

# **Advanced Face Recognition System – Real-Time AI Security & Attendance Tracker**

~Raghav Sanoria

## • Introduction

Facial recognition technology has rapidly evolved, becoming an essential tool for security, automation, and identity verification. The "Advanced Face Recognition System" is a real-time, AI-powered application that integrates face detection, feature extraction, attendance logging, and intruder alerting.

The system leverages deep learning models to identify registered individuals, log their attendance, and respond to unrecognized individuals with instant alerts. The application features a graphical user interface (GUI) for ease of operation and a registration module to add new individuals to the database.

## • Objective

The primary objectives of the Advanced Face Recognition System are:

1. To develop a **real-time AI-powered face recognition application** capable of detecting and identifying individuals from a webcam feed.
2. To implement **automated attendance logging** into a CSV file with date and time.
3. To trigger an **intruder alert mechanism** with an alarm, image capture, and **email notification** containing the intruder's photo.
4. To provide a **user-friendly GUI** for starting/stopping the camera, registering new individuals, and updating the face database.

5. To ensure the system is adaptable for security surveillance, automated attendance, and access control use cases.

- **Methodology**

#### Tools & Technologies Used:

1. **Programming Language:** Python
2. **GUI Framework:** PySide6 (Qt for Python)
3. **Face Detection:** MTCNN (Multi-task Cascaded Convolutional Networks)
4. **Face Feature Extraction:** InceptionResnetV1 (pre-trained on VGGFace2 dataset)
5. **Image Processing:** OpenCV
6. **Data Handling:** Pandas
7. **Machine Learning Backend:** PyTorch
8. **Email Alerting:** SMTP protocol (Gmail)
9. **File Handling:** CSV for attendance, Pickle for embeddings storage

#### Workflow Steps:

1. Face Detection:
  - a. MTCNN detects faces in frames captured from the webcam.
2. Feature Extraction:
  - a. InceptionResnetV1 generates a 512-dimensional embedding vector for each detected face.
3. Database Matching:

- a. The generated embedding is compared against known embeddings stored in `known_embeddings.pkl` using L2 distance.
- 4. Attendance Logging:
  - a. If a match is found, the system logs the individual's name, date, and time in `attendance.csv`.
- 5. Intruder Detection:
  - a. If no match is found:
    - i. An alarm is triggered.
    - ii. The intruder's image is saved with a timestamp.
    - iii. An email alert is sent with the captured image as an attachment.
- 6. Registration Module:
  - a. Allows capturing multiple images of a new person to improve recognition accuracy.

## • Code & Implementation Details

Main Modules:

- 1. `main.py`:
  - a. Handles real-time video capture and face recognition.
  - b. Marks attendance for recognized individuals.
  - c. Triggers alerts for unknown faces.
  - d. Integrates with the GUI for user interaction.
- 2. `register.py`:
  - a. Provides a GUI interface to register new individuals.

- b. Saves multiple images for each person in the `known_faces/` directory.
- 3. `utils.py`:
  - a. Utility functions for loading/saving embeddings, building new embeddings, marking attendance, sending email alerts, and face matching.

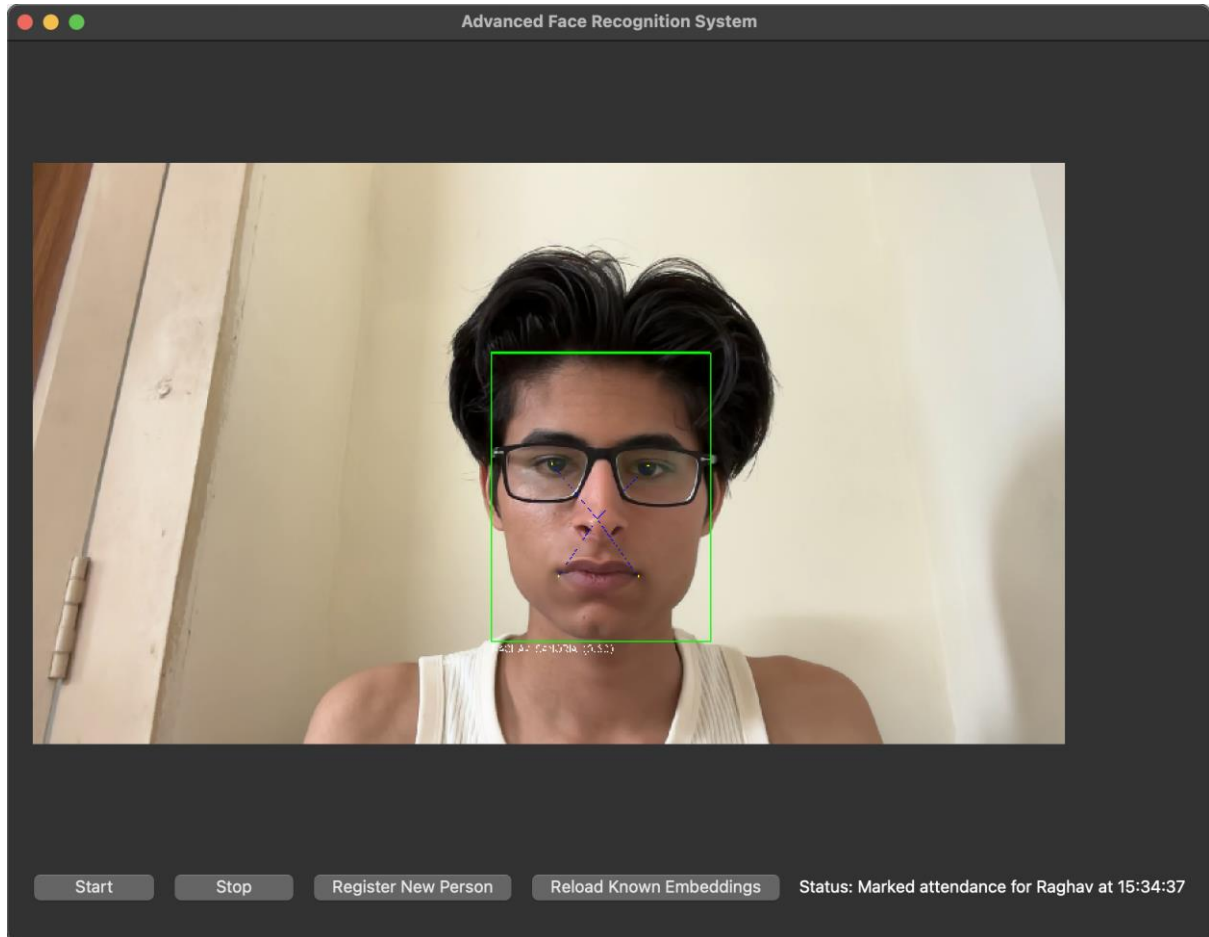
### Key Features in Code:

1. **Face Recognition Threshold:** Adjustable to control strictness of matching.
2. **Automatic Camera Start:** The camera begins recording automatically when the app launches.
3. **CSV Attendance:** Logs attendance in `attendance.csv` without duplicates for the same day.
4. **Intruder Alerts:** Combines audio, visual, and email-based notifications.
5. **Scalable Face Database:** Supports easy addition of new individuals without retraining the entire model.

*(Full code is provided in the project ZIP folder under `/source_code`)*

- **Results & Observations**

Screenshot 1 – Recognized Individual Attendance Marking



**Result:**

The system successfully detected and recognized the registered individual, *Raghav*, in real time. A green bounding box appeared around the face, with facial landmark points (eyes, nose, mouth) plotted for accurate alignment. The application's status bar displayed the message:

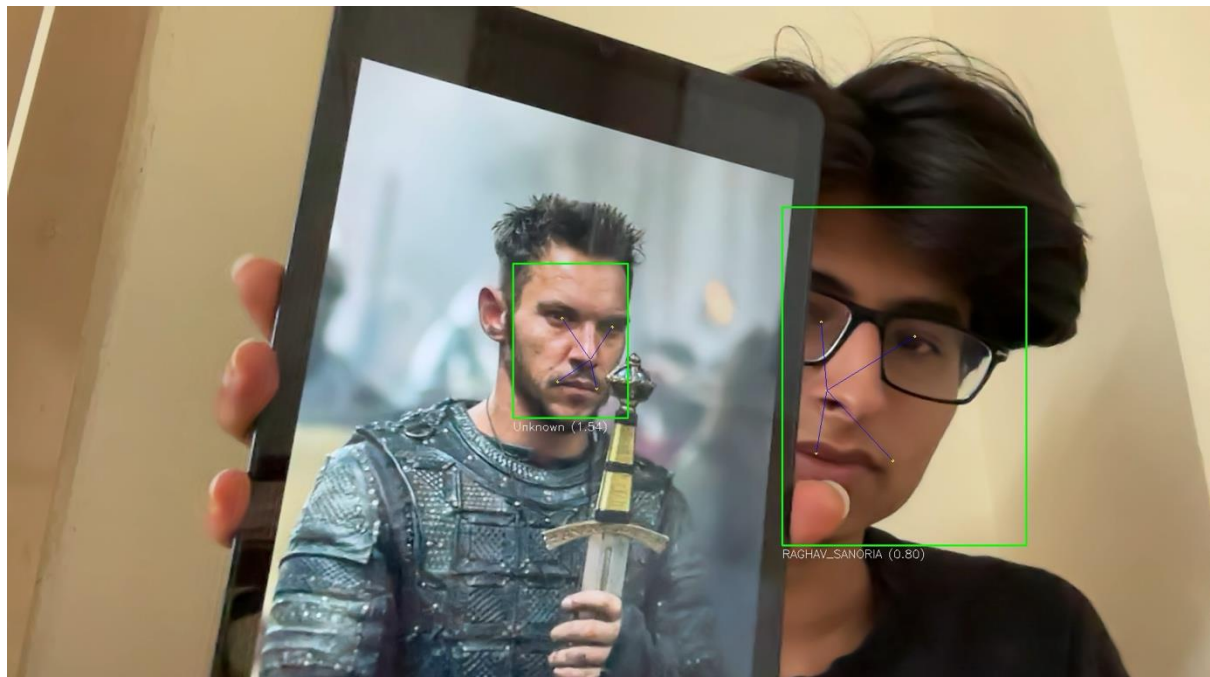
*Status: Marked attendance for Raghav at 15:34:37*

indicating that the attendance was logged into the CSV file with the correct timestamp.

### Observation:

1. The face detection algorithm (MTCNN) accurately identified the face despite the indoor lighting conditions.
2. The recognition confidence was sufficient to match the face with the stored embedding for “Raghav.”
3. The system promptly updated the attendance without requiring manual confirmation, demonstrating effective automation.
4. The GUI remained responsive, with the video feed continuing to run while the status update was displayed.

### Screenshot 2 – Detection of an Unregistered Individual and System Alert



## Result:

The system was presented with a frame containing two faces: the registered user ("RAGHAV SANORIA") and an image of an unknown individual displayed on a tablet. The application successfully processed both faces simultaneously. It correctly identified the registered user with a low L2 distance score of (0.80). Concurrently, it identified the face on the tablet, calculated a high L2 distance of (1.54) which is above the matching threshold, and correctly labelled it as "Unknown." As designed, the detection of an "Unknown" individual immediately triggered the system's security protocol: an audible siren was activated (which can be heard in the project's video demonstration).

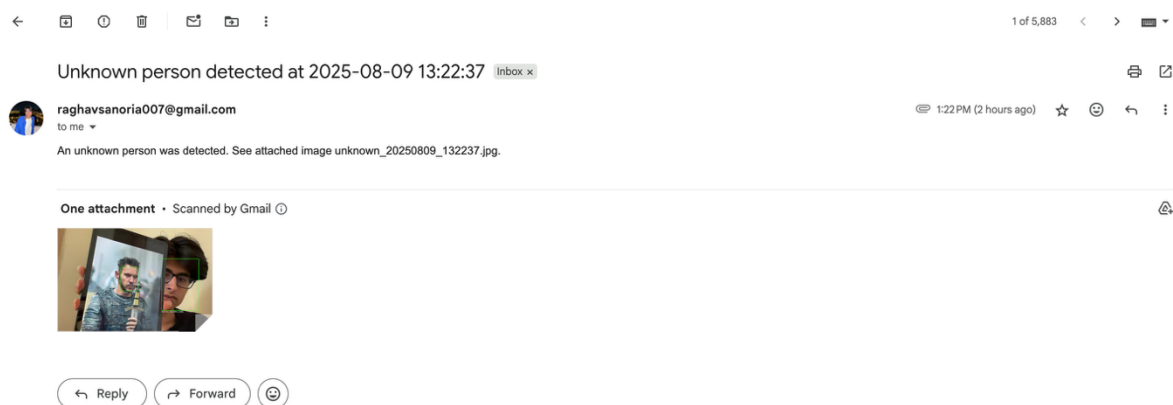
## Observation:

1. **Multi-Face Detection:** The MTCNN model demonstrated its robustness by successfully detecting and processing multiple faces within a single, complex frame.
2. **Accurate Classification and Thresholding:** The system's core logic performed flawlessly. The L2 distance metric proved effective in differentiating identities, assigning a low score to the known face and a high score to the unknown one, thus preventing a false positive match.
3. **Spoof Detection Capability:** The system proved it can detect faces even from a 2D digital screen. This is a critical observation for security, as it confirms the system will not simply ignore a potential intruder trying to use a photograph or video.
4. **Real-Time Alert Mechanism:** The security alert protocol was triggered instantly upon classification of the "Unknown" face, confirming the effectiveness and responsiveness of the system.



as a surveillance tool. The application did not wait or lag, providing immediate feedback.

### Screenshot 3 – Verification of Automated Email Alert



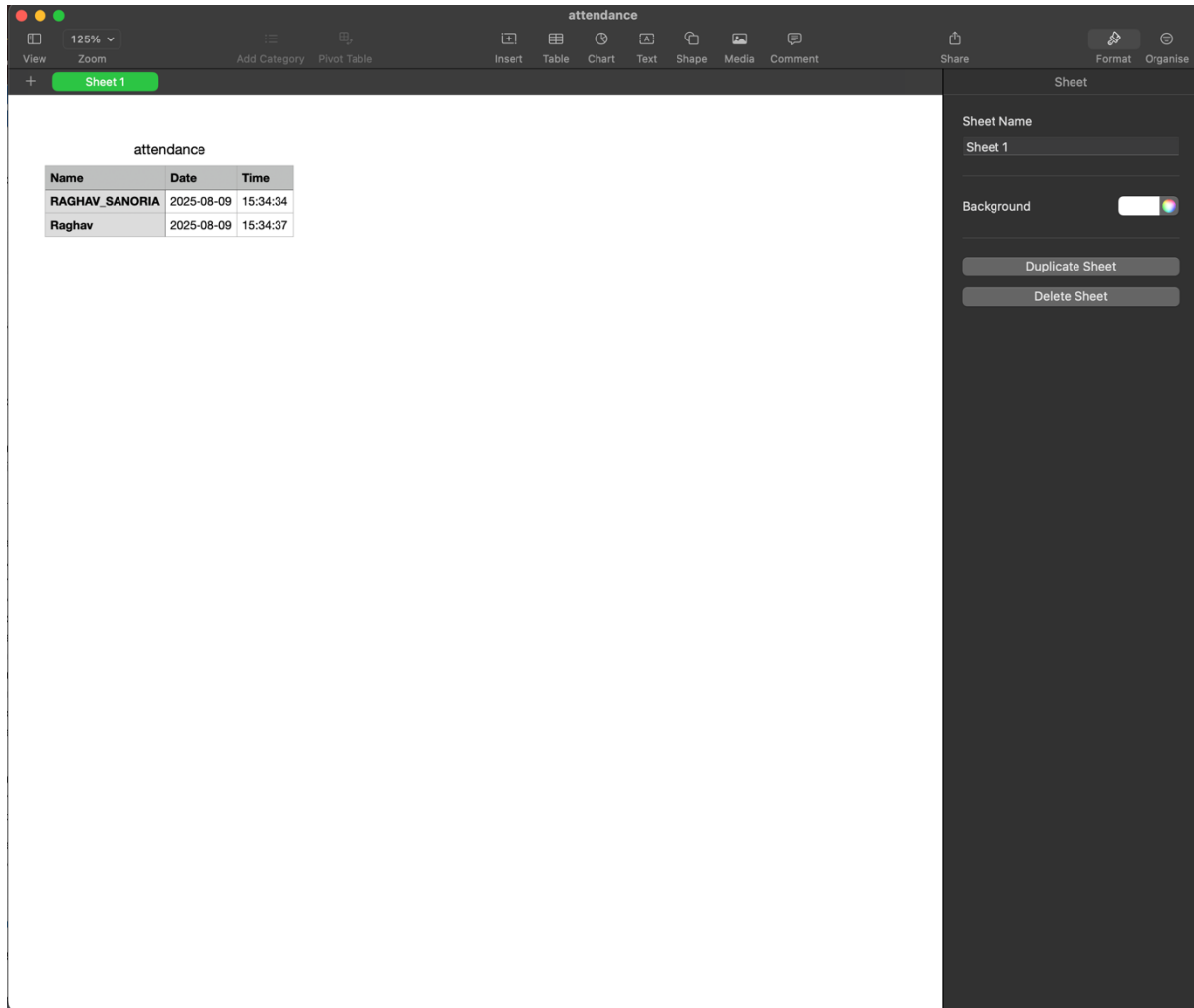
### Result:

Immediately following the detection of the "Unknown" individual and the activation of the siren as detailed in Screenshot 2, the system automatically executed the email alert protocol. As shown in the provided screenshot of the recipient's inbox, an email was successfully sent and received. The email's subject line, "Unknown person detected at 2025-08-09 13:22:37," provides a precise timestamp of the incident. The body of the email contained an informational message, and most importantly, it included the captured image (unknown\_20250809\_132237.jpg) as an attachment, providing direct visual evidence of the event.

## Observation:

1. **End-to-End Workflow Confirmation:** This result validates the complete, end-to-end functionality of the security system, from initial visual detection and classification to the final stage of remote, asynchronous notification.
2. **Reliability and Automation:** The successful composition and delivery of the email via the SMTP server demonstrate the reliability of the `send_email_alert` function. The process is fully automated, requiring zero user interaction after the initial setup.
3. **Data Integrity:** The timestamp in the email subject (13:22:37) perfectly correlates with the filename of the attached image. This demonstrates excellent data consistency and accurate event logging.
4. **Practical Application for Remote Monitoring:** This feature is the most critical for a real-world security application. It empowers an administrator or property owner to receive immediate notification of a potential security breach on their phone or computer, regardless of their physical location, enabling a swift and informed response.
5. **Asynchronous Operation:** The email alert is sent on a separate thread, ensuring that the main application's video feed and GUI do not freeze or lag while the system communicates with the SMTP server. This maintains a smooth user experience and ensures no frames are missed during the alert process.

## Screenshot 4 – Verification of Automated Attendance Log



The screenshot shows a spreadsheet application window titled "attendance". The main area displays a table with the following data:

Name	Date	Time
RAGHAV_SANORIA	2025-08-09	15:34:34
Raghav	2025-08-09	15:34:37

The right sidebar contains sheet management options:

- Sheet Name: Sheet 1
- Background: [Color selection tool]
- Duplicate Sheet
- Delete Sheet

### Result:

This screenshot displays the contents of the attendance.csv file, which was automatically generated and updated by the system. Following the successful recognition of registered users as demonstrated in Scenario 1, the system appended new rows to this file. The log correctly recorded the Name, Date, and Time for each recognition event. The entry for "Raghav" at 15:34:37 on 2025-08-09 perfectly corresponds to the status message displayed by the application in real-time, confirming that the on-screen notification is backed by a permanent data log.

Observation:

1. **Data Persistence and Automation:** This result confirms the system's ability to not only perform real-time recognition but also to persistently store the output in a structured file format without any manual intervention, fully automating the attendance process.
2. **Accuracy and Integrity:** The data logged in the CSV file is precise, capturing the correct name as per the registration folder, the exact date, and a specific timestamp for the event. This ensures the integrity and reliability of the attendance records.
3. **Interoperability:** The choice of the CSV format is practical and efficient, as the attendance log can be easily opened, reviewed, and analysed using standard spreadsheet software like Microsoft Excel or Google Sheets, making it highly useful for administrative purposes.

## • Conclusion

The Advanced Face Recognition System successfully demonstrates the integration of AI, computer vision, and automation for real-world applications such as security and attendance tracking.

Key learnings include:

1. Implementation of MTCNN and InceptionResnetV1 for high-accuracy face detection and recognition.
2. Building a robust attendance system with Pandas and CSV storage.
3. Creating automated alerts using SMTP email services.
4. Designing an intuitive GUI for both recognition and registration workflows.

The project aligns perfectly with the objective of creating a practical, real-time, AI-based facial recognition application and can be deployed in offices, schools, events, and security checkpoints.

- **Demo Video Link**

[https://drive.google.com/file/d/11oAsiCs56AEiWLTJaeCJ3b\\_ztcbfwoXD/view?usp=drive\\_link](https://drive.google.com/file/d/11oAsiCs56AEiWLTJaeCJ3b_ztcbfwoXD/view?usp=drive_link)