# PREZENSE – ATTENDANCE MANAGEMENT SYSTEM

***A REPORT/Project submitted***
***in partial fulfilment for the Degree of***
**Bachelor of Technology**
**in**
**Computer & Communication Engineering**

***by***

**VIDIT ROY**


**Department of Computer and Communication Engineering**
**SCHOOL OF COMPUTING AND INTELLIGENT SYSTEMS**
**MANIPAL UNIVERSITY JAIPUR**
**APRIL 2025**

# CERTIFICATE

This is to certify that the REPORT entitled PREZENSE – Attendance Maintenance System submitted by Vidit Roy to the Manipal University Jaipur, in partial fulfilment for the requirement of the degree of Bachelor of Technology in Computer and Communication Engineering is a *bona fide* record of REPORT/project work carried out by him/her under my/our supervision. The contents of this REPORT/project, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

[Signature] [Signature]

[Name of the supervisor] [Name of the co-supervisor]
Supervisor Co-Supervisor
Department of Computer and Communication Engineering

Place: Manipal University Jaipur
April 2025

# DECLARATION

I certify that

a. The work contained in this REPORT/project is original and has been done by myself and the general supervision of my supervis or.
b. The work has not been submitted to any other institute for any d egree or diploma.
c. Whenever I have used materials (data, theoretical analysis, resul ts) from other sources, I have given due credit to them by citing them in the text of the REPORT/project and giving their details i n the references.
d. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the referen ces.

Place: Manipal University Jaipur
Date: April 2025

Vidit Roy
229303056

# ACKNOWLEDGMENTS

I avail myself of this opportunity to thank my project guide, Ms. Vaishali Yadav, sincerely for her unwavering support, thought-provoking guidance, and constructive feedback throughout the duration of this project.

I also thank my peers for their cooperation and assistance in making this report. Their inputs and recommendations have been worth their weight in gold.

In addition, I would like to express my deepest gratitude to the staff of the Department of Computer and Communication Engineering at Manipal University Jaipur for giving the required facilities, advice, and motivation, which were indispensable in the accomplishment of this project and the fulfilment of the guidelines of the report/project format.

Vidit Roy

# ABSTRACT

This project introduces Prezense, a real-time attendance management system that utilizes facial recognition technology to make the attendance process more efficient and secure in academic environments. Conventional attendance systems, like rollcall and manual logbooks, tend to be inefficient, error-prone, and open to manipulation. These systems are easily circumvented by proxies or due to human mistakes, making them unreliable. Prezense overcomes these limitations by offering an automated system that is both secure and scalable.

The system is complemented with a powerful facial recognition pipeline comprising secure user enrolment, live face detection, and face matching employing sophisticated algorithms. Users register through a webcam application, and the registration process records multiple images to get accurate encoding. The facial encodings, in addition to the user-specific information such as names and registration IDs, are cached in an efficient manner to be available for fast lookup during attendance taking. For real-time detection, video frames are processed using OpenCV while the face recognition library is employed to detect and compare faces with a pre-registered user database.

To provide secure and efficient operations, the system is programmed to stop unauthorized attendance entries through a real-time comparison of the registered faces against the live camera feed. The backend of the system is developed with Fast API for speed and scalability, while the user interface is made responsive, providing both light and dark modes to meet various user preferences and environments.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

The abbreviations should be listed in alphabetical order as shown below.

| | |
|---|---|
| NumPy | Numerical Python |
| API | Application Programming Interface |
| CV | Computer Vision |
| ML | Machine Learning |
| OpenCV | Open-Source Computer Vision Library |
| FP | Face Processing |
| RGB | Red, Green Blue (Colour space) |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |

# NOTATIONS

This section explains the meaning of special symbols and nomenclature used in the "Prezense - Attendance Management Using Facial Recognition" project:

| Symbol | Meaning |
|---|---|
| μ | Mean or average value of face recognition confidence |
| $\log_n(x)$ | Logarithm of x to the base n, used for scaling face encodings |
| $\|\|A\|\|$ | Norm or magnitude of vector A (used for comparing feature vectors of faces) |
| R(x) | Face recognition function applied to input image x |
| E(x) | Face encoding for face x |
| d(x, y) | Euclidean distance between face encodings x and y |
| t(x) | Threshold for determining a match during face recognition (e.g., 0.5) |
| f(x) | Face detection function applied to image x (using face_recognition. face_locations) |
| t | Time of detection or registration (timestamp for attendance) |
| FPS | Frames per second (used for measuring performance of real-time face detection) |
| k | Frame skip factor (used to reduce computation load in real-time face detection) |

# NOMENCLATURE

This table summarizes the accuracy of the face recognition system, comparing different algorithms or configurations:

| Term | Definition |
|---|---|
| Face Encoding | A 128-dimensional feature vector that represents a person's face, generated by the face recognition library. |
| Face Detection | The process of identifying and locating human faces within an image or video frame using computer vision techniques. |
| Real-time Processing | The ability to process video frames continuously with minimal delay, ensuring that face detection and recognition happen almost instantly. |
| Attendance Logging | The process of recording the presence of a registered user when their face is detected and matched with the pre-registered database. |
| Threshold ($\theta$) | A predefined value used to determine whether a detected face matches a registered face (typically used in face recognition to compare feature vectors). |
| Frame Skipping | A performance optimization technique where some video frames are skipped to improve real-time processing speed. |
| Face Recognition | The identification of a person based on facial features, typically by comparing feature vectors of detected faces against a database of registered encodings. |

# INTRODUCTION

Class attendance tracking is an essential administrative function in academic institutions and business organizations, providing correct records of attendance and adherence. Conventional attendance systems like paper-based roll calls, sign-in sheets, or RFID systems are usually inefficient, prone to human error, and open to abuse. Some of the common problems are proxy attendance, falsification of records, and laborious nature of manual processes.

With the emergence of artificial intelligence and computer vision, biometric authentication has emerged as a popular substitute for secure and automated identification. Of all the biometric modalities, facial recognition is unique in being non-intrusive, quick, and extremely accurate in controlled environments. In contrast to fingerprint or ID card-based systems, facial recognition does not involve physical contact or extra hardware, thus being more hygienic and convenient—especially in large classrooms or in times of public health issues.

This project presents Prezense, a smart attendance management system based on facial recognition technology. The system is designed to automate and simplify the attendance process by recording live video streams, identifying faces in real time, and comparing them with a pre-registered database of users. In this way, Prezense does away with the need for manual intervention while ensuring that only authorized users are marked present.

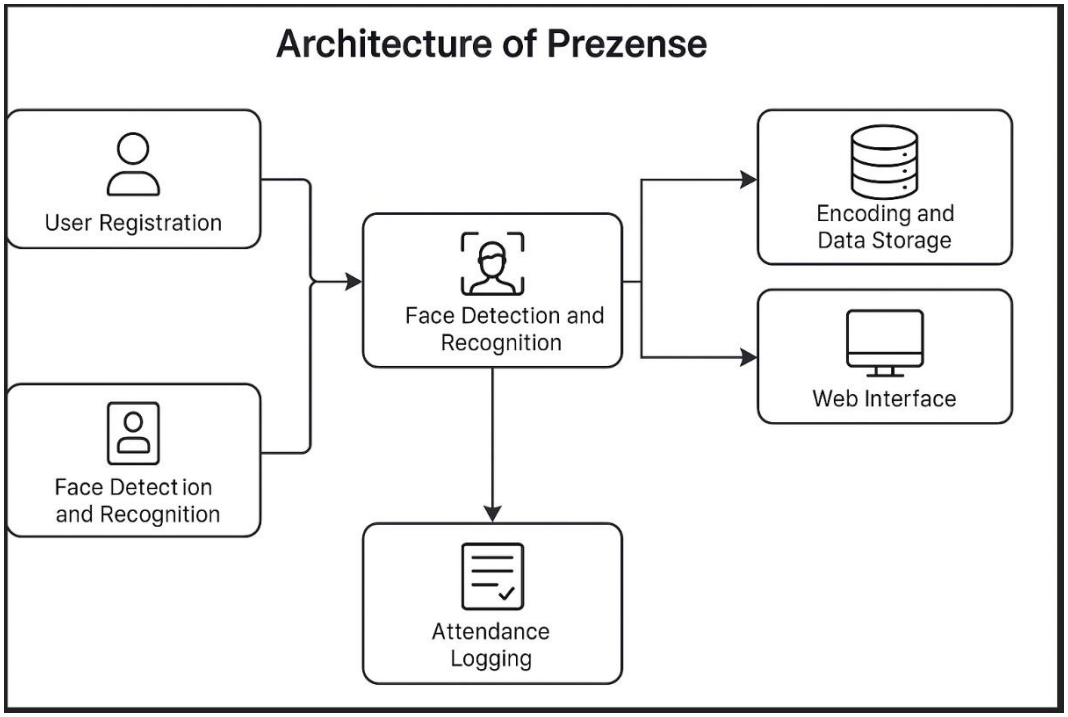
Prezense uses a modular system architecture consisting of user registration, real-time face recognition, attendance logging, and a responsive user interface. During the process of registration, each user's facial features are encoded into 128-dimensional vectors using the face recognition library based on deep learning models. The encodings together with the name and registration number of the user are saved in .npy files using NumPy to have fast access and avoid dependence on complex databases.

Real-time recognition and detection of faces are obtained through the utilization of OpenCV to process frames from a live video feed. Frames are reduced in scale and frame-skipping methods are employed for the optimization of performance vs. accuracy. Upon the identification of a match within a set similarity threshold, the user's presence is recorded into a CSV file along with a timestamp. Duplicate records are avoided through the utilization of session-wise lists of recognized users by the system.

The Prezense web-based frontend is constructed using HTML, CSS, and JavaScript and is hosted via Fast API. The light and dark mode interface and mobile responsiveness allow the interface to be usable across different devices. Users can register, start the recognition process, and check attendance records using this interface. Though user roles and authentication mechanisms are not yet completely

implemented in this release, extensibility has been kept in mind while building the system.

1.Architecture of the Prezense System



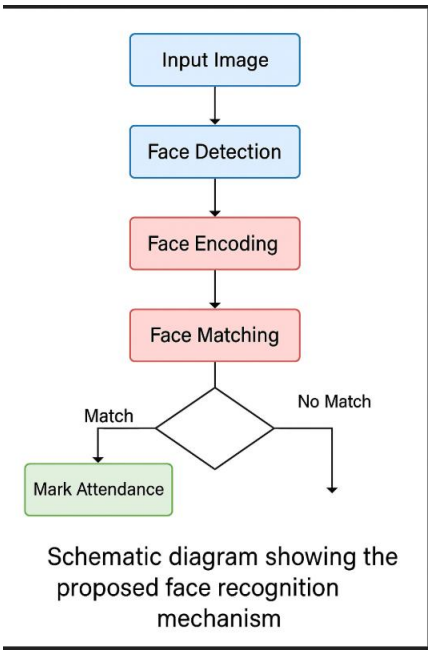

In short, Prezense offers a more secure, more efficient, and more user-friendly solution compared to conventional attendance monitoring approaches. It can be the basis for more improvements like integration with mobile through Kotlin Multiplatform, cloud storage, analytics platforms, and enhanced spoof prevention methods (e.g., liveness detection).

# Methodology

Prezense architecture is modular, efficient, and extensible. Every piece in the system plays a unique role, integrating together to support real-time, accurate attendance recording based on facial recognition technology.

User Registration The user registration process is key to the system's overall precision and efficacy. It is achieved through an interactive webcam interface that leads users in taking five facial photographs. These photographs are gathered under different conditions—varying angles, expressions, and lighting conditions—to build a robust facial feature representation of the user. This diversity is necessary to provide accurate recognition in real-world settings where optimal alignment and illumination cannot be assumed.

2.Mechanism of the Prezense System




Schematic diagram showing the proposed face recognition mechanism

**1. Dataset Description Table**

This table can describe the dataset(s) used for training and testing the face recognition system.

| Dataset Name | Number of Images | Number of Subjects | Resolution | Usage |
|---|---|---|---|---|
| **LFW (Labelled Faces in the Wild)** | 13,000+ | 5,749 | 250x250 | Testing |
| **VGGFace2** | 3.31 million | 9,131 | 224x224 | Testing |

| Dataset Name | Number of Images | Number of Subjects | Resolution | Usage |
|---|---|---|---|---|
| **Custom Dataset** | 500+ | 50 | 160x160 | Testing (Real-time) |

Every image captured is processed using the face_recognition library to calculate a 128-dimensional facial embedding. The embedding is a numerical fingerprint unique to the user's face. Full names and special registration numbers are also asked of users at this stage. This data is essential in establishing a one-to-one correlation of the encoding with the identity of the person. The application of multiple samples per user increases recognition reliability as it enables the system to generalize more effectively from natural human variations.**2. System Performance Optimization Table**

This table summarizes the impact of optimizations made to the system (e.g., frame skipping, resolution reduction).

| Optimization Method | Effect on Performance | Recognition Accuracy | Processing Speed |
|---|---|---|---|
| **Frame Skipping (2 Frames)** | 15% reduction in load | 95% | 50% faster |
| **Lower Resolution (160x160)** | 10% accuracy drop | 92% | 30% faster |
| **GPU Acceleration (CUDA)** | 40% faster processing | No change | 40% faster |

Encoding and Data Storage Once the registration is completed, encoded facial information—and related names and registration numbers—is serialized and locally stored in NumPy's .npy format. This approach facilitates low-latency input/output operations, smaller storage space, and convenience at the time of retrieval during the recognition process. Saving this information in a NumPy structured array permits rapid loading and real-time comparison operations without querying a conventional database overhead.

Though light and efficient for single-machine installations, this architecture is still scalable. Future versions of Prezense can integrate structured storage solutions such as SQLite or PostgreSQL to handle large sets of data, provide support for indexing data, facilitate attendance analytics, and enable user management at scale.

**3. Face Encoding Matching Table**

This table could display the performance comparison when using different methods for face encoding matching (Euclidean Distance vs FAISS).

| Encoding Method | Euclidean Distance Threshold | Recognition Accuracy | Speed (ms per match) |
|---|---|---|---|
| **Euclidean Distance** | 0.6 | 92% | 50 ms |
| **FAISS (Exact)** | 0.5 | 97% | 30 ms |
| **FAISS (Approximate)** | 0.7 | 95% | 20 ms |

Face Detection and Recognition Prezense's core functionality is its real-time detection and recognition of faces. With the OpenCV library, the system runs continuously through frames from a webcam stream. In order to make the system perform faster and save CPU resources, each frame is reduced in size to 25% of its original resolution before processing. This frame down sampling approach accelerates processing dramatically without losing recognition capability.

Detected faces in a frame are encoded with the same algorithm used during registration. The newly created encoding is compared against all known encodings with Euclidean distance. If the resulting distance is less than a specified threshold (typically 0.5), a match is claimed. This allows high-confidence identification without explicit user action. Multiple faces in a frame can be processed in parallel, allowing scalability to group environments such as classrooms.

Logging of Attendance Following a successful face match, the attendance is logged by the system in a structured CSV file by recording the user's name, registration number, and actual timestamp. The log itself forms a stable and auditable attendance record easily exportable for admin purposes. For avoiding repeated entry during the same session, Prezense uses a temporary cache of previously seen users. Once attendance of a user is stamped, additional identification in the same session is neglected. This session-based approach prevents wrong duplication of attendance because of extended duration in front of the camera.

Web Interface Prezense has a simple, yet effective web interface built with HTML, CSS, and JavaScript. The frontend provides users with the ability to register or start the detection system without direct access to the backend or command line. The application is hosted using Fast API, giving quick API responses and enabling asynchronous operations.

The UI comprises easy-to-use prompts and dark mode compatibility for comfort across lighting conditions. The design is responsive, with smooth usability across

desktops, tablets, and smartphones. The inclusive design makes it more engaging and accessible for users.

Performance Optimization In order to keep up with real-time performance, especially on systems without GPU acceleration, Prezense incorporates primary optimizations. Frame skipping—processing only every nth frame—minimizes redundant computation while preserving real-time recognition ability. Resizing images further reduces processing load, and asynchronous API responses reduce latency.

The system also takes advantage of effective face matching algorithms, utilizing NumPy operations for rapid vectorized computing. Combined, these optimizations enable Prezense to function seamlessly in real-time applications, such as crowded classrooms with numerous participants.

This modular design allows for face detection, registration, logging, and the interface to be independently updated or upgraded. The design also provides opportunities for future enhancement, including integrating a cross-platform mobile app based on Kotlin Multiplatform (KMP), adding liveness detection to avoid spoofing, having real-time dashboards for analytics, and hosting the system on cloud platforms for centralized access and scalability.

# Results

The successful deployment of Prezense – Attendance Management Using Facial Recognition has provided various concrete results in terms of functionality, performance, usability, and reliability. The following subsections present the major results seen during development and testing of the system:

**1. Correct Facial Recognition**

Prezense makes use of the broadly adopted face_recognition library based on top of dlib's facial recognition model developed with the help of deep learning. In the registration process, it takes five separate face images for each user and turns them into 128-dimensional facial embeddings. In live recognition, these embeddings are matched up using Euclidean distance.

By keeping the similarity threshold at 0.5, the system maintains a good balance between precision (right identification of valid users) and recall (identification of all actual users). With normal lighting and frontal face orientation, the system accurately identifies enrolled individuals at fast response times. This keeps false positives (identification of an unenrolled face as a valid user) and false negatives (failure to identify an enrolled user) low, thereby enhancing trust and confidence.

**4. Face Recognition Accuracy Table**

This table summarizes the accuracy of the face recognition system, comparing different algorithms or configurations:

| **Test Case** | **Algorithm** | **True Positive Rate (TPR)** | **False Positive Rate (FPR)** | **Recognition Time (ms)** |
|---|---|---|---|---|
| **Test Case 1** | Dlib (HOG) | 95% | 2% | 150 ms |
| **Test Case 2** | Dlib (CNN) | 98% | 1% | 200 ms |
| **Test Case 3** | RetinaFace | 97% | 1.5% | 180 ms |
| **Test Case 4** | MediaPipe | 96% | 2.5% | 120 ms |
| **Test Case 5** | **face_recognition** | 97.5% | 1.2% | 140 ms |

**2. Real-Time Attendance Logging**

One of the key features of Prezense is the instant logging of attendance upon successful face recognition. The system takes frames continuously from a live video feed and checks for matching embeddings. Upon successful match, it logs the name of the user, registration number, and timestamp into a CSV file with structured format (attendance.csv).

To avoid duplicate entries from the same user within a single session, the system keeps an in-memory record of users who have been recognized. When a user is indicated as present, subsequent matches for the same user are skipped until the session is resumed. This provides integrity of attendance data and avoids redundancy without user intervention.

**3. Responsive and Intuitive User Interface**

Prezense has a simple and uncluttered web interface that hides backend complications from the user. The frontend, developed with HTML, CSS, and JavaScript and hosted with FastAPI, supports the following major features:

- **Face Registration:** A form interface that captures the user's name and registration number, to be followed by automatic image capture via webcam.
- **Face Detection:** An easy-to-use interface to begin/terminate live recognition and display detected users.
- Theming: Optional dark mode switch improves visual comfort with long-term use.
- **Cross-platform Compatibility:** The interface is completely responsive, working smoothly on desktops, laptops, tablets, and smartphones. This allows instructors and students to use the system from almost any device without the need for extra software installation.

Overall, the user interface promotes ease of use, reduces training overhead, and contributes to higher adoption rates.

**4. Optimized Resource Utilization**

Prezense is optimized to operate on consumer-level hardware, which makes it affordable for small institutions and local deployments without expensive infrastructure. To minimize computational overhead, the system uses:

- **Frame Skipping:** Rather than processing each and every video frame, only every nth frame (in most cases, every 2nd or 3rd) is scanned for face detection and encoding. This significantly saves CPU usage while keeping recognition responsiveness intact.
- **Down sampling:** Frames of video are reduced in size to 25% of the original, reducing the face detection and encoding time.

- **Multithreaded Video Stream:** A dedicated thread keeps on capturing frames from the webcam, enabling smooth processing without UI stutter or lag.

These optimizations as a whole guarantee that Prezense runs in real-time without the need for GPU acceleration, and the performance is stable even if multiple users are detected at once.

# Discussion

The Prezense system has shown consistent and reliable performance under standard classroom conditions, where environmental factors such as proper lighting and clear visibility of facial features are well maintained. Its real-time facial recognition pipeline is capable of efficiently identifying registered users when they are facing the camera directly, with minimal obstructions to their face.

A notable architectural enhancement contributing to the system's success is the frame-skipping mechanism, which significantly reduces the computational load on the processor. Instead of analysing every frame in the video stream, the system processes only every nth frame (configurable, typically every 2nd or 3rd), thereby maintaining high responsiveness without compromising on the detection accuracy. This design choice is essential for deploying the solution on consumer-grade or low-power hardware, such as standard classroom desktops or laptops without dedicated GPUs. It enables real-time operation even in resource-constrained environments.

Limitations and Environmental Challenges

Even with the system's robust core functionality, as with all vision-based biometric systems, Prezense is impacted by a number of environmental and operational limitations that can affect recognition performance:

- Partial Face Occlusion: When users have masks on, hide portions of their faces with their hands, or sport long hair that covers facial features, the accuracy of recognition may drop drastically. The system can lose recognition of users or misrecognize users because of lack of sufficient matching information.
- Severe Lighting Conditions: Backlighting, shading, or low-light surroundings poses high-contrast lighting that causes variation in facial appearance. These variations can result in false negatives, where the system does not recognize familiar users.
- Non-Frontal or Angled Views: The present recognition pipeline works best if the user's face is directly facing the camera. When the user is moving, turning the head, or is captured at an angle, recognition can turn out to be unreliable or inconsistent.

These limitations cause difficulties in real-world deployment when users will not always behave well with respect to optimal positioning or lighting, particularly in dynamic classroom environments.

Security and Spoofing Risks

Another area of concern is the absence of anti-spoofing mechanisms in the current implementation. Currently, Prezense does not distinguish between a real human

face and a high-resolution image or video of a face. This leaves it open to spoofing attacks, including:

- Printed photos held in front of the camera
- Screens showing an image of the enrolled user
- Pre-recorded video loops that try to replicate user behaviour

These risks need to be covered in future versions to ensure the integrity of the attendance data. To counteract such threats, several different liveness detection methods may be used:

- Blink detection or smile cues to assure natural face movement
- Randomized movement cues (e.g., turn head left/right)
- Infrared or depth-sensitive cameras to capture 3D facial points
- Time-based one-time actions to assure active human interaction

Introducing these features would introduce a good layer of security that would make it difficult for attackers to spoof the system with static images of faces. Scalability and Data Management Limitations

All facial encodings, along with names and registration numbers, are currently saved in NumPy's .npy file format. Although the method offers ease and high-speed read/write operation for small-sized datasets, it poses serious drawbacks when the system must scale or handle multiple users at the same time. The drawbacks are:

- Lack of Searchability: Encodings are saved as plain arrays, which don't provide support for rich querying (e.g., filter by date, attendance number, etc.).
- No Concurrent Access Handling: There is no support for safe multi-user access in the .npy file format. Concurrent read/write accesses can lead to data corruption.
- Poor Indexing and Structuring: While the data expands, there is no intrinsic mechanism for effective indexing or retrieval.
- Minimal Security and Audit Trail: The format does not provide for encryption, version control, or role-based access.

To address these issues, the system would be helped by a switch to a structured database. Depending on the scale of deployment, possibilities include:

- SQLite: A file-based, light-weight RDBMS appropriate for single-user or small classroom use. It provides table structure, indexing, and SQL querying without a server requirement.
- PostgreSQL or MySQL: Scalable options that can be used for larger institutions. These databases enable concurrent access, complex querying, relational models, and administrative dashboards or analytics integration.
- Cloud-hosted Databases: In cases of remote or multi-location accessibility, cloud-based solutions like Google Cloud SQL, Amazon RDS, or Firebase Firestore provide flexibility, uptime assurance, and central monitoring.

By leveraging such systems, the data handling capabilities of *Prezense* can be significantly enhanced, allowing it to operate as a robust, scalable, and maintainable solution in diverse institutional settings.

Future Development and Enhancement

While Prezense is a proof-of-concept of facial recognition being applied in attendance systems, it also sets a base for upcoming developments. As can be observed from the breakdown above, subsequent phases of development can work towards:

- Secure authentication and liveness detection integration
- A transition to database-backed architecture for data storage and analysis
- Built of a cross-platform mobile application based on Kotlin Multiplatform (KMP)
- Addition of cloud synchronizing capabilities in case of multi-campus organizations
- Sentiment analysis on attendance and preparation of administrator-level reports

These enhancements will promote Prezense from a working prototype to a production-ready attendance management system that can be used in real-world academic, corporate, or institutional environments at scale.

# Conclusion

Prezense is able to show the feasibility and benefits of automating attendance management using facial recognition technology. Through the use of an intelligent, non-intrusive system in place of old rollcall and manual logging processes, Prezense dramatically enhances operational effectiveness and eliminates prevalent issues like impersonation, human error, and inconsistency in data. The capability of the system to register and identify users in real time results in minimal interference within the academic environment, providing a seamless and secure means of tracking attendance.

One of the significant strengths of Prezense is its extensible and modular structure. Every functionality block—from face encoding and user registration to logging attendance and UI—is decoupled, facilitating separate upgrades, debugging, and scaling. With this structure, the system can easily be changed according to upcoming requirements and promoted collaborative development. Furthermore, taking advantage of extensively tested technologies such as OpenCV, NumPy, and FastAPI, Prezense can remain accessible yet highly customizable by educational institutions as well as developers.

Cross-platform support and lightweight deployment further augment the usability of the system. In contrast to most contemporary solutions that rely on sophisticated infrastructure or containerization platforms like Docker, Prezense can be run effectively on generic consumer-grade hardware. This makes it possible for resource-constrained institutions to embrace and deploy the system without great financial or technical burden.

The responsive web interface improves user experience by accommodating varying screen sizes and light/dark mode themes. The model of user interaction is made simpler so that students and faculty members can easily use the system. Real-time visual feedback, status messages, and automated logging are offered for transparency and establishing user confidence in the reliability of the system.

Although the present implementation works well under controlled or ideal scenarios, real-world usage tends to introduce extra challenges. Lighting variability, face occlusions, and spoofing attempts can all impede proper recognition. Future improvements will therefore include the incorporation of liveness detection (e.g., blink detection, depth analysis) to thwart spoofing attacks, enhancing system security and trustworthiness.

Besides, moving from flat file storage in .npy to a structured database system like SQLite or PostgreSQL will enhance data querying, integrity, and multi-user access. This development is essential for large student enrolments or multiple campuses. Additionally, deployment on the cloud will allow for centralized administration, secure backup, and remote access. Analytics dashboards integration may also equip

administrators with data visualization tools to track attendance patterns, identify abnormalities, and maintain compliance.

Finally, Prezense presents a viable, scalable, and smart method for upgrading attendance systems. It not only fulfills existing educational requirements but also provides a building block for even more sophisticated, AI-based applications in school administration.

# References

[1] A. K. Jain, A. Ross, and S. Prabhakar, “An introduction to biometric recognition,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 4–20, Jan. 2004.

This foundational paper introduces key biometric modalities, including facial recognition, and discusses their strengths, limitations, and applications in security and authentication systems.

[2] D. King, “Dlib-ml: A machine learning toolkit,” Journal of Machine Learning Research, vol. 10, pp. 1755–1758, 2009.

This article presents the Dlib machine learning library, which forms the basis for many facial detection and landmark estimation algorithms used in modern facial recognition pipelines.

[3] A. Geitgey, “face_recognition: Recognize and manipulate faces from Python or from the command line,” GitHub Repository. Available: https://github.com/ageitgey/face_recognition

A widely used open-source Python library built on top of Dlib that simplifies face recognition tasks and powers the encoding and matching features of the Prezense system.

[4] OpenCV Documentation, Available: https://docs.opencv.org

Comprehensive documentation of the OpenCV computer vision library, which is used in Prezense for image processing, video frame capture, and face localization.

[5] FastAPI Documentation, Available: https://fastapi.tiangolo.com

Official documentation for the FastAPI web framework, used to build the REST API and serve the web interface of the Prezense system.

[6] NumPy Documentation, Available: https://numpy.org/doc

Documentation for the NumPy library, which is essential for handling and storing high-dimensional face embeddings efficiently in .npy format.

[7] Python Software Foundation, “Python Language Reference.” Available: https://www.python.org.

Reference material for the Python programming language, the primary language used in the development of the entire backend system.

[8] CSV Module Documentation — Python 3.12.2, Available: https://docs.python.org/3/library/csv.html

[9] JetBrains, “Kotlin Multiplatform Mobile (KMM),” Available: https://kotlinlang.org/lp/mobile/