ISSN No. 0976-5697

VOLUME 17 SPECIAL ISSUE 1, JULY 2026

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Automating the Attendance System by Face Recognition in Rural Schools Reducing Time and Errors also Making Record Remotly Available

Vijendra Kumar Maurya
Department of Computer Science and Engineering
Geetanjali Institute of Technical Studies (GITS)
Dabok, Udaipur, Rajasthan, India
maurya.vijendra@gmail.com

Rajendra Singh Rao
Scholar Department of CSE
Geetanjali Institute of Technical Studies (GITS)
Dabok, Udaipur, Rajasthan, India
rajendrasinghmanawat@gmail.com

Mahesh Rawal
Scholar Department of CSE
Geetanjali Institute of Technical Studies (GITS)
Dabok, Udaipur, Rajasthan, India
maheshzrawal1333@gmail.com

Nikunj Paliwal
Scholar department of CSE
Geetanjali Institute of Technical Studies (GITS)
Dabok, Udaipur, Rajasthan, India
nikunjpaliwal1212@gmail.com

Abstract: First thing every teacher typically does when they get to school is taken attendance for their first class of the day, and this is still done on a piece of paper, completely manually, and takes a large amount of time for the teacher to decide if a student is present or not. Furthermore, many rural schools continue to use these old ways of taking attendance (In paper Register). As we all know, these types of methods are slow, they can be easily messed up, and therefore very difficult to manage. This research paper provides a method to take attendance automatically using a system developed for rural school systems. Our method can be implemented by any rural school, even those who don't currently have any advanced teaching support or technology. Our system will require cameras to monitor students in classrooms, taking live video of the students and processing this video to identify which students are in class Room based on their identity in the school's database, and automatically updating the database of student attendance.
Implementing this method will eliminate the need for manually taking attendance of classes and save time for both teachers and students. This method will also reduce paper work and will have accurate and more secure records of student attendance maintained for many years. This research paper will provide a solution for the Automatic Attendance System at a very low cost to rural school.

Keywords: Automatic Attendance System, Face Recognition, TensorFlow, Deep Learning, Artificial Intelligence/Machine Learning, Real-time Face Detection, Image Processing, API, Frameworks, Libraries.

I. INTRODUCTION

The education sector, particularly in rural areas, continues to face a significant gap in adopting modern technological solutions, as many schools still rely on traditional paper-based methods for tracking attendance despite the widespread use of advanced tools in other sectors to improve productivity. This gap is largely due to barriers such as limited awareness of technological advancements, lack of funding, and the absence of technical support, making it difficult for rural institutions to adopt high-cost solutions like fingerprint-based systems, which also require multiple devices to manage large student queues efficiently. To address these challenges, a more accessible and cost-effective software-centric approach is proposed in the form of an Automated AI/ML-based face detection attendance system. By leveraging existing infrastructure such as local IP cameras, this system eliminates the need for expensive hardware and transforms simple cameras into efficient attendance tools. The automation of attendance not only reduces manual effort and saves time but also ensures secure data management, thereby enabling rural schools to take an important step toward digital transformation and modernization in education.

The Gap Between Rural Schools and Modern Problem Statement

II. PROBLEM STATEMENT

Here are some technologies and their problems.

2.1.1 Manual Way (Classroom Problem)

i. Manual way consumes more than 10 minutes along with high chances of proxies and errors.
ii. This disturbs the flow while teaching.
iii. Causes indiscipline while taking attendance.
iv. Also students are not sure that there attendance is marked and cannot track it.

2.1.2 Manual Way (Availability issue)

I. Records are stored as Hard-Copy only and exist only copy.
II. No way to access it Remotely.
III. Less transparency as someone could alter the attendance.

2.1.3 Manual Way (summing the attendance)

I. While summing the attendance there is high chance of error and is very time taking.
II. Proper maintaining of hard copy attendance is required.

2.2 Identity Proof

© 2025-2027, IJARCS All Rights Reserved

I. In schools the main and almost the only identity proof is only the ID card.
II. These cards can be lost, broken, altered.
III. So automated system like RFID also not so reliable.

2.3 Other automated system

Like fingerprint, QR scanner, RFID marks attendance one-by-one, so they also become time taking

III. OBJECTIVE OF THE PROJECT

Following objectives are given

3.1 Automation of Processes

The goal of using Machine Learning to replace outdated aper register book for marking attendance.

3.2 Affordable for Rural Schools

By developing a high-quality recognition model using the TensorFlow tool and local camera hardware, we hope to create a low-cost but effective solution to automatic attendance taking system for rural school.

3.3 Time Efficiency

Reduce the time spent on roll calls, allowing teachers to maximize instructional time. This will help teachers to save time for conducting their classes.

IV. SOFTWARE REQUIREMENT SPECIFICATION

4.1 *Development Tools –*

I. VS code – code editor
II. NPM – dependency management for node.js
III. Pip – Python package manager
IV. Git – version control

4.2 Programing Languages –

I. Python 3.10/3.11 – for Anti-Spoofing Model
II. JavaScript (ES6+) – for frontend and backend logics
III. HTML & CSS – for web interface

4.3 Development Frameworks –

IV. FastApi – python API server for anti-spoofing
V. Node.js – Web server and API management
VI. Express.js – for data handling in backend
VII. Websocket – for real-time communication between python(anti-spoofing) and javascript(face recognition).
VIII. React.js – user interface and getting video-feed.

4.4 Python Libraries –

I. OpenCV – image processing and frame handling
II. NumPy – numerical computations
III. PyTorch – deep learning model execution
IV. Torchvision – image transformation and pretrained models
V. Pillow – images manipulation
VI. FastAPI – REST/Websocket API creation
VII. Uvicorn – ASGI server for running FastAPI

4.5 Browser Requiremnet –

I. WebRTC
II. Websockets
III. HTML5 video

V. DETAILED LIFE CYCLE OF THE PROJECT

5.1 Requirement Gathering

This project starts by gathering the requirements for the face anti-spoofing system. This involved identifying the technical requirements necessary for implementing a real-time face authentication system with anti-spoofing capabilities.

The face anti-spoofing system must be capable of capturing real-time video streams from the users camera using browser APIs. The face anti-spoofing system must detect faces in the captured video stream. The face anti-spoofing system must extract frames and transmit them to the backend for verification. The face anti-spoofing system must perform liveness detection to identify attempts such as photos, screen displays or prerecorded videos. The face anti-spoofing system must return an fake classification result to the client interface. The face anti-spoofing system must allow face recognition only after successful liveness verification.

Non-Functional Requirements for the face -spoofing system include real-time processing capability, low latency communication between services, scalability for handling multiple concurrent users and secure data transmission between client and server.

The requirements analysis also determined the need for a -service architecture where browser-based applications interact with backend services through WebSocket communication.

5.2 System Design

The face anti-spoofing system architecture is designed as a three-layer distributed architecture consisting of the client layer, application gateway and machine learning processing layer.

The client layer operates within the web browser. Is responsible for capturing the camera feed using WebRTC APIs. Frames from the video stream are extracted using the HTML Canvas API. Captured frames are encoded into Base64 image format. Transmitted to the backend server using WebSocket communication.

The application gateway layer is the Node.js server that acts as an intermediary between the client and the machine learning module. This layer performs tasks such as managing WebSocket connections from clients receiving image frames from the browser forwarding frames to the Python -spoofing service and returning classification results to the frontend.

The machine learning processing layer is the -spoofing module implemented as a Python WebSocket service. The service performs operations such as receiving image frames from the gateway server decoding Base64 image data converting images into NumPy arrays performing preprocessing using OpenCV, running -spoofing inference and returning classification results.

5.3 Development

The face anti-spoofing system development followed an Agile methodology, where the project was implemented in stages with continuous testing and refinement.

The development was divided into modules

I. Iteration 1 was camera integration.
II. Iteration 2 was frame extraction and transmission.
III. Iteration 3 was WebSocket gateway implementation.
IV. Iteration 4 was -spoofing service development.
V. Iteration 5 was system integration.

© 2025-2027, IJARCS All Rights Reserved

5.4 Testing and Validation

Testing was performed to ensure functionality and system stability under real-time conditions. Functional testing verified system functionalities including camera initialization, frame capture and transmission WebSocket communication and anti-spoofing inference results. Performance testing measured system latency and processing time to ensure real-time operation validation testing evaluated the -spoofing module using different input conditions such as live human faces, printed photos and digital images displayed on screens.

5.5 Deployment

The face anti-spoofing system is deployed as a -service web application consisting of two backend services and a browser-based client interface. Following development components are uses.

I. Client Interface: Web Browser
II. Gateway Server: Node.js
III. Anti-Spoof Service: Python
IV. Communication: WebSockets

5.6 Maintenance and Updates

Post-deployment maintenance ensures that the face anti-spoofing system remains operational, secure and efficient.Maintenance activities include updating machine learning models to improve detection accuracy fixing bugs and resolving system errors updating dependencies and libraries and monitoring server performance and network latency.

Future updates may include integrating liveness detection techniques, improved face recognition algorithms and enhanced system scalability, for large-scale deployments of the face anti-spoofing system.

VI. SYSTEM PLANNING

System planning for the proposed face authentication system that prevents spoofing involves defining how it will be set up allocating computer resources estimating infrastructure costs and coordinating with system stakeholders. The system is designed as a web-based architecture with three layers: client interface, gateway server and machine learning service.

6.1 Identification of Target Regions

The system is designed for use in web-based authentication environments like login portals, attendance systems or access control platforms. The pilot deployment targets. Institutions with camera-enabled devices and stable internet connectivity.

The system operates entirely through a web browser. Therefore requires

I. A device with a browser that supports WebRTC
II. A web camera for video capture
III. Internet connectivity for WebSocket communication

During pilot deployment the system is expected to operate in environments such as

I. institutions
II. Secure login portals
III. Identity verification systems

The system architecture allows deployment on local servers or cloud-based infrastructure enabling scalability for multiple users and concurrent authentication requests.

6.2 Resource Allocation

The proposed system follows a -service architecture where different modules handle specific tasks.

Client-Side Resources (Frontend).The client interface is built using web technologies. Runs directly in the users browser. Key components include are

I. HTML5 video element for real-time camera feed
II. Canvas API for capturing frames from the video stream
III. WebSocket client for transmitting captured frames to the backend

Frames are extracted every 500 milliseconds from the video stream using the Canvas API and converted to Base64 encoded JPEG images before transmission.

Required technologies:

I. HTML5
II. JavaScript
III. WebRTC (for camera access)
IV. WebSocket protocol

Gateway Server (Node.js)

The Node.js server acts as a communication gateway between the frontend client and the Python anti-spoofing service.

Responsibilities of the Node.js layer include:

I. Handling WebSocket connections from clients
II. Forwarding frames to the Python anti-spoofing service
III. Returning classification results back to the client

The gateway server is implemented using:

I. Node.js runtime
II. WS WebSocket library

The gateway architecture ensures that the Python ML service remains isolated allowing scaling and better system stability.

Machine Learning Service (Python Anti- Module)

The anti-spoofing detection module is implemented in Python and runs as a separate WebSocket service.

6.3 Budget Estimation

The estimated budget for the pilot implementation of the face authentication system that prevents spoofing is ₹16 lakhs, which includes infrastructure setup, server resources, development and operational costs.

The major cost components include

I. Component: Server infrastructure (GPU/CPU servers) Cost: ₹6,00,000
II. Component: Camera and hardware equipment, Estimated Cost: ₹2,00,000

© 2025-2027, IJARCS All Rights Reserved

III. Component: Software development and integration Estimated Cost: ₹4,00,000
IV. Component: Machine learning model training Estimated Cost: ₹2,00,000
V. Component: Deployment and maintenance Estimated Cost: ₹2,00,000

The infrastructure budget accounts for high-performance servers capable of processing real-time video frames and machine learning inference.

6.4 Stakeholder Involvement

The successful deployment of the face authentication system that prevents spoofing requires coordination among stakeholders.

Technical Development Team responsible for:

I. Developing frontend and backend components
II. Integrating the machine learning -spoofing module
III. Maintaining the server infrastructure

System Administrators for:

I. Server configuration
II. Network security
III. Monitoring of services

End Users

End users interact with the face authentication system that prevents spoofing through a web browser and provide camera input for authentication.

Institutional Authorities / Organizations

Organizations deploying the face authentication system that prevents spoofing are responsible for:

I. Providing infrastructure support
II. Ensuring compliance with data privacy regulations
III. Monitoring operational performance

Collaboration, among these stakeholders ensures deployment, efficient system maintenance and continuous improvement of the face authentication system that prevents spoofing.

VII. SYSTEM DESIGN

1. Data Acquisition Layer

The Data Acquisition Layer is what gets the video from the users device. It uses the browser to get to the camera and gets a video stream from the users webcam. This video stream is shown in a video player on the webpage. Each frame of the video is taken out and used. The Data Acquisition Layer uses the Canvas API to get these frames. Each frame is changed into a format so it can be sent over the internet. This happens at intervals so the system does not get too busy. The Data Acquisition Layer uses things like HTML5 Video API and WebRTC camera access and Canvas API and JavaScript image encoding. This layer makes sure the system always gets the video it needs to check if someone is real.

2. Communication Layer

The Communication Layer is what helps the different parts of the system talk to each other. It sends the video frames from the users browser to the server in time. The system uses something called WebSocket to do this. WebSocket is better than ways because it lets the system send video frames quickly without having to ask for them all the time. The server gets these frames. Sends them to another service that checks if someone is real or not. This layer has to do a things

I. It has to handle users at the same time
II. It has to send video frames in time
III. It has to send the video to the service that checks if someone is real
IV. It has to send the result to the user

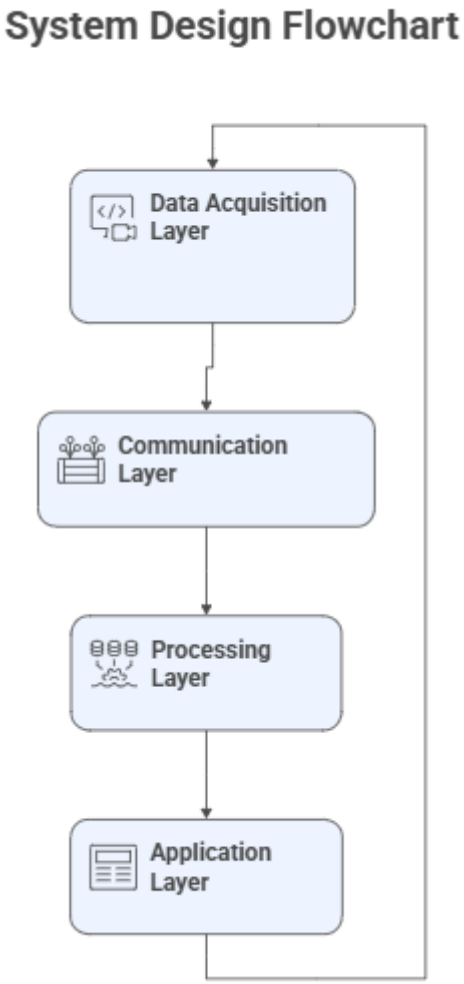

Fig7- System Design Cycle

3. Processing Layer

The Processing Layer is what checks if someone is real or not.It gets the video frames from the server. Does a few things:

- It changes the video frames back into pictures
- It changes these pictures into a format that the computer can understand
- It uses a library called OpenCV to get ready for the check
- It runs a check to see if someone is real or not
- It says if the person is real or not

© 2025-2027, IJARCS All Rights Reserved

This layer is the core of the system and uses things like Python and OpenCV and NumPy and AsyncIO and WebSocket server. It checks the video. Says if someone is real or trying to trick the system.

4. Application Layer

The Application Layer is what the user sees and talks to.it lets the user do a thing:

I. It lets the user allow the system to use the camera
II. It shows the user the video from the camera
III. It shows the user if someone is real or not

When the system checks a video frame it sends the result back to the users browser. Shows it on the webpage. The webpage shows the user if the person is real or not so the user can be sure they are safe. This layer uses things, like JavaScript and React.js and HTML5 and CSS.It makes sure the user can talk to the system and get the results in time.

VIII. TECHNICAL IMPLEMENTATION AND CODING

8,1. Camera Integration and Frame Acquisition

The client-side application gets video from the users webcam using WebRTC. It shows the video inside an HTML5 video element.

Frames are taken from the video now. Then using the HTML Canvas API. These frames are turned into Base64 encoded JPEG images to send to backend services easily. Here are the key technologies used

I. HTML5 Video API
II. camera access
III. JavaScript Canvas API
IV. Base64 image encoding

The frame capture system keeps sending image samples to the backend without using much processing power.

8,2. Real-Time Communication using WebSockets

The system uses WebSockets for communication between the client and backend servers. A Node.js WebSocket server gets image frames from the browser client. Sends them to the Python anti-spoofing service. Using WebSocket's is good because it has following advantage.

I. It has latency
II. It streams image frames continuously
III. It handles real-time data well

The Node.js server uses the ws WebSocket library to manage client connections at once.

8.3. Backend Gateway Implementation (Node.js)

The Node.js server is like a messenger between the browser and the machine learning service. It does these things

IV. Accepts connections from clients
V. Gets Base64 encoded frames from the browser
VI. Sends frames to the Python -spoofing service
VII. Gets classification responses
VIII. Sends results back to the client

This setup separates the client app from the machine learning service making the system more flexible and able to handle users.

8.4. Anti-Spoofing Processing Service (Python)

The main detection part is a Python WebSocket server that does image processing and spoof detection.

When it gets an image frame it:

I. Decodes the Base64 image
II. Turns the image into a NumPy array
III. Preprocesses the image using OpenCV
IV. Runs the anti-spoofing detection
V. Says if the frame is Live or Spoof

The Python service uses communication to handle many frame requests at the same time.

Key libraries used:

I. OpenCV (cv2) for image decoding and preprocessing
II. NumPy for data processing
III. AsyncIO for the event loop
IV. WebSockets library for communication

8.5. Frontend User Interface

The frontend is made with JavaScript. Shows the camera feed and system output in real-time.Users can:

I. Turn on the webcam
II. See the video feed
III. Check anti-spoofing results

When the Python service decides if a frame is live or spoof it sends the result back, to the browser through the Node.js server and shows it to the user. Technologies used-

I. HTML5
II. JavaScript
III. CSS
IV. WebSocket client API

IX. RESULT AND DISCUSSION

9.1 Unit Testing

We did unit testing to check if each part of the system works correctly. We tested each part on its own to make sure it works properly before we put everything together.

Here are some examples of the unit tests we did:

I. We tested if the camera starts up correctly so the browser can use the webcam with WebRTC APIs.
II. We tested if we can get frames from the video stream using the Canvas API.
III. We tested if we can turn frames into Base64 format correctly.
IV. We tested the Python part to make sure it can decode and process images using OpenCV correctly.

Unit testing helps us make sure each part of the system does what it is supposed to do without any mistakes.

© 2025-2027, IJARCS All Rights Reserved

9.2 Integration Testing

We did integration testing to check how different parts of the system work together.

I. How the browser client talks to the Node.js WebSocket server
II. How the Node.js gateway sends data to the Python -spoofing service
III. If we get the classification results from the Python service

We paid special attention to making sure frames are sent and received correctly in real-time over the WebSocket pipeline.

Integration testing helps us make sure all parts of the system work together correctly and stay in sync when things are happening in time.

9.3 System Testing

System testing checks the application to see how it works under real conditions.

We did the following system-level tests:

I. We tested the process from when the camera captures a picture to when we get the final classification result.
II. We tested how long it takes to get a response after we send a frame.
III. We tested if the system can handle users at the same time.
IV. We tested what happens when the network is interrupted. We send invalid image data.

System testing showed us that the whole system works reliably and meets the performance requirements.

9.4 Real-Time Performance Testing

We did real-time performance testing to see how well the system can process video frames.

We tested the system with types of input such as:

I. Live human faces
II. Printed photographs
III. Images on computer screens

The system was able to tell if frames were live or spoofed which shows that the anti-spoofing detection pipeline works well.

Performance testing also showed us that the system can process frames at intervals, without significant delay, which gives users a smooth real-time experience.

X. PROJECT SCREENSHOTS

1. Asking for camera permission

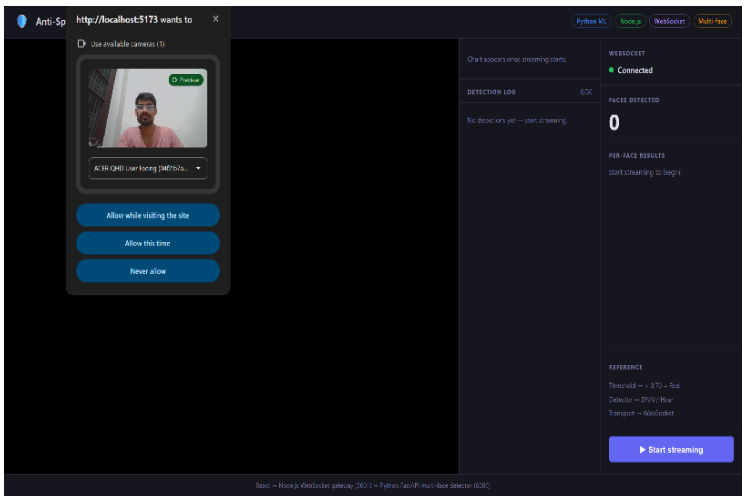

Fig 10.1 while first visit it asks for camera permission

2. Landing UI

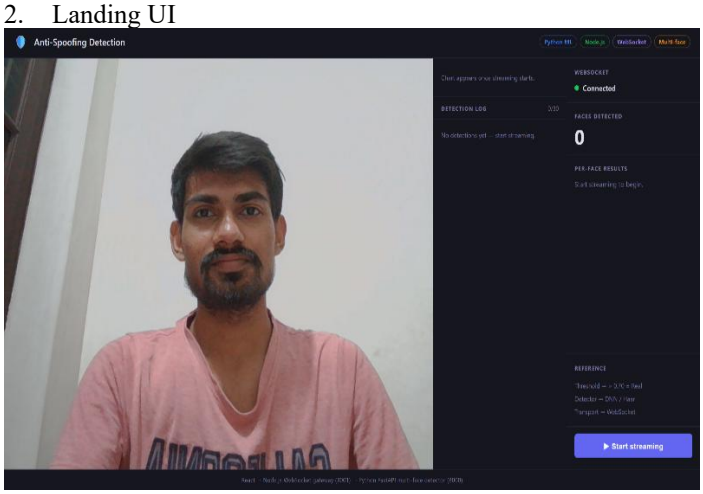

Fig 10.2 after giving the camera permission (landing page)

3. After clicking on “Start streaming” it start detecting faces

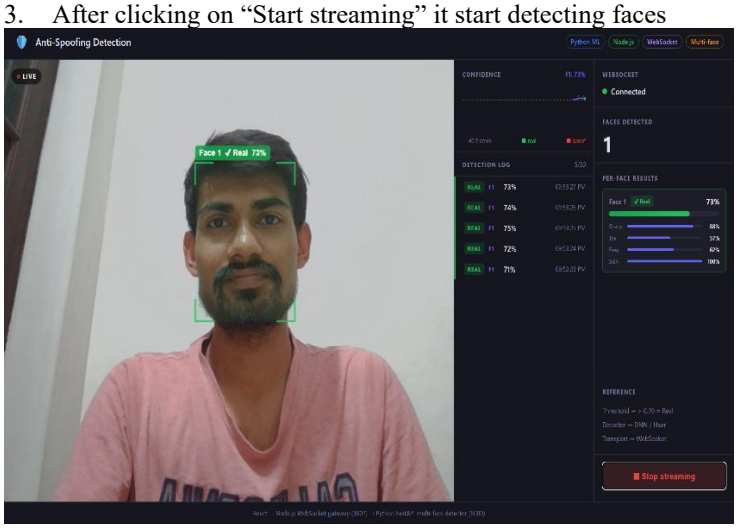

Fig 10.3 detecting faces from camera feed (since it is real face it is showing real)

4. Working with multiple face and Working of antispoofing

i. For a face to be real the minimum threshold is 0.70 confidence of the model which we used.
ii. 0.70 is strict number so that we can prevent system from detecting a real face from a spoof photo even in a single frame

© 2025-2027, IJARCS All Rights Reserved

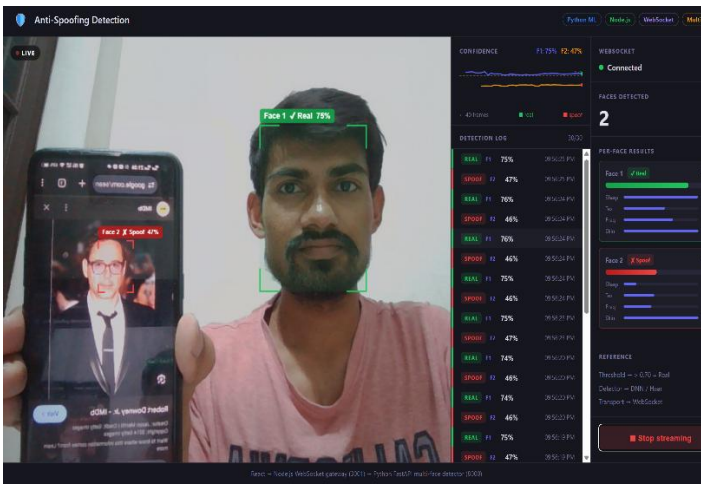

Fig 10.4 detects multiple faces and differentiate between real and spoof faces

XI. CONCLUSION

The education sector, particularly in rural areas, continues to face a significant gap in adopting modern technological solutions, as many schools still rely on traditional paper-based methods for tracking attendance despite the widespread use of advanced tools in other sectors to improve productivity. This gap is largely due to barriers such as limited awareness of technological advancements, lack of funding, and the absence of technical support, making it difficult for rural institutions to adopt high-cost solutions like fingerprint-based systems, which also require multiple devices to manage large student queues efficiently. To address these challenges, a more accessible and cost-effective software-centric approach is proposed in the form of an Automated AI/ML-based face detection attendance system. By leveraging existing infrastructure such as local IP cameras, this system eliminates the need for expensive hardware and transforms simple cameras into efficient attendance tools. The automation of attendance not only reduces manual effort and saves time but also ensures secure data management, thereby enabling rural schools to take an important step toward digital transformation and modernization in education.

Abbreviations and Acronyms

I. API – Application Programming Interface
II. HTML – HyperText Markup Language
III. CSS – Cascading Style Sheets
IV. WebRTC – Web Real-Time Communication
V. REST – Representational State Transfer
VI. RFID – Radio Frequency Identification
VII. AI – Artificial Intelligence
VIII. ML – Machine Learning
IX. IP – Internet Protocol

REFERENCES

[1] N. Soni, M Kumar and G. Mathur, "Face Recognition using SOM Nural Network with Different Facial Feature Extraction Techniques, "International Journal of computer pplications, vol 76, no.3, pp 7-11, 2013.

[2] Md. Sajid Akbar, Abu Musa AI Ashray, Pronob Saker, Jia Uddin, Ahmad Tamim Mansoor. Face Recognition and RFID Verified Attendance System. International Conference on Computer and Information Technology (ICCIT). 2018; 21(4):42-47

[3] Abin Abraham, Mehul Bapse, Yash Kalaria, Ahmer Usmani. Face Recognition Base Attendance System. International Journal of Engineering Research & Technology (IJERT). 2020; 9940:200-205.

[4] Abshish Jadhav, Dhwaniket Kamble, Santosh B. Rathod, Sumita Kuamr, Pratima Kadam, Mohammad Dalwai. Attendance Management System Using Face Recognition. International Journal of Emerging Technologies and Innovative Research (JETIR). 2034;11(2):45-50.

[5] Li, Q., Zhang, Y., & Chen, T. (2021). Facial recognition-based automatic attendance system using convolutional neural networks. International Journal of Intelligent Systems, 36(3), 1104-1115.

[6] Patel, R., & Kumar, S. (2020). Face detection using deep learning algorithms: A review. International Journal of Computer Applications, 182(7), 15-22.

[7] Jain, P., Gupta, M., & Jain, S. (2020). Automatic Attendance System Using Face Recognition . International Journal of Advanced Computer Science and Applications, 11(6), 107-113.

[8] Mariyam Mahboob, Nounamn Mohsin Abdul Qadir. Attendance Management System Using Face Recognition. International Research Journal of Modernization in Engineering Technology and Science (IRJMETS). 2024; 6(4): 122-128.

[9] Samridhi Dev, Tushar Patnaik. Student Attendance System Using Face Recognition. International Journal of Scientific & Technology Research (IJSTR). 2020,9(3): 4402-4405.

[10] Sharma, S., & Patel, V. (2021). Comparative study of face detection algorithms in automatic attendance systems. Journal of Machine Learning & AI Research, 12(1), 25-32.

[11] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM Computing Surveys, vol. 35, no. 4, pp. 399–458, 2003.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2001, pp. 511–518.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815–823.

[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[15] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in Proc. British Machine Vision Conference (BMVC), 2015, pp. 1–12.

[16] Maurya, Vijendra Kumar, et al. "Adopting blockchain technology for smart farming and food

© 2025-2027, IJARCS All Rights Reserved

security." International Conference on Data Science and Applications. Singapore: Springer Nature Singapore, 2023.

© 2025-2027, IJARCS All Rights Reserved