جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

ENSAM
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

# Projet Industrie 4.0

## Système Intelligent de Présence avec Reconnaissance Faciale

---

**Encadré par :** Pr.Mohamed Hosni

**Préparé par :**

- Oussama Fahim
- Salma Bourkiba
- Hasna Jhabli
- Meryem Filali Ansari
- Ouiame Bellaknich
- Mariam Lakhsassi
- Fatima El fadili
- Salma Oulkiass

Juin 2025

# CONTENTS

# LIST OF FIGURES

CHAPTER 1

# INTRODUCTION GÉNÉRALE SUR LE PROJET

## 1.1 Problématique

Dans un contexte académique où l'efficacité, la transparence et la fiabilité de la gestion des présences sont devenues des enjeux majeurs, les méthodes traditionnelles telles que les feuilles de présence papier ou les appels manuels se révèlent chronophages, sujettes à l'erreur, et peu adaptées à l'ère numérique actuelle. Ces procédés manuels peuvent engendrer des falsifications (présence fictive, signature pour autrui), un manque de traçabilité, et une perte de temps précieux au début de chaque séance.

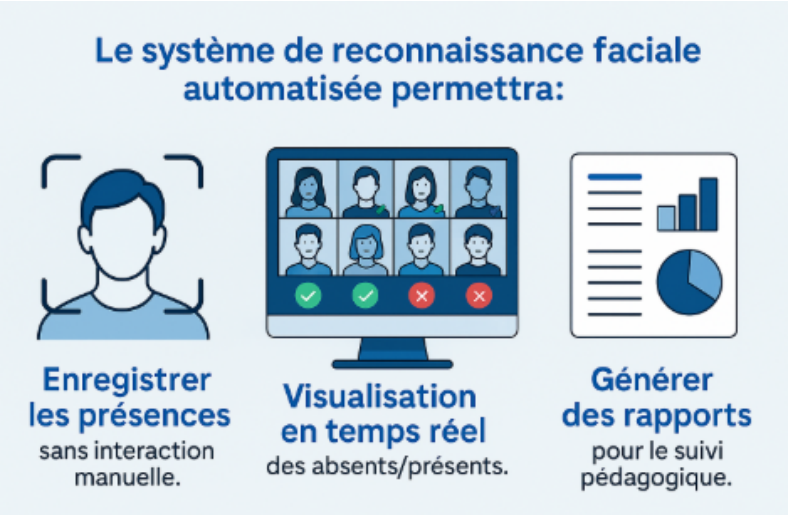
Parallèlement, la révolution de l'Industrie 4.0, avec ses technologies intégrées telles que l'Internet des Objets (IoT), l'intelligence artificielle (IA), et l'automatisation intelligente, ouvre de nouvelles perspectives pour moderniser les processus pédagogiques et administratifs. Cela soulève une question centrale :

**Comment concevoir et déployer un système intelligent, fiable et sécurisé de reconnaissance faciale capable d'automatiser l'enregistrement des présences dans un environnement éducatif, tout en garantissant rapidité, scalabilité, respect de la vie privée, et assistance en cas d'échec de détection ?**

## 1.2 Introduction à la solution

L'entrée dans l'ère de l'Industrie 4.0 marque une transformation profonde des processus industriels et organisationnels, en intégrant les technologies numériques les plus avancées. Elle repose sur la convergence entre l'automatisation, l'intelligence artificielle, l'Internet des Objets (IoT), la cyber-physicalité et la connectivité en temps réel, créant ainsi des systèmes intelligents capables d'interagir avec leur environnement, d'apprendre et de s'optimiser de manière autonome.

Dans ce contexte, notre projet s'inscrit pleinement dans cette dynamique d'innovation technologique. Il propose la conception et la mise en œuvre d'un Système Intelligent de Suivi de Présence basé sur la reconnaissance faciale, destiné à automatiser la gestion des présences dans un environnement éducatif, tout en assurant fiabilité, gain de temps, et traçabilité. Cette solution vise à remplacer les méthodes manuelles traditionnelles (feuilles de présence papier, appels nominaux), souvent sources d'erreurs, de fraudes ou de pertes de données, par une approche moderne, rapide et sécurisée.





Le cœur du système repose sur l'intégration de caméras intelligentes connectées à un Raspberry Pi embarquant des algorithmes de détection et reconnaissance faciale en temps réel. Grâce à l'intelligence artificielle et au machine learning, il devient possible d'identifier automatiquement les élèves à leur arrivée, d'enregistrer leur présence sans intervention humaine, et de générer des feuilles de présence dynamiques et exploitables à tout moment. En cas d'échec de reconnaissance, le système capture automatiquement l'image non identifiée, permettant une validation manuelle a posteriori par l'enseignant via une interface graphique (dashboard).

Le projet se distingue également par sa dimension IoT (Internet des Objets), dans la mesure où les composants matériels et logiciels sont connectés, capables de communiquer en temps réel et de s'adapter dynamiquement à l'environnement. Cette architecture distribuée permet un traitement local rapide (edge computing) tout en offrant une interface centralisée pour la gestion pédagogique.

Ce projet présente donc un triple intérêt :

- **Technologique**, en mobilisant des briques avancées d'IA, d'électronique embarquée et de visualisation de données.
- **Pédagogique**, en offrant une solution concrète aux problématiques réelles de suivi des élèves.

### 1.2.1 Objectifs de projet

**Promouvoir une pédagogie innovante et responsabilisante**

L'objectif principal du projet est d'accompagner la transformation digitale de l'environnement pédagogique en intégrant des technologies de l'Industrie 4.0 dans la vie académique quotidienne. L'automatisation de la gestion de présence via reconnaissance faciale permet non seulement de gagner en efficacité et en fiabilité, mais aussi d'instaurer une culture de ponctualité, de transparence et d'engagement. L'enseignant devient un acteur du pilotage intelligent des présences et l'étudiant est responsabilisé dans sa démarche d'assiduité.

- **Intégrer concrètement les technologies de l'Industrie 4.0**
  Ce projet vise à mettre en pratique des concepts clés de l'Industrie 4.0 tels que l'intelligence artificielle embarquée, l'Internet des Objets (IoT), le traitement temps réel, et la vision par ordinateur. En implémentant un système fonctionnel avec Raspberry Pi, caméra et FaceNet, le projet permet aux étudiants d'aborder concrètement les notions de systèmes intelligents distribués, pipeline de données et interaction homme-machine, dans une optique d'innovation et de prototypage rapide.
- **Permettre une exploitation intelligente et sécurisée des données**
  Au-delà de la simple détection de présences, l'un des objectifs clés est de transformer les données de présence en informations exploitables pour l'administration pédagogique, grâce à des rapports générés automatiquement, et une analyse de l'assiduité dans le temps.

### 1.2.2 Périmètre du Projet

Ce projet s'inscrit dans une démarche d'automatisation intelligente de la gestion de présence des étudiants, dans un contexte pédagogique où l'efficacité, la traçabilité et la réduction de l'intervention humaine sont essentielles. Le périmètre fonctionnel du projet englobe l'ensemble des composants nécessaires à la détection, l'enregistrement, la validation et la consultation des présences grâce à la reconnaissance faciale. Voici les principales fonctionnalités couvertes par le système :

1. **Reconnaissance faciale automatisée**
   Dès l'entrée des étudiants dans la salle, une caméra connectée à un système embarqué (Raspberry Pi) capture les visages en temps réel. Un algorithme de reconnaissance faciale (FaceNet pour l'identification) permet de reconnaître automatiquement l'identité de chaque étudiant sans intervention humaine. Cette reconnaissance repose sur un modèle pré-entraîné, associé à une base de données d'images de visages encodées.

2. **Enregistrement automatique des présences**
   Lorsqu'un visage est reconnu, le système procède à l'enregistrement automatique de la date et de l'heure dans une base de données sécurisée. Cela élimine la nécessité d'une saisie manuelle ou de l'utilisation de feuilles de présence papier, tout en garantissant l'intégrité et la fiabilité des données.

3. **Génération automatique de la liste de présence**
   À la fin de chaque session, une liste récapitulative des présences est générée automatiquement. Ce document peut être exporté aux formats CSV, et transmis à l'enseignant. Cette fonctionnalité assure une traçabilité fluide, rapide et conforme aux exigences académiques et administratives.

4. **Gestion des cas d'anomalies (échecs de reconnaissance)**
   En cas d'échec de reconnaissance (flou, luminosité faible, absence d'encodage préalable. . . ), le système capture et enregistre temporairement l'image non reconnue dans un répertoire dédié. L'enseignant, via le tableau de bord, peut alors:

   - Valider manuellement la présence s'il reconnaît l'étudiant,
   - Refuser la validation si la personne est non identifiable ou absente.

## 1.3 Exigences Fonctionnelles et Non Fonctionnelles

### 1.3.1 Exigences Fonctionnelles

Le système doit répondre aux besoins suivants afin d'assurer une gestion automatisée et fiable des présences :

- **Enregistrement initial des étudiants** : le visage de chaque étudiant doit être capturé via une caméra et associé à un identifiant unique comprenant le nom, le prénom et un identifiant numérique.
- **Reconnaissance faciale en temps réel** : le système doit détecter et reconnaître automatiquement le visage des étudiants dès leur entrée dans la salle. La capture et le traitement de l'image doivent être réalisés en moins de 10 secondes.
- **Enregistrement des présences** : à chaque reconnaissance réussie, le système doit automatiquement consigner les informations suivantes : identifiant de l'étudiant, nom, date, heure de passage.
- **Rapports de présence** : le système doit générer des rapports exploitables et exportables au format CSV afin de permettre un suivi pédagogique et administratif.

### 1.3.2 Exigences Non Fonctionnelles

Le système doit également respecter des critères de qualité pour garantir sa robustesse et sa fiabilité :

- **Temps de réponse** : le délai de traitement par étudiant (de la détection à l'enregistrement) doit être inférieur ou égal à 10 secondes.
- **Scalabilité** : le système doit maintenir une réactivité constante même en cas de forte affluence.
- **Sécurité des données** : toutes les informations personnelles, en particulier les images faciales, doivent être stockées sous forme cryptée afin de garantir la confidentialité des données.
- **Fiabilité et disponibilité** : le système doit fonctionner de manière fiable au moins 99% du temps, avec des sauvegardes régulières des données. La capture d'image doit être réalisée en moins de 5 secondes pour garantir une expérience fluide.

## 1.4 Composants et Technologies Utilisés

| **Catégorie** | **Composants / Technologies** |
|---|---|
| **Système d'exploitation** | Raspberry Pi OS |
| **Langage de programmation** | Python 3 (langage principal du projet) |
| **Bibliothèques de traitement d'image** | • OpenCV (traitement d'image)<br>• MTCNN (détection de visages)<br>• FaceNet (génération d'embeddings)<br>• SVM (classification des visages) |
| **Gestion de données** | Pandas (gestion et export des fichiers CSV) |
| **Matériel** | • Raspberry Pi 4<br>• Caméra<br>• Écran LCD<br>• Écran HDMI<br>• Alimentation adaptée (5V, 3A pour Raspberry Pi) |

**Table 1.1:** Composants matériels et technologies logicielles utilisées dans le projet

CHAPTER 2

# ETUDE ET CONCEPTION DU PROJET

## 2.1 Architecture du Projet

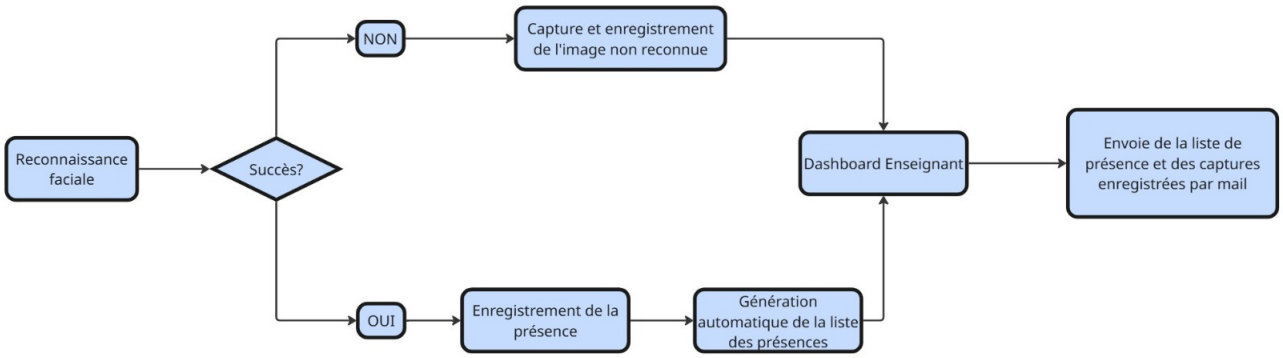



**Figure 2.1:** L'architecture du projet

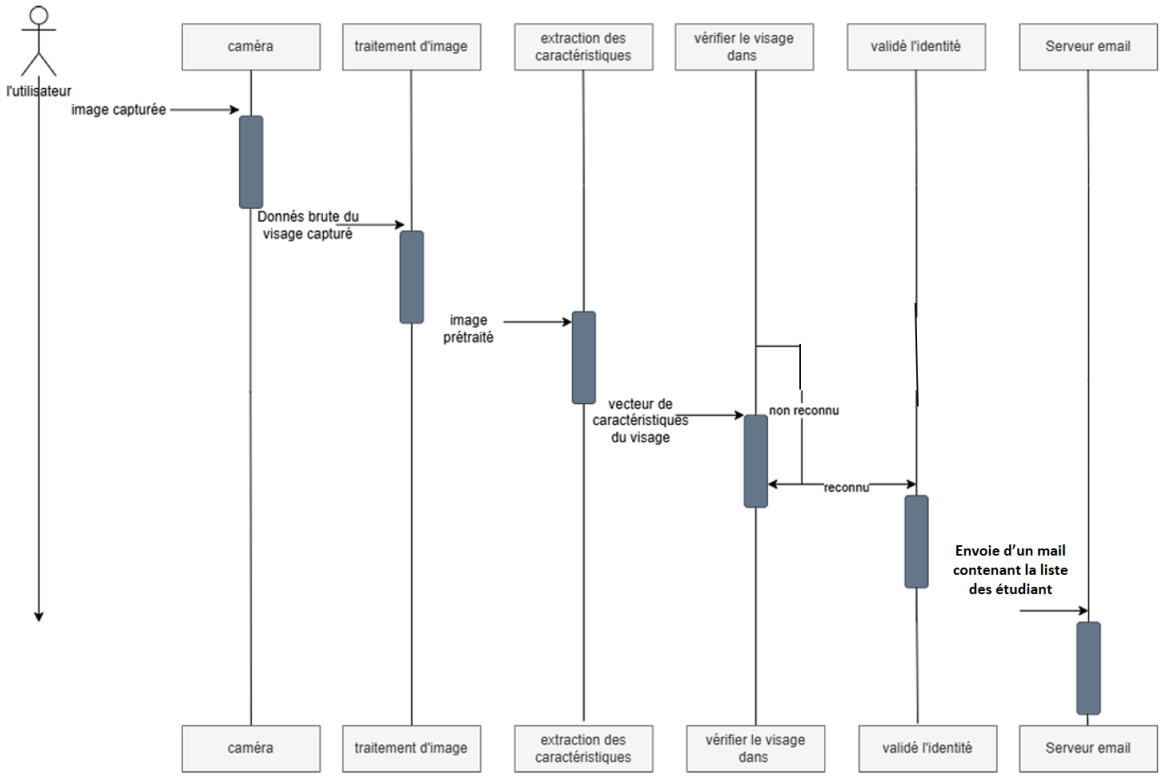



**Figure 2.2:** Diagramme de séquence

Cette architecture décrit un système automatisé de gestion des présences par reconnaissance faciale, avec une intervention manuelle possible en cas d'échec. Le processus se décompose comme suit :

- **Étape 1 : Reconnaissance Faciale**
  Le système tente d'identifier un visage capturé. Deux issues sont possibles :
  - **Succès** : La présence est enregistrée automatiquement.
  - **Échec** : L'image non reconnue est sauvegardée pour traitement ultérieur.
- **Étape 2 : Gestion des Anomalies**
  En cas d'échec, l'image est transmise à un **dashboard enseignant**, permettant :
  - Une validation manuelle de la présence (par exemple, en associant manuellement l'image à un étudiant).
- **Étape 3 : Génération de la Liste**
  Les données validées (automatiquement ou manuellement) alimentent une liste de présence:
  - La liste sera envoyée à l'enseignant après l'entrée des étudiants( après une 30 min de l'horaire de l'entrée)

**Fonctionnalités clés :**

Le dashboard offre une interface centralisée pour :

- Corriger les erreurs du système automatique.
- Surveiller les statistiques de présence en direct.

CHAPTER 3

# RÉALISATION DU PROJET

Ce chapitre décrit la mise en œuvre technique du projet, depuis l'installation de l'environnement de développement jusqu'à l'intégration finale des différentes composantes. Nous abordons la configuration du Raspberry Pi 4, l'installation des dépendances nécessaires, ainsi que le développement des modules de reconnaissance faciale, de gestion de la base de données, et de l'interface utilisateur. Chaque module a été testé et optimisé afin d'assurer un fonctionnement fluide et efficace du système. Enfin, nous présentons les tests réalisés pour valider la précision de la reconnaissance faciale et l'efficacité des emails et dashboard envoyés

## 3.1 Composants et Matériels

Pour assurer le bon fonctionnement de notre système de reconnaissance faciale pour le contrôle d'accès, nous avons intégré plusieurs composants matériels essentiels :

1. **Raspberry Pi 4**

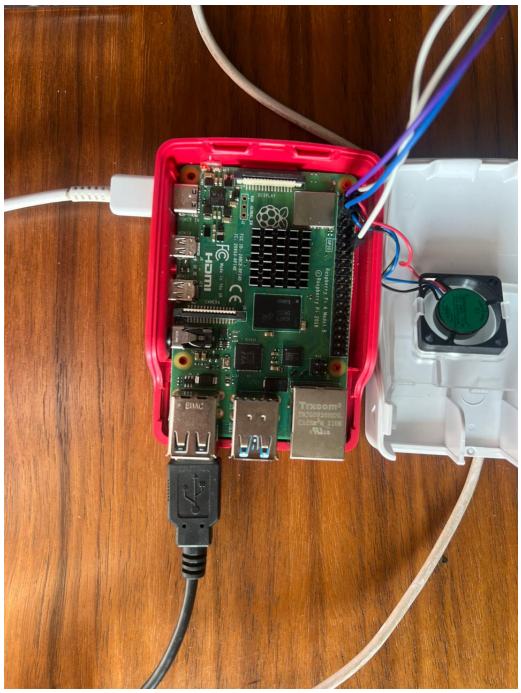


**Figure 3.1:** Raspberry Pi 4

La Raspberry Pi 4 est le cœur du système. Il s'agit d'un micro-ordinateur compact et puissant, capable d'exécuter des algorithmes d'intelligence artificielle, de traitement d'image et de reconnaissance faciale. Dans ce projet, la Raspberry Pi 4 est responsable de la gestion de la webcam, de l'exécution du code de détection et de reconnaissance des visages, ainsi que de la communication avec la base de données pour enregistrer les présences et absences.

2. **Webcam**

**Figure 3.2:** Webcam

La webcam joue le rôle de capteur d'image. Elle est utilisée pour capturer en temps réel les visages des étudiants à l'entrée de la salle. Ces images sont ensuite transmises à la Raspberry Pi pour le traitement.

3. **Écran LCD**


**Figure 3.3:** Ecran LCD

L'écran LCD est utilisé pour l'affichage en temps réel des informations relatives à la reconnaissance. Par exemple, il peut afficher le nom de l'étudiant reconnu, la date et l'heure de l'enregistrement, ou un message de confirmation de présence. Il permet également d'afficher des messages d'erreur ou d'indication (ex. 'Visage non reconnu', 'Mail envoyé au Professeur'). Cet écran offre ainsi une interface utilisateur simple et directe.

## 3.2 Les Etapes Concrètes de Réalisation

### 3.2.1 Installation du système d'exploitation

- Télécharger « Raspberry Pi OS » depuis le site officiel.
- Utiliser « Raspberry Pi Imager » ou « balenaEtcher » pour flasher l'OS sur la carte SD.
- Insérer la carte SD dans le Raspberry et le démarrer avec clavier, souris et écran.

### 3.2.2 Configuration de base

- Mise à jour du système
- Activer la caméra via raspi-config

- Installer les bibliothèques nécessaires :
  bash
  sudo apt install python3-pip
  pip install opencv-python facenet-pytorch pandas torch torchvision joblib RPLCD scikit-learn

### 3.2.3 Connexion à distance avec VNC

- Activer VNC dans raspi-config > Interface Options > VNC.
- Installer *RealVNC Viewer* sur le PC.
- Se connecter à l'adresse IP du Raspberry Pi.

## 3.3 Construction de modèle

### 3.3.1 Pipline de construction du modèle

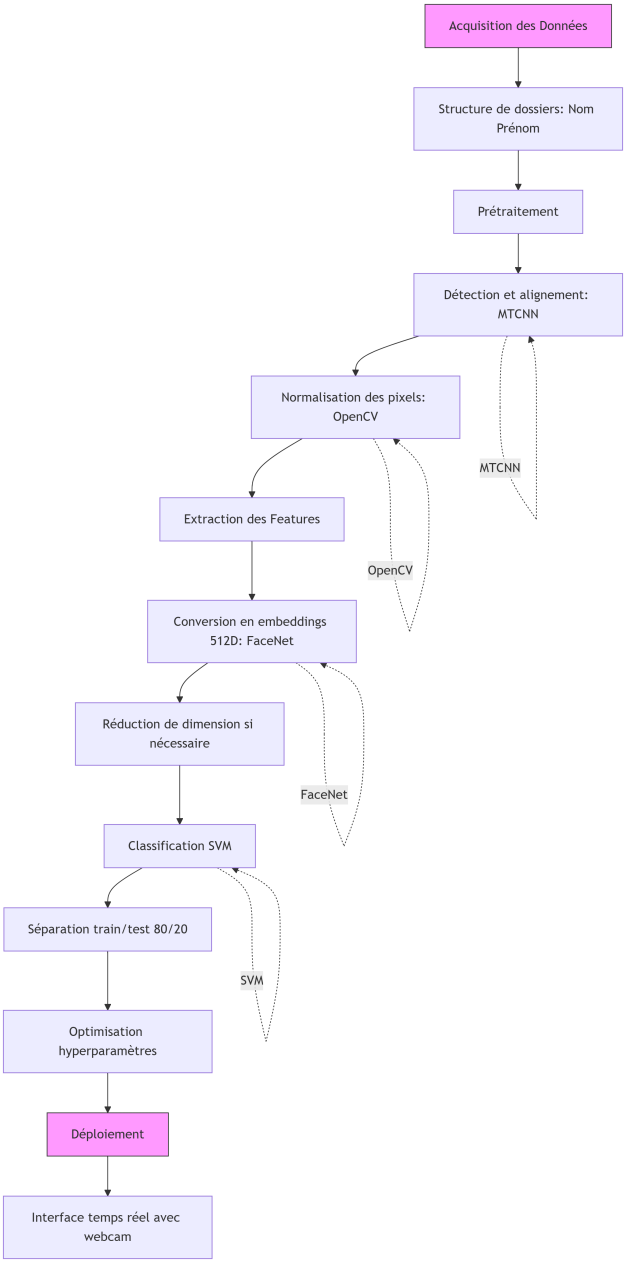



**Figure 3.4:** Pipeline de construction de modèle

**Explication :** Ce pipeline décrit un système complet de reconnaissance faciale, structuré en 5 phases claires :

**1. Acquisition des Données :** Les images des visages sont collectées et organisées dans une structure standardisée de dossiers selon nom `Nom_Prénom` pour chaque étudiant, garantissant une base de données cohérente pour les étapes suivantes.

**2. Prétraitement**

- **Détection et alignement** : Le modèle MTCNN localise les visages et les aligne pour une analyse uniforme.
- **Normalisation** : OpenCV ajuste les pixels (luminosité, contraste) pour réduire les variations parasites.

**3. Extraction des Features**

- **Embeddings 512D** : FaceNet convertit chaque visage en un vecteur de caractéristiques numériques.
- **Réduction de dimension** : Optionnelle (ex: PCA), pour optimiser l'efficacité du modèle.

**4. Classification (SVM)**

- **Séparation 80/20** : Les données sont divisées en ensembles d'entraînement (80%) et de test (20%).
- **Optimisation** : Réglage des hyperparamètres du SVM (ex: noyau, pénalité) pour maximiser la précision.

**5. Déploiement :** Une interface temps réel (webcam) permet d'utiliser le modèle entraîné pour reconnaître les visages dans des conditions réelles.

### 3.3.2 Acquisition et labeling des Données

**Structuration de la Base de Données Visuelles**

Les images des visages des étudiants sont collectées et organisées de manière structurée afin de garantir la cohérence et la fiabilité de la base de données utilisée pour la reconnaissance faciale. Chaque étudiant possède un répertoire individuel nommé selon la convention `Nom_Prénom`, facilitant ainsi le traitement automatisé et l'identification précise lors des étapes d'entraînement et d'inférence du modèle. Cette organisation standardisée permet non seulement d'assurer la traçabilité des données collectées, mais également de simplifier la gestion des mises à jour (ajout ou suppression d'étudiants) et l'intégration avec les algorithmes de génération d'embeddings. Grâce à cette structuration rigoureuse, le système peut exploiter efficacement l'ensemble des images disponibles pour garantir un haut niveau de précision dans la reconnaissance faciale.

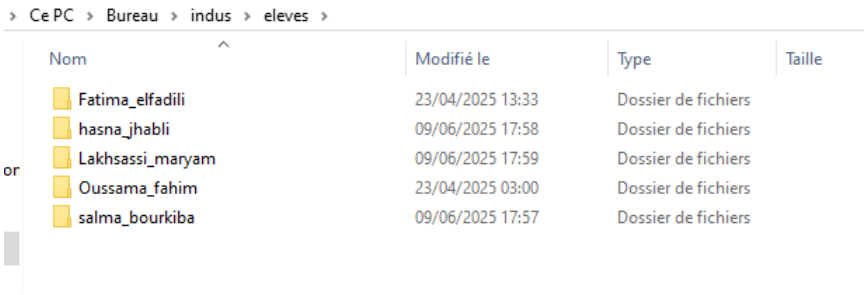



**Figure 3.5:** Dataset strurturée

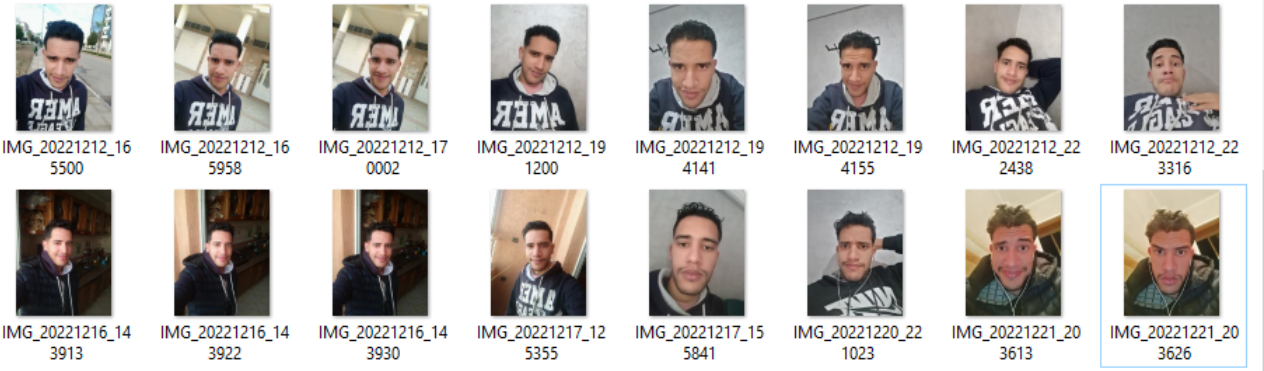



**Figure 3.6:** exemple des images de dataset de l'étudiant Oussama fahim

## Labeling des Données

**Listing 3.1:** Labeling des données

```
DATASET_PATH = "eleves/"

# Cr er un dataframe pour stocker les informations des images
def create_dataset(dataset_path):
    images = []
    labels = []

    for person_name in os.listdir(dataset_path):
        person_path = os.path.join(dataset_path, person_name)

        if os.path.isdir(person_path):
            for image_name in os.listdir(person_path):
                image_path = os.path.join(person_path, image_name)

                # V rifier que c'est bien un fichier image
                if image_path.lower().endswith(('.png', '.jpg', '.jpeg')):
                    images.append(image_path)
                    labels.append(person_name)

```

```
    return pd.DataFrame({'image_path': images, 'label': labels})

df = create_dataset(DATASET_PATH)
print(f"Dataset cr avec {len(df)} images")
print(df.head())
```

**Fonctionnalité du Code et Rôle du Labeling**

- **Fonction principale:** Ce script Python constitue une **pipeline de préprocessing** pour un système de reconnaissance faciale. Sa fonction première est d'organiser un dataset d'images selon une structure standardisée, en générant un DataFrame contenant :
  - Les chemins d'accès absolus aux images
  - Les labels correspondants (noms des personnes)
- **Mécanisme de labeling** :
  Le système de labeling joue un rôle **critique** pour la phase d'entraînement du modèle, avec deux caractéristiques essentielles :
  - **Structure hiérarchique** : Utilisation des noms de dossiers (`Nom_Prénom`) comme identifiants uniques
  - **Validation automatique** : Filtrage strict des extensions (`.png/.jpg/.jpeg`) pour garantir l'intégrité des données
- **Sortie structurée** :
  Le DataFrame produit suit le format suivant :

| image_path | label |
|---|---|
| eleves/Oussama_fahim/photo1.jpg | Oussama_fahim |
| eleves/Fatima_elfadili/photo2.png | Fatima_elfadili |

- **Intégration dans le pipeline** :
  Ce module s'insère dans la chaîne de traitement comme **premier maillon** :
  1. Fournit les métadonnées nécessaires aux étapes suivantes (détection MTCNN, embedding FaceNet)
  2. Permet un suivi traçable des échantillons grâce à l'association image/label
- **Contrôle qualité** :
  Les messages de sortie (`print`) offrent une **vérification immédiate** :
  - Nombre total d'images chargées

– Aperçu des 5 premières entrées (vérification visuelle du format)

**Impact sur le système** : Cette étape conditionne directement la performance du modèle final en garantissant :

- Une **correspondance exacte** entre images et identités
- L'**élimination proactive** des fichiers non-images
- Une **base cohérente**

### 3.3.3 Prétraitement des Images pour la Reconnaissance Faciale

Le prétraitement des images est une étape cruciale pour garantir la qualité et la fiabilité des données d'entrée utilisées dans le processus de reconnaissance faciale. Le code suivant montre comment les images de visages sont automatiquement détectées, recadrées, normalisées et préparées avant d'être utilisées pour l'entraînement du modèle ou la classification.

**Listing 3.2:** Prétraitement des images de visages avec MTCNN

```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
mtcnn = MTCNN(keep_all=True, device=device)

# Fonction de pr traitement avec d tection des visages
def preprocess_image(image_path, target_size=(160, 160)):
    try:
        # Charger l'image
        img = Image.open(image_path).convert('RGB')

        # D tection et alignement du visage
        boxes, probs = mtcnn.detect(img)

        if boxes is not None:
            # Prendre la bo te avec la plus haute probabilit
            box = boxes[np.argmax(probs)]

            # Recadrer le visage
            face = img.crop(box)
            face = face.resize(target_size)

            # Convertir en numpy array et normaliser
            face_array = np.array(face).astype('float32')
            face_array = (face_array - 127.5) / 128.0

            return face_array
```

```
        else:
            return None
    except Exception as e:
        print(f"Erreur lors du traitement de {image_path}: {e}")
        return None

# Appliquer le prtraitement    toutes les images
processed_images = []
valid_labels = []

for idx, row in df.iterrows():
    processed_img = preprocess_image(row['image_path'])
    if processed_img is not None:
        processed_images.append(processed_img)
        valid_labels.append(row['label'])

# Convertir en numpy arrays
X = np.array(processed_images)
y = np.array(valid_labels)

# Encoder les labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Sauvegarder le label encoder pour plus tard
np.save('label_encoder.npy', label_encoder.classes_)
```

**Explication du Code :** Ce script réalise le prétraitement des images en plusieurs étapes :

- **Chargement du modèle MTCNN** : le modèle `MTCNN` est initialisé pour la détection des visages. Il est capable de repérer plusieurs visages sur une image, mais ici on garde le visage avec la plus haute probabilité.
- **Chargement et détection du visage** : chaque image est lue et convertie en RGB. MTCNN détecte les boîtes englobantes (`boxes`) et leurs probabilités associées (`probs`).
- **Recadrage et redimensionnement** : le visage le plus probable est extrait, recadré selon la boîte détectée, puis redimensionné à la taille standard de `160x160` pixels.
- **Normalisation** : les pixels de l'image du visage sont normalisés pour centrer les valeurs autour de zéro, ce qui améliore la stabilité du modèle en aval (FaceNet).
- **Traitement de toutes les images** : la fonction est ensuite appliquée à toutes les images de la base. Seules les images valides sont retenues.

- **Encodage des labels** : les étiquettes textuelles (noms des étudiants) sont encodées en entiers via `LabelEncoder`, facilitant l'apprentissage.
- **Sauvegarde des labels** : les classes du `LabelEncoder` sont sauvegardées pour permettre une correspondance ultérieure lors de la phase de prédiction.

**Rôle du Prétraitement des Images:** Le prétraitement des images joue un rôle fondamental dans la qualité globale du système de reconnaissance faciale. Il permet de :

- **Standardiser les images** : en garantissant une taille et un format identiques pour toutes les images, le modèle de reconnaissance faciale (FaceNet) peut apprendre des représentations cohérentes.
- **Aligner les visages** : la détection précise des visages permet de centrer et d'aligner les traits importants (yeux, nez, bouche), ce qui est essentiel pour obtenir des embeddings robustes.
- **Réduire le bruit** : en ne gardant que les visages valides et en supprimant les images mal détectées, on améliore la qualité des données et la fiabilité de la classification.
- **Faciliter l'apprentissage du modèle** : les images normalisées et standardisées permettent au classificateur (SVM) d'apprendre des frontières de décision plus efficaces.

Grâce à ce prétraitement rigoureux, le système peut atteindre de meilleures performances en reconnaissance faciale, même dans des conditions variées (variations de luminosité, angles, expressions faciales).

### 3.3.4 Extraction des Features

L'extraction des *features* (ou représentations vectorielles) constitue une étape clé dans le processus de reconnaissance faciale. Une fois les images de visages prétraitées, elles sont transformées en vecteurs d'embeddings à l'aide du modèle `FaceNet`. Ces embeddings permettent de représenter chaque visage sous forme numérique dans un espace vectoriel, facilitant ainsi la comparaison et la classification.

Le code suivant illustre le chargement du modèle `FaceNet` et le processus d'extraction des embeddings :

**Listing 3.3:** Extraction des embeddings faciaux avec FaceNet

```
1 # Chargement du mod le FaceNet
```

```
def load_facenet_model():
    # Utiliser la version PyTorch de FaceNet
    resnet = InceptionResnetV1(pretrained='vggface2').eval().to(device)
    return resnet

facenet = load_facenet_model()

# Fonction pour extraire les embeddings
def get_embedding(face_pixels, model):
    # Convertir en tensor PyTorch
    face_tensor = torch.tensor(face_pixels.transpose(2, 0, 1)).unsqueeze(0)
        .float().to(device)

    # G n rer l'embedding
    with torch.no_grad():
        embedding = model(face_tensor)

    return embedding.cpu().numpy().reshape(-1)

# G n rer les embeddings pour toutes les images
embeddings = np.array([get_embedding(face, facenet) for face in X])
```

**Explication du Code :** Le processus d'extraction des features repose sur les étapes suivantes :

- **Chargement du modèle FaceNet** : la fonction `load_facenet_model()` initialise un modèle `InceptionResnetV1`, pré-entraîné sur le jeu de données `VGGFace2`. Ce modèle permet de produire des embeddings de haute qualité pour les visages.
- **Transformation en tensor** : chaque image de visage prétraitée est convertie en `tensor PyTorch` avec un réarrangement des axes requis par le modèle (canaux en premier).
- **Extraction des embeddings** : le modèle `FaceNet`, en mode évaluation (sans entraînement), génère un embedding pour chaque visage sous la forme d'un vecteur flottant.
- **Conversion et stockage** : les embeddings sont convertis en vecteurs `numpy` et regroupés dans un tableau pour un traitement ultérieur (classification par SVM).

**Rôle de l'Extraction des Features:** L'extraction des embeddings joue un rôle central dans le système de reconnaissance faciale :

- **Représentation vectorielle des visages** : chaque visage est représenté par un vecteur dense dans un espace de grande dimension (512 dimensions pour le modèle utilisé), ce qui facilite les comparaisons entre visages.
- **Invariance aux variations** : les embeddings capturent les caractéristiques discriminantes d'un visage tout en étant relativement invariants aux variations d'éclairage, de posture ou d'expression.
- **Facilitation de la classification** : grâce à cette représentation vectorielle standardisée, les modèles de classification tels que le `SVM` peuvent apprendre des frontières de décision efficaces pour différencier les identités.
- **Robustesse du système** : la qualité des embeddings détermine directement la précision globale du système de reconnaissance faciale.

En résumé, l'étape d'extraction des features permet de transformer les images de visages brutes en vecteurs numériques exploitables par les algorithmes de classification, garantissant ainsi la performance et la robustesse du système final.

### 3.3.5 Classification (SVM)

Une fois les embeddings extraits pour chaque visage, il est nécessaire de les classifier afin d'identifier l'individu correspondant à chaque vecteur. Pour cela, un modèle de Support Vector Machine (`SVM`) est utilisé. Le code suivant illustre l'entraînement et l'évaluation de ce classifieur.

**Listing 3.4:** Classification des embeddings avec SVM

```
# S paration des donn es en ensemble d'entra nement et de test
X_train, X_test, y_train, y_test = train_test_split(
    embeddings, y_encoded, test_size=0.2, random_state=42)

# Cr er un simple classifieur (SVM serait meilleur en production)
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

classifier = SVC(kernel='linear', probability=True)
classifier.fit(X_train, y_train)

#  valuer  le classifieur
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy du classifieur: {accuracy:.2f}")

# Sauvegarder le classifieur
```

```
18 import joblib
19 joblib.dump(classifier, 'face_classifier.pkl')
```

**Explication du Code** Le processus de classification est réalisé en plusieurs étapes :

- **Séparation des données** : les embeddings sont divisés en un ensemble d'apprentissage (`train`) et un ensemble de test (`test`) avec un ratio 80/20 afin d'évaluer les performances du classifieur sur des données inédites.
- **Création du classifieur SVM** : un modèle de `SVC` (Support Vector Classifier) avec un noyau linéaire est instancié. L'option `probability=True` permet de calculer les probabilités d'appartenance à chaque classe, ce qui est utile pour des décisions plus fines.
- **Entraînement du modèle** : le classifieur est entraîné sur les embeddings de l'ensemble d'apprentissage.
- **Évaluation du modèle** : la précision (`accuracy`) du modèle est mesurée sur l'ensemble de test, ce qui donne une estimation objective de sa capacité de généralisation.
- **Sauvegarde du modèle** : le classifieur entraîné est sauvegardé sous forme de fichier `pkl`, afin de pouvoir être réutilisé en production sans nécessiter un nouvel entraînement.

**Pourquoi le choix du SVM exactement ?** Le choix du `SVM` (Support Vector Machine) pour la classification des embeddings faciaux repose sur plusieurs considérations techniques et pratiques :

- **Efficacité sur des espaces de grande dimension** : les embeddings produits par `FaceNet` sont des vecteurs de 512 dimensions. Le `SVM` est particulièrement performant dans ce type d'espace, où il peut trouver des frontières optimales de séparation.
- **Robustesse avec peu d'échantillons par classe** : contrairement aux réseaux de neurones qui nécessitent de grandes quantités de données, le `SVM` fonctionne efficacement même lorsque peu d'exemples sont disponibles pour chaque individu (ce qui est souvent le cas dans des contextes académiques ou industriels).
- **Modèle rapide à entraîner** : le temps d'apprentissage d'un `SVM` avec noyau linéaire est très raisonnable, ce qui permet une itération rapide et facilite l'adaptation du système.
- **Capacité de généralisation** : les `SVM` sont bien connus pour leur capacité à généraliser correctement sur des données non vues, ce qui est crucial pour une reconnaissance faciale fiable.

- **Simplicité d'intégration en production** : un `SVM` entraîné peut être facilement intégré dans un pipeline existant, avec une faible empreinte mémoire et une rapidité d'inférence en temps réel, adaptée aux contraintes matérielles d'un système embarqué (par exemple sur Raspberry Pi).

En résumé, le `SVM` constitue un excellent compromis entre simplicité, performance et robustesse pour la classification des embeddings faciaux dans ce projet.

### 3.3.6 Déploiement du Système de Reconnaissance Faciale en Temps Réel

Dans cette section, nous présentons le déploiement du système complet de reconnaissance faciale en temps réel, tel que mis en œuvre dans le code présenté précédemment. Le système a été conçu pour fonctionner initialement sur une machine de développement (PC) en vue de valider son bon fonctionnement avant le déploiement sur la plateforme embarquée Raspberry Pi.

**Architecture générale** Le pipeline déployé en temps réel repose sur l'architecture suivante :

- **Capture vidéo en continu** via une webcam connectée (USB ou intégrée), avec un rafraîchissement en temps réel de la fenêtre d'affichage.
- **Détection des visages** frame par frame à l'aide du détecteur MTCNN, offrant à la fois détection rapide et alignement des visages.
- **Extraction des embeddings** des visages détectés grâce au modèle pré-entraîné FaceNet (InceptionResnetV1), générant un vecteur numérique pour chaque visage.
- **Classification des visages** par un modèle SVM entraîné en amont, permettant d'identifier l'individu correspondant ou de signaler un visage inconnu.
- **Gestion intelligente des présences** avec suivi temporel : chaque détection valide est confirmée après une période de stabilisation de 2 secondes, évitant les détections furtives ou erronées.
- **Enregistrement des résultats** : à l'issue de la session de détection (durée paramétrée à 30 minutes dans notre implémentation), un fichier CSV est automatiquement généré contenant la liste des présences pour archivage et exploitation ultérieure.

**Exemples de résultats sur webcam** Voici un exemple illustrant les résultats de la reconnaissance faciale en temps réel sur webcam, avant déploiement final sur Raspberry Pi :

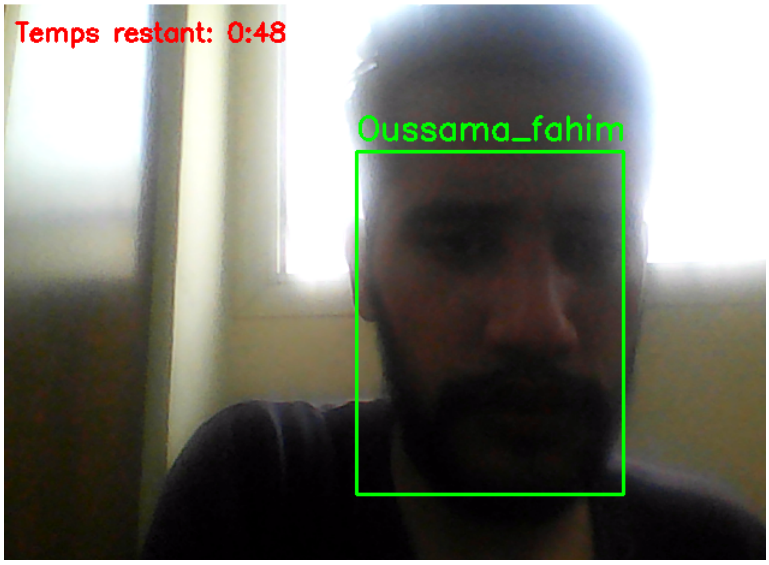



**Figure 3.7:** Exemple

### 3.3.7 Montage du Système de Reconnaissance Faciale

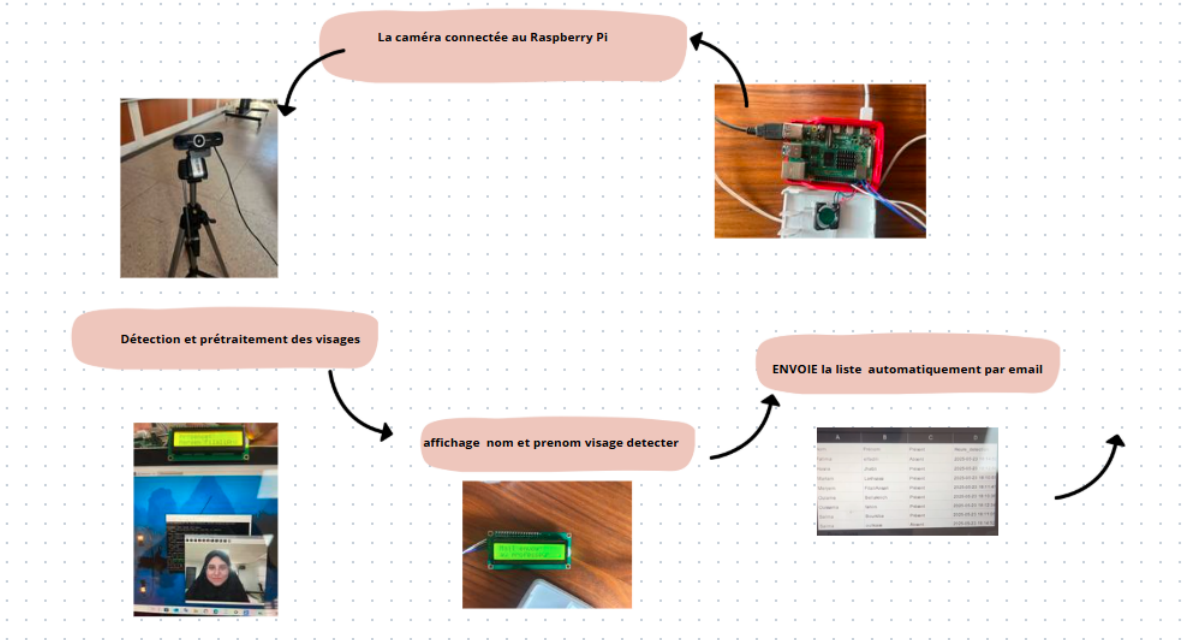



**Figure 3.8:** Montage du Système de Reconnaissance Faciale

L'image ci-dessus illustre l'assemblage complet de notre système de reconnaissance faciale basé sur Raspberry Pi. On y distingue le Raspberry Pi 4 connecté à une caméra positionnée de manière à capturer les visages des étudiants en salle. L'écran LCD relié au Raspberry Pi

affiche en temps réel les résultats de la reconnaissance, notamment les noms des personnes identifiées. Le tout est alimenté par une source 5V adaptée, assurant une autonomie et stabilité du système. Ce montage compact et intégré permet une capture fluide, un traitement instantané et une visualisation claire, facilitant ainsi la gestion automatique des présences.

### 3.3.8 Envoi automatique d'emails après détection faciale

Après la détection faciale à l'aide du Raspberry Pi, une fonctionnalité a été mise en place pour automatiser le suivi de présence à travers l'envoi d'e-mails. Cette étape vise à informer responsable des résultats de la reconnaissance, en joignant des pièces justificatives. Fonctionnalité mise en œuvre : Dès qu'un visage est détecté :

L'algorithme identifie la personne à partir de la base de données.

Un fichier Excel est automatiquement généré, contenant trois colonnes :

Nom de la personne

Statut (Présent, Absent ou Non reconnu)

Heure de détection

En parallèle :

Une capture d'image du visage détecté est enregistrée automatiquement.

Pour les personnes non reconnues ou mal détectées, une mention spéciale est ajoutée dans le fichier Excel.

Ce fichier est ensuite joint automatiquement à un e-mail, accompagné des images capturées lors de la non reconnaissance, puis envoyé à l'enseignant pour assurer un suivi fiable et traçable de la présence.

### 3.3.9 Résultats de projet

Le processus de reconnaissance faciale s'est achevé avec succès, validant l'efficacité du système mis en place. Une fois le programme exécuté, la caméra connectée au Raspberry Pi commence à capturer des images en temps réel. Celles-ci sont analysées à l'aide de l'algorithme de reconnaissance faciale .

Lorsque le système identifie une correspondance entre le visage détecté et un visage connu, il procède à l'affichage du nom de la personne reconnue sur l'**écran LCD** connecté au Raspberry

Pi. P. L'affichage du nom sur l'écran LCD permet une interaction matérielle claire et pratique, en séparant le traitement visuel du traitement textuel.


**Figure 3.9:** Visualisation du visage détecté

Une fois le projet finalisé, les résultats obtenus montrent que le système de reconnaissance faciale fonctionne correctement. Lorsque la caméra est activée, elle détecte automatiquement le visage de la personne se tenant devant l'appareil.

L'image capturée est ensuite comparée à celles enregistrées dans la base de données. Si une correspondance est trouvée, le nom de la personne apparaît à l'écran.

Une fois l'exécution de l'application terminée, un fichier Excel est généré automatiquement. Ce fichier contient la liste des personnes reconnues et non reconnues par le système, avec leur statut de présence.

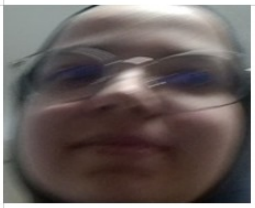
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Inconnu_12 | | Inconnu | 2025-06-13 22:05:36 | | | | |
| Inconnu_13 | | Inconnu | 2025-06-13 22:05:40 |  | | | |
| Mariam | Lakhsassi | Présent | 2025-06-13 22:05:43 | | | | |
| Fatima | elfadili | Absent | N/A | | | | |
| Hasna | Jhabli | Absent | N/A | | | | |
| Meryem | FilaliAnsari | Absent | N/A | | | | |

**Figure 3.10:** Extrait du fichier Excel généré automatiquemen

## 3.4 Interface pour les enseignants

Dans le but de faciliter l'interaction avec notre système de reconnaissance faciale, nous avons développé une interface graphique à l'aide de Streamlit, une bibliothèque Python permettant la création rapide d'applications web interactives. Cette interface permet d'effectuer différentes opérations liées à la gestion des présences de manière intuitive et visuelle.

### 3.4.1 Page de détection de présence

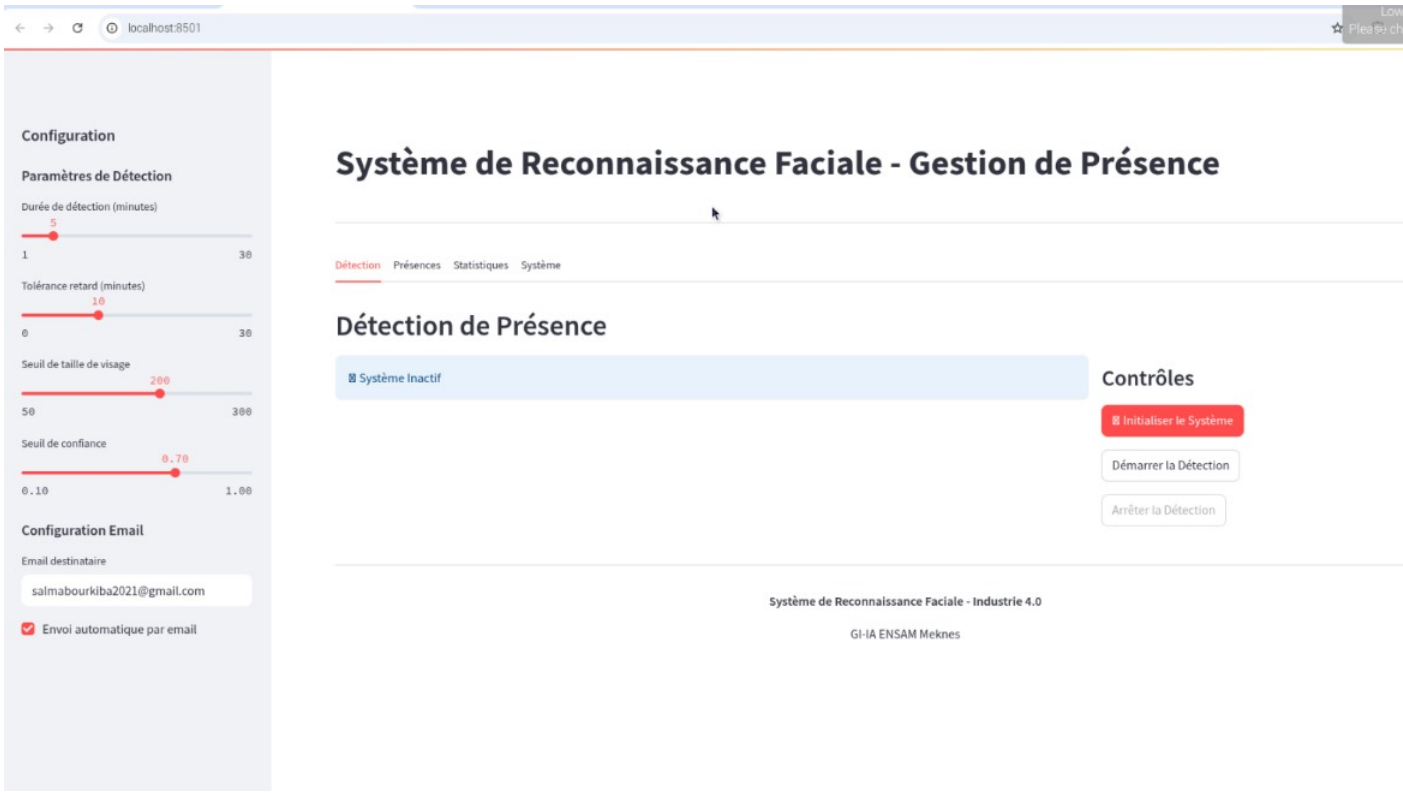



**Figure 3.11:** Page de détection de présence

La page principale permet aux enseignants de lancer ou arrêter le système de reconnaissance faciale. L'utilisateur peut :

- Initialiser le système
- Démarrer ou arrêter la détection de présence en temps réel
- Visualiser l'état du système (actif/inactif)

Plus on a ajouter une section de configuration des paramètres de détection à gauche danslaquelle l'enseignant peut ajuster les paramètres de détection:

- Durée de détection: période pendant laquelle le système reste actif pour détecter les visages
- Tolérance de retard (minutes) : délai autorisé pour considérer un étudiant comme retardataire.
- Seuil de taille de visage : filtre les visages trop petits ou trop éloignés pour améliorer la précision.

- Seuil de confiance : niveau de confiance minimal requis pour valider une reconnaissance faciale.
- Configuration Email : permet d'entrer une adresse pour l'envoi automatique des rapports de présence.

### 3.4.2 Page des Statistiques et Analyses

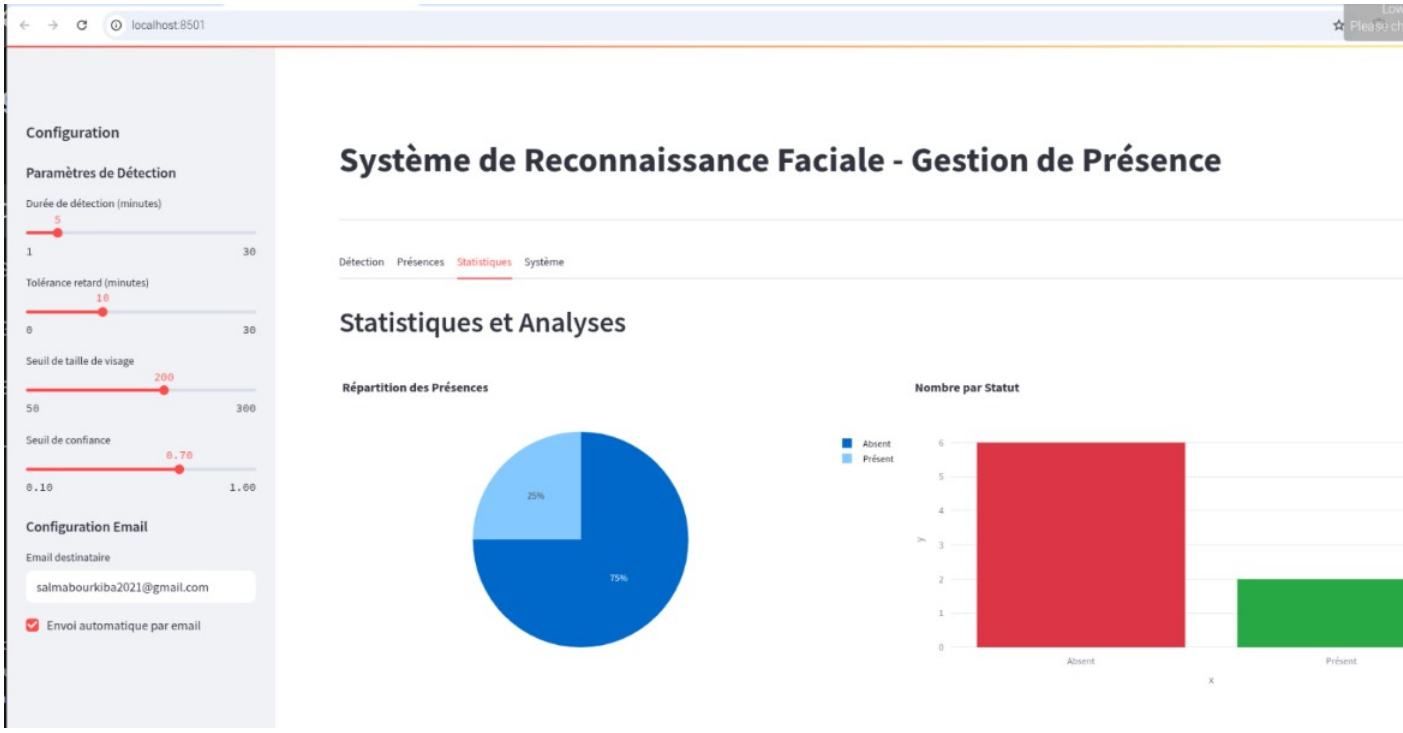



**Figure 3.12:** Page des statistiques

Cette section présente les résultats de détection sous forme de graphiques : -Un camembert affichant la répartition entre les étudiants présents et absents -Un histogramme montrant le nombre d'étudiants par statut (présent/absent). Cela va permettre aux enseignants d'avoir une vue d'ensemble rapide et compréhensible du taux d'absentéisme dans la classe

# CONCLUSION

Ce projet de reconnaissance faciale basé sur le Raspberry Pi 4 démontre la possibilité de créer un système intelligent, autonome et efficace pour la gestion de la présence. Grâce à l'intégration de la caméra, des bibliothèques Python telles qu'OpenCV , et de modules de traitement de fichiers et d'envoi d'e-mails, l'application permet non seulement de détecter et identifier les visages en temps réel, mais aussi de générer automatiquement un rapport de présence sous forme de fichier Excel et de l'envoyer par e-mail accompagné des captures d'image. Cette solution offre une alternative pratique et moderne aux méthodes traditionnelles de pointage manuel. Elle peut être appliquée dans plusieurs contextes, notamment le milieu scolaire, les entreprises, ou encore les événements, avec des avantages en termes de gain de temps, réduction des erreurs humaines, et suivi automatisé.