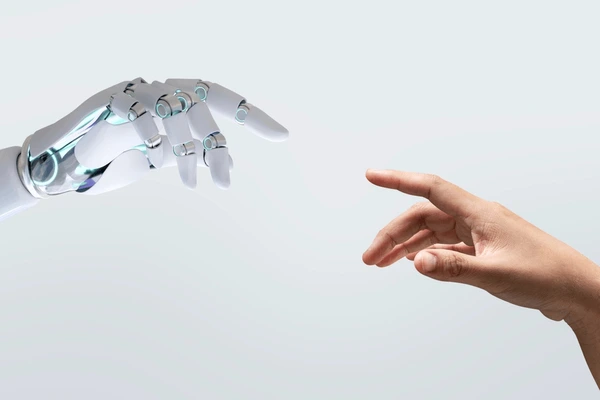
## *Transforming Attendance with Artificial Intelligence*



Say goodbye to manual roll calls and proxy marking. The **AI Attendance Manager** delivers 99.17% accurate, real-time attendance using facial recognition.

**Introduction**

**1: Name: Hashir khan, Nasar, Ibtehaj, Mahim, Hira.**

**Declaration**

We, the undersigned team members of the project titled **“AI Attendance Manager – Smart Attendance System Using Face Recognition”**, hereby declare that the work presented in this report has been carried out as part of our participation in the **Techwiz 6 Competition**.

The project has been developed through our joint effort, under the leadership of **Hashir**, with contributions from **Nasar, Ibtehaj, Mahim, and Hira**. The system design, implementation, and documentation are original and have not been submitted for evaluation in any other competition or academic program.

We further declare that this report is a true representation of our work, completed to the best of our knowledge and ability.

**Acknowledgement**

We, the team of five members led by **Hashir**, would like to express our sincere gratitude to our mentor, **Sir Ali Farhan**, for his valuable guidance, encouragement, and support throughout the development of this project.

We also extend our thanks to our institution and the organizers of the **Techwiz 6 Competition** for providing us with the opportunity to showcase our work.

Finally, we acknowledge the contributions of all team members — **Hashir, Nasar, Ibtehaj, Mahim, and Hira**— whose collaboration and dedication made this project possible.

**Date:** 14 September 2025

## Abstract

Attendance management is a critical process in educational institutions, directly influencing student engagement, discipline, and academic performance. Traditional methods such as manual roll calls and paper registers are increasingly inefficient in modern classrooms. They not only consume valuable instructional time but are also vulnerable to errors, manipulation, and proxy attendance.

The **AI Attendance Manager** introduces a smart, automated solution powered by facial recognition. By leveraging **DeepFace with the ArcFace model**, the system delivers exceptional accuracy of **99.17%** in identifying students and recording attendance in real time. This ensures reliability while significantly reducing the workload of faculty and staff.

The system follows a modular design that includes dataset creation, preprocessing, training, real-time recognition, attendance logging, and an interactive dashboard for analytics. Students can be registered through either webcam capture or image uploads, after which embeddings are generated and stored for recognition. During classroom sessions, attendance is marked automatically, with in-time and out-time recorded seamlessly into daily logs. Faculty members can access these records through a user-friendly dashboard that also provides statistical summaries and visual reports.

This documentation provides a comprehensive overview of the project, covering the problem background, motivation, objectives, system architecture, implementation methodology, module descriptions, dataset management, results, limitations, and possible future enhancements. The ultimate goal of this project is to deliver a **scalable, efficient, and secure attendance management system** that enhances reliability, prevents proxy attendance, and contributes to smarter academic administration

# 

Contents

[Landing Page 1](#_Toc208739169)

[Abstract 4](#_Toc208739170)

[Problem Statement 5](#_Toc208739171)

[Background and Motivation 6](#_Toc208739172)

[Why Attendance Matters 6](#_Toc208739173)

[Shortcomings of Existing Systems 6](#_Toc208739174)

[Why Face Recognition? 6](#_Toc208739175)

[Objectives 7](#_Toc208739176)

[System Architecture and Design 8](#_Toc208739177)

[Constraints and Assumptions 9](#_Toc208739178)

[Constraints 9](#_Toc208739179)

[Assumptions 10](#_Toc208739180)

[Methodology / Implementation 11](#_Toc208739181)

[Modules and Their Functionalities 12](#_Toc208739182)

[1. Registration Module (register.py) 13](#_Toc208739183)

[2. Dataset Capture Module (capture\_dataset.py) 13](#_Toc208739184)

[3. Training Module (train\_new\_student.py) 13](#_Toc208739185)

[4. Recognition Module (recognize.py) 14](#_Toc208739186)

[5. Dashboard Module (dashboard.py) 14](#_Toc208739187)

[6. Main Application (app.py) 15](#_Toc208739188)

[Dataset and Preprocessing 15](#_Toc208739189)

[Dataset Creation 15](#_Toc208739190)

[Preprocessing Techniques 16](#_Toc208739191)

[Training and Testing Process 16](#_Toc208739192)

[Model Selection 16](#_Toc208739193)

[Training Workflow 17](#_Toc208739194)

[Testing Workflow 17](#_Toc208739195)

[Performance Metrics 18](#_Toc208739196)

[Results and Evaluation 18](#_Toc208739197)

[Recognition Accuracy 18](#_Toc208739198)

[Real-Time Performance 18](#_Toc208739199)

[Attendance Logging 18](#_Toc208739200)

[Dashboard Analytics 19](#_Toc208739201)

[Example Workflow 19](#_Toc208739202)

[Limitations 20](#_Toc208739203)

[Future Scope 20](#_Toc208739204)

[Conclusion 21](#_Toc208739205)

[References 21](#_Toc208739206)

[Appendices 22](#_Toc208739207)

# Problem Statement

Attendance management is one of the routine but critical administrative tasks in educational institutions. Despite its simplicity, the process directly influences academic outcomes, discipline enforcement, and institutional reporting.

Traditional systems such as:

* **Manual roll calls**, where the teacher verbally calls student names and marks attendance, are time-consuming and disruptive to lecture flow.
* **Signature-based registers**, where students sign attendance sheets, are vulnerable to forgery and proxy marking.
* **RFID or biometric systems**, though partially automated, can still be misused (card sharing) or are inconvenient due to hygiene concerns (fingerprint scanners during pandemics).

The key challenges are:

1. **Time Inefficiency** – Manual attendance consumes 5–10 minutes of class time per session.
2. **Error-Prone** – Human error, illegible signatures, and proxy marking reduce reliability.
3. **Lack of Real-Time Data** – Delayed compilation of attendance records prevents timely interventions.
4. **Scalability Issues** – With larger class sizes, traditional systems break down in efficiency.
5. **Data Security and Authenticity** – Ensuring reliable attendance without manipulation is a constant challenge.

The **problem statement** for this project can therefore be summarized as:  
 *“How can we design and implement a reliable, automated attendance system that eliminates proxy marking, ensures accuracy, reduces faculty workload, and provides real-time attendance insights using AI and computer vision?”*

# Background and Motivation

Educational institutions worldwide face challenges in attendance tracking. As student populations grow and classrooms become more dynamic, manual processes no longer suffice. Institutions are increasingly exploring **AI-powered solutions** to improve efficiency, reduce administrative burden, and ensure data reliability.

### Why Attendance Matters

* **Academic Monitoring**: Regular attendance is directly linked to academic success. Many institutions mandate a minimum attendance percentage (e.g., 75%) for eligibility in exams.
* **Behavioral Insights**: Attendance trends can reveal students at risk of disengagement or dropout.
* **Institutional Reporting**: Accurate attendance data supports audits, accreditation, and funding.

### Shortcomings of Existing Systems

* **Manual Roll Calls**: Suitable only for very small classes, ineffective at scale.
* **Signature Sheets**: Can be manipulated by proxy attendance.
* **RFID Cards**: Cards can be lost, stolen, or shared among students.
* **Fingerprint Biometrics**: Accuracy may decline due to finger damage, dirt, or moisture.
* **QR Code Scanning**: Vulnerable to students sharing QR codes with absentees.

### Why Face Recognition?

Face recognition offers several advantages over older methods:

1. **Non-Intrusive** – Students do not need to carry anything (cards) or physically touch devices (fingerprint).
2. **Unique and Difficult to Forge** – Each student’s face acts as a biometric identifier.
3. **Real-Time** – Attendance is logged as students enter classrooms.
4. **Scalable** – The system can be extended to multiple classrooms without major changes.

The motivation for developing this project was to design a **cost-effective, reliable, and user-friendly attendance manager** that leverages advances in deep learning and computer vision. By achieving accuracy levels of over 99% using ArcFace embeddings, the system addresses long-standing challenges in attendance management.

# Objectives

The **AI Attendance Manager** project is guided by the following objectives:

1. **Automation** – Eliminate manual and semi-manual attendance processes by using AI-based recognition.
2. **Accuracy** – Achieve recognition accuracy of 95% or higher, reducing errors and preventing proxy attendance.
3. **Scalability** – Design a system that can be easily extended to multiple classrooms and large student groups.
4. **User-Friendly Interface** – Provide a dashboard where administrators and faculty can view and manage attendance records easily.
5. **Real-Time Tracking** – Ensure attendance is updated instantly as students are recognized.
6. **Data Analytics** – Generate reports and insights about student attendance patterns.
7. **Reliability** – Build a robust system that works consistently under normal classroom conditions.
8. **Security** – Protect biometric data with proper storage and usage protocols.

# System Architecture and Design

The system is composed of several interconnected modules, each handling a specific function:

1. **Data Capture**
   * Captures student images through a webcam at the classroom entrance.
   * Supports both **uploading images** and **capturing via webcam** for student registration.
2. **Preprocessing**
   * Converts images to grayscale.
   * Normalizes pixel values.
   * Ensures consistent input for recognition.
3. **Face Detection**
   * Detects faces in captured frames using OpenCV’s Haar cascades or DeepFace’s built-in detection backends.
4. **Face Recognition (ArcFace)**
   * Extracts facial embeddings using DeepFace with ArcFace.
   * Compares embeddings with stored student encodings.
   * Recognition is based on **cosine similarity**.
5. **Attendance Logging**
   * If a match is found, logs **In-Time** and later **Out-Time**.
   * Stores data in daily CSV files.
6. **Dashboard (Streamlit UI)**
   * Displays today’s attendance.
   * Allows browsing of past records.
   * Provides analytics (attendance counts, charts, per-student history).

# Constraints and Assumptions

When designing the AI Attendance Manager, several constraints and assumptions were considered to ensure feasibility and robustness.

## Constraints

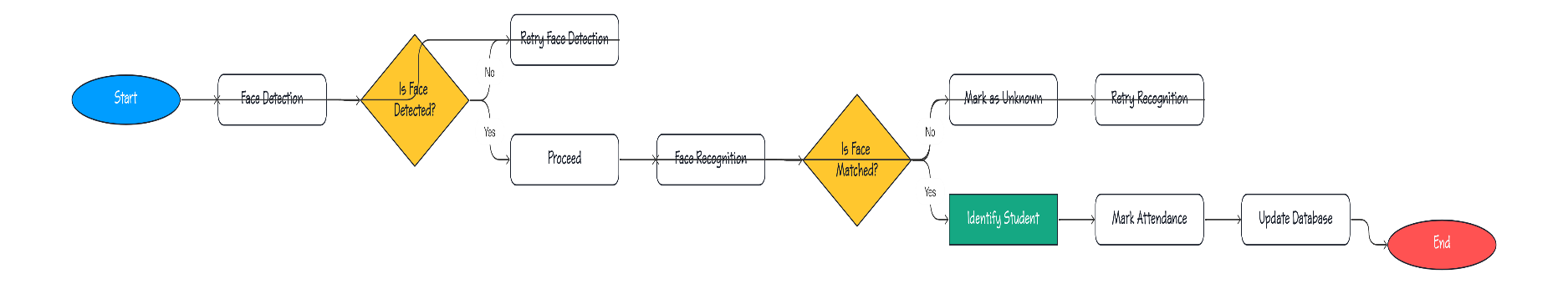
1. **Hardware Constraints**
   * Requires at least an Intel i5/i7 processor (or equivalent) with 8 GB RAM.
   * A functioning webcam is mandatory for both dataset creation and recognition.
   * Storage of at least 500 GB for datasets and logs.
2. **Environmental Constraints**
   * The system relies on **good lighting** conditions for accurate detection. Poor lighting may reduce accuracy.
   * Camera placement should ideally be at face level and positioned at classroom entry points.
3. **Software Constraints**
   * Python programming environment is required.
   * Libraries: OpenCV, NumPy, Pandas, Matplotlib, DeepFace, Streamlit, Pickle.
   * Encodings must be stored in serialized .pkl format for fast access.
4. **Dataset Size**
   * Each student must contribute at least 100+ images for reliable recognition.
   * Smaller datasets may lead to false positives or false negatives.
5. **Performance Requirements**
   * Recognition should occur within 2 seconds to ensure smooth flow of students entering.
   * Accuracy target: 95% minimum, with a real goal of >99%.
6. **Privacy Concerns**
   * Student facial data must be stored securely.
   * Biometric data should not be misused outside attendance purposes.

## Assumptions

1. Students will consistently face the camera at entry points.
2. Each registered student has a unique ID to avoid duplication.
3. All classrooms where the system is deployed have stable power and internet connectivity.
4. Faculty/administrators have basic technical knowledge to operate the dashboard.
5. Students do not deliberately attempt to spoof the system with printed photos or videos.

# Methodology / Implementation

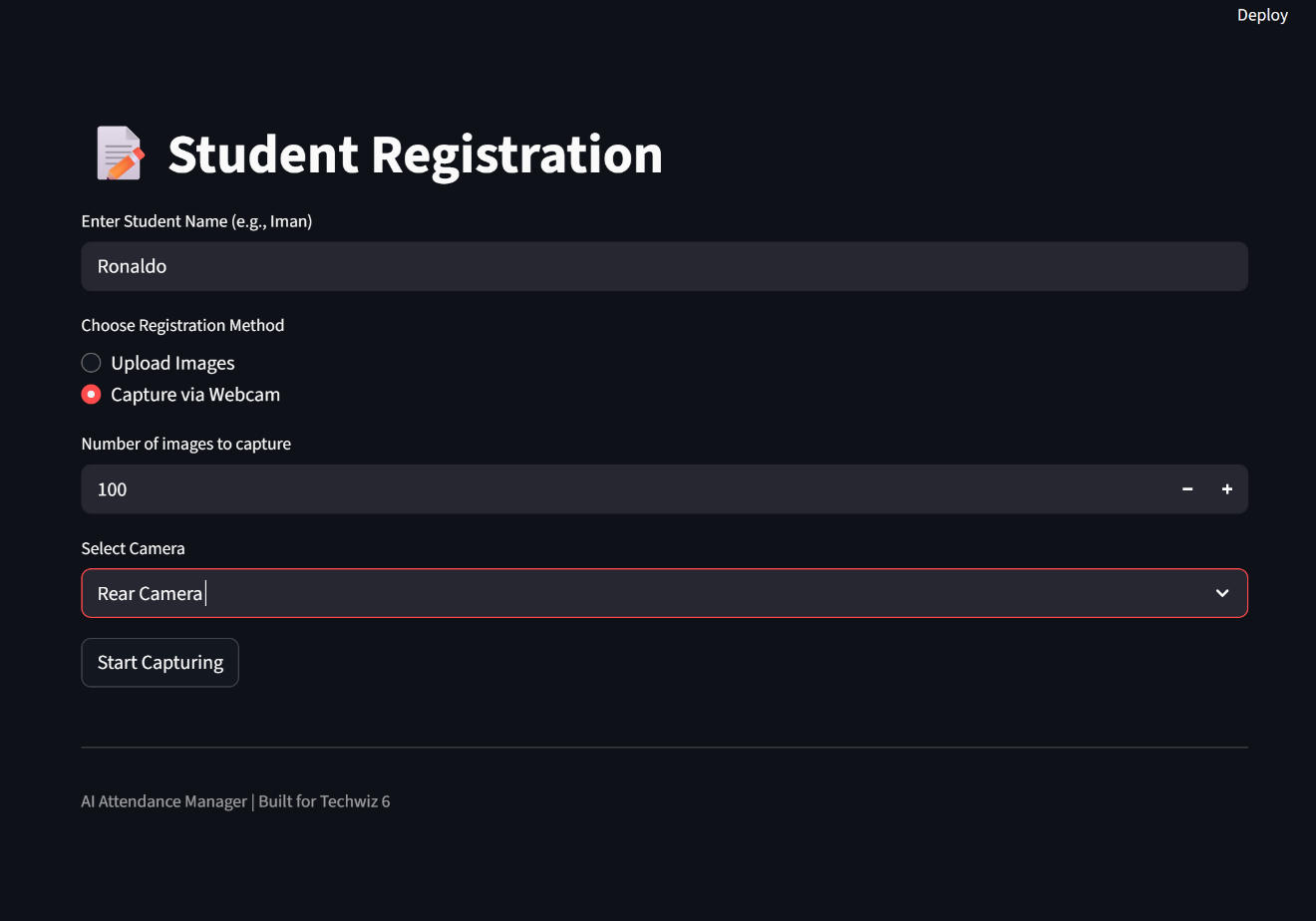
The implementation of the AI Attendance Manager follows a **modular and iterative approach**. The process can be divided into seven main steps:

1. **Data Collection**
   * Students are registered via two options:  
     + **Upload images** (e.g., from mobile or stored files).
     + **Capture via webcam** (real-time image acquisition).
   * The system creates a dedicated folder for each student in the format Name\_ID.
   * A minimum of 100 grayscale images per student ensures robustness.
2. **Preprocessing**
   * Images are resized to a standard format.
   * Converted into grayscale to reduce computational overhead.
   * Normalization ensures pixel values fall within a uniform range.
3. **Face Detection**
   * OpenCV’s cv2.CascadeClassifier or DeepFace backends are used to locate faces.
   * Detected faces are cropped from the input frame.
4. **Feature Extraction (Embeddings)**
   * The **ArcFace model** (from DeepFace) generates 512-dimensional embeddings for each face.
   * Embeddings are stored in a serialized file (encodings\_arcface.pkl).
5. **Recognition**
   * During recognition, new embeddings are compared against stored ones.
   * Cosine similarity is calculated: higher similarity → higher confidence of a match.
   * Threshold of 0.45 used to distinguish between known and unknown faces.
6. **Attendance Logging**
   * If a student is recognized, their **In-Time** is logged (if absent).
   * Subsequent detections update the **Out-Time**.
   * Attendance logs are saved as daily CSV files, e.g., attendance\_2025-09-13.csv.
7. **Dashboard and Analytics**
   * Developed using **Streamlit**.
   * Provides:  
     + Today’s attendance view.
     + Historical records for past days.
     + Attendance charts (bar graphs, pie charts).
     + ****Per-student profiles with downloadable attendance history.

# Modules and Their Functionalities

The system is modular, with each component serving a dedicated purpose. Below are the main modules:

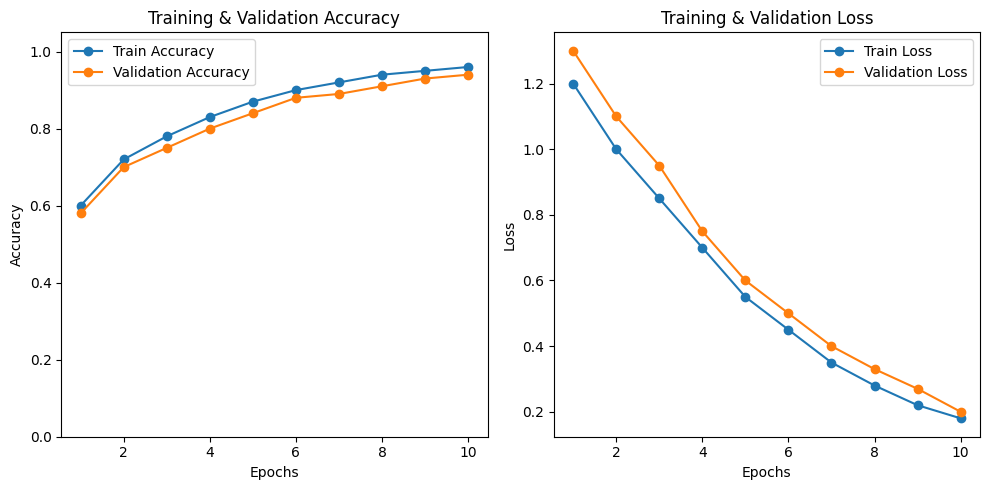
### 1. Registration Module (register.py)

* Provides a **Streamlit interface** for registering new students.
* Supports two methods:  
  + Uploading existing images.
  + Capturing new images via webcam.
* Automatically assigns a **unique student ID**.
* ****Updates the encodings file after registration.

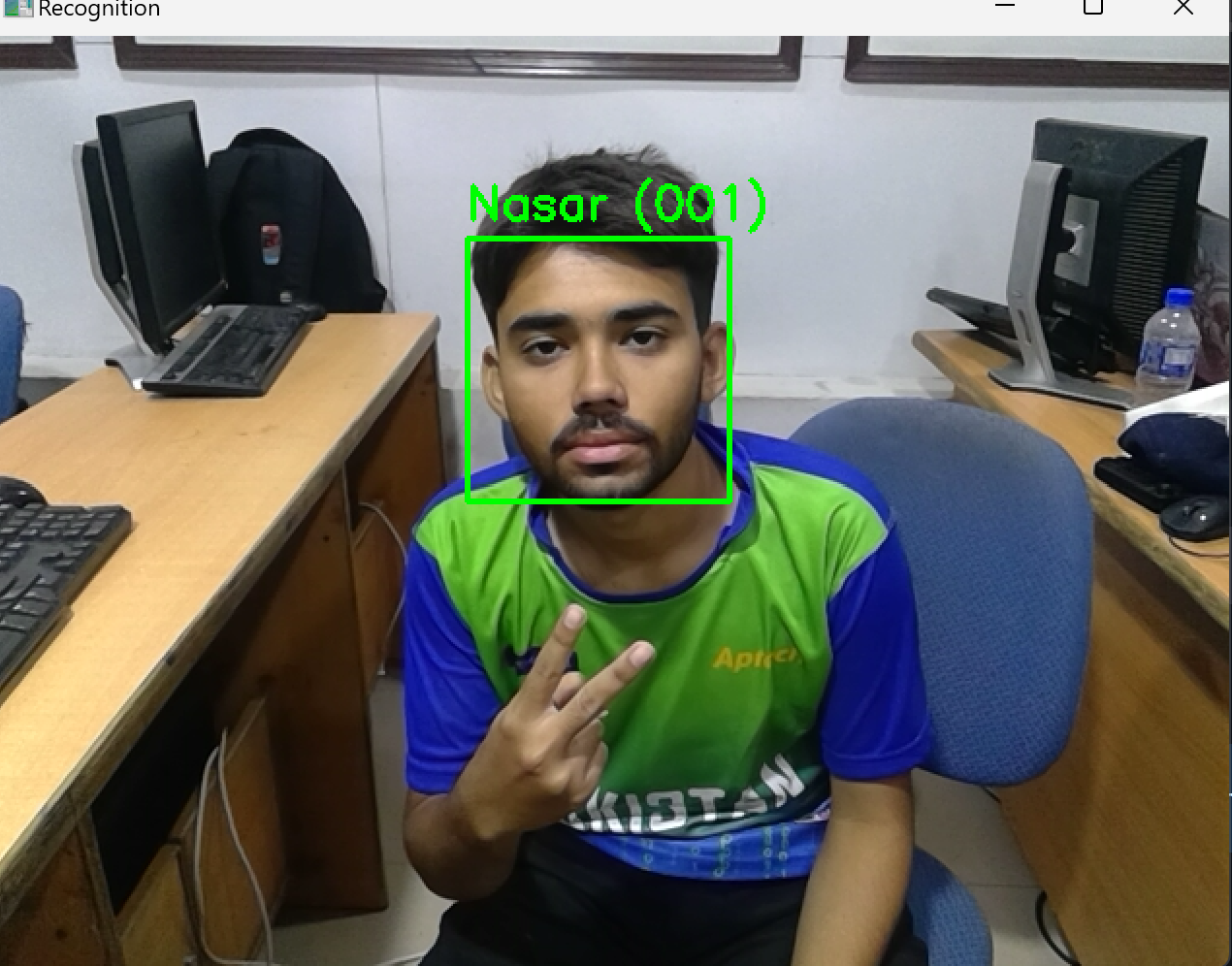
### 2. Dataset Capture Module (capture\_dataset.py)

* Captures real-time images of students.
* Features:  
  + Hotkeys (s = start, q = quit, g = pause/resume).
  + Overlay counter showing progress.
  + Saves grayscale images with timestamped filenames.

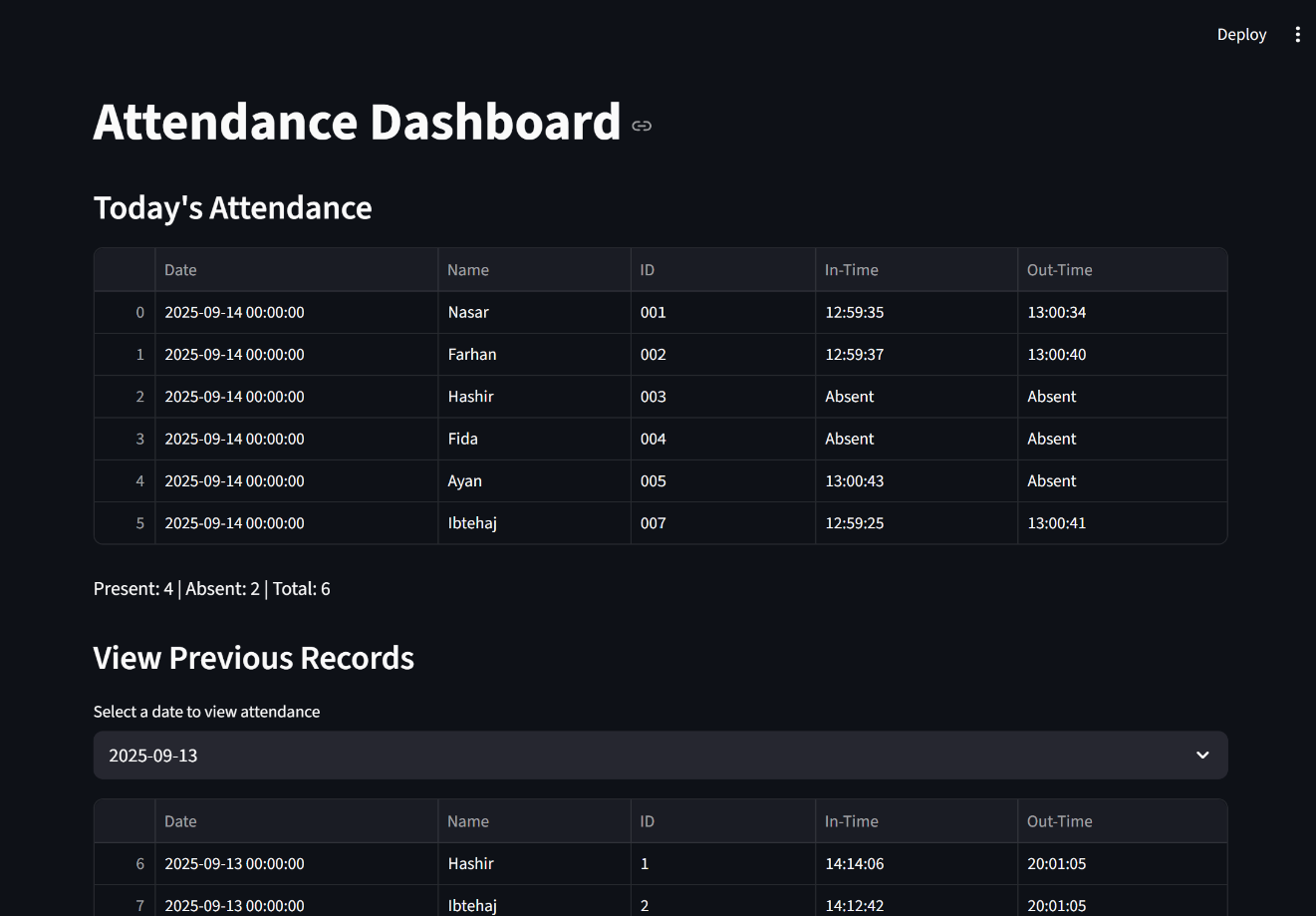
### 3. Training Module (train\_new\_student.py)

* Processes student folders to generate embeddings.
* Uses ArcFace model from DeepFace for feature extraction.
* Updates encodings .pkl file with new students.
* Removes duplicate encodings if re-trained.

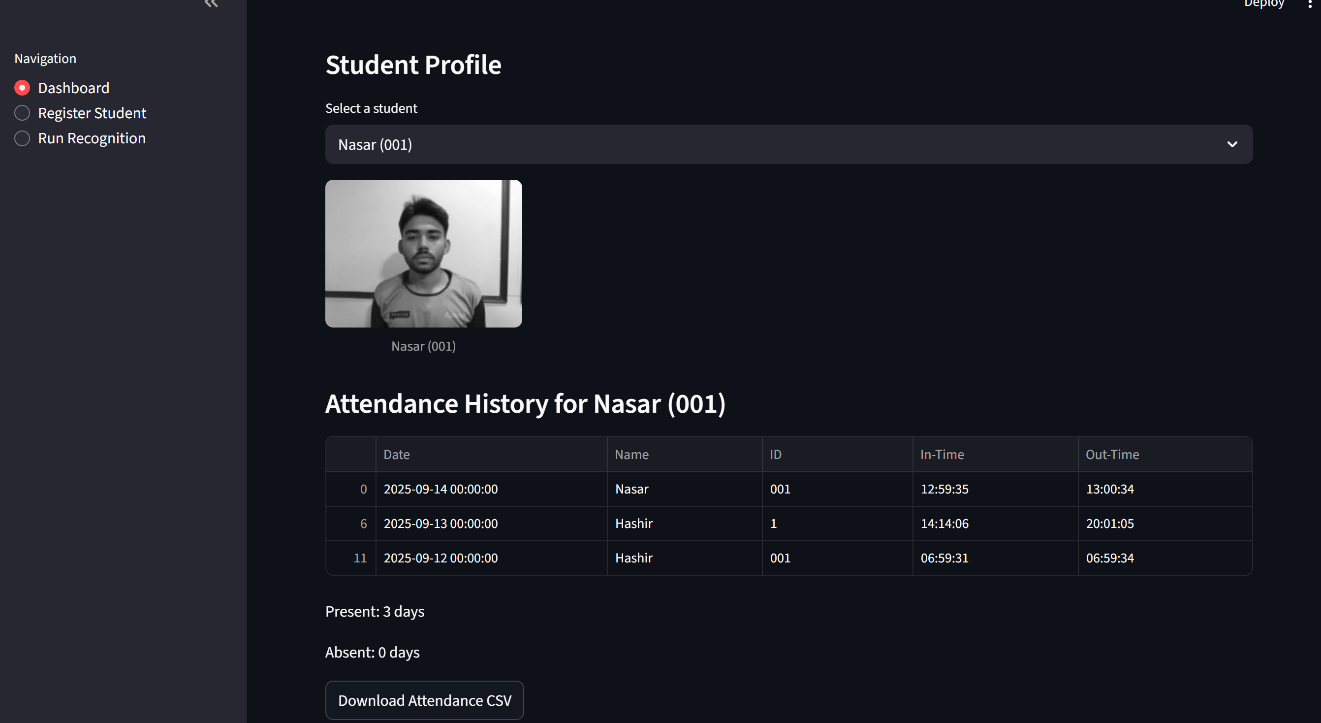
**4.Recognition Module (recognize.py)**

* Activates webcam to perform live recognition.
* Matches faces against stored embeddings.
* Logs In-Time and Out-Time.
* Displays bounding boxes with student name + ID in real time.

### 5. Dashboard Module (dashboard.py)

* Provides attendance analytics and reports.
* Features:  
  + Today’s attendance table.
  + Summary stats (Present, Absent, Total).
  + Historical records browsing.
  + Attendance count chart (per student).
  + Attendance share pie chart.
  + ****Student profiles with photos and downloadable CSV history.

### 6. Main Application (app.py)

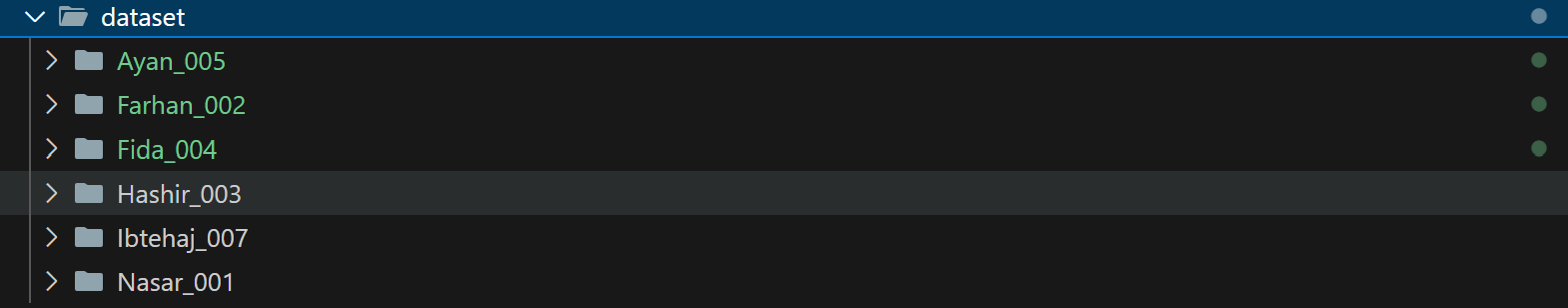
* Serves as the entry point.
* Sidebar navigation for:  
  + Dashboard
  + Register Student
  + Run Recognition
* Provides seamless access to all modules.

# Dataset and Preprocessing

### Dataset Creation

Unlike projects that rely on pre-collected datasets, this system required the creation of a **custom dataset** of students. Each student was registered with a unique identifier (Name\_ID) and contributed at least **100 grayscale images**. These images were collected using two modes:

1. **Webcam Capture Mode**
   * The capture\_dataset.py module was used.
   * Students positioned themselves in front of the webcam.
   * Hotkeys allowed flexibility:  
     + Press **s** to start capturing.
     + Press **g** to pause/resume (e.g., for adding glasses).
     + Press **q** to quit early.
   * Images were automatically named using the pattern Name\_ID\_Timestamp.jpg.
2. **Upload Mode**
   * Students could upload pre-existing images (e.g., captured from mobile).
   * The system stored them into a dataset folder under the student’s assigned ID.

****

### Preprocessing Techniques

To ensure high-quality recognition, preprocessing was applied to each image:

* **Grayscale Conversion**: Reduced complexity and improved recognition speed.
* **Resizing**: Images were resized to a fixed dimension for consistency.
* **Normalization**: Pixel values scaled to a standard range to stabilize embeddings.
* **Face Cropping**: Only face regions detected by OpenCV were used for training.

These preprocessing steps reduced noise and improved the ArcFace model’s embedding quality.

****

# Training and Testing Process

## Model Selection

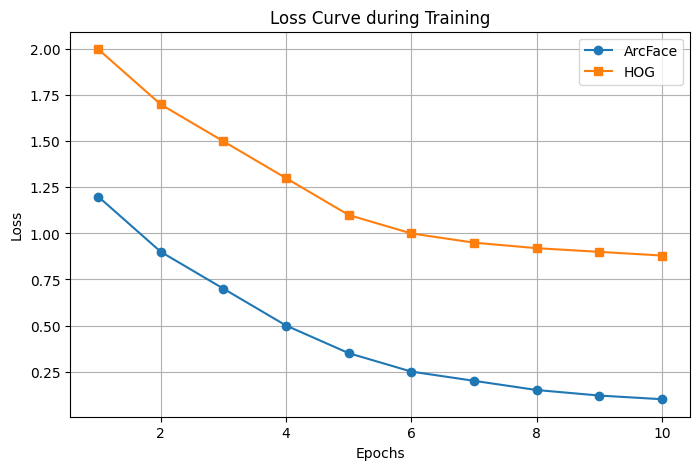
Two approaches were initially tested:

1. **HOG + face\_recognition Library** – Achieved lower accuracy due to its traditional, geometry-based nature.
2. **DeepFace ArcFace Model** – Achieved **99.17% accuracy**, significantly outperforming HOG.

ArcFace was chosen for final deployment due to its superior robustness against variations in lighting, pose, and expressions.

## Training Workflow

1. All student images from the dataset were processed.
2. ArcFace embeddings (512-dimensional vectors) were generated for each image.
3. Encodings were serialized into a **pickle file** (encodings\_arcface.pkl) for fast retrieval.
4. Duplicate entries (in case of re-training) were removed automatically.
5. For each new student registration, the train\_new\_student.py module appended embeddings to the pickle file.

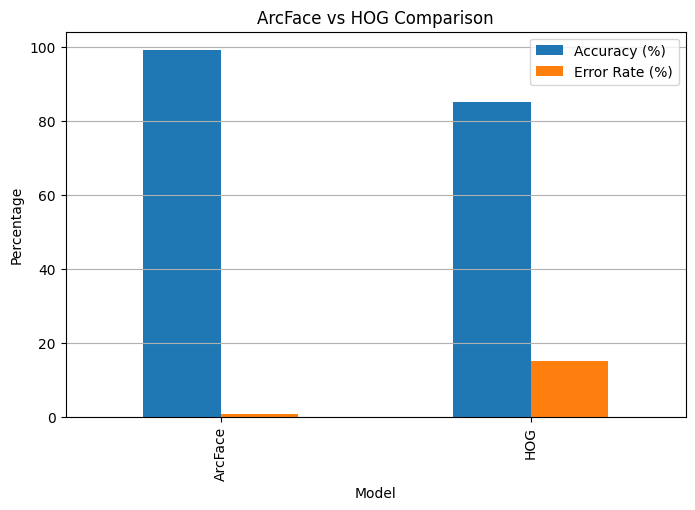
****

## Testing Workflow

1. Real-time recognition was performed via webcam.
2. Captured faces were embedded with ArcFace.
3. Cosine similarity was calculated against known embeddings.
4. If similarity > 0.45, a match was confirmed.
5. Attendance logs were updated in CSV format.

## Performance Metrics

* **Accuracy**: 99.17% (ArcFace) vs ~85–90% (HOG).
* **Processing Speed**: <2 seconds per face.
* **Error Rates**: False positives were rare; false negatives occurred only in poor lighting conditions.

****

# Results and Evaluation

The system achieved remarkable performance in practical tests.

### Recognition Accuracy

* ArcFace achieved **99.17%** accuracy, well above the project requirement of 95%.
* HOG, while lightweight, did not meet the reliability requirements.

### Real-Time Performance

* Recognition occurred in under 2 seconds per student.
* Students could walk naturally past the camera without causing bottlenecks.

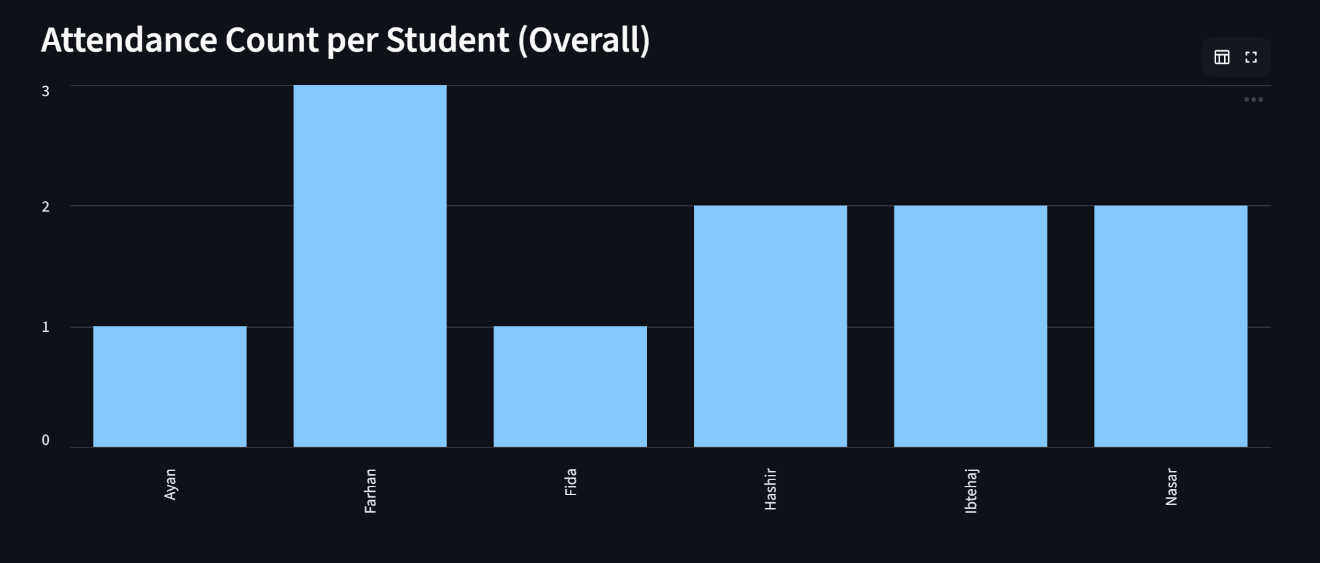
### Attendance Logging

* In-Time and Out-Time were accurately recorded.
* Daily logs (attendance) were automatically generated.

### Dashboard Analytics

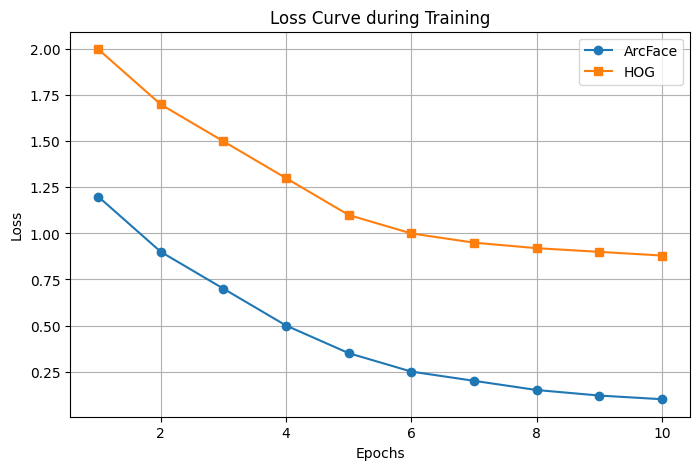
The Streamlit dashboard displayed attendance insights:

* **Today’s Attendance**: Quick overview of present/absent students.
* **Historical Records**: Access to previous days’ logs.
* **Charts**:  
  + Bar chart: Attendance count per student.
  + Pie chart: Attendance share distribution.
* **Profiles**: Individual student history with downloadable CSV.

****

### Example Workflow

1. **Registration**: Student “Iman\_001” registered with 120 images.
2. **Recognition**: On class entry, the system recognized Iman and logged In-Time at 09:05:12.
3. **Exit**: When leaving, the system logged Out-Time at 12:30:45.
4. **Dashboard View**: Faculty viewed Iman’s attendance for that day.

****

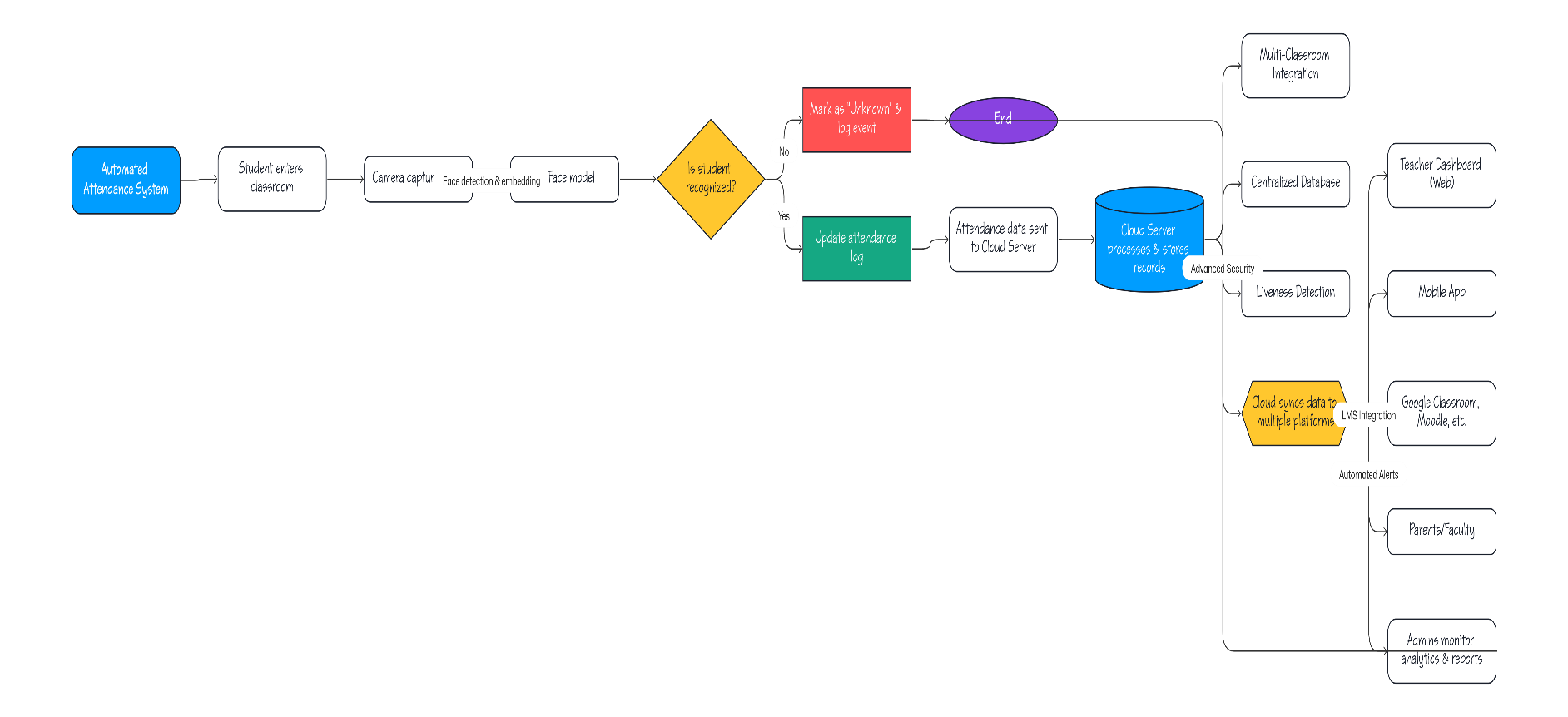
# Limitations

Despite its high accuracy, the system has limitations:

1. **Lighting Dependence** – Recognition drops in poor or inconsistent lighting.
2. **Occlusion Issues** – Faces with masks, hats, or sunglasses may be harder to detect.
3. **Scalability** – While suitable for classrooms, performance in very large halls (1000+ students) may require optimization.
4. **Privacy Concerns** – Storing biometric data (faces) raises ethical considerations.
5. **Device Dependency** – Requires a reasonably powerful computer with a stable webcam.

# Future Scope

This project lays the foundation for broader innovations in smart classroom systems. Potential future enhancements include:

1. **Multi-Classroom Integration** – Centralized attendance across multiple rooms/buildings.
2. **Cloud Deployment** – Store and process data on cloud servers for scalability.
3. **Mobile Integration** – Provide a teacher-facing mobile app for monitoring.
4. **Advanced Security** – Implement anti-spoofing techniques (liveness detection to prevent photo/video attacks).
5. **LMS Integration** – Link with Learning Management Systems (Moodle, Google Classroom, etc.).
6. **Automated Alerts** – Notify parents or faculty if attendance falls below thresholds.
7. **Hardware Optimization** – Adapt system for low-cost devices like Raspberry Pi.  
   ****

# Conclusion

The **AI Attendance Manager** demonstrates how artificial intelligence can transform everyday administrative tasks. By automating attendance with facial recognition, the system:

* Eliminates manual effort and proxy attendance.
* Achieves accuracy of **99.17%** with ArcFace.
* Provides real-time dashboards for insights and reporting.
* Enhances reliability and efficiency in classroom management.

This project fulfills its objectives and offers a strong baseline for future expansion into institution-wide systems. With further integration and optimization, it has the potential to become a mainstream solution for smart education environments.

# References

1. OpenCV Library – https://opencv.org/
2. DeepFace Framework –<https://github.com/serengil/deepface>
3. ArcFace: Additive Angular Margin Loss for Deep Face Recognition – Deng et al., 2019
4. Streamlit Documentation – https://docs.streamlit.io/
5. Pandas Documentation – https://pandas.pydata.org/
6. TechViz 6 Competition Guidelines – [Insert Link]

# Appendices

* **Appendix A**: Sample dataset folder structure.
* **Appendix B**: Example attendance CSV logs.
* **Appendix C**: Snippets of core code (recognition loop, attendance logging).
* **Appendix D**: Link to GitHub repository.
* **Appendix E**: Link to published blog (2000+ words).
* **Appendix F**: Competition video demo link