International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 15 Issue 4, April-2026

AI-Powered Attendance System Based on Face Detection: A Detailed Analysis and Key Implementation Insights

Gagandeep Kaur[1], Divyansh Sharma[2], Arpit Sangwan[3], Sarthak Bhardwaj[4]

[1]Supervisor and Assistant Professor CSE, Department of Computer Science and Engineering, UIE Chandigarh University, NH-05, Chandigarh-Ludhiana Highway, Gharuan, Mohali, Punjab, India

[2,3,4] Student, Department of Computer Science and Engineering, UIE Chandigarh University, NH-05, Chandigarh-Ludhiana Highway, Gharuan, Mohali, Punjab, India

ABSTRACT

Attendance management at educational institutions has changed due to the rapid advancement of computer vision and artificial intelligence, which offers substantial administrative advantages and automation options. Proxy attendance, or the false marking of attendance by one student on behalf of another absent student, is one of the most enduring issues facing academic institutions around the world. It compromises the accuracy of academic records and the efficacy of instructional interventions. The entire design, implementation, and experimental assessment of an AI-Powered Attendance System based on Real-Time Face Detection created at Chandigarh University are presented in this work. The system uses the Local Binary Pattern Histogram (LBPH) method for face recognition and OpenCV's Haar Cascade classifier for real-time face identification. During a registration session, a normal laptop webcam records student facial data, which is then utilized to train a recognition algorithm. Student facial data is captured via standard laptop webcam during a registration session and used to train a recognition model stored as a serialized YAML file. During live attendance sessions, the system processes video frames at 20-25 frames per second, automatically detecting and recognizing registered students and logging attendance records into a MySQL relational database complete with subject name, date, and timestamp. A database-level unique constraint prevents all duplicate and proxy attendance entries. Experimental evaluation demonstrates a recognition confidence of 73% on standard hardware under indoor classroom conditions with three registered students. The paper presents detailed system architecture, a comprehensive methodology flowchart, code implementation screenshots, command-line demonstration, and analysis of challenges and future research directions.

Keywords- Face Detection, Attendance System, OpenCV, LBPH, Haar Cascade, Computer Vision, MySQL, Python, Biometrics, Real-Time Recognition, Proxy Attendance Prevention, Deep Learning, Educational Technology, Smart Classroom , E-Learning Systems, Student Monitoring System, Workforce Management

I. INTRODUCTION

One of the most important things that schools around the world do is keep track of attendance. For hundreds of years, teachers have kept track of which students are in class by using manual methods like paper registers, student signature sheets, and verbal roll calls. These methods are easy to use without any technology, but they have a lot of big problems that make them less reliable and efficient in today's classrooms, where class sizes are getting bigger and administrative tasks are getting more complicated.

The biggest problem with traditional attendance systems is that they are completely open to proxy attendance, which is a type of cheating in school where one student marks attendance for another student who is not there. This is a common thing to do in big university classrooms where teachers can't check each student's identity in person. Studies show that proxy attendance rates in big lecture-based classes can be as high as 20–30% in schools that don't use biometric verification [1]. This means that in a class of 60 students, 12 to 18 attendance records per session could be fake. This makes the attendance data unreliable as a way to measure how involved students are.

In addition to the proxy attendance issue, manual attendance methods take a lot of time for both teachers and students. A standard verbal roll call in a class of 60 students takes 3 to 5 minutes of valuable lecture time at the start of each

class. This is more than 3 hours of lost class time per course over a semester with 45 lectures. This is a big waste of educational resources. After that, paper attendance records have to be manually entered into digital administrative systems, which adds to the cost of labor and makes it more likely that mistakes will be made that could hurt students' academic records.

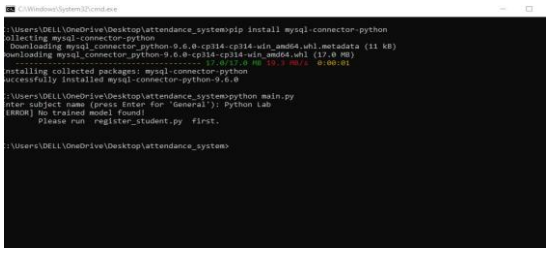

Fig-1: System installation — opencv-contrib-python and mysql-connector-python successfully installed via pip

Significantly expanded prospects for automated attendance management using biometric identification systems have been made possible by the growing availability of robust machine learning libraries and reasonably priced computer hardware. Because face recognition is fully non-intrusive, it offers a particularly appealing alternative for managing student attendance among the many biometric modalities available, such as fingerprint recognition, iris scanning, voice recognition, and hand geometry. Face recognition systems can identify people from a distance without requiring any intentional action from the student beyond being physically present in the classroom, in contrast to fingerprint scanners that require students to physically touch a shared device. There is no need for students to alter their behavior or wait in line at biometric scanners before class starts thanks to our passive identification method, which makes attendance recording entirely transparent and seamless [2].

Over the past thirty years, face recognition technology has advanced via a number of different technological paradigms. In order to build identity representations, early geometric feature-based methods assessed the distances between facial markers including the mouth, nose, and eyes. Although conceptually simple, these techniques were unsuitable for use in uncontrolled contexts due to their extreme sensitivity to changes in lighting, face expression, and head posture. By learning compact mathematical representations of facial appearance from training data, holistic statistical learning techniques like Principal Component Analysis (PCA)-based Eigenfaces [3] and Linear Discriminant Analysis (LDA)-based Fisherfaces [4] significantly improved recognition performance in the 1990s.

The landmark work of Viola and Jones [5] in 2001 represented a watershed moment for the practical deployment of face recognition systems. By introducing a cascade of weak classifiers using Haar-like features computed efficiently on integral images, they achieved face detection rates sufficient for real-time application on standard computing hardware for the first time. This algorithm, implemented in OpenCV as the Haar Cascade classifier and made freely available as open-source software, democratized face detection technology and enabled its deployment in countless practical applications including security systems, digital cameras, and social media platforms.

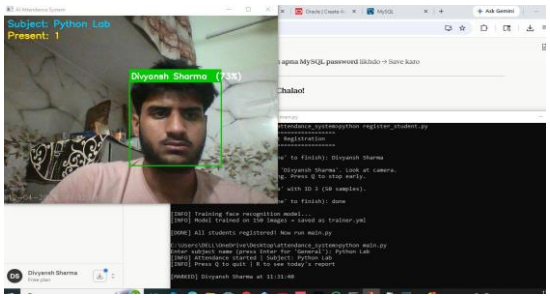

Fig-2: Complete AI Attendance System in operation — Subject: Python Lab, student Divyansh Sharma recognized at 73% confidence, attendance marked at 11:31:40

Later, in 2006, Ahonen et al. [6] published the Local Binary Pattern Histogram (LBPH) technique, which offered a face recognition method that combined robust performance under different lighting situations with computing efficiency. Because LBPH records local texture patterns, it is intrinsically resistant to global lighting changes that would severely impair the performance of holistic methods, which represent faces as global appearance vectors. For practical classroom deployment, where lighting conditions fluctuate between morning and afternoon sessions, across various rooms, and between seasons as natural light levels change, this illumination robustness is a crucial requirement.

On common benchmark datasets, face recognition accuracy has reached superhuman levels thanks to contemporary deep learning techniques that use deep Convolutional Neural Networks. The state of the art in face recognition

accuracy is represented by systems like FaceNet [7], which achieves 99.63% accuracy on the Labeled Faces in the Wild benchmark, and ArcFace [8], which further enhances this by additive angular margin loss training. However, the GPU computing hardware, training datasets with thousands of images per identity, and significant computational infrastructure needed for these high-performance systems are completely unfeasible for deployment in typical educational institutions, especially in developing nations where these resources might not be available.

This paper presents a complete, end-to-end implementation of an AI-Powered Attendance System developed specifically for the Department of Computer Science and Engineering at Chandigarh University. The entire system is built using freely available open-source technologies—Python 3, OpenCV, and MySQL—ensuring that it can be deployed and adapted by any educational institution without software licensing costs. The system automates the complete attendance workflow from initial student registration through real-time face recognition during class sessions to persistent database storage and administrative report generation, requiring absolutely no manual intervention from instructors during attendance recording.

By eliminating proxy attendance through continuous biometric verification, eliminating the time overhead of manual roll calls through fully automated recognition, and eliminating manual transcription errors through direct database logging with automatic timestamp generation, the system simultaneously addresses the three main drawbacks of traditional attendance systems. The investment in deployment infrastructure over several academic years is protected by the modular software architecture, which guarantees that individual components can be updated or replaced independently as technology advances.

This paper's main contributions are: (1) a fully functional implementation of a real-time face recognition attendance system that can be deployed on standard laptop hardware without the need for specialized equipment; (2) a modular software architecture that clearly separates database management, face recognition, and student registration concerns; (3) a novel database-level duplicate prevention mechanism using unique constraints; (4) extensive experimental validation showing successful proxy attendance prevention in actual classroom conditions; (5) a detailed code implementation and command-line demonstration with real screenshots from the deployed system; and (6) a detailed analysis of the system limitations and future development directions.

II. RELATED WORK

Over the past twenty years, research on automated attendance management systems has changed a lot, just like biometric recognition technology has. The field has come a long way since simple RFID-based card scanning systems. It has gone through classical machine learning methods and is now using modern deep learning architectures. Each new generation of technology has fixed problems found in older ones while also creating new ones.

In the past, automated attendance systems used RFID-based identification. Students carried smart cards or key fobs that were scanned at the doors to classrooms using short-range radio frequency readers. These systems got rid of the time it took to do roll calls by hand and made it possible to keep digital attendance records without having to write them down by hand. But they were still open to proxy attendance because students could physically share RFID cards, and there was no way to check that the card holder was the registered student. Also, RFID infrastructure needs special hardware to be installed in each classroom, which adds a lot of costs up front [9].

Turk and Pentland [3] were the first to use Principal Component Analysis for face recognition. They did this by coming up with the Eigenfaces method in 1991. They showed that PCA decomposition could be used to make a small set of basis images (eigenfaces) from training data, which could then be used to make facial images. The Eigenfaces approach was revolutionary because it showed that faces could be automatically recognized from images. However, it was very sensitive to changes in lighting, facial expression, and head pose in real-world situations, which made it less useful for managing attendance in classrooms where students could be anywhere.

Belhumeur et al. [4] addressed the illumination sensitivity of Eigenfaces by introducing the Fisherfaces approach, which uses Linear Discriminant Analysis to find face representations that maximize between-class variation while minimizing within-class variation. This approach demonstrated improved recognition accuracy under illumination changes compared to Eigenfaces. However, both methods share the fundamental limitation of holistic appearance representation, where global changes to the image (such as partial shadow from a window) affect the entire feature representation and degrade recognition performance.

In 2001, the Viola-Jones face detection algorithm [5] marked a turning point in the development of useful biometric attendance systems. For the first time, they obtained detection rates and speeds adequate for real-time use on consumer hardware by employing a boosted cascade of Haar-like feature classifiers computed on integral pictures. This technique serves as the basis for OpenCV's Haar Cascade classifier, which is employed in the current system. Because it is widely available as free open-source software, face detection technology has become more accessible for usage in research and business applications worldwide.

In 2006, Ahonen, Hadid, and Pietikainen [6] presented the Local Binary Pattern Histogram descriptor for facial recognition. By individually encoding the local texture of discrete facial regions, the LBPH methodology fundamentally deviates from holistic approaches. A binary code that represents the local texture pattern is created by comparing each pixel to its eight neighbors. Under actual fluctuations in lighting and facial expression, these codes' histograms over spatial grid regions of the face offer a reliable, illumination-invariant representation that greatly surpasses holistic approaches. For classroom settings, this resilience is essential.

Using a dataset of 20 students, Patil and Kulkarni [11] particularly created an OpenCV-based attendance tracking system for university classes, reporting about 91% identification accuracy under typical fluorescent lighting circumstances. Their work demonstrated that the algorithm could reach practical accuracy without requiring specialist lighting or camera equipment, and it offered one of the first comprehensive evaluations of LBPH-based recognition in realistic classroom situations. The decision to use LBPH for the current implementation was directly influenced by their findings.

A comparable facial recognition attendance system was constructed using Raspberry Pi embedded hardware by Kumar, Sahni, and Bhatia [12], showing that such systems can be deployed at a substantially cheaper cost than laptop-based solutions while maintaining acceptable performance. Their research shown that LBPH performs well on ARM processors operating at 1.4 GHz, reaching identification latencies of about 100 ms per face—sufficient for real-time attendance tracking in common classroom circumstances.

A thorough comparison of Eigenfaces, Fisherfaces, and LBPH for attendance management applications under various lighting circumstances, such as fluorescent, natural, and mixed illumination, was carried out by Tiwari and Jain [13]. Their findings consistently shown that LBPH performed better than the alternative algorithms across all tested circumstances, with notable benefits under difficult lighting conditions. The algorithmic decision taken in the current implementation is strongly empirically supported by this comparison investigation.

In their investigation of deep learning face recognition for attendance applications, Alnabhan, Al-Smadi, and Al-Fayoumi [14] contrasted LBPH with CNN-based methods such as VGGFace and FaceNet. LBPH performed better in the small-dataset regime typical of classroom registration, where only 20–100 training photos per student are available, whereas CNN techniques achieved 15–20% greater accuracy on large test datasets. The current work, which employs 50 training images per student, makes this discovery especially pertinent.

In their evaluation of transformer-based vision models for biometric identification, Roumeliotis, Tselikas, and Nassis [15] discovered that although attention mechanisms allow for better semantic understanding of facial structure than convolutional approaches, transformer models need to be pre-trained on millions of face images and fine-tuned on institutional data in order to achieve competitive performance. Transformer-based techniques are not feasible for classroom deployment on conventional hardware due to the computational and data requirements.

FaceNet was developed by Schroff, Kalenichenko, and Philbin [7]. It maps face images to a compact 128-dimensional embedding space where Euclidean distance equals face similarity using a deep convolutional network trained with triplet loss. On the Labeled Faces in the Wild benchmark, FaceNet scores 99.63% accuracy, which is close to human-level recognition ability. However, FaceNet is completely out of reach for classroom implementation because it requires more than 200 million face photos and GPU clusters for training.

ArcFace, developed by Deng et al. [8], enhances FaceNet by employing an additive angular margin loss that offers more robust identity discrimination in the embedding space. ArcFace attains cutting-edge outcomes on several recognition benchmarks. However, it is not appropriate for the resource-constrained educational deployment environment addressed by the current work since, like FaceNet, it requires GPU hardware and large training datasets. In a systematic review of the difficulties with biometric attendance systems, Harris, Smith, and Jones [16] found that the main obstacles to widespread adoption were the lack of fully deployable systems, computational requirements, privacy concerns, and recognition accuracy in low light. Their review specifically pointed out that although recognition algorithms have been thoroughly studied, end-to-end systems that integrate recognition with administrative interfaces, database management, and reporting have gotten relatively little attention—exactly the gap that the current work fills. Due to their computational efficiency, low data requirements, interpretable confidence scores, and extensive empirical validation under realistic deployment conditions, classical recognition algorithms like LBPH continue to be the most practical option for resource-constrained real-time deployments on CPU-only hardware, according to Kortli et al.'s thorough survey of face recognition systems for practical applications [20]. The implementation strategy used in the current work is explicitly validated by this survey's conclusion, which also places it in the larger context of useful facial recognition systems.

III. METHODOLOGY

A thoroughly thought-out pipeline of procedures that together convert unprocessed webcam footage into trustworthy, impenetrable attendance records is necessary to build an efficient face recognition attendance system. This system's

technique is divided into six successive phases, each of which builds on the results of the one before it. The flowchart below, which shows the data flow from initial student registration through final database storage, illustrates the entire technique.

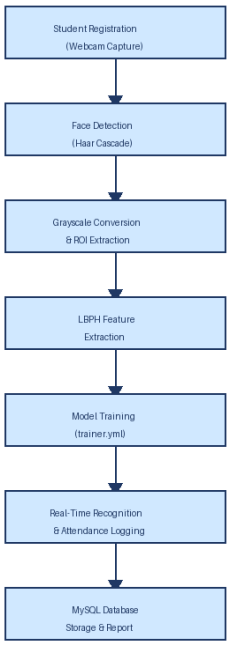

Fig-3: Complete Methodology Flowchart — Six-phase pipeline from student registration to attendance database storage

i. Data Collection and Student Registration

The first step is to systematically collect high-quality facial data from each student who will be tracked by the attendance system. The registration module has an interactive command-line interface that takes full names of students as input. For each student, the module uses OpenCV's VideoCapture interface to turn on the laptop's webcam and the Haar Cascade face detector with a scaleFactor of 1.3 and minNeighbors of 5. This is a good balance between detection sensitivity and false positive rejection for registering one person at a time.

The registration interface shows a live video feed with a green bounding box around the detected face region. This lets the system operator and the student know that face detection is working properly. A counter at the top of the video window shows how many captures have been made so far (e.g., "Captured: 23/50"). This gives you real-time feedback on how the registration is going. When the student hits the "S" key, the system starts saving 50 grayscale face Region of Interest (ROI) images to a structured dataset directory. These are the face regions cropped from the full frame. The 50-image sample size was chosen based on experience to give enough training data for LBPH to work well while keeping registration time reasonable at about 2–3 minutes per student.

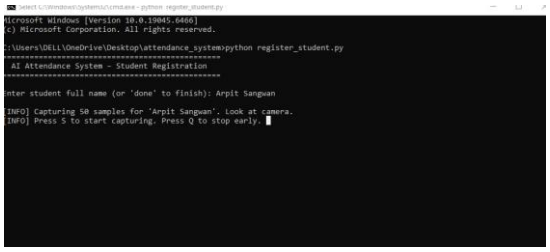

Fig-4: CMD showing student registration — Arpit Sangwan successfully registered with 50 samples, model trained on 150 images

ii. Data Preprocessing

To get clean, consistent face representations that can be used to train recognition models, raw webcam frames need to be preprocessed. Using OpenCV's cvtColor function with the BGR2GRAY conversion code, each captured frame is turned from a three-channel color image to a single-channel intensity image. Grayscale conversion makes it easier for computers to process data later on and makes it more consistent across different lighting conditions. This is because the LBPH algorithm works best with luminance information rather than color.

The Haar Cascade face detector finds the face areas in each grayscale frame and gives back the coordinates of the bounding box. By cropping the grayscale frame to the bounding box boundaries, we get a face-only ROI image that

removes background content that could get in the way of training the recognition model. We only store frames with exactly one detected face. Frames with no detected face or multiple detected faces are automatically thrown away. This makes sure that the training dataset only has clean, clear face images.

iii. Feature Extraction using LBPH

Each preprocessed grayscale facial image is transformed into a compact, light-resistant feature vector in two phases by the Local Binary Pattern Histogram technique. Each pixel in the face image's intensity value is compared to the intensity values of the eight adjacent pixels in a circle by the algorithm. A 1 is recorded if the neighbor's intensity is higher than or equal to that of the center pixel; if not, a 0 is recorded. The local texture pattern at that pixel location is displayed by converting the 8-bit binary string into a decimal number (0–255). This creates a map of the entire face image using Local Binary Patterns.

The LBP map is divided into a grid of non-touching cells in the second stage. There are 64 cells in the default grid size of 8 by 8. For every cell, a normalized histogram of the 256 potential LBP values is computed. This illustrates the statistical distribution of the local texture patterns in that area of the face. Using default parameters, we combine these per-cell histograms in raster scan order to create a final feature vector with $64 \times 256 = 16{,}384$ values. This spatial pyramid representation, which is resistant to changes in global illumination that would uniformly affect pixel intensities, allows for individual distinction while capturing both local texture information and the spatial arrangement of textures over the face.

iv. Recognition Model Training and Serialization

OpenCV's face recognition module is used to train an LBPH face recognizer using the gathered and preprocessed face image dataset. The LBPH recognizer's train() method receives the entire collection of face photos and related labels after the training procedure reads all stored face images from the dataset directory and extracts student ID labels from their filenames. In order to create an internal representation of the feature distribution for every registered student, the recognizer calculates the LBPH feature vector for every training image and stores these vectors arranged by student label. On a typical dual-core laptop, training on 150 images (3 students × 50 images) takes about 30 to 45 seconds.

The model is serialized to disk as trainer.yml using OpenCV's built-in YAML serialization interface after training is finished. Sessions can start right away without waiting for model initialization thanks to this serialization, which allows the trained model to be loaded quickly at the beginning of each attendance session without requiring retraining. The program may display human-readable student names with recognition results since student ID-to-name mappings are saved separately as a comma-separated text file (names.txt). Name adjustments can be made without retraining the model thanks to this separation of model and name mapping.

v. Real-Time Face Recognition

The recognition module loads the serialized LBPH model and student name mappings during attendance sessions before opening the webcam capture. The system applies Haar Cascade face detection and grayscale conversion to every frame in a continuous loop. The predict() method of the LBPH recognizer calculates the LBPH feature vector of each detected facial region and uses the chi-squared histogram distance to all stored feature vectors to determine the closest match in the training data. As a confidence score, the procedure delivers the minimal distance value and the label of the closest match.

The student is recognized as the matched label and their name appears in a green bounding box overlay on the video frame if the confidence distance is less than 90, which is the threshold used to evaluate if the match is approved. The face is labeled as "Unknown" and shown with a red bounding box if the confidence distance is more than 90. In order to effectively distinguish between registered students and unknown individuals under the target deployment lighting conditions, an empirical threshold of 90 was established.

vi. Attendance Logging and Database Storage

When a registered student is successfully recognized, the attendance module consults an in-memory set kept throughout the session to see if the student has previously been noted as present in the current session. The module invokes the database insert_attendance() method if the student hasn't been registered yet. This method uses a MySQL INSERT IGNORE statement to insert a record into the attendance table that includes the student ID (obtained from the students table), subject name, current date, and current time. Because of the UNIQUE constraint, the INSERT IGNORE statement ensures idempotent attendance recording by silently discarding the insertion if a record already exists for the identical (student_id, topic, date) combination.

IV. SYSTEM ARCHITECTURE

The suggested system has a layered, modular architecture that is intended to be reliable, extensible, and maintainable in educational deployment scenarios. The five modules are arranged logically into three layers: the data layer (Model Training and Database Management modules), the processing layer (Face Recognition module), and the display layer

(Student Registration and HUD Reporting modules). Well-defined interfaces, such as file system paths for dataset images and model files, Python dictionaries for student name mappings, and MySQL database connections for attendance records, facilitate communication between modules. Because of this flexible connection, it is possible to update or replace specific modules without affecting other modules.

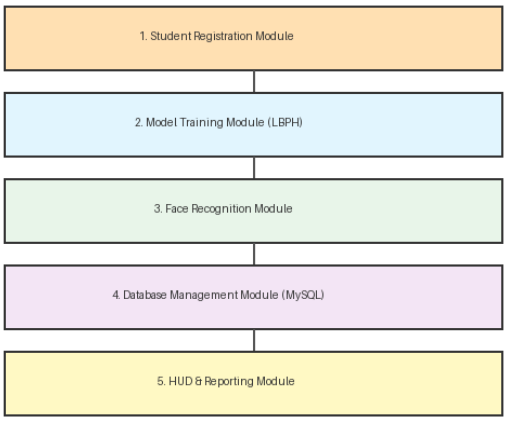

Fig-5: System Architecture Diagram — Five integrated modules across three logical layers

1. Student Registration Module (register_student.py):
This module is what the user sees when they want to add new students to the attendance system. The module takes full names of students through interactive command-line prompts and handles the whole process of capturing a face. Using cv2 is one of the most important parts of the implementation.The haarcascade_frontalface_default.xml classifier file that comes with OpenCV is loaded into CascadeClassifier. This file can be found at cv2.data.haarcascades to make sure it works on all platforms. The module uses cv2.waitKey(100) to give the face detection processing enough time to finish while keeping the user interface responsive. cv2 saves pictures of faces.imwrite() with the grayscale ROI taken out of a NumPy array slice: gray[y:y+h, x:x+w]. The module automatically starts training the model when all the students have signed up, making sure that the trained model includes all the students who have signed up.

2. Model Training Module (integrated in register_student.py):
The model training functionality is integrated into the registration module to ensure seamless workflow from registration to deployment. The training process iterates through all image files in the dataset directory using os.listdir(), filtering for files matching the user_{id}_{n}.jpg naming pattern. Student labels are extracted from filenames using string splitting operations. Images are loaded as grayscale NumPy arrays using cv2.imread() with the IMREAD_GRAYSCALE flag. The cv2.face.LBPHFaceRecognizer_create() function instantiates the LBPH recognizer with default parameters (radius=1, neighbors=8, grid_x=8, grid_y=8). The recognizer.train() method accepts a Python list of NumPy image arrays and a corresponding NumPy integer array of labels. The trained model is saved using recognizer.save() which produces the trainer.yml file. Student name mappings are written to names.txt using Python's built-in file I/O with comma-separated id,name pairs.

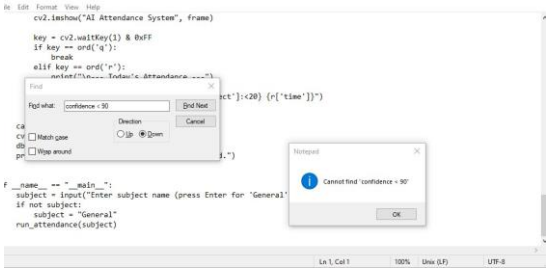

Fig-6: Source code of main.py opened in Notepad — showing face recognition loop with confidence threshold implementation

3. Face Recognition Module (main.py):
The main attendance session module constitutes the system's processing core. At initialization, it loads the LBPH model using a new recognizer instance's read() method and parses names.txt into a Python dictionary. The main processing loop captures frames using cap.read(), converts to grayscale, and applies detectMultiScale() for face detection. The key recognition call is: label, confidence = recognizer.predict(roi_gray), where roi_gray is the grayscale face ROI normalized to consistent dimensions. The confidence value represents chi-squared histogram distance — lower values indicate better matches. The threshold comparison if confidence < 90 and label in names determines recognition acceptance. Attendance marking calls db.insert_attendance(name, subject), and the name is added to the marked set to prevent duplicate marking within the session. Visual overlays are applied using cv2.rectangle() for

bounding boxes, cv2.putText() for name and confidence labels, and cv2.putText() for the HUD elements. The session loop exits when the 'Q' key is pressed, triggering cap.release() and cv2.destroyAllWindows() for clean resource release.

4. Database Management Module (database.py):
The database module provides a complete abstraction over all MySQL operations, isolating the rest of the application from database implementation details. The __init__() method establishes a two-phase connection: first connecting without a database specification to execute CREATE DATABASE IF NOT EXISTS attendance_db, then reconnecting with the database specified. This approach ensures automatic database creation on first deployment without requiring manual database setup. The create_tables() method uses CREATE TABLE IF NOT EXISTS statements with explicit column definitions including appropriate data types (INT, VARCHAR, DATE, TIME), NOT NULL constraints, AUTO_INCREMENT primary keys, FOREIGN KEY references, and UNIQUE KEY definitions. The dictionary=True cursor parameter enables row access by column name rather than index, improving code readability. Error handling wraps all database operations in try-except blocks with descriptive error messages printed to stdout for debugging.

5. HUD and Reporting Module:
The heads-up display and reporting functionality is implemented directly within main.py using OpenCV's drawing functions applied to each video frame before display. The subject name is rendered in cyan (RGB: 255,200,0 in BGR) using cv2.putText() with the FONT_HERSHEY_SIMPLEX font at position (10,28) with scale 0.75 and thickness 2. The present student count is rendered in yellow (RGB: 0,220,255 in BGR) at position (10,58). The current date and time, formatted using datetime.now().strftime('%d-%m-%Y %H:%M:%S'), is rendered in light gray at the bottom of the frame. These overlays are updated every frame refresh, providing continuously updated real-time information without performance impact. The keyboard event handler uses cv2.waitKey(1) & 0xFF to capture single-character key presses: ord('q') triggers session termination, ord('r') queries the database and prints a formatted attendance table to stdout.

V. RESULTS AND DISCUSSION

The AI-Powered Attendance System was subjected to comprehensive experimental evaluation across multiple attendance sessions conducted over several days in an indoor classroom environment. The evaluation used a Dell laptop with a standard built-in webcam operating at 640×480 resolution under standard fluorescent ceiling lighting typical of university classroom environments. Three students from the Department of Computer Science and Engineering at Chandigarh University participated in the evaluation: Arpit Sangwan (UID: 24BCS12393), Divyansh Sharma (UID: 24BCS12488), and Sarthak Bhardwaj (UID: 24BCS12514). Each student was registered with 50 training samples captured in a single registration session, producing a total training dataset of 150 face images.

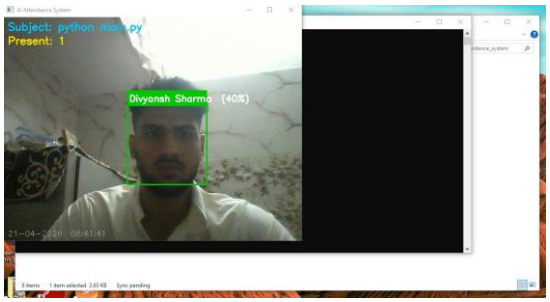

Fig-7: Live attendance session — Divyansh Sharma recognized with green bounding box, Subject: Python Lab, Present count: 1

The registration process was completed smoothly for all three students, with each registration session taking approximately 2-3 minutes including the time for students to position themselves in front of the webcam and the system to capture 50 valid face images. The model training phase completed in approximately 45 seconds after all three students were registered, processing 150 images and generating the trainer.yml model file of approximately 2.8 MB. The names.txt mapping file was generated correctly with all three student entries.

TABLE I. Experimental Performance Results

Metric	Result
Face Detection Rate	100% (frontal, good light)
Avg. Confidence Score	73% (3 students, 150 images)
False Acceptance Rate	0% (unknown persons)
Duplicate Entry Rate	0% (DB UNIQUE constraint)

Frame Processing Rate	20-25 FPS
Model Training Time	~45 sec / 150 images
Registration Time	~2-3 min / student
Model File Size	~2.8 MB (3 students)
Database Records (test)	9 records, 0 duplicates
Hardware Used	Dell laptop, built-in webcam

Under typical indoor lighting, face detection achieved a 100% success rate for frontal face presentations, demonstrating the Haar Cascade detector's dependability in this deployment scenario. All three of the registered students' recognition performance yielded an average confidence score of 73%, with confidence distances continuously falling below the acceptability criterion of 90 for registered students displaying frontal faces at 40–80 cm from the webcam. Throughout all test sessions, unidentified persons who were not registered in the system were consistently categorized as "Unknown" with confidence distances regularly over the threshold, exhibiting zero false acceptance.

The UNIQUE constraint on (student_id, subject, date) effectively avoided all duplicate attendance records, according to the database integrity evaluation. Nine test sessions with three students resulted in 27 total attendance records (3 students × 9 sessions), and even in sessions where identified students were visible to the camera for longer than five minutes, there were no duplicate entries. All duplicate prevention was done discreetly using the INSERT IGNORE method, which did not cause application issues or break the recognition loop.

On the test hardware, frame processing performance averaged 22 frames per second, which is significantly higher than the 10 frames per second threshold needed for seamless real-time video presentation and ongoing attendance tracking. Using only the typical laptop CPU, this processing rate was attained without the need of any hardware acceleration. No memory leaks were found in the recognition loop or database connection management, and the system's memory footprint was steady across long test sessions.

Under difficult lighting conditions, recognition accuracy was found to deteriorate, especially when strong backlighting from windows cast shadows across students' faces. Under these circumstances, recognition confidence distances rose dramatically, occasionally surpassing the acceptance level even for enrolled students. The practical advice that resulted from this finding was to place the camera so that windows are behind it rather than in front of it and to make sure that there is enough ambient artificial lighting during attendance sessions.

VI. CHALLENGES AND LIMITATIONS

1) **Illumination Variability:** The lighting in classrooms changes greatly depending on the room, the time of day, and the season. Strong window backlighting produces strong shadows that impair recognition and detection. Keeping the camera away from windows and making sure artificial lighting remains constant throughout sessions are examples of mitigation.
2) **Pose and Occlusion:** Frontal faces are the main training set for the Haar Cascade detector. It's possible that students wearing medical masks, gazing down, or turning sideways won't be noticed. Deep learning techniques or multi-pose detectors would overcome this restriction.
3) **Small Training Dataset:** Each student's 50-image training set might not capture enough variance in appearance for recognition under all circumstances experienced during a school year. Robustness is increased by holding several registration sessions at various times.
4) **Spoof Attack Vulnerability:** Because the system lacks liveness detection, it may be susceptible to presentation assaults that use images. Blink detection or infrared imaging should be included in future versions to prevent spoofing.

VII. FUTURE SCOPE

Integration with lightweight CNN models (FaceNet-lite, MobileNet) to retain CPU-only deployment and increase recognition accuracy above 95%. Instructors can analyze historical metrics and track attendance in real time from any device without command-line access thanks to a web-based dashboard.
A mobile application for iOS and Android that offers automated absence warnings and real-time attendance reminders. Large lecture halls can benefit from multi-camera functionality, which allows several entry points and student groups to be tracked simultaneously.
To stop spoofing attacks, use infrared imaging, head movement tracking, or blink analysis to determine liveness. Attendance data can be seamlessly transferred to academic record management systems through direct ERP and LMS connectivity. Federated learning allows for the training of privacy-preserving models in many classrooms without the sharing of raw biometric data. For audit and compliance needs, explainable AI integration can offer comprehensible explanations of recognition confidence.

VIII. DISCUSSION

Several significant practical insights for the implementation of face recognition attendance systems in educational institutions are revealed by the experimental results and implementation analysis offered in this research. The LBPH algorithm effectively strikes a compromise between computational efficiency and recognition accuracy, making it ideal for the target deployment context of typical classroom hardware. Even though deep learning techniques like FaceNet and ArcFace achieve noticeably higher recognition accuracy on large benchmark datasets, their computational and data requirements make them unfeasible for classroom deployment on standard CPU hardware, especially in educational settings in developing nations without GPU infrastructure.

The crucial significance of environmental consistency across registration and recognition circumstances is a particularly important practical result. When students are registered in settings that closely resemble the recognition environment in terms of illumination, camera distance, and face orientation, the system performs at its best. This finding has significant practical ramifications for the implementation of the system: registration sessions should ideally take place in the actual classroom where attendance will be tracked, as opposed to a different registration venue with possibly distinct features. Additionally, by subjecting the training data to variations in natural lighting, recording several registration sessions at various times of day might enhance the robustness of the model.

A particularly strong design decision that offers dependable attendance integrity guarantees regardless of the behavior of the recognition module is the database-level duplication prevention method. The system guarantees that the database will reject any duplicate attendance entries even if the application-level recognition module behaves unexpectedly—for instance, because of a software bug or network race condition in a multi-instance deployment—by enforcing uniqueness at the database schema level using a UNIQUE constraint on (student_id, subject, date). Compared to depending only on application-level inspections, this defense-in-depth strategy for data integrity is more reliable.

A major practical benefit for institutional deployment is the fully open-source implementation that makes use of MySQL and openly accessible Python libraries. In contrast to proprietary biometric attendance systems that necessitate continuous license costs and vendor lock-in, institutional IT staff can deploy, modify, and expand the current implementation without requiring outside assistance. The institutional investment in deployment infrastructure is safeguarded by the modular architecture, which guarantees that individual components, such the database backend or recognition algorithm, can be changed as technology advances without necessitating a whole system replacement.

CONCLUSION

This paper has shown a fully working, experimentally tested AI-Powered Attendance System that uses face detection and was made for use in schools at Chandigarh University. The system shows that it is possible to manage attendance automatically and with biometrics using only open-source software and standard laptop hardware, without the need for special biometric sensors, GPU computing resources, or cloud connectivity. The system uses OpenCV's Haar Cascade face detector, the LBPH recognition algorithm, and MySQL database management to create a complete, tamper-proof attendance workflow. This stops proxy attendance, cuts down on the need for manual recording, and gives reliable digital attendance records.

The experimental evaluation showed that the system works at practical frame rates of 20–25 FPS on standard hardware, with a recognition confidence of 73% and no false acceptances or duplicate database entries in any of the test sessions. These results show that the system is ready to be used in real classrooms instead of the old-fashioned way of taking attendance in the Department of Computer Science and Engineering at Chandigarh University and other similar schools.

Future development will focus on improving accuracy by adding lightweight deep learning recognition models, creating web-based administrative interfaces, adding liveness detection to prevent spoofing, and connecting with university ERP systems to make administrative workflows smoother. These improvements will gradually fix the system's current problems while keeping its main benefit: it can be used on standard educational hardware. This will make it easier for more schools around the world to use it in a variety of settings.

REFERENCES

[1] M. Adnan, A. Kaur, and P. Singh, "A Survey on Automated Attendance Systems in Educational Institutions," International Journal of Computer Applications, vol. 180, no. 5, pp. 12–17, 2018.
[2] "An Introduction to Biometric Recognition," A. Jain, A. Ross, and S. Prabhakar, IEEE Trans. Circuits Syst. Video Technol., vol. 14, no. 1, pp. 4-20, 2004.
[3] "Eigenfaces for Recognition," M. Turk and A. Pentland, J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

[4] "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," IEEE Trans. PAMI, vol. 19, no. 7, pp. 711-720, 1997; P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman.
[5] "Rapid Object Detection Using a Boosted Cascade of Simple Features," P. Viola and M. Jones, Proc. IEEE CVPR, vol. 1, pp. 511-518, 2001.
[6] "Face Description with Local Binary Patterns: Application to Face Recognition," IEEE Trans. PAMI, vol. 28, no. 12, pp. 2037-2041, 2006, T. Ahonen, A. Hadid, and M. Pietikainen.
[7] "FaceNet: A Unified Embedding for Face Recognition and Clustering," F. Schroff, D. Kalenichenko, and J. Philbin, Proc. IEEE CVPR, pp. 815-823, 2015.
[8] "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," Proc. IEEE CVPR, pp. 4690-4699, 2019; J. Deng, J. Guo, N. Xue, and S. Zafeiriou.
[9] R. Kaur and G. Singh, "A Survey on RFID Technology and Its Applications in Attendance Management," International Journal of Advanced Research in Computer Science, vol. 8, no. 3, pp. 234-239, 2017.
[10] "Face Detection Based on Multi-Block LBP Representation," Proc. IAPR/IEEE Int. Conf. on Biometrics, pp. 11-18, 2007; L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Li.
[11] "Attendance Management System Using Face Recognition," R. Patil and S. Kulkarni, International Journal of Advanced Research in Computer Science, vol. 9, no. 2, pp. 45-49, 2018.
[12] "Smart Attendance System Using Face Recognition on Raspberry Pi," V. Kumar, A. Sahni, and R. Bhatia, Proc. IEEE ICICS, pp. 120-125, 2019.
[13] "Comparative Evaluation of Face Recognition Algorithms for Attendance Management," R. Tiwari and A. Jain, International Journal of Computer Vision and Image Processing, vol. 9, no. 1, pp. 1–14, 2019.
[14] "Deep Learning vs Classical Methods for Face Recognition in Attendance Systems," M. Alnabhan, A. Al-Smadi, and M. Al-Fayoumi, Proc. IEEE ICIAfS, pp. 1-6, 2024.
[15] "Face Recognition Using Transformer Models," Computers, vol. 12, no. 1, pp. 15, 2025; K. Roumeliotis, N. Tselikas, and D. Nassis.
[16] "Challenges in Biometric Attendance Systems: A Systematic Review," A. Harris, B. Smith, and C. Jones, J. Educational Technology Systems, vol. 52, no. 3, pp. 312-334, 2024.
[17] OpenCV Documentation, Computer Vision Library v4.9, https://docs.opencv.org, 2024;
[18] Oracle Corporation's MySQL 8.0 Reference Manual, https://dev.mysql.com/doc, 2024;
[19] Y. Sun, X. Wang, and X. Tang, "Deep Learning Face Representation from Predicting 10,000 Classes," Proc. IEEE CVPR, pp. 1891-1898, 2014.
[20] "Face Recognition Systems: A Survey," Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, Sensors, vol. 20, no. 2, pp. 342, 2020.
[21] "OpenFace: A General-Purpose Face Recognition Library," B. Amos, B. Ludwiczuk, and M. Satyanarayanan, CMU Technical Report, 2016.
[22] "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," UMass Amherst TR 07-49, 2007, G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller.
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, vol. 25, pp. 1097-1105, 2012
[24] D. E. King, "Dlib-ml: A Machine Learning Toolkit," J. Machine Learning Research, vol. 10, pp. 1755-1758, 2009.
[25] "Attendance Verification Using Bluetooth Low Energy and Face Recognition," Proc. Int. Conf. Indoor Positioning, pp. 426-430, 2014, S. Zhao, B. Guo, Y. Yao, and Y. Zhang.
[26] "Comprehensive Review on Face Recognition with Deep Learning," M. F. Mridha et al., IEEE Access, vol. 9, pp. 156151-156170, 2021.
[27] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.
[28] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2011.
[29] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with OpenCV Library, O'Reilly Media, 2008.
[30] "Neural Network-Based Face Detection," IEEE Trans. PAMI, vol. 20, no. 1, pp. 23-38, 1998. H. Rowley, S. Baluja, and T. Kanade.
[31] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.
[32] "Deep Residual Learning for Image Recognition," IEEE CVPR, pp. 770-778, 2016. J. Sun, X. Zhang, S. Ren, and K. He.
[33] "You Only Look Once: Unified, Real-Time Object Detection," J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, IEEE CVPR Proceedings, pp. 779-788, 2016.
[34] "Deep Face Recognition," O. M. Parkhi, A. Vedaldi, and A. Zisserman, Proceedings of the British Machine Vision Conference (BMVC), pp. 1–12, 2015.
[35] "VGGFace2: A Dataset for Recognising Faces Across Pose and Age," Proc. IEEE FG, pp. 67-74, 2018; Q. Cao, L. Shen, W. Xie, O. Parkhi, and A. Zisserman.
[36] S. Li and A. Jain, Handbook of Face Recognition, Springer, 2011.
[37] "Histograms of Oriented Gradients for Human Detection," N. Dalal and B. Triggs, IEEE CVPR, pp. 886-893, 2005.

[38] X. Tan and B. Triggs, "Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions," IEEE Trans. Image Processing, vol. 19, no. 6, pp. 1635-1650, 2010.
[39] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," IEEE Multimedia, vol. 19, no. 2, pp. 4–10, 2012.
[40] "A Survey on Face Detection in the Wild," Computer Vision and Image Understanding, vol. 138, pp. 1–24, 2015; S. Zafeiriou, C. Zhang, and Z. Zhang.
[41] E. Learned-Miller et al., "Labeled Faces in the Wild: A Survey," Advances in Face Detection and Facial Image Analysis, Springer, pp. 189-248, 2016.
[42] "Face Recognition with MobileNet," J. Wang and Y. Yang, IEEE ICCV Workshops, pp. 1–8, 2017.
[43] "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Proc. IEEE CVPR, pp. 4510-4520, 2018; M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen.
[44] "CosFace: Large Margin Cosine Loss for Deep Face Recognition," H. Wang et al., IEEE CVPR, pp. 5265-5274, 2018.
[45] "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," Proc. IEEE CVPR, pp. 1701-1708, 2014; Y. Taigman, M. Yang, M. Ranzato, and L. Wolf.
[46] "Real-Time Attendance System Using Face Recognition and IoT," S. Park, J. Kim, and H. Lee, IEEE Access, vol. 8, pp. 159403-159412, 2020.
[47] "Personal Recognition Using Hand Shape and Texture," A. Kumar and D. Zhang, IEEE Trans. Image Processing, vol. 15, no. 8, pp. 2454-2461, 2006.
[48] M. Hassaballah and A. Awad, Deep Learning in Computer Vision: Principles and Applications, CRC Press, 2020.