path in the database. This way, every time I need to open the image, I can just use the open function from PIL and the file path.

All in all, I have learnt a lot from my first python project and ended up being satisfies with the result.

## F. Evidence of working program

### I. Testing

1. Entering new data to be saved to the database

- Normal data entered (Expected: msg box success save and added in the data list, entry form is cleared)

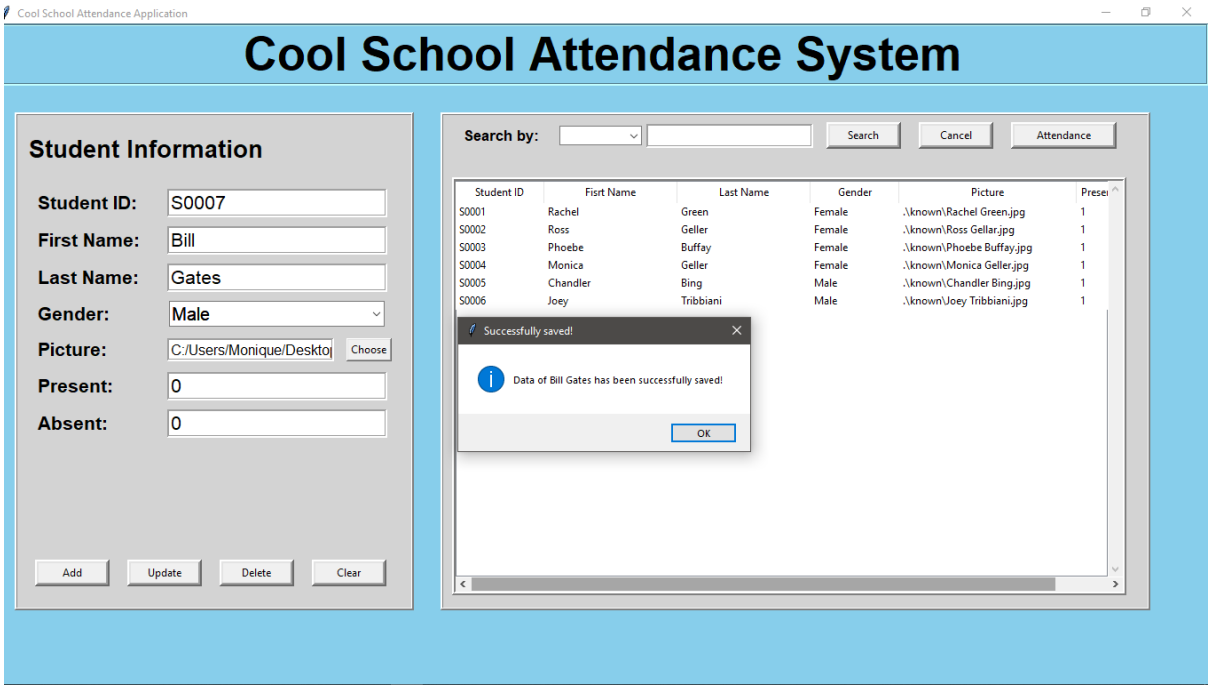


**Figure 26**: The message box appears correctly

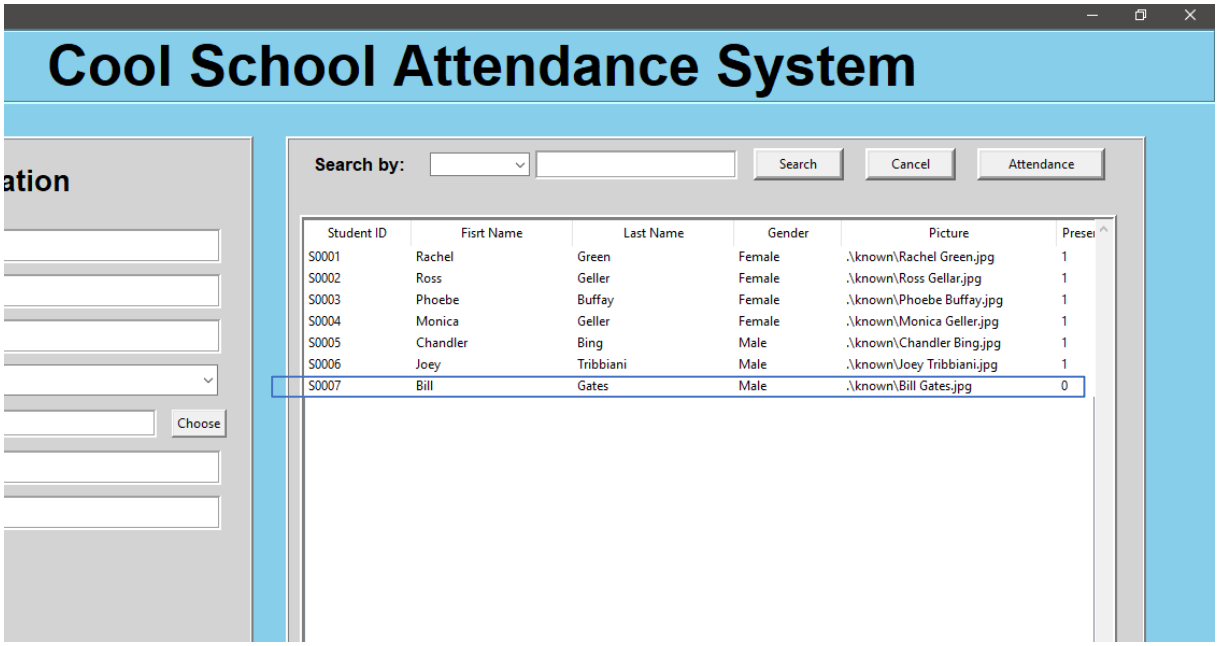


**Figure 27**: The data is immediately updated in the data list on the right side

- Abnormal data entered (Expected: Error message is shown according to the respective errors)

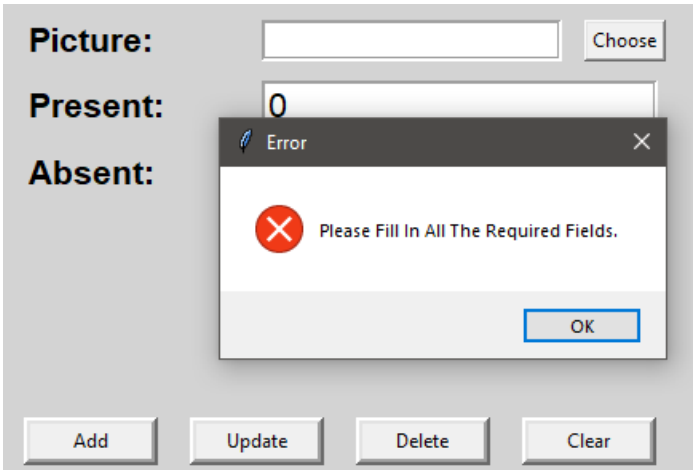


**Figure 28**: Error message when there are empty fields

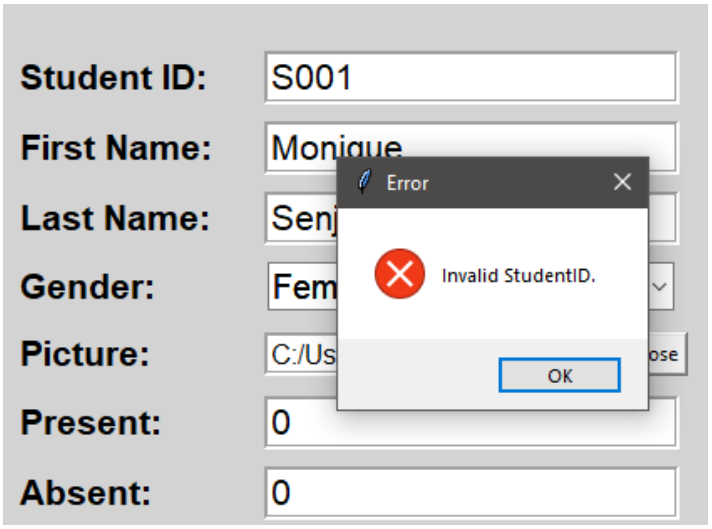


**Figure 29**: Error message when the student id is not according to the format and is not unique

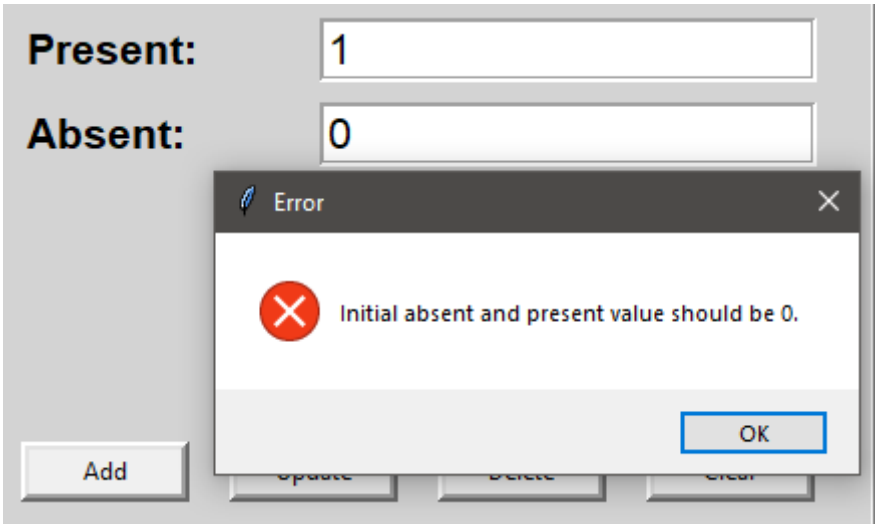


**Figure 30**: Error message when the initial absent and present value is not 0

2. Updating already existing data
   - Normal data (Expected: msg box success save and added in the data list, entry form is cleared)

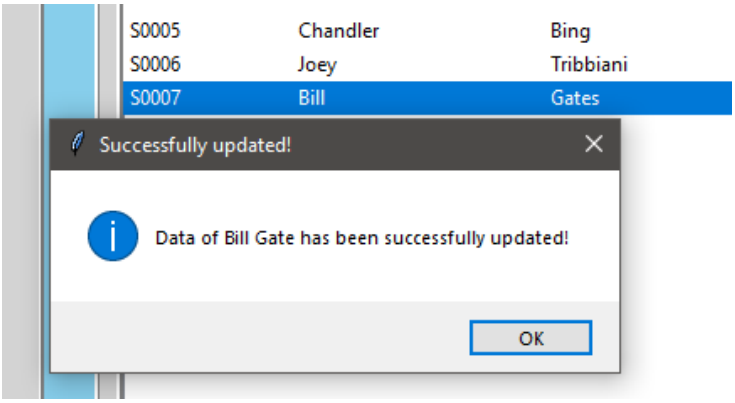


**Figure 31**: Message is shown that it is successful

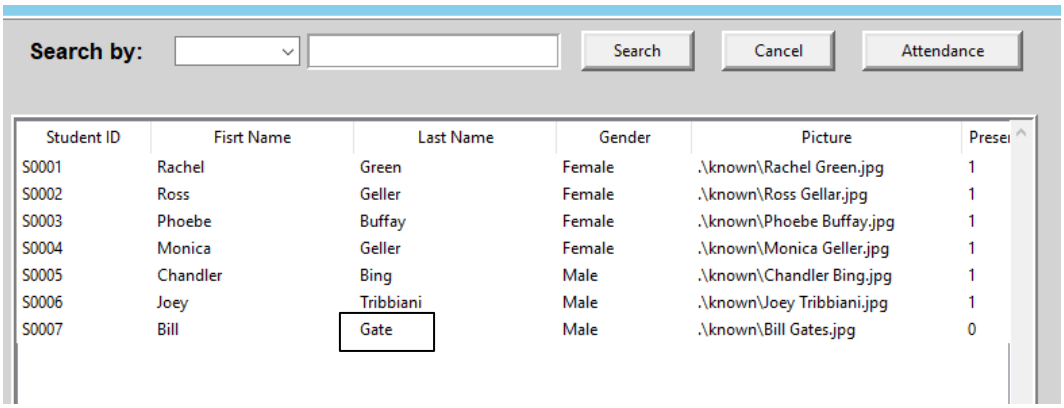


**Figure 32**: The data is updated in the data list

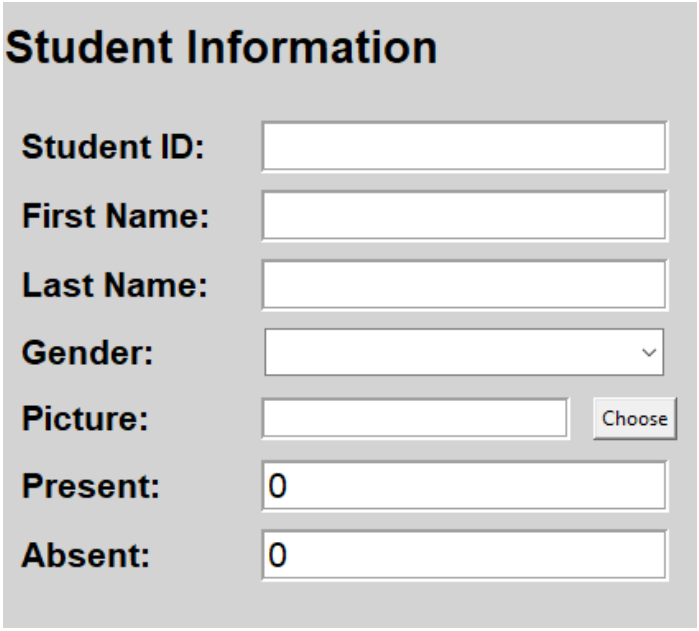


**Figure 33**: Entry form is cleared

- Abnormal data (Expected: Errors to be shown according to the respective errors that is from the validation, this is the same as the add new test case)

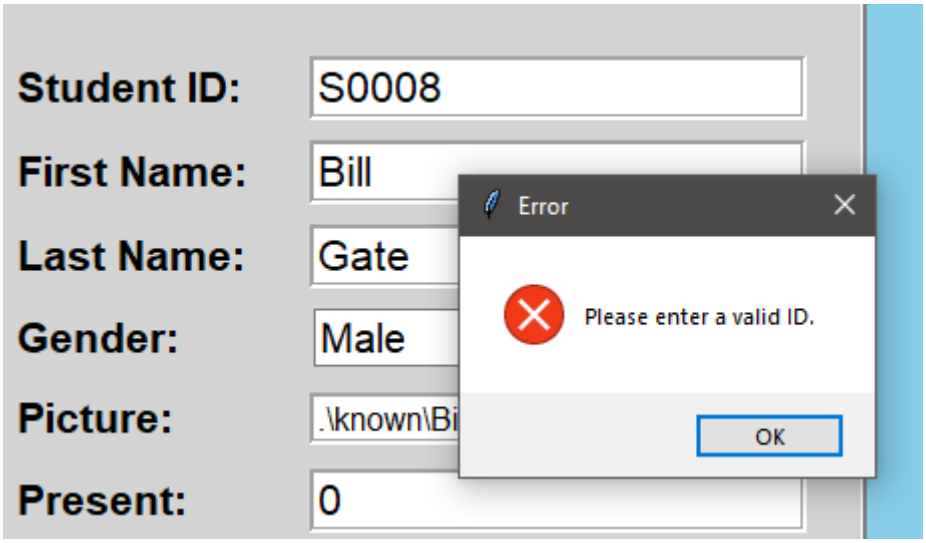


**Figure 34**: The update failed as the ID is not registered in the database

3. Deleting the entry in the database
   - Normal data (Expected: msg box appears asking for confirmation, msg box appears if the deletion is successful, data list is updated and entry form is cleared)

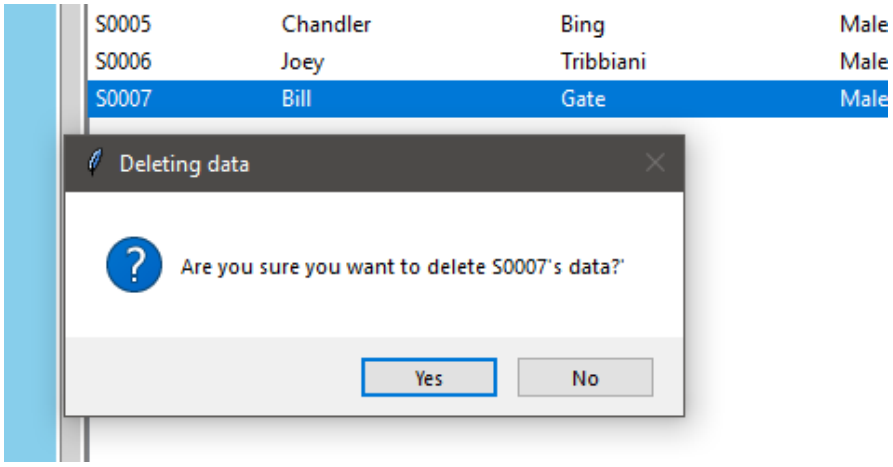


**Figure 35**: Confirmation box asking a yes or no answer

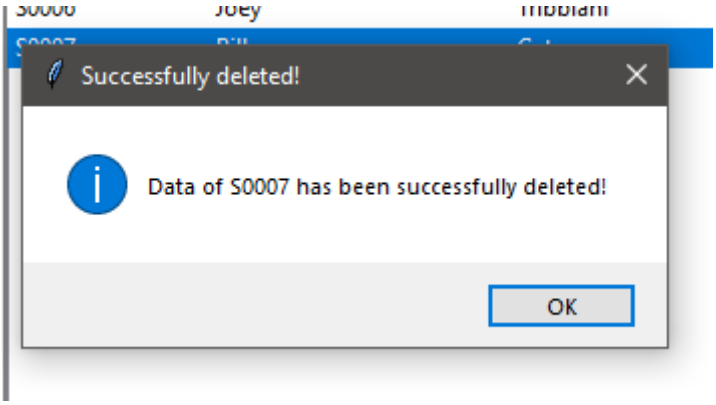


**Figure 36**: Message box showing it is successful

| Student ID | Fisrt Name | Last Name | Gender | Picture | Prese |
|---|---|---|---|---|---|
| S0001 | Rachel | Green | Female | .\known\Rachel Green.jpg | 1 |
| S0002 | Ross | Geller | Female | .\known\Ross Gellar.jpg | 1 |
| S0003 | Phoebe | Buffay | Female | .\known\Phoebe Buffay.jpg | 1 |
| S0004 | Monica | Geller | Female | .\known\Monica Geller.jpg | 1 |
| S0005 | Chandler | Bing | Male | .\known\Chandler Bing.jpg | 1 |
| S0006 | Joey | Tribbiani | Male | .\known\Joey Tribbiani.jpg | 1 |

**Figure 37**: Data list is updated immediately

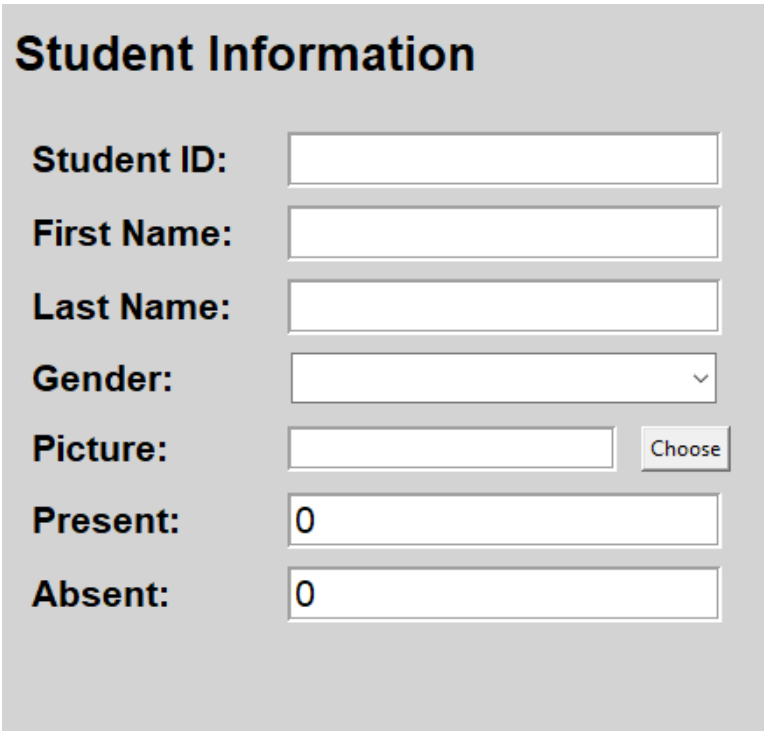


**Figure 38**: Form entry is cleared

- Abnormal data (Expected: Errors to be shown according to the respective errors that is from the validation, this is the same as the update test case where the id is invalid)

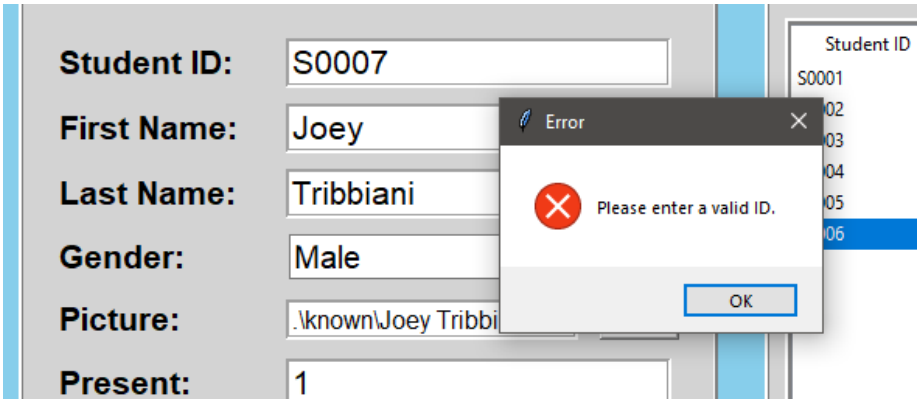


**Figure 39**: The ID cannot be found in the database

4. Searching the data in the data list
   - Normal data (Expected: Data is shown according to the search filter, when cancel is clicked, all data will be shown)

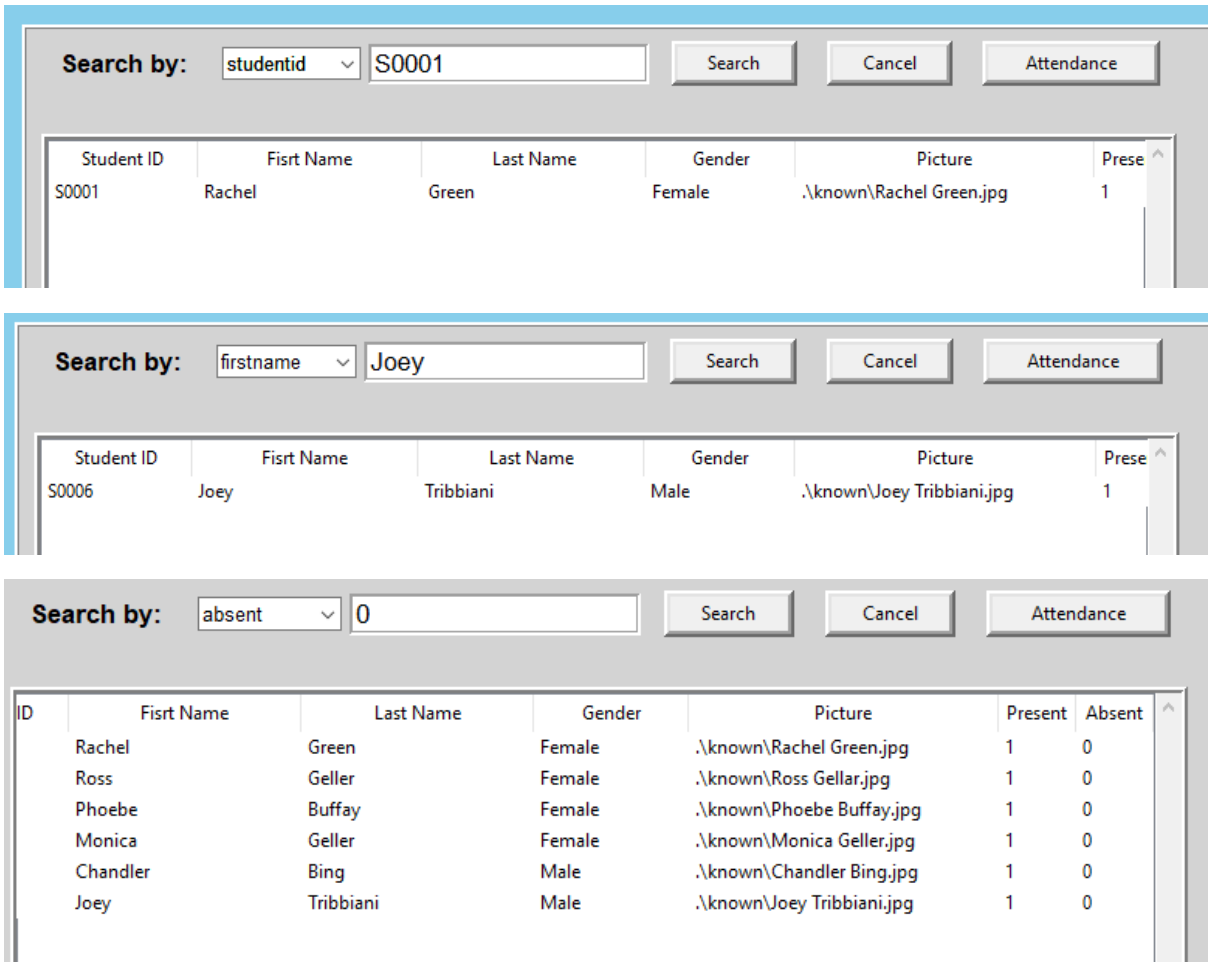


**Figure 40**: The data list is immediately updated when the search button is clicked

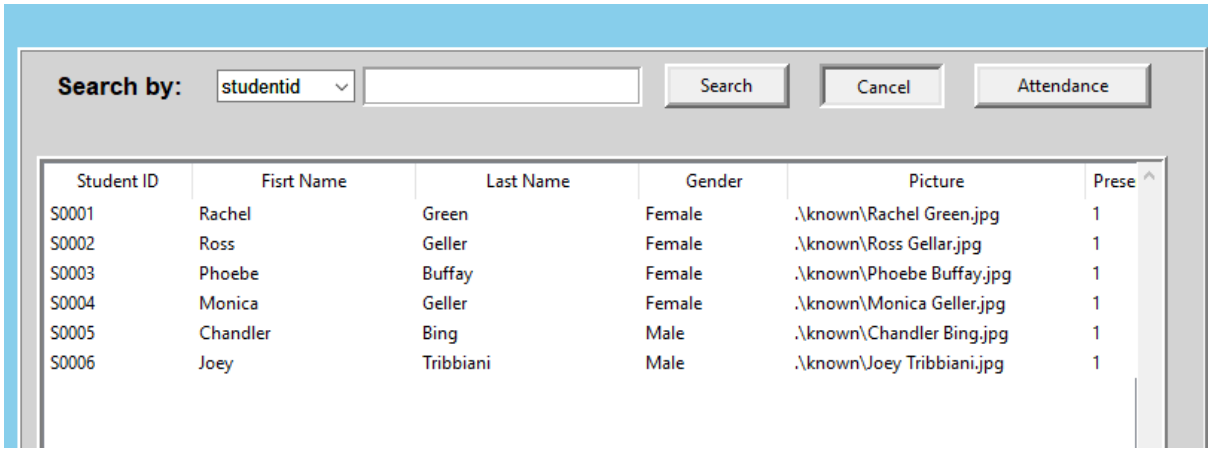


**Figure 41**: The data list is immediately updated when the cancel button is clicked

5. Attendance taking

- Normal data (Expected: msg box success save and the data is updated)

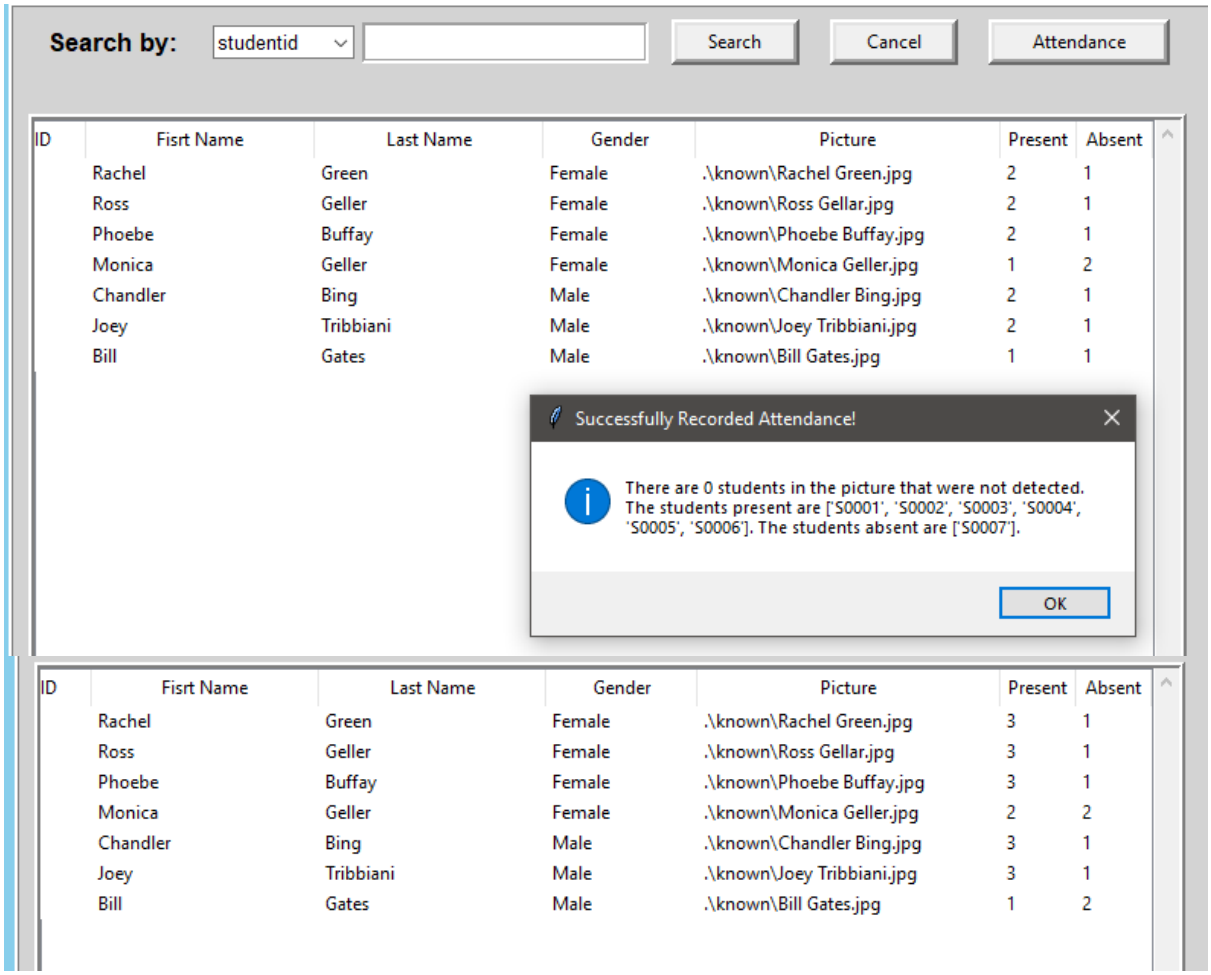


**Figure 42**: A message box of the details are shown and the present absent numbers change accordingly

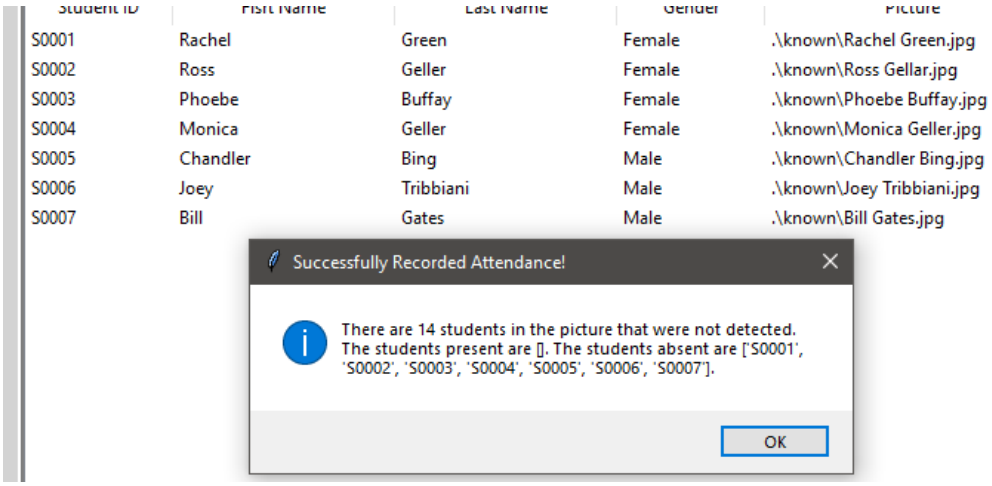


**Figure 43**: If the picture uploaded doesn't contain any of the students, it will mention how many unrecognized faces there are.

## Credits

- Adam Geitgey
  - Facial-recognition library v1.2.2
    **https://github.com/ageitgey/face_recognition**
- OS
- PIL
- Tkinter
- SQLite3
- All images (Test samples) credited to the owner

## Bibliography

Face recognition with OpenCV, Python, and deep learning. (2018, June 18). *PyImageSearch*. **https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/**

Geitgey, A. (n.d.). *Face-recognition: Recognize faces from python or from the command line* [Python]. Retrieved January 9, 2021, from **https://github.com/ageitgey/face_recognition**

Geitgey, A. (2020, September 24). *Machine learning is fun! Part 4: Modern face recognition with deep learning*. Medium. **https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78**

*How to create a database in python using sqlite3—Data to fish*. (n.d.). Retrieved January 9, 2021, from **https://datatofish.com/create-database-python-using-sqlite3/**

*How to use Pillow (Pil: Python imaging library) | note.nkmk.me*. (n.d.). Retrieved January 9, 2021, from **https://note.nkmk.me/en/python-pillow-basic/**

Os module in python with examples. (2016, November 21). *GeeksforGeeks*. **https://www.geeksforgeeks.org/os-module-python-examples/**

*Tkinter—Python interface to tcl/tk—Python 3. 9. 1 documentation*. (n.d.). Retrieved January 9, 2021, from **https://docs.python.org/3/library/tkinter.html**