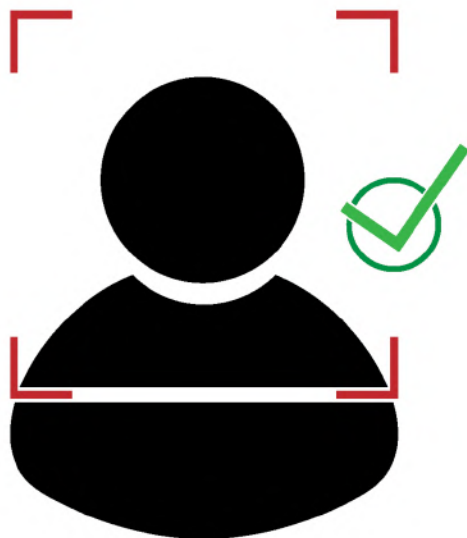


# *Smart Absence*



*Project by :*  
**EL KACHOUANI TAOUFIK**

## **Table of Content :**

<b>Introduction</b>	<b>2</b>
The project :	2
The story behind :	2
My mission :	3
<b>The Solution</b>	<b>4</b>
The approach :	4
Workplan :	5
<b>Implementation</b>	<b>6</b>
Tools :	6
Facial recognition algorithm :	6
LBPH Algorithm :	6
Conception and Application :	7
General conception and database :	7
Main window :	9
Add seance window :	9
Recognition window :	10
Export window :	11
Options window :	11
Mechanism of the application :	12
Additions :	13
The logo :	13
Setup script :	13

# 1. Introduction

This document contains an explanation of the project, the story behind this project, answering the question how I ended up making it, the steps followed to construct it and solve all the problems that came along the making, and the constructions on how to use the program properly.

## 1. The project :

Let's start with a brief explanation of the project, Smart Absence came as a solution to one of the problems we had noted in our school. Some lecture classes can have up to 300 students, with this large number of students, it's hard to manage their absence. The traditional way implies to call the names from a list, and whoever hasn't answered is noted as absent. On average, this method takes more than 30 minutes from the lecture's time. As a solution, we thought of a system based on facial recognition, that will scan the students' faces before entering the lecture, and it will provide different ways to extract the lists. Also, this solution will provide a direct digital data stored of students, and their presence behaviour for the administration to monitor and analyse.

## 2. The story behind :

This section discusses how I ended up making this project. I always wanted to use Python as an oriented object programming language, so when I had the chance to work on this project I took it. I was a representative of my master class on the bureau of students in our school, under the ENFORCERS party. During this we had a meeting with one of the clubs in the school, OpenMind, it was a club of robotics and tech, and was then chaired by Hamza Saber, who had the idea of working on projects as solutions for our school's problems. I remember coming late to the online meeting, and I asked him to do a recap of the first projects, Smart Absence was one of them. I had the idea at first to contact a fellow colleague of mine to ask him if he would like to work on this idea, because he did a similar one as a lecture project while we were taking the image processing lecture. I had contacted him, and he refused to work on it because he had more work and responsibilities so he couldn't add something else. Then I saw the opportunity, what I wanted was there, a project would be solved using OOP Python. After that online meeting, I had discussed the idea with Hamza in detail, and he was glad that I was interested, so he asked if I would like to carry on and work on it. At that time I was working on my graduation project, I wasn't sure if I could do them both, but I took the challenge, and started the conception process. After several meetings we fixed the objectives to work on.

I want to thank you all, the ENFORCERS party, OpenMind Club, the national school of applied science Khouribga and whoever had a contribution in this work, for this opportunity and experience.

### **3. My mission :**

I was charged to develop a Python based application that will work under a Raspberry Pi 4. The objectives were to assure the recognition of students, keep the records on a database to extract afterwards and assure a sort of flexibility and reliability. Let me cite that at this stage of my career, my developpement skills were primitive and basic, if you are a developer reading this, I will be honored to know your opinions.

## 2. The Solution

As cited before, the problem of keeping track of presence, especially when there is a large number of students, is time consuming. On our part we had to come up with a solution based on our skills, studies and interests, I mean a digital technological way to solve the problem.

### 1. The approach :



Our approach is to use facial recognition systems using Python, implemented on Raspberry Pi computers, the system may contain many of the computers interconnected to work in parallel and update the database. This system will have a web based application, to extract the data by different granularity, date, class or lecture etc..

To further explain our vision, every classroom will have a unit on the door that contains a Raspberry Pi computer with a touch screen and a web camera. The

responsible of each class that will take a lecture on that classroom should enter his class's data, the professor and the module of the lecture. The application then starts the detection and recognition phase, each student should position in front of the camera, and the system will do the rest, in the meanwhile, it keeps recording the present students on a database. For data extraction, we had the vision of having a web based application that can be accessed from anywhere on the school's network to extract the presence list.

Essentially, this was our vision, this work is the first version therefore it may not cover all the features discussed earlier.

## 2. Workplan :

I always trusted one way of working, and it is by taking the work I have to do as a big problem that needs a solution. This big problem can be divided into small problems, and by solving the small ones you can sum them up and solve your main big nasty problem. In this case the big problem was the development of an application, so the small problems for me where (by order of importance) :

### 1. A facial recognition algorithm.

The main objective here is to found a suitable algorithm that can be reliable and fast for recognizing students' faces.

### 2. Application conception and database.

This step's objective in general, is making a MCV conception to, starting with the database, the classes, and the UI view.

### 3. Starting coding.

This phase is the last one, to start summing the solutions found before, into one application.

### 3. Implementation

Before I start talking about the implementation, and as I cited before, my development skills are basic at this time. Also I haven't provided UML schemes, maybe in the next updates. And I want to add that this is the first version, it's not the final one. But I welcome you to share with me your opinion, suggestions and ideas to improve this project more.

#### 1. Tools :

What's easier and simpler to use for a problem like this than Python, specially with OpenCV, that contains a large number of algorithms and solutions for image processing problems, and for the GUI, I used Qt5 under the wrapping of PyQt5. For the database, I had chosen to work with SQLite3. These were our main tools to start with, also Python is pre-installed on Rasbian, the Raspberry Pi operating system.

#### 2. Facial recognition algorithm :

The recognition is quite different from detection, facial detection has the objective of locating where the face is in a picture. The recognition on the other side, is to identify the face and map it to an ID or a name.

This is the first small problem to solve before starting the actual application coding. As said before, the objective of this step is to search for a suitable algorithm its main characteristics should be :

- Algorithm that would be trained with a small dataset.
- Fast recognition.
- Models can be updated.

After some research, I had decided to go with the LBPH (Local Binary Patterns Histograms) facial recognition algorithm. It's already implemented on OpenCV, you can call it easily. I was also satisfied by its results.

##### a. LBPH Algorithm :

Let's dig deep into it and try to understand how it works. First of all it's full name is Local Binary Patterns Histograms algorithm, it's a combination of Local Binary Patterns (LBP), a texture classification algorithm, and histograms of oriented gradients (HOG) descriptor algorithm.

The first algorithm (LBP) was described back in 1994 by T. Ojala, M. Pietikainen, T. Mehpaa from Oulu University in Finland. It's main idea is to describe the neighborhood of image

elements using binary codes, describing small-scale appearance (textures) of the image. In other words is an image operator which transforms an image into an array or image of integer labels, these labels directly or their statistics are used for further analysis. The LBP detects microstructures such as edges, lines, spots, flat areas, which can be estimated by the histogram.

The second algorithm (HOG), is a feature descriptor, mainly used for object detection, it's essential is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms.

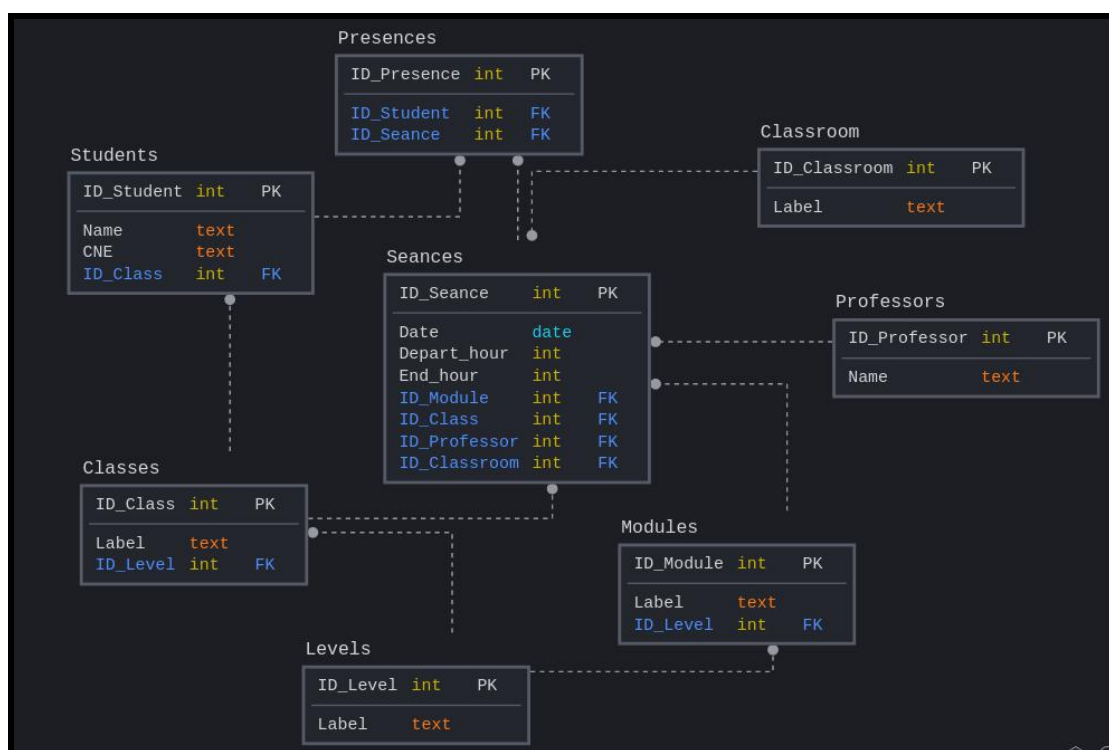
The combination of both algorithms has proved its capability of recognizing faces, based on a pre-trained model.

### 3. Conception and Application :

#### a. General conception and database :

For the conception of how the mechanism of the application will be, and by using oriented object programming paradigme, there is no better way than the Model, View and Control (MVC) model for its flexibility, and code organisation.

The Model part, consists of models, classes that are the elements interacting with each other in the application. It's the structure, how every entity/element should be represented.





The figure below represents the database's schema, it consists of 8 tables, each one will be represented as a class. The 8 tables are the main actors in our application and they are :

- ★ Seances : it's for storing a seance, which signifies the lecture with its class, classroom, date and time, professor, module.
- ★ Presences : is where present students are stored.
- ★ Students : the list of students, each student has a name, CNE (Code nationale de l'étudiant, National Code of the student) and a class id.
- ★ Classes : contain the list of school's classes by each level.
- ★ Levels : is the list of levels.
- ★ Modules : is the list of modules of each level.
- ★ Classrooms : contain the list of school's classrooms.
- ★ Professors : is the list of professors, each professor has a name.

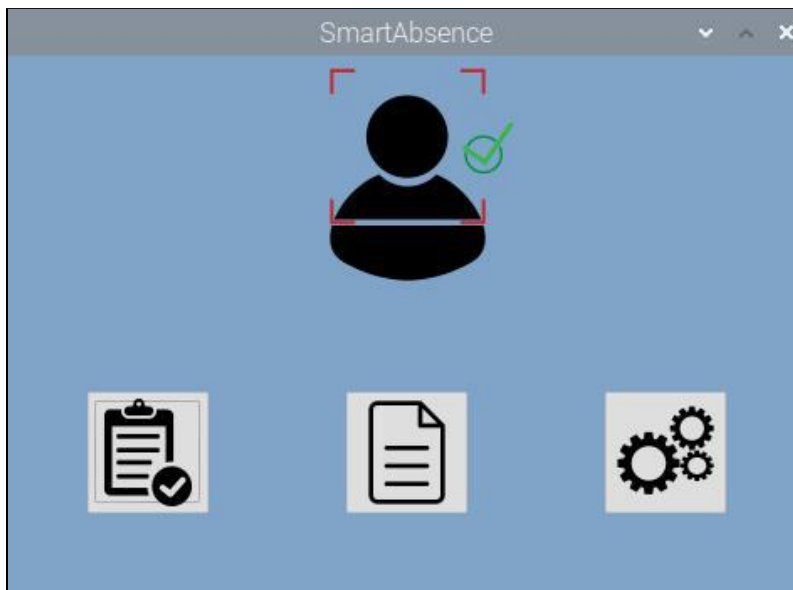
In addition to the 8 classes, the model part of the application also contains a model class for the facial recognition algorithm. This class controls the training, updating and recognition using the algorithm discussed before.

The Control part, lets just call it data access classes, in our case, is the DAO, data access object, or data access classes, it's a set of classes that manage the connection with the database. For each of the 8 models, there is an equivalent in the controles classes, that provides methods to add, update, delete, search and list. The classes are named with the suffix DAO, so there is for example StudentDAO, that provides :

- AddStudent : a method to add a student object to the database.
- DeleteStudent : a method to delete a student object from the database.
- UpdateStudent : a method to update a student object in the database.
- GetStudent : a method to search for a student object in the database.
- GetStudentByCNE : a method to search for a student object in the database by it's CNE.
- ListStudents : a method to list all students in the database.
- ListStudentsByClass : a method to list all students of a specific class in the database.

In general that's the structure of a DAO class, all the other actors have the same classes and mainly the same methods.

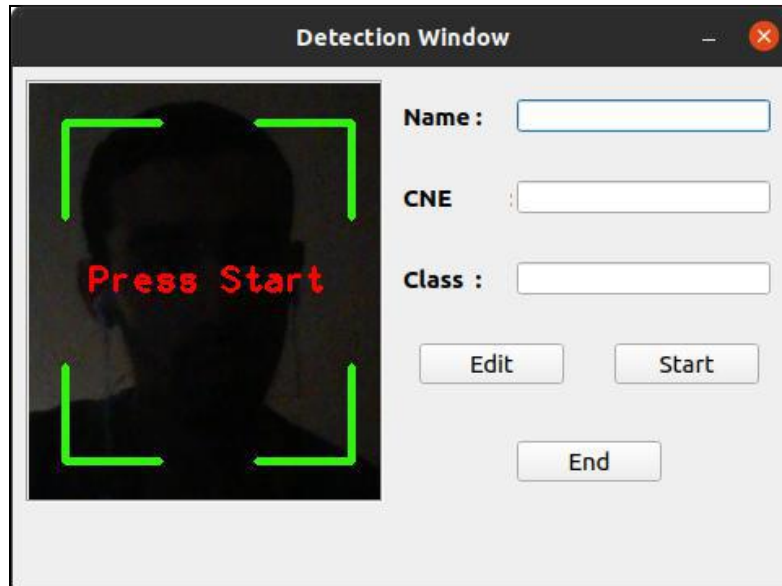
The last part is the View, I chose to work with PyQt5, a graphical user interface library, it's the intermediate between the user and the application. I tried to keep it simple and clear.

**Main window :**

This figure represent the main window, it redirect the user to the functionality he want to use, there is 3 buttons, the first one is to start a new seance, and recognition, the second one is for extracting data, the third button drive the user to more functionalities like data importation and exportation or training the facial recognition models.

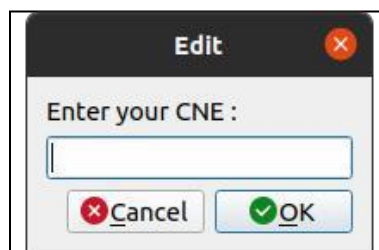
**Add seance window :**

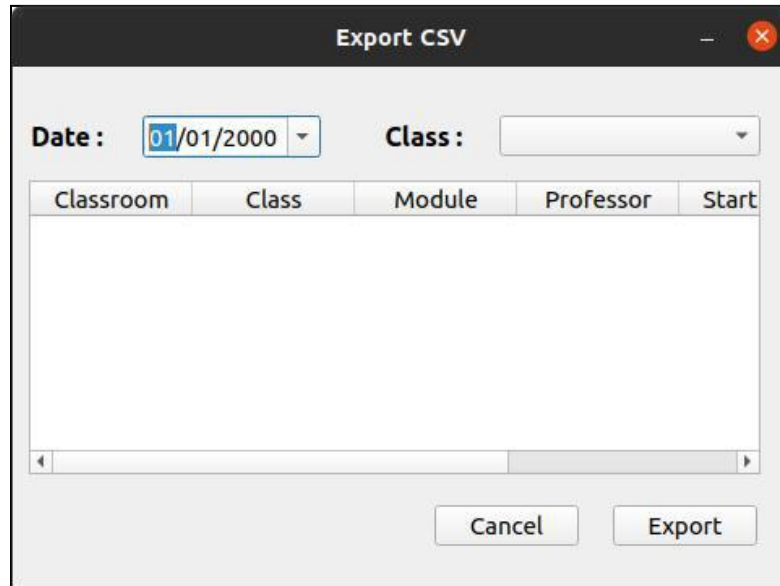
By clicking the first button, a new window shows with attributes of a seance to add, and the start button takes the user to a new window, for recognition.

**Recognition window :**

Before going into the recognition window, let's first talk about the recognition mechanism. The application has to have a model of faces for each class already prepared, this model is trained using a minimum of 5 pictures of each student in the class. So when a seance has been added, the recognition starts by loading the class's recognition model to compare the upcoming students' pictures. The recognition is based on taking 5 pictures with an interval of 500ms between each two, each one will be compared to the class's model, the recognised student is the one who appears in most of the pictures tested.

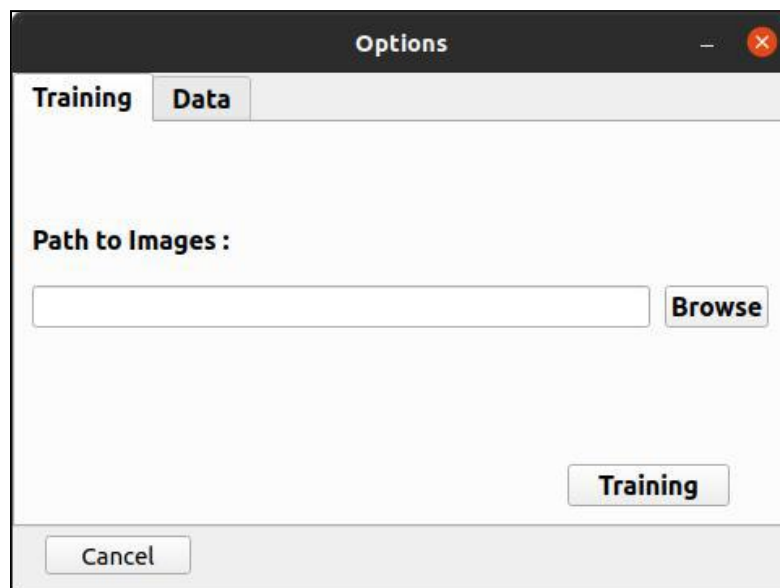
The recognition window consists of the camera view area, text boxes that show the detected student's data. Also there are three buttons, Start to start the recognition for every student positioned in front of the camera. The Edit button is for updating the model, by asking him to enter his CNE, so that if a student has been miss recognised, he can update the recognition model. Updating will be done in the end based on the 5 pictures taken for the miss recognised student. The End button is for indicating that the operation is done, and there are no more students to recognise. If there are any miss recognised students it will trigger the update method.



**Export window :**

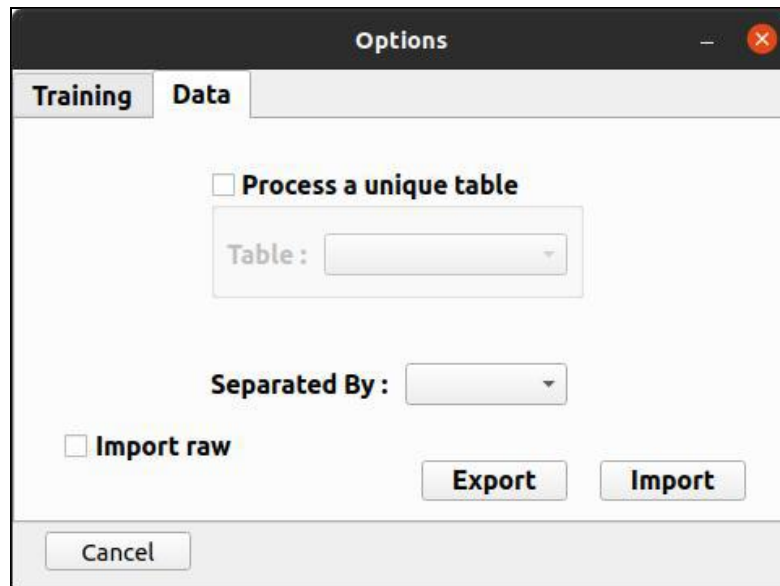
The 'Export CSV' window features a title bar with a close button. It contains two search filters: 'Date' with a dropdown menu showing '01/01/2000' and 'Class' with an empty dropdown. Below these is a table with five columns: 'Classroom', 'Class', 'Module', 'Professor', and 'Start'. The table is currently empty. At the bottom right, there are 'Cancel' and 'Export' buttons.

The Export window contains a table to show seances, and two attributes to search by, the date and class. By specifying a date, the application automatically will search for seances with the same day date, and you can add a class to narrow down the results of the search. The Export button will allow you to export the chosen seance from the list into a CSV file.

**Options window :**

The 'Options' window has a title bar with a close button and two tabs: 'Training' and 'Data'. The 'Data' tab is selected. It contains a label 'Path to Images :', a text input field, and a 'Browse' button. At the bottom right, there is a 'Training' button, and at the bottom left, there is a 'Cancel' button.

The last window is the options window, it contains two tabs, the first one is for training, by providing a path to a folder containing a student's pictures



The data tab provides access to more data related options, such as exporting or importing database tables to CSV, so that you can keep archives of tables as data security measurements. You can either import or export a single table by choosing the table's name from the Combo box. By default the Export button will export all the tables to separated CSV files.

The Import raw option is still in development, the idea behind it is to make it possible to import raw data.

### **Mechanism of the application :**

The application have a directories and filenames specifications :

- The application folder should contain the following directories :
  - Models : it's where to store the facial models for each class.
  - Database : it contains the database files.
  - icons : it's a collection of icons in png format used in the application.
  - DataSet : it's for storing images taken by the application for updating the model.
  - Data : optional folder to contain exported data.
- The facial models of each class should be named after the class ID.
- The CSV files to import data (not raw import) should be named after each table in the database.
- The training should be done by choosing the path to a dataset that contains folders each one of them named after the class ID, also contains folders with

student ID as a name, and the pictures inside should be named by student ID underscore picture ID. Example : **1001/1000/1000\_0.png** is the path to the first picture of student **1000** that is in class **1001**.

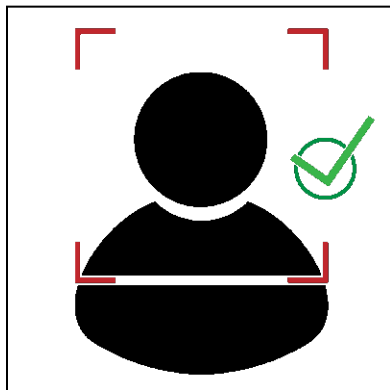
To understand the mechanism I prefer to put the application in a use case. The first version has no remote database or archiving server, it will be a single unit for testing and improving the UI/UX or the back end code.

At first the user should fill the database with static data, that is the students list, classes, levels, models, professors and classrooms. Then he should train the algorithm, by providing a minimum of 5 pictures for each student and respecting the pictures and directories instructions explained up. After all that the user can easily start a new seance for recognition.

### **Additions :**

In this section I want to discuss another side of the project.

#### **The logo :**



The logo has a form of the upper part of a person that is framed by a sort of square corners colored in red. It stands for scanning, detection or recognition, and I added the green symbol that signifies the correctness.

#### **Setup script :**

You will found a shell script with the code files named setup.sh, it's for installing the required packages and libraries, it also move the necessary files to the directory **/opt/Smart\_Absence**, after that it will create the database by running a python script named **db\_create.py**.