**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.

# ***Impacts***

The application of an AI-based attendance system is significant both locally and globally. Locally, it increases efficiency by automating the manual process of recording attendance, doing away with human error, and promoting accountability among students and employees alike. It also optimizes institutional resource use, simplifies communication via automatic notifications, and provides independence to faculty members in maintaining attendance records on their own.

Worldwide, this system promotes the modernization of educational and corporate organizations, enables the generation of insightful data, and provides scalability for future applications in areas like healthcare, conferences, and government services. It also introduces to the forefront important discussions on privacy, ethical AI applications, and data security, contributing to the global dialogue on responsible technology deployment. As institutions adopt these systems, they not only enhance operational efficiency but also foster economic development and facilitate improved AI literacy across different segments of the population.

# ***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.

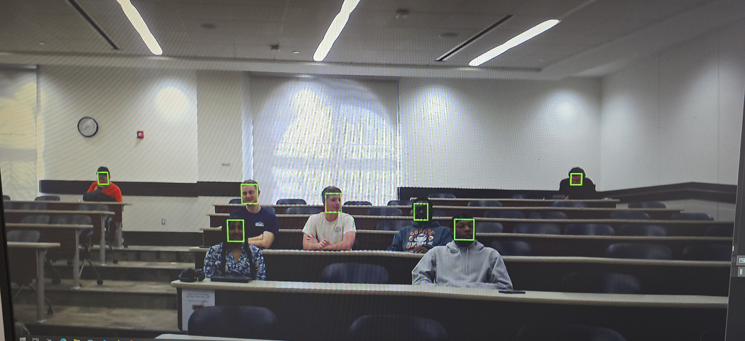
During testing, we encountered a significant difference in performance between webcam and smartphone inputs. The **webcam-generated face crops were of poor quality**, often lacking sufficient detail for recognition (see **Figure 1**). These low-resolution "blob head" crops failed to retain defining facial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

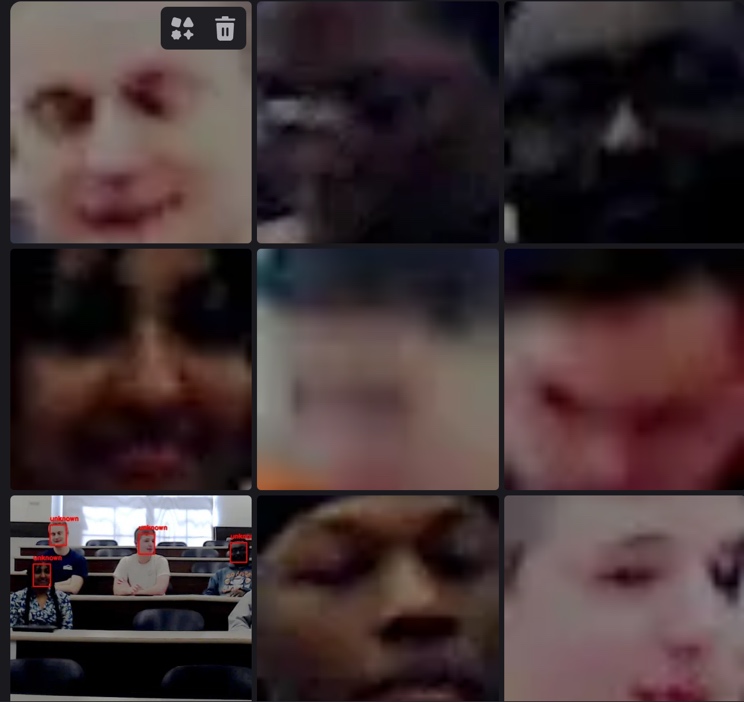
To mitigate this, we switched to using a **smartphone camera**, which produced much clearer and more detailed face crops (see **Figure 3**). With higher-resolution inputs, DeepFace successfully recognized individuals with high accuracy.

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 4**).

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

**Figure 1**: Initial Scanning with limited Data as of Stage 1

**Figure 2**: Low-quality “blob head” crops from webcam detection as of Stage 2.

**Figure 3**: High-quality face crop from smartphone input as of Stage 2.

**Figure 4**: Final system output showing recognition results and attendance marking as of stage 2.

## **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 5:** 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**

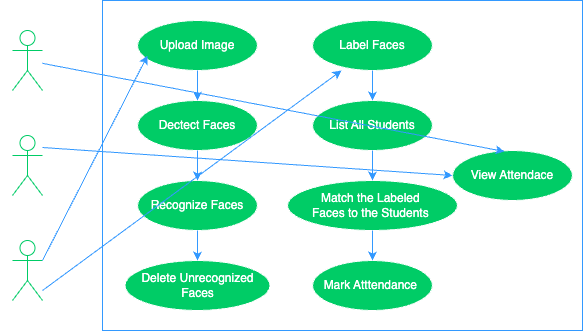
To demonstrate our Attendance System, we created 4 additional diagrams: Class Diagram, Use Case Diagram, ER Diagram, and Sequence Diagram.

**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.

A screenshot of a computer program

AI-generated content may be incorrect.

**Figure 6:** 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 7:** 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 8:** 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 9:** 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.

## **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.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 10:** Database Mockup in Excel for Stage 1.

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. Also we used Flask as it allows for communication with MySQL database. Flask will also enable attendance checker to run the actual face detection scripts which will then update database

A screenshot of a computer

AI-generated content may be incorrect.` 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.

**Figure 11:** How to create the database and list of the Tables.

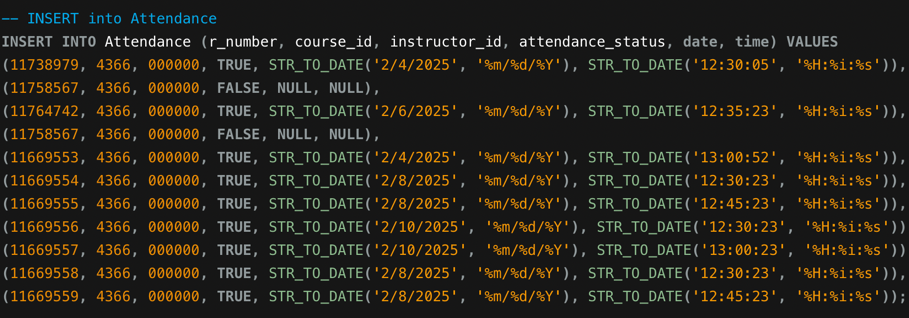
The Courses table stores course details, with course\_id as the primary key and instructor\_id uniquely identifying each instructor. The table ensures that every course has a name and an assigned instructor. An INSERT statement adds a sample course, "Senior Capstone Project," taught by Victor Sheng.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

A blue screen with white text

AI-generated content may be incorrect.

**Figure 12:** 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.

****

**Figure 13:** 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 a boolean status for attendance tracking. Unlike Courses, which primarily store static course details, Attendance dynamically records daily participation with timestamps and formatted outputs for readability.

A screenshot of a computer program

AI-generated content may be incorrect.For stage 3, we added filler data for potential students, as we don’t have their data, as it’s private information. They have first, middle, and last name, R# number, Accurate Instructor and Teaching Assistant Information, and Attendance record.

A screen shot of a computer screen

AI-generated content may be incorrect.

**Figure 13:** Accurate andFiler Data of Students

**A screen shot of a computer

AI-generated content may be incorrect.**

**Figure 14:** Accurate andFiler Data of Courses

**Figure 15:** Accurate andFiler Data of Attendance

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.

A black screen with white text

AI-generated content may be incorrect.

**Figure 16:** Output of the Database, showing accurate information and proper formatting.

## **AI Attendance code, Local Host code, and Image Database.**

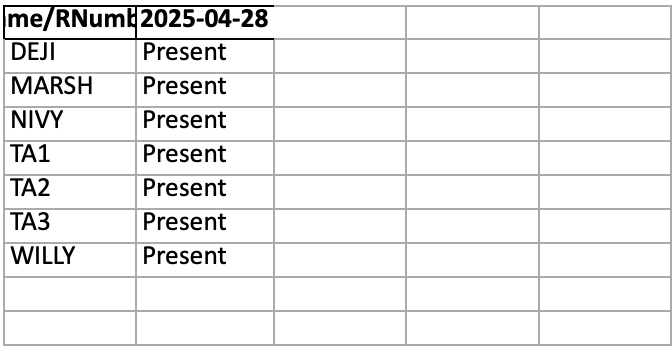
The AI-powered attendance tracking system is designed to streamline and automate the process of recording student attendance using facial recognition. By integrating state-of-the-art computer vision models with a web-based interface, the system eliminates manual entry, reduces errors, and enhances administrative efficiency. It supports both localhost and online implementations, offering real-time recognition, record management, and visual verification features that make it suitable for academic institutions seeking scalable and reliable attendance solutions.

At the core of the system is a dual-model approach combining **YOLOv8** for real-time face detection and **DeepFace** for accurate facial recognition using the **Facenet** model. When an image is uploaded, YOLOv8 detects and crops individual faces, which are then compared against a pre-stored database of student images. A confidence threshold of 0.6 ensures that only reliable matches are recorded, effectively managing challenges such as variable lighting, facial angles, or occlusions. Attendance results are dynamically updated and logged in an Excel sheet for export, providing a clear, administrative-ready record.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Figure 17:** Yolov8 and Deep Face Integration

**Figure 18:** Image Database

**Figure 19:** Excel Sheet of Attendance Results

A screenshot of a computer

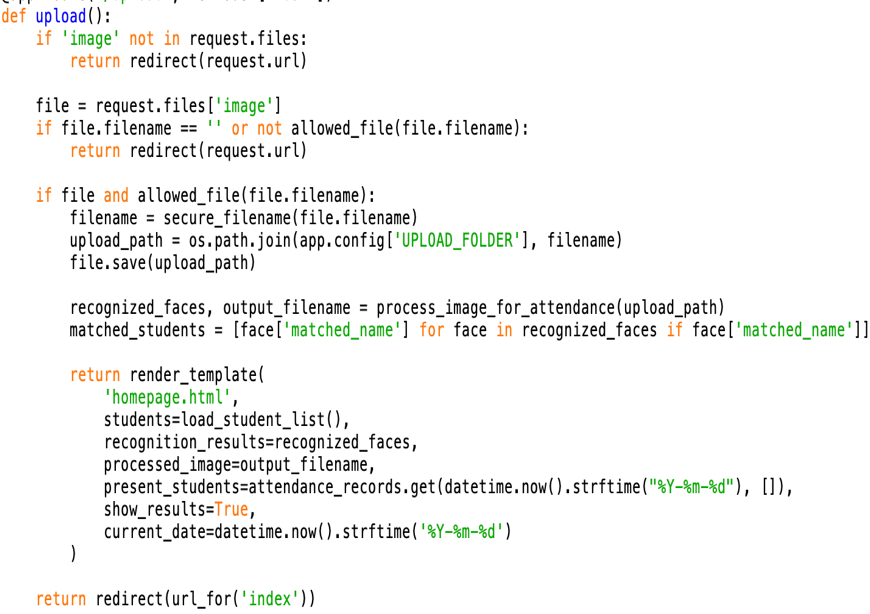
AI-generated content may be incorrect.The system supports both a public-facing website and a local server (localhost) version, each tailored for different use cases. The **website version** features an organized layout with tools for uploading new images, reviewing current attendance, and referencing past records, making it ideal for tracking long-term attendance histories. The **localhost demo**, by contrast, emphasizes rapid recognition and immediate feedback, displaying processed images alongside real-time attendance statistics with options to submit or export data. This makes it optimal for real-time classroom use.

**Figure 20:** Localhost Demo

****Both implementations rely on a temporary but actively growing dataset of student images, which are stored locally in designated folders such as uploads, database, and recognized\_faces. Any new images processed are added to this dataset, allowing the system to evolve. Unrecognized faces are stored in a separate unrecognized\_faces folder, enabling administrators to review and update the database as needed. This structure ensures the system maintains high accuracy while remaining adaptable and easy to maintain.

**Figure 21:** recognized\_faces Folder****

**Figure 22:** unrecognized\_faces Folder



**Figure 23**: code to add the unrecognized faces to the database and allow detection

By combining advanced AI models with practical backend tools like Flask, OpenCV, and pandas, the attendance system delivers a responsive, secure, and intelligent solution for modern classrooms. Whether used through the website for comprehensive tracking or the localhost for on-the-spot recognition, the system ensures reliable data capture, privacy compliance, and operational ease, laying a strong foundation for future enhancements such as student behavior analytics or integration with institutional management systems.

## **Visual Website**

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 (refer to **Figure 27**).

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 (refer to **Figure 27**).

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 (refer to **Figure 27**).

A screenshot of a computer

AI-generated content may be incorrect.The entire HTML website was hosted on **PythonAnywhere**, a cloud-based platform that provided the necessary infrastructure to deploy the application publicly. PythonAnywhere handled everything from storing HTML files, managing the Python virtual environment, executing Flask backend logic, and maintaining MySQL database access. The platform’s server-side capabilities, including an allocation of 4 GB RAM, were sufficient to run the attendance system smoothly, even during resource-intensive tasks like image processing and database transactions. This hosting solution made it easy to deploy, test, and showcase the full functionality of the project online.

**Figure 24:** First Prototype of the Website.

**A screenshot of a computer

AI-generated content may be incorrect.**The initial prototype of this attendance system was developed with the primary goal of integrating the frontend interface with the backend and establishing a functional connection to the database. This version focused on ensuring that user interactions—such as uploading attendance images, selecting students, and submitting records—could be effectively processed and reflected in the database through Flask routes and SQL queries. The frontend was intentionally kept simple but structured, using HTML, CSS, and embedded JavaScript to provide dynamic behaviors while allowing data retrieved from the backend (via Jinja templating) to populate the interface in real time. This foundational version served as a critical step in validating the core functionality and data flow between the user interface, server logic, and database storage before adding more advanced features or visual polish.

**Figure 25:** Second Prototype of the Website.

The second version of the attendance system was focused on scaling the application to handle larger datasets more efficiently while also improving the visual structure and layout of the HTML. Enhancements were made to ensure that more student records and attendance data could be processed and displayed without compromising performance or readability. This involved optimizing how student lists and attendance tables were rendered, including adding scrollable areas, limiting viewport height, and applying consistent sizing and spacing to elements. The interface was also visually refined with a more balanced color scheme, better font usage, and clearer separation of functional sections like the sidebars and main content. These refinements not only improved the overall user experience but also laid the groundwork for a cleaner, more scalable system capable of supporting real-world classroom or institutional use cases.

The overall look of the second prototype remained the same, but its functionality largely increased because of many of the back-end and some of the front-end changes. This prototype is also where the connection from the website to the backend AI model took place and proved to be the most challenging.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 26:** Third Prototype of the Website.

The third version of the attendance system was the finalized build used for the project presentation, and it introduced several crucial improvements aimed at both reliability and user experience. One of the key fixes addressed an intermittent issue with the database connection—if the initial attempt to connect failed, the system was modified to automatically retry and re-establish the connection, greatly enhancing stability. Additionally, user feedback was significantly improved through the introduction of a processed image display that visually confirmed attendance recognition results: faces found in the database were boxed in green, while unknown or unmatched faces were highlighted in red. This gave immediate, intuitive feedback on which students were successfully identified. Furthermore, the image processing pipeline was optimized for speed, reducing the time between submission and result, which was critical for demonstrating real-time functionality during the live presentation. These enhancements made the third version not only more robust and visually informative but also much more responsive and presentation-ready.

A group of people in a classroom

AI-generated content may be incorrect.

**Figure 27:** Current and Future Prototype of the Website.

Future prototypes of the attendance system could incorporate several enhancements to further improve usability and scalability. One potential upgrade would be the addition of a loading bar or visual indicator during image uploads and processing, providing users with a clearer, more engaging experience while waiting for results—particularly useful for larger images or slower connections. Another significant improvement could be the introduction of an instructor login system, which would allow for managing multiple classes under separate user profiles. This change would require a more sophisticated database design to associate students, attendance records, and image data with specific instructors and course sections. Additionally, a student-facing portal could be developed to let students track their own attendance history, offering transparency and encouraging accountability. While these features would make the system more comprehensive and closer to production-level readiness, they were not essential for the scope of this project. Most of these enhancements are more about building on existing infrastructure than solving novel problems—they're relatively straightforward to implement but would require careful database restructuring and UI adjustments, making them more tedious than technically challenging.

## **Localhost Demo [HTML]**

Like the Website, the Localhost bridges the functionality between the front end and the back end of the web-based attendance system. The application displays data retrieved from a MySQL database as well as allows user input, which is sent back to the backend for processing. This intuitive and responsive UI is developed using HTML and CSS, and rendered using Flask templates.

On the localhost demo, the UI displays real-time attendance results after processing an uploaded classroom image. It shows a list of present students on the left panel, pulling data dynamically based on face recognition outcomes. Each student's ID is listed with their attendance status ("Present"), while the right panel summarizes the overall attendance statistics, including the total number of students present and absent. Users can manually submit new attendance records or export the full attendance list with dedicated buttons. At the center, the interface presents the processed image with detected faces highlighted, offering clear visual confirmation of recognized students. This functionality is powered by Flask routing and backend logic, which handles image processing, face recognition, and data updates efficiently. The integration of Jinja2 templating ensures dynamic updates without requiring page reloads, providing a seamless and responsive user experience.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 27**: Localhost Demo

A group of people sitting in a lecture hall

AI-generated content may be incorrect.

**Figure 28:** Zoom-in of Detected Photo

The localhost demo focuses heavily on real-time recognition results, presenting a streamlined and modern interface where present and absent students are listed clearly alongside an immediate visual output of the processed classroom image. It emphasizes quick access to actions like submitting manual attendance or exporting full records, making it highly efficient for active session management. In contrast, the website version offers a more detailed and structured layout, with sections for both past attendance records and current attendance results. It provides a historical view, allowing users to review previous attendance alongside the current session. The website also includes the ability to upload and process new images, but places greater emphasis on record-keeping and retrieval. While both versions are built around image-based attendance verification, the localhost version is optimized for speed and immediacy, whereas the website adds a layer of depth and archival functionality for broader attendance tracking over time.

In conclusion, both the localhost and website interfaces serve important yet distinct roles within the AI-based attendance system. The localhost interface is ideal for real-time classroom environments where speed and quick feedback are crucial. Meanwhile, the website complements this by offering extended functionality for managing and reviewing historical attendance data. Together, they form a comprehensive solution that addresses both the immediate and long-term needs of automated attendance tracking through intuitive design and efficient backend integration.

# ***Possible Updates***

To make the system more efficient and scalable, the database should be periodically updated by adding filler or placeholder data, ensuring it reflects the latest enrollment and class rosters. When a new student is added, their data—such as name, student ID, and course enrollment—can be seamlessly integrated into the system by importing it from an official list, such as a CSV file exported from institutional records. This approach minimizes manual entry errors and ensures consistency across sessions.

In addition, implementing an automated email notification system would enhance communication between the system and its users. This feature would allow students to receive real-time updates regarding their attendance status, and administrators or instructors could be alerted if irregularities are detected, such as frequent absences or face mismatches. These notifications not only foster accountability but also help maintain transparency and trust in the system.

Another critical feature would be the ability to grant professors access to manually update attendance records through the web interface. This is particularly important in cases where the system fails to recognize a student due to obstructions or lighting issues in the uploaded image. Manual adjustments ensure that such discrepancies can be resolved promptly without compromising the integrity of the attendance logs.

Furthermore, introducing a visual indicator such as a loading bar during image processing would significantly improve the user experience. It provides feedback to users while they wait, reassuring them that the system is actively working, which is especially important during real-time classroom usage.

Collectively, these enhancements—automated data integration, real-time notifications, professor access for manual updates, and responsive UI elements—will make the attendance system more dynamic, accurate, and user-friendly. They will also contribute to its adoption in varied academic settings by addressing both the technical and practical needs of instructors and students alike.

# ***Conclusion***

In summary, this stage of the AI-based attendance system effectively demonstrates the capability of facial recognition technology to identify individuals from a dataset and match them in real time, even when presented with angled or slightly varied images. Through careful testing, we observed that the system performs well under different lighting conditions and facial orientations, confirming its robustness and reliability. This not only validates the underlying YOLOv8 model and our face-matching approach but also sets the foundation for more advanced features in future stages, such as attendance logging, multi-face tracking, and real-time database integration.

Moreover, this stage highlights the potential of AI to streamline routine processes like attendance management, offering institutions a faster, more secure, and contactless alternative to traditional methods. The flexibility and efficiency showcased here demonstrate the system’s potential for scalability across different environments such as schools, workplaces, or events. As we move toward integrating the MySQL database and enhancing the UI for user-friendliness, the insights gained from this testing phase will guide us in refining both the technical and functional aspects of the system. Ultimately, this project aims to not only automate attendance but also contribute to the broader adoption of ethical, privacy-conscious AI in everyday applications.

**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>