**Efficient Attendance Monitoring Using**

**AI-Based Face Detection**

***Nicholas Marshall, Nivedita Prabhu, Frank Diabour, Matthew Willingham, Ayodeji Adeogun***

***Computer Science Students   
Texas Tech University   
Lubbock, United States***

Abstract— This project presents the design and implementation of an AI-based attendance system that leverages facial recognition technology to streamline and secure the attendance process. Traditional attendance systems are often prone to inefficiencies, errors, and manipulation. By integrating machine learning techniques and computer vision, the proposed system captures and recognizes facial features in real-time to accurately record student attendance. The system architecture includes image preprocessing, face detection using Haar cascades, and face recognition through Local Binary Patterns Histograms (LBPH). Developed with Python and OpenCV, the system demonstrates high accuracy in identifying registered individuals and is scalable for deployment in educational institutions. This work highlights the effectiveness of AI-driven solutions in improving administrative efficiency and ensuring data integrity.

**Keywords*:* AI-based attendance system, facial recognition, machine learning, computer vision, Haar cascades, Local Binary Patterns Histograms (LBPH), Python, OpenCV, educational institutions, data integrity.**

# ***Introduction***

In educational institutions and corporate environments, attendance tracking remains a critical administrative task. Traditional methods such as manual roll calls and sign-in sheets are inefficient, error-prone, and susceptible to manipulation. With the growing availability and accuracy of artificial intelligence technologies, particularly facial recognition, there is an opportunity to automate attendance in a way that is both efficient and secure.

This project proposes the development of an AI-based attendance system using facial recognition to automate and enhance the reliability of attendance tracking. Leveraging Python and OpenCV, the system uses computer vision techniques to detect and recognize faces in real-time. While the solution offers improved efficiency and accuracy, it also presents challenges such as handling varied lighting conditions, facial occlusions, and addressing privacy concerns. This project explores the design, implementation, and performance of such a system, aiming for a robust solution suitable for real-world use.

# ***Motivation***

Manual attendance methods are time-consuming and susceptible to inaccuracies, including proxy attendance and human error. An AI-based attendance system eliminates the need for traditional roll calls or sign-in sheets by automating the process through facial recognition, thereby enhancing efficiency and integrity. This ensures that only physically present students are marked, saving valuable class time and allowing instructors to focus on teaching.

However, such systems face challenges, particularly in maintaining high accuracy under varied conditions—such as poor lighting, facial occlusion (e.g., glasses, masks), or lookalike individuals. These issues can lead to misidentification, but they can be mitigated by training the model with diverse datasets, capturing multiple facial angles, and using confidence thresholds to reduce error margins.

Security and privacy are also significant concerns. Facial data must be handled ethically and securely, which can be achieved by encrypting sensitive information, limiting the storage of biometric data to essential embeddings, and ensuring student consent before data collection.

This system is highly applicable in environments such as large lecture halls, training programs, and workplaces where efficient and reliable attendance tracking is essential. The impact of deploying such technology is considerable—it reduces administrative overhead, ensures fair and tamper-proof attendance records, and improves the overall educational experience for both instructors and students. Additionally, seamless integration with backend databases and user interfaces, combined with real-time processing through optimized AI models like YOLOv8, ensures performance and scalability in real-world use cases.

# ***Research***

### **Overview**: Using the technology and implementing this system will be necessary to make this project a success. We will further develop: Data Collection and UI help, Model Training and Image Augmentation, UML Diagrams and Descriptions, Database System [Store Attendance], and System Integration [Model + Schema + UI]

## **Model Training and Image Augmentation**

For this project, we did not conduct custom model training. Instead, we used **YOLOv8n-face**, a lightweight, pre-trained object detection model optimized for face detection. This allowed for a faster and more efficient implementation without the need for computationally expensive training.

While YOLOv8n-face handled the initial detection, we integrated **DeepFace** for facial recognition. The workflow was as follows:

1. Use YOLOv8n-face to detect and crop faces from each video frame.
2. Pass the cropped face to DeepFace for identity verification.

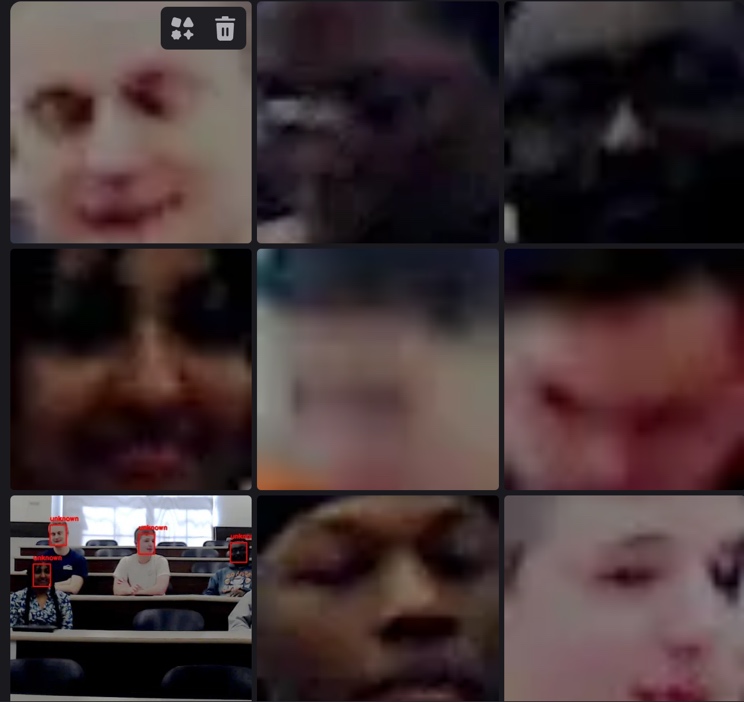
During testing, we encountered a significant difference in performance between webcam and smartphone inputs. The **webcam-generated face crops were of poor quality**, often lacking sufficient detail for recognition (see **Figure 1**). These low-resolution "blob head" crops failed to retain defining facial features, resulting in low DeepFace accuracy.

To mitigate this, we switched to using a **smartphone camera**, which produced much clearer and more detailed face crops (see **Figure 2**). With higher-resolution inputs, DeepFace successfully recognized individuals with high accuracy.

The system was tested with a small group database. DeepFace correctly:

* Identified known individuals and marked them as “Present”.
* Rejected individuals not in the database with the message “Does not exist” (see **Figure 3**).

No training or data augmentation was needed, as both detection and recognition relied on robust, pre-trained models.

**Figure 1**: Low-quality “blob head” crops from webcam detection.

**Figure 2**: High-quality face crop from smartphone input.

**Figure 3**: Final system output showing recognition results and attendance marking.

## **Data Collection and help with UI**

Data collection involved capturing **1–3 clear images** per individual using a smartphone. These reference images were stored locally to serve as the face database for DeepFace recognition.

**To simulate real-world use, we created a live UI where:**

* YOLOv8n-face detects faces in real-time.
* Cropped faces are passed to DeepFace.
* The system returns and displays the person’s name and status.

**The UI displays consist of:**

* A bounding box and label around each detected face.
* Text output indicating if the person is “Present” or “Does not exist”.
* A group of men crossing a street

  AI-generated content may be incorrect.A real-time log of attendees.

**Figure 4:** A previous example of a group of men with the sensor picking up what they are and their position.

These visual cues helped validate system performance and ensure that only enrolled individuals could be marked as present.

The switch to a smartphone camera played a crucial role in making the system practical, showing the importance of high-quality input in computer vision pipelines.

## **UML Diagrams and Descriptions**

To demonstrate our Attendance System, we created 4 additional diagrams: Class Diagram, Use Case Diagram, ER Diagram, and Sequence Diagram.

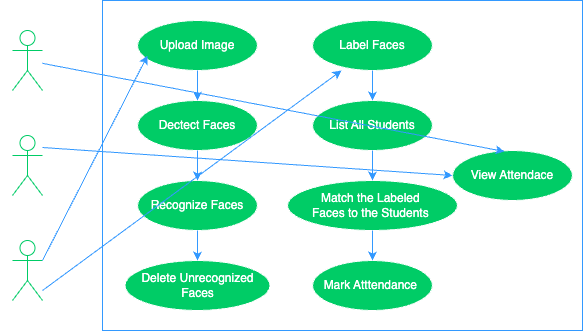
**Class Diagram**: The YOLO-based face recognition system is structured around core classes that facilitate its functionality. Central to this system is the FaceRecognitionSystem, which oversees the detection process through the YOLOModel and manages the recognition aspect with the DeepFaceRecognizer. Complementing these functions is the ImageProcessor, which plays a crucial role in preprocessing tasks, such as cropping detected faces to ensure accurate recognition. This architecture highlights the associations within the system, demonstrating how the detection and recognition modules interact seamlessly to deliver effective face recognition capabilities. In the future, we want to show how this connects to the MySQL Database and HTML website.

A screenshot of a computer program

AI-generated content may be incorrect.

**Figure 5:** Class Diagram explaining the YOLOModel and DeepFaceRecognizer.

**Use Case Diagram:** The face recognition system allows users to interact seamlessly through a variety of use cases. Initially, a user can "Upload Image" to the system, which then proceeds to "Detect Faces" within the uploaded image. Following this, the system can "Recognize Faces" and "Label Faces" accordingly. Teachers are empowered to take photos of the class or individual students, enabling them to "List All Students" and subsequently "Match the Labeled Faces to the Students" for accurate identification. Attendance can be efficiently managed as teachers "Mark Attendance" and can "View Attendance," making necessary modifications if attendance has not been marked correctly. Students also can view their attendance records. Meanwhile, developers have the functionality to take photos, label them, and mark, modify, and view attendance, ensuring a comprehensive and organized system for all users involved.



**Figure 6:** Use Case Diagram showing how the “actors” connect to what they’re able to use and access.

**ER Diagram:** The database design comprises three essential models: Students, Courses, and Attendance records. In this structure, a one-to-many (1:N) relationship is established between Students, identified by the r\_number, and Attendance, which also utilizes the r\_number to track student attendance. Similarly, a one-to-many (1:N) relationship exists between Courses, denoted by course\_id, and Attendance, where course\_id associates attendance records with specific courses. This design ensures database consistency by implementing Primary Keys (PK) and Foreign Keys (FK), thereby maintaining the integrity and accessibility of the data across these interrelated models.

**Figure 7:** ER diagram showing the parts of the database and how it connects to its entities.

**Sequence Diagram:** The website facilitates user interaction by enabling the uploading of pictures for attendance and the viewing of attendance records. It incorporates a MySQL database to store and manage attendance data, which is utilized for both retrieval and updates. An AI model, equipped with a pre-trained system, is employed to verify attendance from the uploaded pictures.

The AI model is developed using training data, which is subsequently deleted following the training process to ensure data integrity. This integrated attendance workflow allows seamless interaction between the website, the AI model, and the attendance database, thereby streamlining the management and verification of attendance records.

A diagram of a computer program

AI-generated content may be incorrect.

**Figure 8:** Sequence Diagram showing the flow of the system.

The presented diagrams illustrate the architecture and workflow of the AI-based attendance system, showcasing how various components—such as face detection, recognition modules, and database integration—interact seamlessly. These visuals help clarify the system’s functionality, from capturing real-time input to recording accurate attendance, reinforcing the efficiency and reliability of the proposed solution.

## A group of green ovals with blue text AI-generated content may be incorrect.**Database System [Store Attendance]**

In the initial phase of Stage 1, the data was consolidated into a single set of tables and columns; however, it became necessary to separate this information into distinct categories for improved clarity. To facilitate easier coding, I have opted to utilize a True/False format instead of a checkbox, as this aligns more accurately with the boolean variable requirements.

Additionally, We are in the process of modifying certain values to ensure they are more representative of the data at hand. Given the current lack of access to a comprehensive database encompassing all individuals in this class, We are solely relying on our group information for the data in our database.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 9:** New Database Mockup in Excel.

` We created a database named RECORDS and structured it to manage student attendance. The schema includes three main tables: Students, which stores student details with a unique roll number (r\_number); Courses, which holds course information along with the instructor’s details; and Attendance, which tracks student attendance using foreign keys linking to the Students and Courses tables. The Attendance table uses an AUTO\_INCREMENT primary key and includes fields for attendance status, date, and time. The database ensures data integrity with constraints like PRIMARY KEY, UNIQUE, and FOREIGN KEY relationships.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 10:** How to create the database and list of the Tables.

A blue screen with white text

AI-generated content may be incorrect.The Courses table stores course details, with course\_id as the primary key and instructor\_id uniquely identifying each instructor. The table ensures that every course has a name and an assigned instructor. An INSERT statement adds a sample course, "Senior Capstone Project," taught by Victor Sheng. The SELECT queries display the table's contents, labeling the output for clarity.

**Figure 11:** The Courses Table, inserting data, and setting up the output.

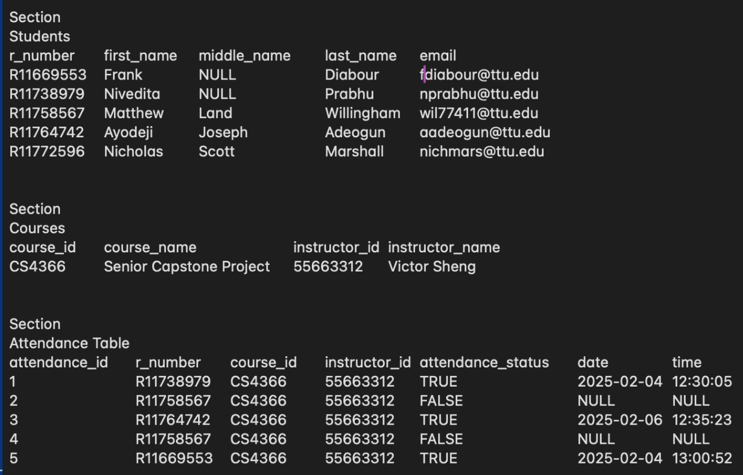
The Attendance table tracks student attendance, using attendance\_id as an AUTO\_INCREMENT primary key. It references the Students and Courses tables via foreign keys (r\_number, course\_id, and instructor\_id) to maintain data integrity. The attendance\_status field is a boolean, defaulting to FALSE, with date and time fields for recording attendance details. An INSERT statement logs an attendance entry, converting date and time strings into the appropriate format. The SELECT query retrieves attendance records, formatting student and course IDs with prefixes and displaying attendance\_status as TRUE or FALSE for clarity.

A screenshot of a computer program

AI-generated content may be incorrect.

**Figure 12:** The Attendance Table, inserting data, and setting up the output.

Both tables enforce data integrity using primary and foreign keys, but while Courses assigns each course a unique instructor, Attendance links students to courses and tracks their presence. Attendance also incorporates automatic ID generation (AUTO\_INCREMENT) and boolean status for attendance tracking. Unlike Courses, which primarily stores static course details, Attendance dynamically records daily participation with timestamps and formatted outputs for readability.

The output successfully displays structured student, course, and attendance data, reinforcing the database's relational integrity. The Students table correctly associates unique student records with emails, while the Courses table links each course to an instructor. Overall, the database provides a well-organized framework for tracking attendance but may require validation to ensure complete and accurate records.

**Figure 13:** Output of the Database, showing accurate information and proper formatting.

## **System Integration [Model + Schema + UI]**

The user interface (UI) plays a critical role in bridging the functionality between the front end and the back end of the web-based attendance system. It is designed not only to present data retrieved from a MySQL database but also to allow user input, which is transmitted back to the backend for processing and storage. Implemented using HTML and CSS, and rendered via Flask templates, the UI delivers an intuitive and responsive experience tailored for educational administrators.

On the homepage, the UI displays a list of students retrieved from the Students table, allowing the user to click on any name to view individual attendance records. This functionality is dynamically handled using route-based Flask logic, which pulls data from the database and injects it into the homepage.html template using Jinja2 templating syntax.

The interface also includes a dedicated form for marking attendance. Through this form, instructors can submit data including student ID, course ID, date, time, and attendance status. These inputs are handled by a POST request and passed to the backend, where they are validated and stored in the Attendance table.

A screenshot of a computer

AI-generated content may be incorrect.A standout feature of the UI is its support for image uploads, enabling visual identification for attendance verification. Users can select and preview images on the same page before submitting. A JavaScript event listener enhances interactivity by revealing a preview of the uploaded image and the submission button only after a file is chosen The UI instantly reflects changes to the system, such as new attendance records, providing immediate visual feedback without requiring a page refresh. This is achieved through tight integration with Flask’s routing and MySQL queries, making the system both responsive and efficient.

**A screenshot of a computer

AI-generated content may be incorrect.Figure 14:** First Prototype of the Homepage.

A screenshot of a computer

AI-generated content may be incorrect.**Figure 15:** Second Prototype of the Homepage.

**Figure 16:** Third and Current Prototype of the Homepage.

The most challenging aspect of successful image capturing will be suitable equipment that allows for high enough quality images to be graded. Some current solutions to the problem might be grading based on multiple frames (capturing a video) or simply taking multiple pictures of closeups. The website might be chosen for this purpose, or it might be moved to the backend.

# ***Summary***

The document outlines the development of an AI-based attendance system utilizing facial recognition to enhance efficiency and security in attendance tracking. It addresses the limitations of traditional methods by implementing machine learning and computer vision techniques for real-time facial feature recognition. The system's architecture encompasses image preprocessing, face detection with Haar cascades, and face recognition using Local Binary Patterns Histograms (LBPH). Developed with Python and OpenCV, the system achieves high accuracy in identifying registered individuals and offers scalability for deployment in educational institutions. This work demonstrates the potential of AI-driven solutions in improving administrative processes and ensuring data integrity.

References:

[1] U. Essien and Godwin Ansa, “Attendance Based Systems for Face Recognition using a Camera Rail Approach,” vol. 3, no. 1, pp. 14–28, Oct. 2024, Available: <https://www.researchgate.net/publication/384765708_Attendance_Based_Systems_for_Face_Recognition_using_a_Camera_Rail_Approach>

[2] Atlassian, “Importing Excel Data into MySQL | Atlassian,” Atlassian, 2025. <https://www.atlassian.com/data/databases/excel-to-mysql>

[3] D. DSouza, “Setting Up Flask MySQL Integration: 4 Easy Steps | Hevo,” hevodata, Apr. 28, 2023. <https://hevodata.com/learn/flask-mysql>

[4] R. Kothari, D. Joshi, and M. Shah, "Real-Time Face Recognition System Using Deep Learning," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, 2020.

[5] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, 2016, doi: [10.1109/LSP.2016.2603342](https://doi.org/10.1109/LSP.2016.2603342).

[6] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2015. [Online]. Available: <https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/>

[7] S. Serengil and A. Ozpinar, "DeepFace: Lightweight Face Recognition and Facial Attribute Analysis Framework for Python," GitHub, 2020. [Online]. Available: <https://github.com/serengil/deepface>

[8] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, doi: [10.1109/CVPR.2014.220](https://doi.org/10.1109/CVPR.2014.220).

[9] Ultralytics, "YOLOv8 Documentation," 2023. [Online]. Available: <https://docs.ultralytics.com>