Implementation

In this section, we will explain concretely the implementation of our solution by detailing the realization of each component of our project. This description will include the technical aspects and the methodological choices that guided our work, offering a complete and precise vision of our approach.

It is important to note that the employees whose names and images appear during this chapter have given their consent for the use of their data for demonstration purposes. However, for security reasons, no actual IP addresses or configuration data of the installed cameras will be displayed.

1 Face detection

The goal at this level is to detect the presence of a face and capture images of it. In order to achieve this, we coupled OpenCV for reading video streams to the pre-trained Haar Cascade Frontal Face model for detecting faces in these images. [14]

We used detection and capture at 2 different levels according to which the number of frames to be captured varies.

First, we aim to collect a large number of images for a given employee, in order to use them on the one hand for training the face recognition model and on the other hand as verification images at the verification function of the system.

Then, when the system is running, we want to capture a single image every time a new face is detected. This image will serve as input for the facial recognition component.

Thus, by recording the fields precisely framed by the Haar Cascade model, we will have obtained from video images one or more images presenting only the face which appears there.

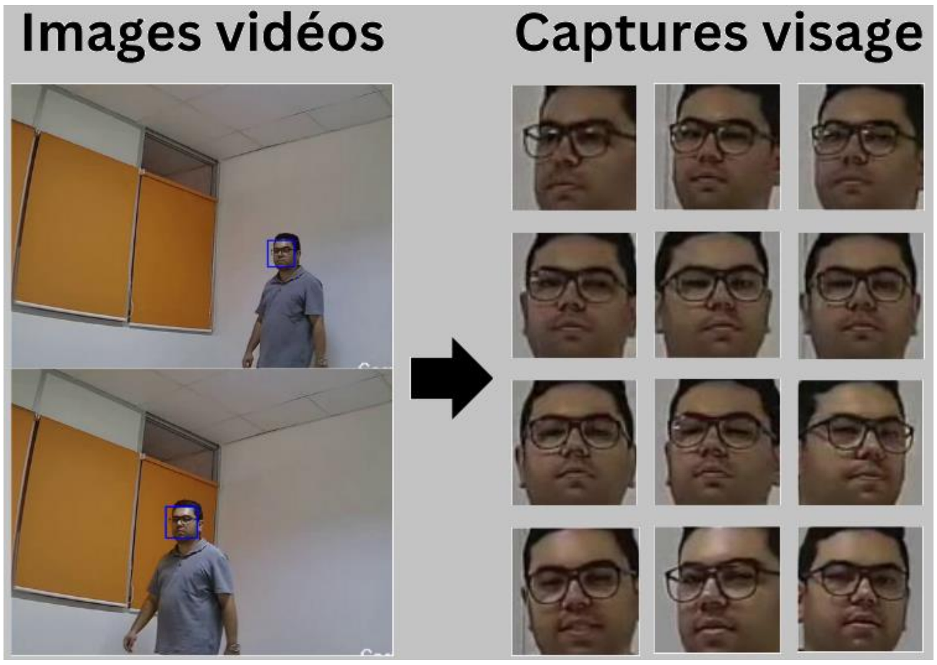


Figure1: Face detection and extraction from video images

2 Facial recognition

In this section, we will present the implementation of the facial recognition component. We will start by explaining the method of creating and training the Siamese model. Then, we will detail the verification function integrated into the system, which uses this model to verify the identity of the detected employee.

2.1 Creation of the Siamese model

The Siamese model requires specific data for effective training. We already have a large number of previously recorded facial images, which we will divide equally into positive and anchored images.

In addition to these images, we will use a large number of negative images from the LFW (Labeled Faces in the Wild) database, a public database containing over 13,000 images of faces of famous and unknown people, collected under various conditions. [15]

This combination of data will allow the model to learn to effectively distinguish between the faces of employees and those of other individuals.

Figure2: Extract facial images from LFW database

To ensure consistent and optimized input to the model, it is crucial to perform image preprocessing that includes resizing to uniform sizes and scaling pixel values between 0 and 1.

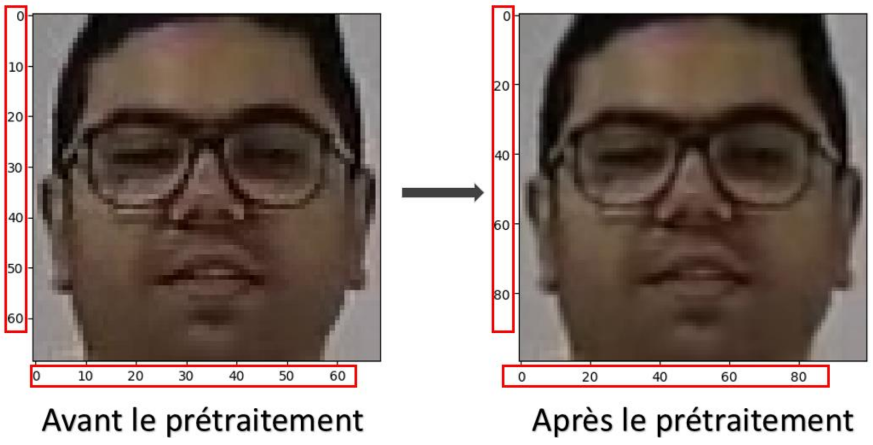
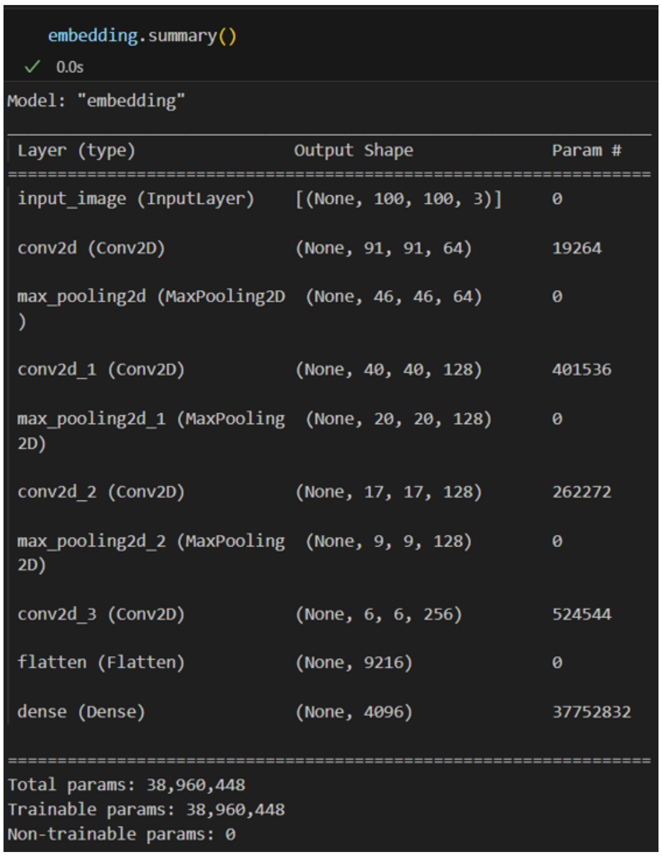


Figure3: Result of the image preprocessing function

The model is composed of three main parts: the Embedding Network, the distance part and the classification part.

The “Embedding” function defines a convolutional neural network model to create image embeddings, which are low-dimensional vector representations used for image comparison in a Siamese model. Defining this function results in a model with a convolutional neural network (CNN) that takes 100x100x3 images as input and produces an embedding vector of size 4096. It is composed of multiple blocks of convolutional and max-pooling layers, followed by a dense layer. The model has a total of 38,960,448 trainable parameters, as shown in the figure below.



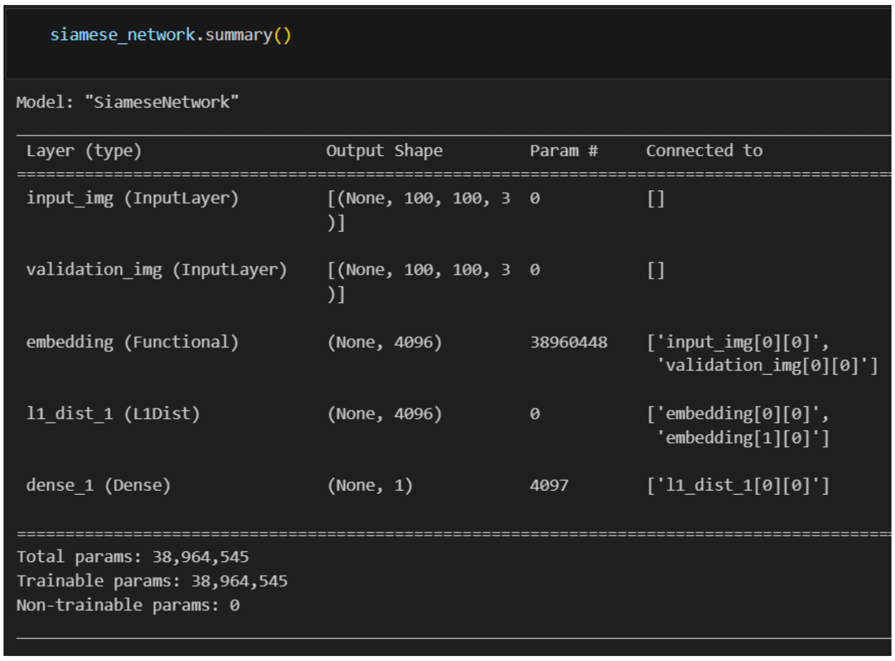
```
embedding.summary()
✓ 0.0s
Model: "embedding"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d (Conv2D)             (None, 91, 91, 64)        19264

 max_pooling2d (MaxPooling2D  (None, 46, 46, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 40, 40, 128)       401536

 max_pooling2d_1 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_2 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         524544

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 4096)              37752832

=================================================================
Total params: 38,960,448
Trainable params: 38,960,448
Non-trainable params: 0
```

Figure4: Interpretation of the “embedding” layer

Next, we move on to the distance part of the model which calculates the distance between the two embeddings (the input image embedding and the validation image embedding). This part uses a custom L1Dist layer which calculates the Manhattan distance (also called L1 distance) between the two embedding vectors.

Finally, the last part of the model is the classification which takes as input the calculated distance and predicts whether the two images are similar or not. This part consists of a Dense layer with a single output and a sigmoid activation which predicts the similarity between the two images.

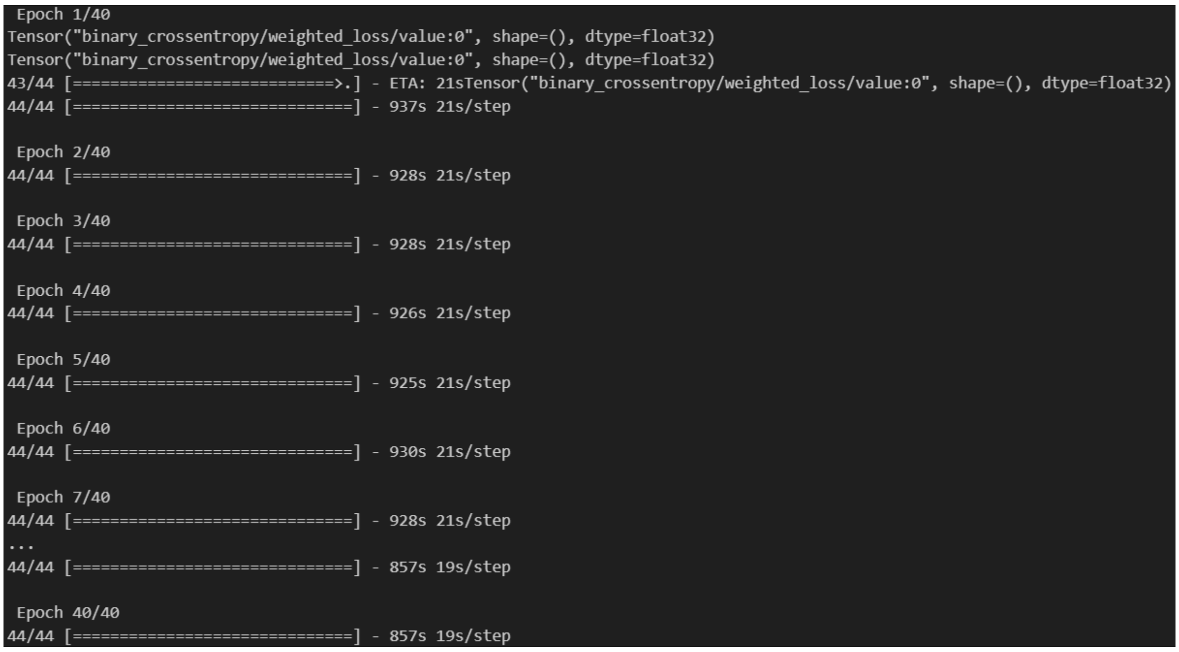
Once we define the L1Dist layer and the Dense layer, we obtain a model that consists of two 100x100x3 image inputs, processed by an embedding sub-model sharing the same weights. The resulting embeddings are then compared by an L1Dist layer that calculates the L1 distance between them. Finally, a dense layer produces the final output of the network, indicating the similarity between the two images.



```
siamese_network.summary()

Model: "SiameseNetwork"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_img (InputLayer)         [(None, 100, 100, 3  0           []
                                )]

 validation_img (InputLayer)    [(None, 100, 100, 3  0           []
                                )]

 embedding (Functional)         (None, 4096)         38960448    ['input_img[0][0]',
                                                                  'validation_img[0][0]']

 l1_dist_1 (L1Dist)             (None, 4096)         0           ['embedding[0][0]',
                                                                  'embedding[1][0]']

 dense_1 (Dense)                (None, 1)            4097        ['l1_dist_1[0][0]']

==================================================================================================
Total params: 38,964,545
Trainable params: 38,964,545
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure5: Interpretation of the Siamese network

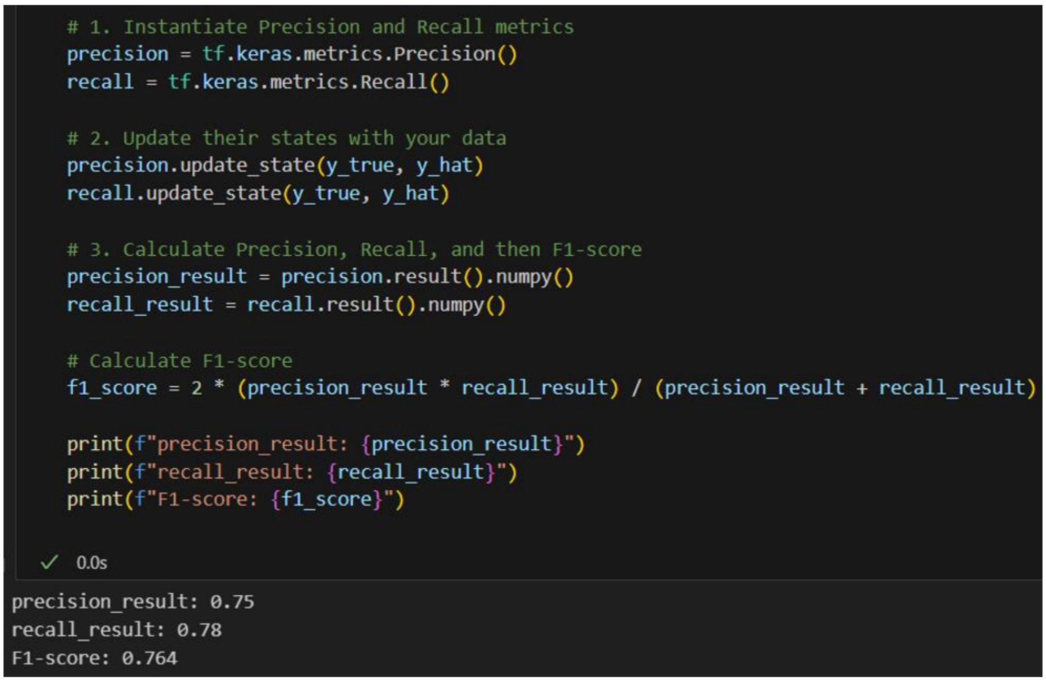
After setting up the necessary prerequisites, we define a training function and specify a number of epochs. An epoch represents a complete cycle through the entire training dataset. [16]
The figure below illustrates the training process with the fixed number of epochs.



```
 Epoch 1/40
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(), dtype=float32)
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(), dtype=float32)
43/44 [============================>.] - ETA: 21sTensor("binary_crossentropy/weighted_loss/value:0", shape=(), dtype=float32)
44/44 [==============================] - 937s 21s/step

 Epoch 2/40
44/44 [==============================] - 928s 21s/step

 Epoch 3/40
44/44 [==============================] - 928s 21s/step

 Epoch 4/40
44/44 [==============================] - 926s 21s/step

 Epoch 5/40
44/44 [==============================] - 925s 21s/step

 Epoch 6/40
44/44 [==============================] - 930s 21s/step

 Epoch 7/40
44/44 [==============================] - 928s 21s/step
...
44/44 [==============================] - 857s 19s/step

 Epoch 40/40
44/44 [==============================] - 857s 19s/step
```

Figure6: Siamese model training process

After training the model, we can now evaluate it. The three main metrics are recall, precision and F1-score.



```
# 1. Instantiate Precision and Recall metrics
precision = tf.keras.metrics.Precision()
recall = tf.keras.metrics.Recall()

# 2. Update their states with your data
precision.update_state(y_true, y_hat)
recall.update_state(y_true, y_hat)

# 3. Calculate Precision, Recall, and then F1-score
precision_result = precision.result().numpy()
recall_result = recall.result().numpy()

# Calculate F1-score
f1_score = 2 * (precision_result * recall_result) / (precision_result + recall_result)

print(f"precision_result: {precision_result}")
print(f"recall_result: {recall_result}")
print(f"F1-score: {f1_score}")
```

✓ 0.0s

```
precision_result: 0.75
recall_result: 0.78
F1-score: 0.764
```

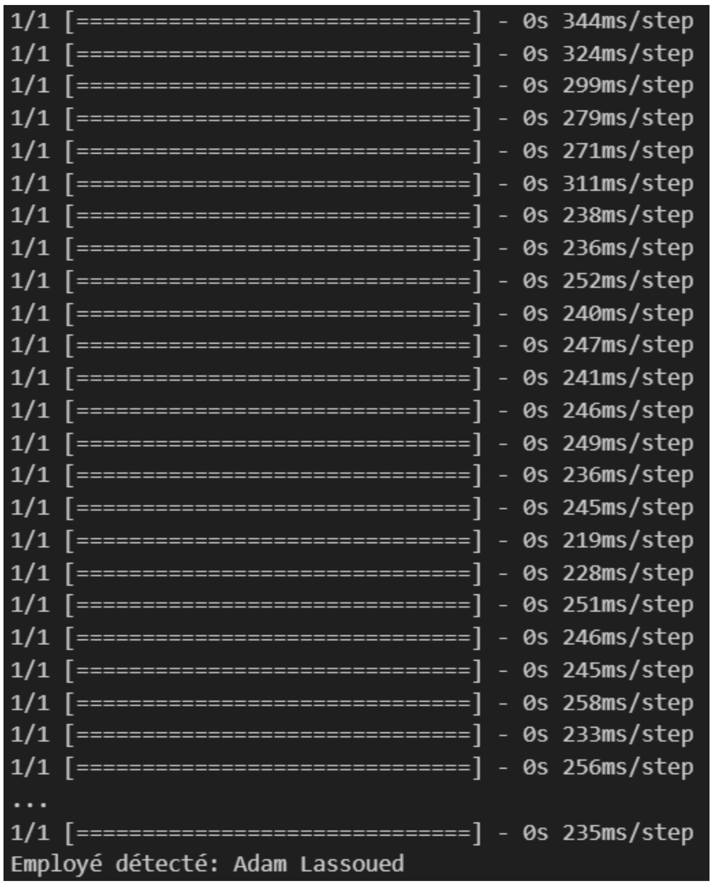
Figure 27: Calculation of evaluation metrics

With precision and recall values of 0.75 and 0.78, the model exhibits reasonable ability to avoid erroneous predictions while identifying a majority of positive instances. This suggests a balanced performance where the model is relatively accurate in its positive predictions while still managing to capture a significant proportion of the true positive cases.

2.2 Facial recognition within the system

The verification function is essential in our system to be able to perform the task of facial recognition. It takes as input the Siamese model, as well as detection and verification thresholds. The verification images used are from the collected data as described in the face detection section.

This feature compares facial embeddings extracted from verification images with those of already registered employees. If the similarity between the embeddings exceeds the specified verification threshold, the employee's identity is confirmed.



```
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 324ms/step
1/1 [==============================] - 0s 299ms/step
1/1 [==============================] - 0s 279ms/step
1/1 [==============================] - 0s 271ms/step
1/1 [==============================] - 0s 311ms/step
1/1 [==============================] - 0s 238ms/step
1/1 [==============================] - 0s 236ms/step
1/1 [==============================] - 0s 252ms/step
1/1 [==============================] - 0s 240ms/step
1/1 [==============================] - 0s 247ms/step
1/1 [==============================] - 0s 241ms/step
1/1 [==============================] - 0s 246ms/step
1/1 [==============================] - 0s 249ms/step
1/1 [==============================] - 0s 236ms/step
1/1 [==============================] - 0s 245ms/step
1/1 [==============================] - 0s 219ms/step
1/1 [==============================] - 0s 228ms/step
1/1 [==============================] - 0s 251ms/step
1/1 [==============================] - 0s 246ms/step
1/1 [==============================] - 0s 245ms/step
1/1 [==============================] - 0s 258ms/step
1/1 [==============================] - 0s 233ms/step
1/1 [==============================] - 0s 256ms/step
...
1/1 [==============================] - 0s 235ms/step
Employé détecté: Adam Lassoued
```

Figure7: Running the check function

3 Control

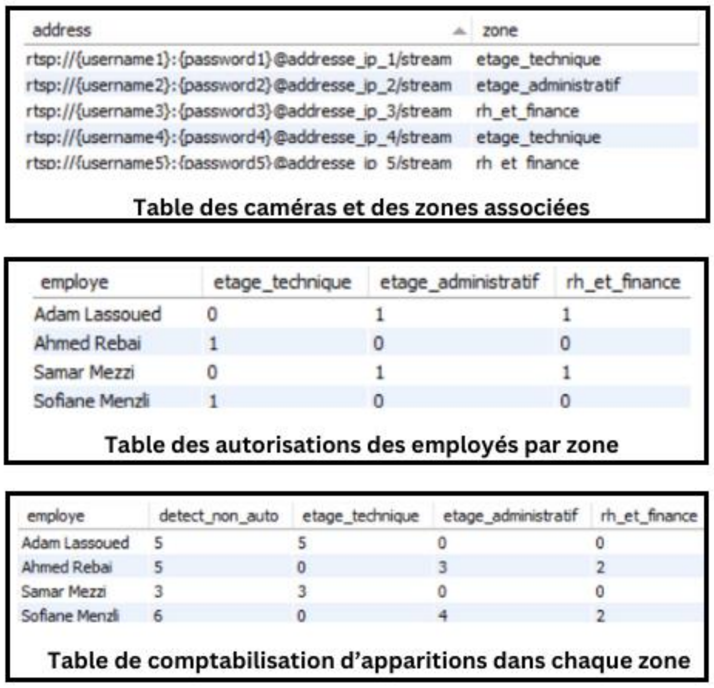
Once an employee is detected and recognized, the control component checks whether that employee is authorized to access the area monitored by the source camera.

In case of unauthorized presence, the control component of the system must archive the data related to the violation in several forms. First, it records an image of the entire frame where the employee appeared for precise visual documentation. Then, it updates the log file that records the details of each violation detected, thus ensuring complete traceability of events. In addition, it updates the database table that counts the number of appearances of each employee in unauthorized areas.

We will start by exploring the architecture of the MySQL database which mainly consists of three essential tables:

- a table that associates each camera with a specific area
- a table that details the allowed and disallowed areas for each employee, with allowed areas represented by "1"s and disallowed areas represented by "0"s

- a table that records the number of times an employee appears in each zone (allowed zones are always 0)



address	zone
rtsp://{username1}:{password1}@addresse_ip_1/stream	etage_technique
rtsp://{username2}:{password2}@addresse_ip_2/stream	etage_administratif
rtsp://{username3}:{password3}@addresse_ip_3/stream	rh_et_finance
rtsp://{username4}:{password4}@addresse_ip_4/stream	etage_technique
rtsp://{username5}:{password5}@addresse_ip_5/stream	rh_et_finance

Table des caméras et des zones associées

employe	etage_technique	etage_administratif	rh_et_finance
Adam Lassoued	0	1	1
Ahmed Rebai	1	0	0
Samar Mezzi	0	1	1
Sofiane Menzli	1	0	0

Table des autorisations des employés par zone

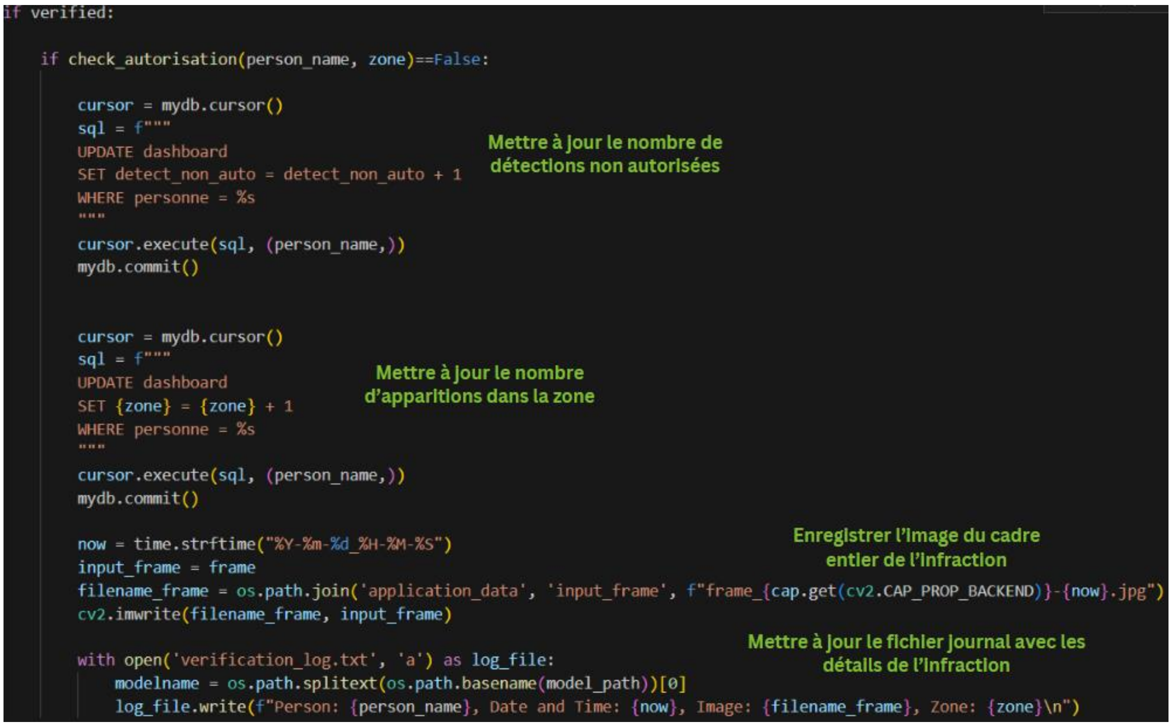
employe	detect_non_auto	etage_technique	etage_administratif	rh_et_finance
Adam Lassoued	5	5	0	0
Ahmed Rebai	5	0	3	2
Samar Mezzi	3	3	0	0
Sofiane Menzli	6	0	4	2

Table de comptabilisation d'apparitions dans chaque zone

Figure8: Overview of database tables

The verification task is performed by two closely related functions: the first checks the area assigned to the source camera upon system startup, while the second function checks whether the employee is authorized to the same area once detected and recognized.

Once a person is identified and it is verified that he is not authorized to appear in the area of the camera that detected him, the system must record the details of the violation event in its various data forms.



```
if verified:

    if check_autorisation(person_name, zone)==False:

        cursor = mydb.cursor()
        sql = f"""
        UPDATE dashboard
        SET detect_non_auto = detect_non_auto + 1
        WHERE personne = %s
        """
        cursor.execute(sql, (person_name,))
        mydb.commit()


        cursor = mydb.cursor()
        sql = f"""
        UPDATE dashboard
        SET {zone} = {zone} + 1
        WHERE personne = %s
        """
        cursor.execute(sql, (person_name,))
        mydb.commit()

        now = time.strftime("%Y-%m-%d_%H-%M-%S")
        input_frame = frame
        filename_frame = os.path.join('application_data', 'input_frame', f"frame_{cap.get(cv2.CAP_PROP_BACKEND)}-{now}.jpg")
        cv2.imwrite(filename_frame, input_frame)

        with open('verification_log.txt', 'a') as log_file:
            modelname = os.path.splitext(os.path.basename(model_path))[0]
            log_file.write(f"Person: {person_name}, Date and Time: {now}, Image: {filename_frame}, Zone: {zone}\n")
```

Mettre à jour le nombre de détections non autorisées
Mettre à jour le nombre d'apparitions dans la zone
Enregistrer l'image du cadre entier de l'infraction
Mettre à jour le fichier journal avec les détails de l'infraction

Figure9: Data archiving operations

4 The data visualization interface

In this last section, we will discuss the interface for viewing archived data. Given the critical nature of this data, access to it is strictly reserved for a security officer, with a unique identifier and a password. Thus, the first action to take to access the data is to authenticate.

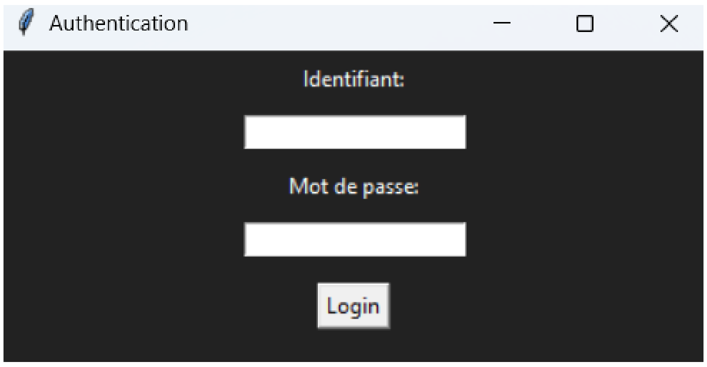
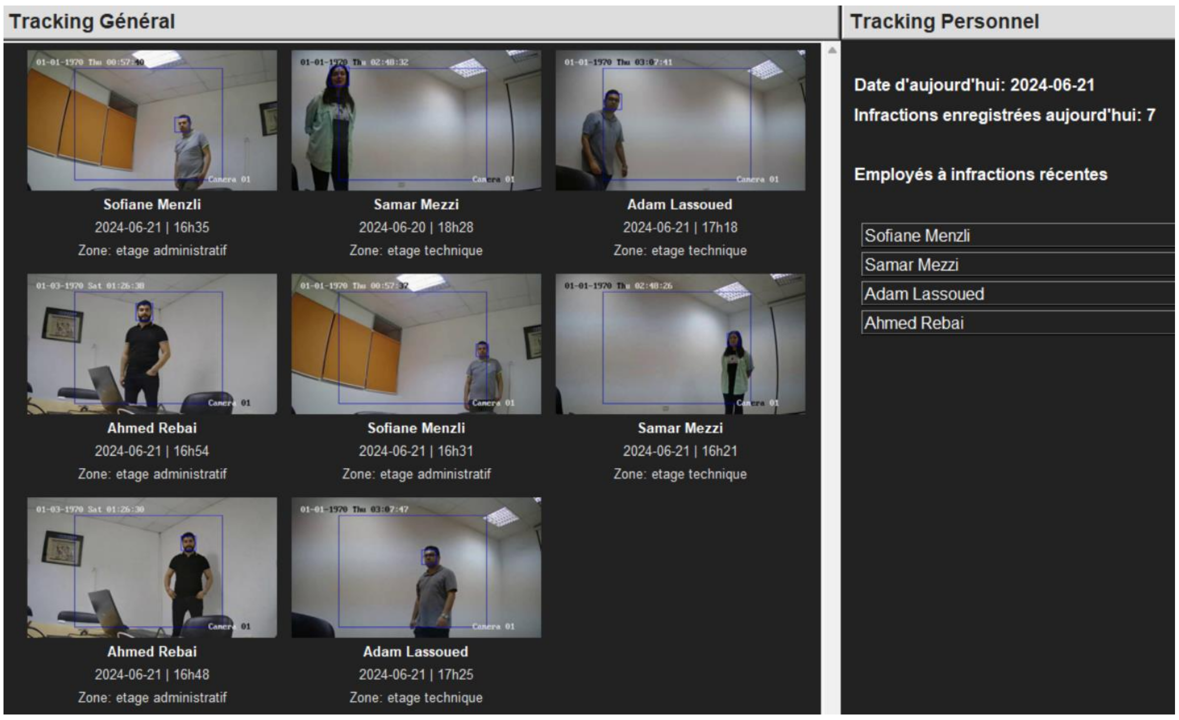


Figure10: Authentication window

Once authenticated, the user accesses the main interface of the system. This interface is structured in two distinct frames:

- The left frame: It is dedicated to "general tracking": it presents a series of identified, time-stamped and localized images of recent offenses, organized in descending chronological order.

- The right frame: It displays the current date, the number of violations recorded for the day as well as the list of employees who committed violations, also listed in descending chronological order.


Tracking Général

Sofiane Menzli
2024-06-21 | 16h35
Zone: etage administratif

Samar Mezzi
2024-06-20 | 18h28
Zone: etage technique

Adam Lassoued
2024-06-21 | 17h18
Zone: etage technique

Ahmed Rebai
2024-06-21 | 16h54
Zone: etage administratif

Sofiane Menzli
2024-06-21 | 16h31
Zone: etage administratif

Samar Mezzi
2024-06-21 | 16h21
Zone: etage technique

Ahmed Rebai
2024-06-21 | 16h48
Zone: etage administratif

Adam Lassoued
2024-06-21 | 17h25
Zone: etage technique

Tracking Personnel

Date d'aujourd'hui: 2024-06-21
Infractions enregistrées aujourd'hui: 7

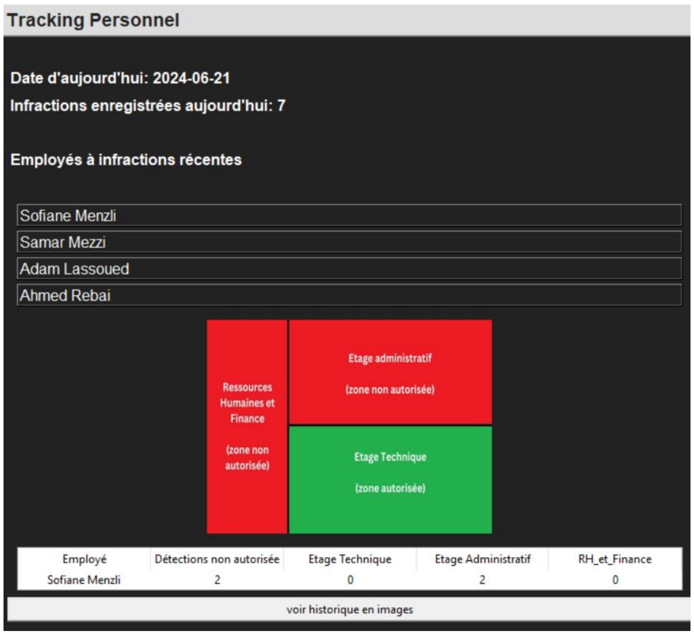
Employés à infractions récentes

Sofiane Menzli
Samar Mezzi
Adam Lassoued
Ahmed Rebai

Figure11: Overview of the data visualization interface

The blank field below the employee list is dedicated to displaying specific details about a selected employee. Indeed, when the user selects an employee, three elements appear:

- a mini map showing the authorized and prohibited areas for this employee
- a table showing the number of times the employee appeared in unauthorized areas
- a button to view personal history of offenses in images.


Tracking Personnel

Date d'aujourd'hui: 2024-06-21
Infractions enregistrées aujourd'hui: 7

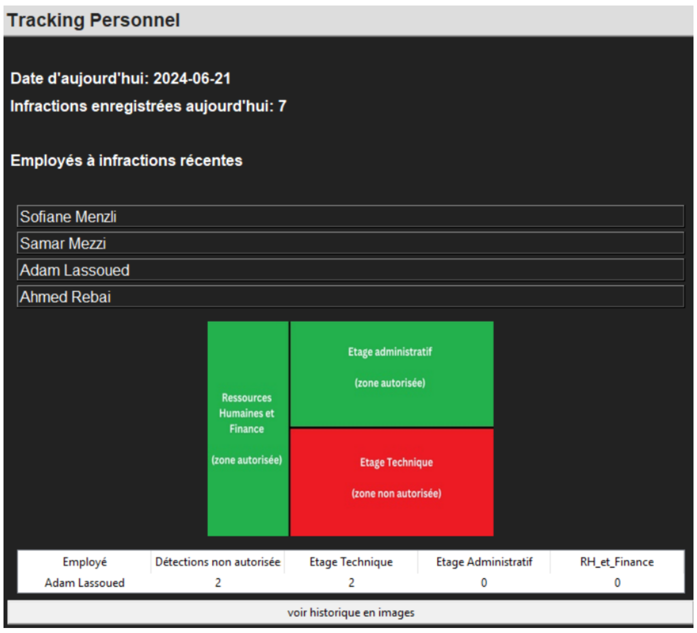
Employés à infractions récentes

Sofiane Menzli
Samar Mezzi
Adam Lassoued
Ahmed Rebai

Employé	Détections non autorisée	Etage Technique	Etage Administratif	RH_et_Finance
Sofiane Menzli	2	0	2	0

voir historique en images

Figure12: Example 1 of a selected employee


Tracking Personnel

Date d'aujourd'hui: 2024-06-21
Infractions enregistrées aujourd'hui: 7

Employés à infractions récentes

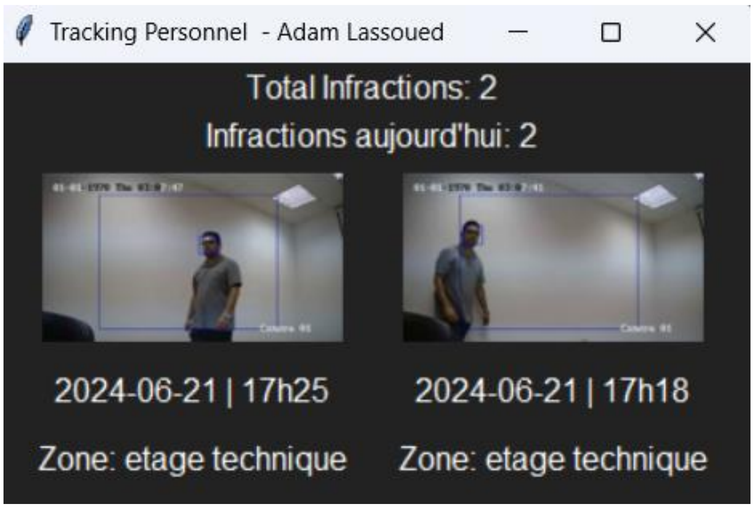
Sofiane Menzli
Samar Mezzi
Adam Lassoued
Ahmed Rebai

Employé	Détections non autorisée	Etage Technique	Etage Administratif	RH_et_Finance
Adam Lassoued	2	2	0	0

voir historique en images

Figure13: Example 2 of a selected employee

To view the complete personal history of the selected person, including images and a summary of the total and daily number of infractions, simply click on the "View history in images" button. This will open a pop-up containing all this information.


Tracking Personnel - Adam Lassoued

Total Infractions: 2

Infractions aujourd'hui: 2

2024-06-21 | 17h25

Zone: etage technique

2024-06-21 | 17h18

Zone: etage technique

Figure14: Personal History Pop-up Window

Conclusion

In conclusion, this project resulted in the realization of a complete system and an intuitive graphical interface, thanks to rigorous planning and iterative implementation by sprints. The use of a well-chosen software and hardware environment was decisive in ensuring the stability and efficiency of the development. Finally, user-centered design principles made it possible to create an optimal user experience that met initial expectations.