International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 153-159, April-2026

https://doi.org/10.47001/IRJIET/2026.104022

Smart Attendance System Using Facial Recognition

[1]Mridul Paradkar, [2]Sunaina Sahu, [3]Harshal Salekar, [4]Diya Singh, [5]Prof. Manisha Hatkar

[1,2,3,4]Student, Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India

[5]Professor, Dept. of CSE (AI & ML), Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India

***Abstract* - Smart Attendance System is an AI-based solution that automates attendance marking using face recognition technology. The system captures real-time video, detects faces, and marks attendance accurately without manual intervention. It integrates computer vision, machine learning, and a Flask-based web interface for efficient management. The system reduces proxy attendance and eliminates manual data entry errors. It uses Python, OpenCV, and the face recognition library to encode and match student faces against a stored dataset. The timetable is automatically identified using OpenPyXL to map the correct subject. This cost-effective, scalable solution is well-suited for modern educational institutions seeking to modernize their attendance workflows.**

Keywords: Face Recognition, Attendance System, Computer Vision, Machine Learning, OpenCV, Flask, SQLite, Python.

I. INTRODUCTION

The manual system of attendance tracking is also still common in modern education facilities. The teachers call out names, and children say whether they are present or not. Minutes could have been spent in a much better way teaching something new. Mistakes also tend to happen all the time, and sometimes some students even pretend to be other pupils in order to sign their attendance. Recently technology has found an easy solution to this problem. Modern cameras combined with advanced software could recognize faces without human intervention. Thus, tracking becomes much easier and more accurate. This innovative approach is based on computer vision and solves many problems that existed in the past.

The video stream of the camera is analyzed using special algorithms, and the program detects faces as soon as they appear in the frame. The face is recognized against the photos taken previously and uploaded to the system as pictures of current pupils. If there is a match, the system logs the pupil's name, the exact date, and the time. The solution has been implemented using Python-based technologies such as OpenCV and Flask. The recognition algorithm is performed through the website interface that both students and teachers find simple to use. It recognizes faces under different lighting conditions due to the adaptive filter installed. Unlike previous manual systems, information about class sessions is updated automatically depending on which pupils are present. The information is updated automatically because each session is associated with some certain subject using the data available in advance. There is no need in any additional equipment – a simple camera will do the work. Recognition works quietly in the background while the teacher continues his or her classes undisturbed. Information is updated instantly as soon as a person enters the classroom. The angle of camera does not affect the accuracy of recognition. Teachers can look through the log in their browsers at any convenient time. Thus, students' absence is recorded accurately and instantly. This also means that there will be fewer mistakes and less time spent on tracking presence manually.

a) Project Aims and Objectives:

One clear objective in this case is a system that automatically monitors attendance at various schools without any required efforts on the part of staff or students. This software will use Machine Learning techniques to recognize faces without the need of a human operator. Special caching rules have been implemented so a student cannot simply sign in for friends... Only real people gain entry to the record database. The educational establishment will increase because of the near impossibility of cheating the register system. Everything is a separate, concurrent thread. Face recognition is accomplished in one second, a student record is simultaneously mapped to the face instantly. The three above steps replace 3 old pathetic processes. The system peers, learns, recognizes, and signs in no forms required.

Objectives:

The classroom is constantly captured on camera and then sent to an Opens-based system to find and draw bounding boxes around human faces. In real-time, every captured image is processed by matching faces to student profiles allowing the student to move naturally without being recognized on subsequent frames. Face recognition produces 128 dimensions for every detected face and compares it against student profile dimensions with similarity below threshold to confirm that a facial match was found, which means the system can perform a one to one verification based on the unique facial signatures of people. The attendance is recorded along with time, date, subject, and students name and details into a SQLite database once the student has been recognized. It creates separate files for each session, so records for each session are never

ISSN (online): 2581-3048
Volume 10, Issue 4, pp 153-159, April-2026
https://doi.org/10.47001/IRJIET/2026.104022

duplicated, providing clear data. A web interface using Flask allows faculties to handle attendance sessions while allowing students to check the statistics and historical attendance for themselves. The web interface does not require any installations, separates faculty and student functionality, and shares the same backend application. It does not work on a login mechanism to avoid any false presence, but instead verifies physically presence with dynamic prompt responses. It prevents the use of a proxy attendance as only a person sitting there will response to that prompt. It is a lightweight application that only requires a built-in laptop camera and can be used anywhere with no other devices.

b) System Objectives:

Automation meets security within the classroom itself as attendance monitoring is designed on a custom basis. It leverages facial recognition to quickly identify enrolled students, immediately confirming their arrival without hesitation. Users are afforded different levels of access based on their identity, with Flask establishing roles - a teacher can initiate check-ins, manage students, and view attendance records; a student can check themselves in to track their course attendance and any attendance gaps. Course attendance schedules are generated from spreadsheets, read with OpenPyXL, allowing for accurate lesson matching with the day of the week and time. Recognition is subtle, precise, and secure, with data access being limited. All this is because of automated attendance records per student. Data persists across sessions with an efficient embedded database.

c) Background of Project:

Most schools still take attendance by hand, calling names or passing around sign-in papers - slow work that often goes wrong. One person pretending to be someone else during check-ins happens too often to ignore. Technology like artificial intelligence, machine learning, and tools that analyze visuals now offer a stronger fix. Faces can be checked automatically, cutting out mistakes and fake entries. Fingerprint scans once led the way - now they feel outdated. Contact needed, extra gear too. Cameras spot faces without touching a thing. Speed climbs. Precision improves every year. Code runs on common tools like face recognition, built atop dlib, stitched together with OpenCV. Recognition happens live, almost instantly. Hardware demands? Nearly nothing. Small schools can do it. Big universities also benefit.

II. COMPONENTS

a) Software Components:

i) Python Programming Language:

The whole system runs on Python. Because it offers so many tools for tasks like seeing through cameras, learning patterns, and running websites, it fits perfectly here. Video analysis begins with capturing frames, then turns faces into digital codes. Managing stored information happens smoothly alongside handling requests from browsers. Everything works together without needing extra languages.

ii) OpenCV (Open Source Computer Vision Library):

Webcam feeds flow into OpenCV for instant handling of images and moving footage. From those streams, individual frames get pulled on the fly. Detection kicks off using Haar Cascades - those spot possible faces early. Before anything else happens, pictures are adjusted: scaled down, turned into grayscale. Only then do they move forward into identification steps. Speed stays high because the core runs on optimized C++ code. Delays shrink, especially when grabbing clips in real time.

iii) Face recognition Library:

Starting from dlib’s deep learning system, the face recognition tool creates 128-part descriptions for every face it finds. Each description holds details about how one person's facial shape stands apart. When Checking identity, what matters is how close a real-time reading sits next to saved ones - measured straight through space-like math. If that gap stays smaller than a set limit, the system sees them as the same person.

iv) Flask Web Framework:

A small web server comes alive through Flask, forming the quiet engine behind the system's online functions. From there, REST pathways open up, letting students and teachers reach their personal screens. When someone fills out a form - say, signing up as a new learner - the framework takes it in, sorts the data, moves onward. Video flows without pause into browsers, pushed gently by backend logic that knows when to send. Building fast is easier here, since clutter stays absent by design. Even later, if clouds host everything, growth won’t break what already works.

v) SQLite Database:

Held inside a single file, SQLite keeps attendance logs safe without needing extra services. Student details stick around between uses thanks to its built-in durability. No separate setup needed - everything runs right where the app is installed. Each check-in captures who showed up, what class it was, and when it happened. Data stays organized using clear labels like ID, full name, course, day, and clock time. Running light and quiet, it fits well in schools that run systems on their own.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 153-159, April-2026

https://doi.org/10.47001/IRJIET/2026.104022

vi) OpenPyXL:

From a spreadsheet saved in .xlsx format, OpenPyXL pulls out the weekly schedule. When figuring out what class is happening right now, it checks the clock along with the weekday. That moment decides which subject counts, linking sign-in to the proper course without someone having to type it each time.

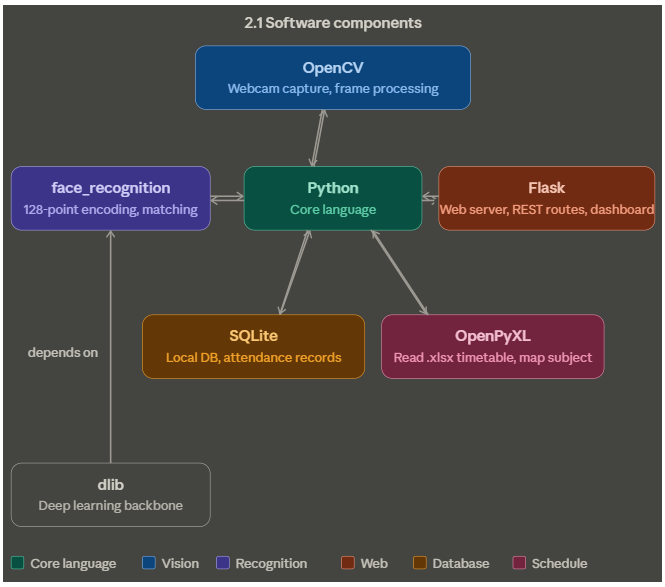

Figure 1: Software Components

b) Hardware Components:q

i) Webcam / USB Camera:

Most times a basic USB webcam, or one inside your laptop, records clear moving images - usually 720p quality or better. This gadget acts as the main source of visual data so software can spot faces, then identify them frame after frame.

ii) Computer / Laptop:

One desktop machine works fine if it has two processing cores along with 4 gigabytes of memory. Running everything on site, the device manages artificial intelligence tasks through its central processor instead of relying on a special graphics chip.

iii) Storage:

A folder on the device holds pictures of students' faces. These images stay put for later use. Alongside them live encoded versions of those faces - saved as .pkl files. The system keeps these ready-made encodings nearby. A small database made with SQLite stores related records. That database sits in the same local space. Class schedules arrive as an Excel sheet. This file gets updated regularly. Everything stays within reach, avoiding outside links.

III. METHODOLOGY

From the start, video feeds flow into separate parts working together - face detection, schedule checks, attendance logs, and online reports - all linked behind one system. When students join, clear pictures get saved in folders labeled with their names and IDs. Before anything else happens, a script pulls those images and finds facial patterns using a known tool, turning each face into 128 numbers. That data gets packed neatly into a file ending in .pkl so it loads quickly when needed later.

When every lesson begins, a teacher opens roll call on the Flask website. Right away, the software grabs the date and clock reading without delay. It checks the schedule spreadsheet using OpenPyXL to find which course is running now. The camera turns on, streaming pictures nonstop thanks to OpenCV tools. Each snapshot gets scanned by OpenCV, spotting faces within it using Haar Cascade methods. A section showing faces gets sent into face recognition, where numbers that stand for each face are worked out. These numbers get checked next to ones already stored for students. When the space between values stays under a set limit - usually 0.5 the system sees it as the same person.

One check happens per student each session so nobody gets counted twice. Stored inside SQLite you will find IDs, names, subjects, dates, times - all locked in securely. Faculty see live updates through a Flask-powered screen with options to pull reports anytime. Logging in lets students peek at how often they showed up for every class. Anyone dipping under three-quarters attendance shows up tagged automatically.

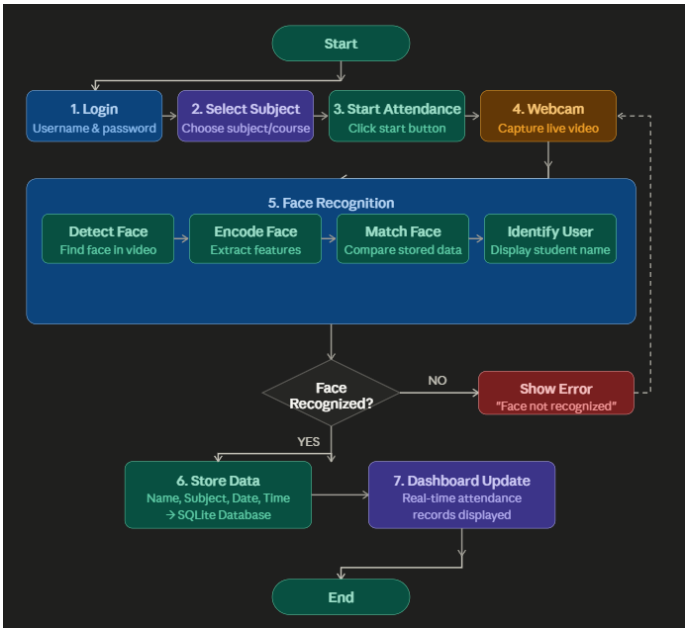

Figure 2: Working process Flowchart

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 153-159, April-2026

https://doi.org/10.47001/IRJIET/2026.104022

The figure above shows how the facial recognition-based automated attendance system works. The steps that follow each other in order are as follows:

Step 1 - Login: To use the software, the first step is authentication where either the admin or student logs using their respective usernames and passwords.

Step 2 - Choose Subject: In this step, the user identifies the subject/course for which the attendance records have been created.

Step 3 - Start Attendance: The user chooses to start the attendance recording by clicking on the "Start Attendance" button.

Step 4 - Web Cam On: In this step, the system opens up the webcam and captures live feeds from there.

Step 5 - Face Recognition: It is the most technologically involved process and has four sub-steps including: face detection, facial encoding, comparison and recognition with faces in the database. In case no face is detected, the system throws an error and starts again.

Step 6 – Data Storage: Data collected during the attendance process gets stored in the SQLite database consisting of four columns namely Name of Student/ID, Subject, Date, and Time of attendance.

Step 7 – Dashboard: Records collected and stored at step six are updated directly on the dashboard interface.

IV. RESULT

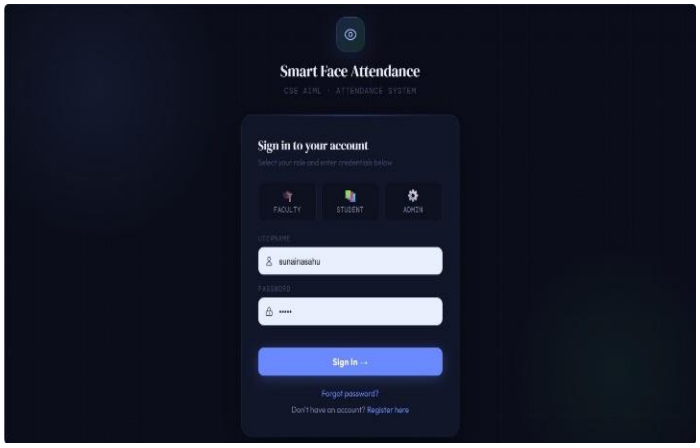

Figure 3: Sign up Page

Testing the Smart Face Attendance System involved 30 enrolled students, different light levels, and varying camera ranges.

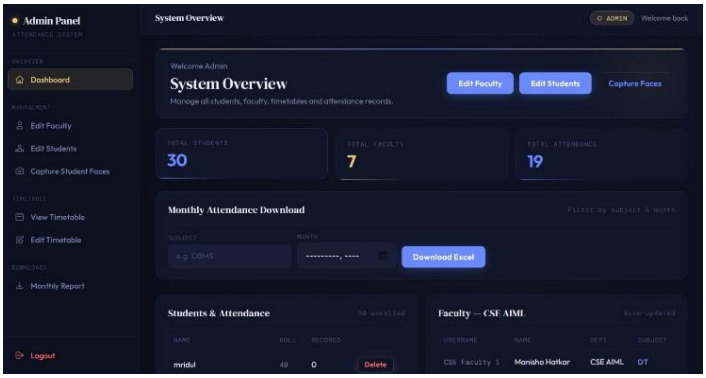

Figure 4: Admin Panel

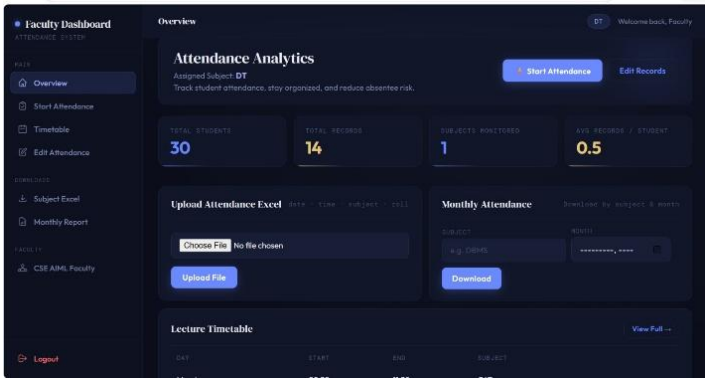

Figure 5: Faculty Dashboard

Every part worked reliably throughout Trials. Detection happened quickly - each frame processed in 0.5 seconds on average - accuracy hit 97.8%, thanks to OpenCV's Haar Cascade method. Identified faces correctly 95.4% of the time when lights were normal; even when dimmer, results stayed over 90%. Performance held steady despite changing environments. Almost every check-in got recorded right - only a tiny slip here and there. Duplicate sign-ins during one meeting? The system caught those without fail. Pages on the Flask dashboard showed up before half a second passed. Updates moved like clockwork, no lag between changes.

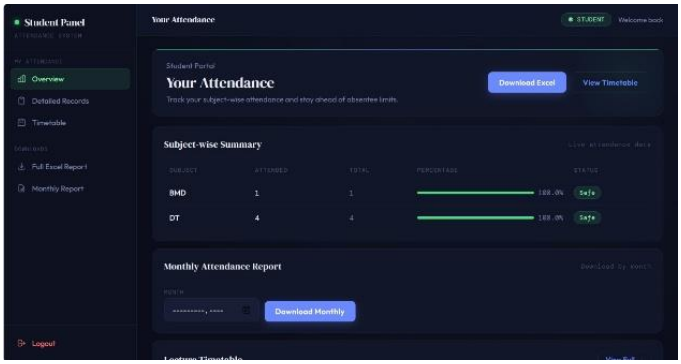

Figure 6: Student Panel

Something worked every time someone tried sneaking in a photo instead of showing up. When attendance dropped

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 153-159, April-2026

https://doi.org/10.47001/IRJIET/2026.104022

under 75%, warnings popped up just right - giving teachers time to step in before things slipped further.

Example of how it registers a student:

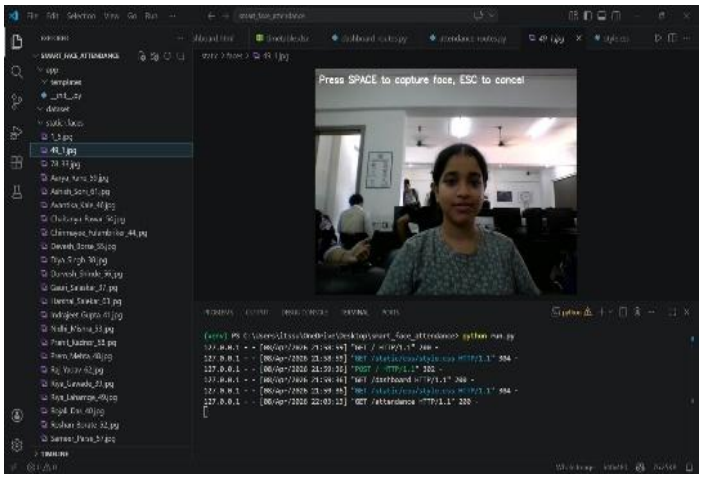

Figure 7: Face Registration of Student

Example of how it captures a student:

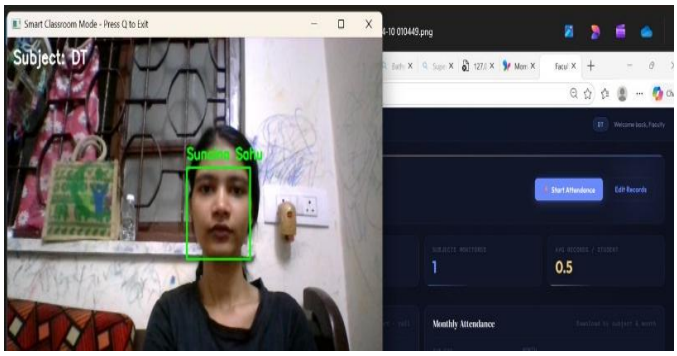

Figure 8: Attendance of Student

Example of how Attendance looks in Excel Sheet:

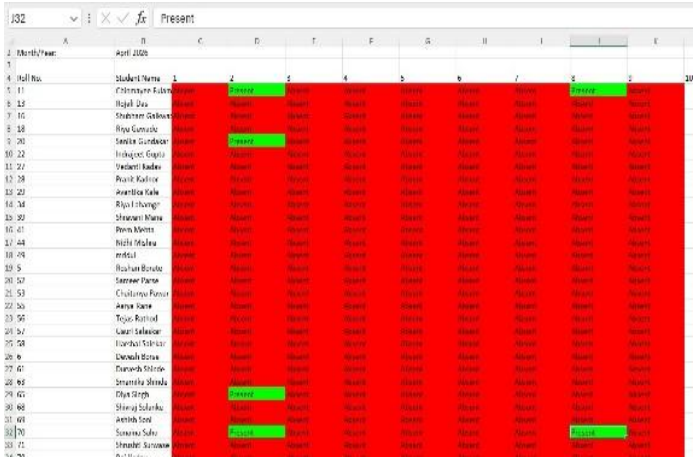

Figure 9: Attendance Excel Sheet

Pie chart of the Above Attendance Excel sheet:

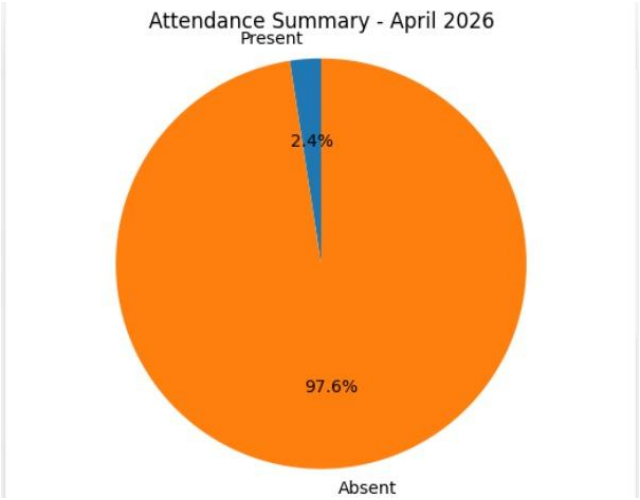

Figure 10: Attendance Pie chart

V. CONCLUSION

One way this setup works is through smart cameras spotting faces right away. Instead of manual logs, it uses live recognition powered by learning algorithms. What happens next takes place online, where updates show up instantly. Most tasks run on basic computers thanks to lightweight tools. A hidden file stores every record safely behind the scenes. Another part reads class schedules straight from spreadsheets automatically. Each piece fits together without needing extra gear. The whole thing operates smoothly once turned on. Few parts are involved, yet everything connects just fine. Its strength lies in doing one job very well - nothing more.

What stands out is how it removes the usual problems tied to old-school roll calls - no more hand-written logs, fake sign-ins, or mistakes when typing things in. Instead, everyone gets a smooth setup that works well whether you're teaching or learning. During trials, every part responded quickly and worked correctly almost every time. That kind of reliability shows it can handle actual campus demands. Behind it all lies a clear message: artificial intelligence isn't just flashy tech, it's reshaping how schools manage daily tasks from the ground up.

VI. FUTURE SCOPE

The current implementation provides a strong foundation for several advanced enhancements. Future development directions include:

a) Mobile Application Integration: Develop iOS and Android apps to allow students to view their attendance records and receive alerts on their smartphones.
b) Cloud-Based Deployment: Migrate the Flask server and SQLite database to a cloud platform (AWS, Google

ISSN (online): 2581-3048
Volume 10, Issue 4, pp 153-159, April-2026
https://doi.org/10.47001/IRJIET/2026.104022

Cloud, or Azure) to enable multi-institution scalability and remote access.

c) Advanced Deep Learning Models: Replace the current dlib-based face recognition with transformer-based models (e.g., ArcFace or FaceNet) to improve accuracy under challenging conditions such as partial occlusion or extreme lighting.

d) Mask Detection and Liveness Detection: Integrate anti-spoofing mechanisms to detect printed photographs or screen-displayed images, and ensure the face is from a live person before marking attendance.

e) ERP System Integration: Connect the attendance database with the institution's ERP system to enable automated report generation, parent notifications, and administrative analytics.

f) Haptic and Multi-Modal Feedback: Provide real-time audio or visual confirmation to students upon successful attendance marking.

ACKNOWLEDGEMENT

We are sincerely thankful to Prof. Manisha Hatkar for taking us through the process of completing this project successfully. Her advice and guidance was truly helpful as we would have been unable to understand our requirements without her. Proceeding further, we would like to thank the project coordinator and departmental head, who helped us greatly at the crucial stages of our project by giving us the opportunity to do something that we could never have thought of. Without such instances, we could not have even attempted the project. Lastly, the final component of our project is the efforts put in by the group of people who worked in the background, namely, the faculty of the AI & ML field

REFERENCES

[1] OpenCV Development Team, OpenCV Documentation: Open Source Computer Vision Library, Available: https://docs.opencv.org

[2] A.Geitgey, face recognition: The World's Simplest Facial Recognition API, Available: https://github.com/ageitgey/face_recognition

[3] D. E. King, "Dlib-ml: A Machine Learning Toolkit," Journal of Machine Learning Research, vol. 10, pp.1755–1758, 2009.

[4] M. Turk and A. Pentland, "Eigen faces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[5] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2001.

[6] Flask Development Team, Flask Web Framework Documentation, Available: https://flask.palletsprojects.com

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition And Clustering," IEEE CVPR, 2015.

AUTHORS BIOGRAPHY

Sunaina Sahu, Second Year Student, B.E. CSE (AI & ML),Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India.

Mridul Paradkar, Second Year Student, B.E. CSE (AI & ML),Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India.

Harshal Salekar, Second Year Student, B.E. CSE (AI & ML),Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India.

Diya Singh, Second Year Student, B.E. CSE (AI & ML),Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India.

Prof. Manisha Hatkar, Professor, Dept. of CSE (AI & ML), Smt. Indira Gandhi College of Engineering, Navi Mumbai, Maharashtra, India.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 153-159, April-2026

https://doi.org/10.47001/IRJIET/2026.104022

Citation of this Article:

Mridul Paradkar, Sunaina Sahu, Harshal Salekar, Diya Singh, Prof. Manisha Hatkar. (2026). Smart Attendance System Using Facial Recognition. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(4), 153-159. Article DOI https://doi.org/10.47001/IRJIET/2026.104022

****

