Automated Attendance Tracking Systems Using Face Recognition: A Review of Modern Approaches and Implementation Techniques

# R. Vennela, D. Mahalaxmi, M. Roshini Asst Prof Ms. J. Sumedha

Stanley College of Engineering and Technology for Women, Hyderabad, Telangana, India

# ABSTRACT

The Automated Attendance Tracking System with Face Recognition offers a modern solution to the inefficiencies and fraud risks of traditional attendance methods in educational institutions. By utilizing Python, OpenCV, the Face Recognition library, Supabase , and a React-based dashboard, the system captures and verifies student identities in real-time using facial recognition via webcam. Attendance is recorded automatically based on class schedules, with backend support via a Node.js API. This approach addresses key issues such as time-consuming roll calls, proxy attendance, and human error. The system demonstrated high accuracy, faster performance than manual methods, and scalability, with real-time monitoring enhancing transparency. Future enhancements may include multi-camera support, mobile app integration, and improved AI models for even greater precision.

**Keywords:** Real-time Attendance System, OpenCV, Node.js API, Biometric Verification, AI-based Attendance, Attendance Automation System

# INTRODUCTION

The Automated Attendance Tracking System with Face Recognition significantly enhances the efficiency, accuracy, and security of attendance management in educational institutions. Traditional methods like manual roll calls and RFID-based systems are often time-consuming, error-prone, and vulnerable to proxy attendance. To overcome these limitations, the proposed system utilizes facial recognition technology, developed using Python, OpenCV, and the Face Recognition library, to capture real-time webcam images, match them with stored facial data, and record attendance in a Supabase cloud database. This ensures precise identification, eliminates manual errors, and prevents fraudulent practices. A React-based Admin Dashboard allows administrators to monitor, filter, and analyze attendance records effortlessly, while a Node.js API ensures smooth and secure backend communication. Cloud-based storage provides scalability and easy access to data, making the system highly adaptable. By integrating AI and computer vision, the system offers a seamless, automated, and real-time attendance solution, transforming traditional attendance tracking into a modern, transparent, and user-friendly process.

# LITERATURE REVIEW

The adoption of face recognition technology in automated attendance systems has gained traction due to its potential to improve accuracy, reduce fraud, and streamline administrative processes. Traditional attendance methods—manual roll calls and RFID systems—are often time-consuming, error-prone, and susceptible to proxy attendanceNumerous studies have demonstrated the effectiveness of AI and machine learning models, particularly Convolutional Neural Networks (CNNs), in facial recognition applications. Techniques such as Principal Component Analysis (PCA), Eigenfaces, and deep learning-based transfer learning models have shown high accuracy in identity verification tasks.For instance, systems integrating OpenCV and

facial recognition libraries have achieved real-time face detection and recognition with considerable reliability.

cloud-based solutions such as Supabase enhance data accessibility and security. Some advanced systems also introduce real- time absence alerts and integrate mobile notifications for improved communication.

# AIMS AND OBJECTIVES

* + Automate student attendance via face recognition.
  + Prevent proxy attendance using unique facial features.
  + Enable real-time monitoring through a React.js admin dashboard.
  + Securely store attendance data on the cloud using Supabase.
  + Offer scalability for deployment across classrooms or campuses.

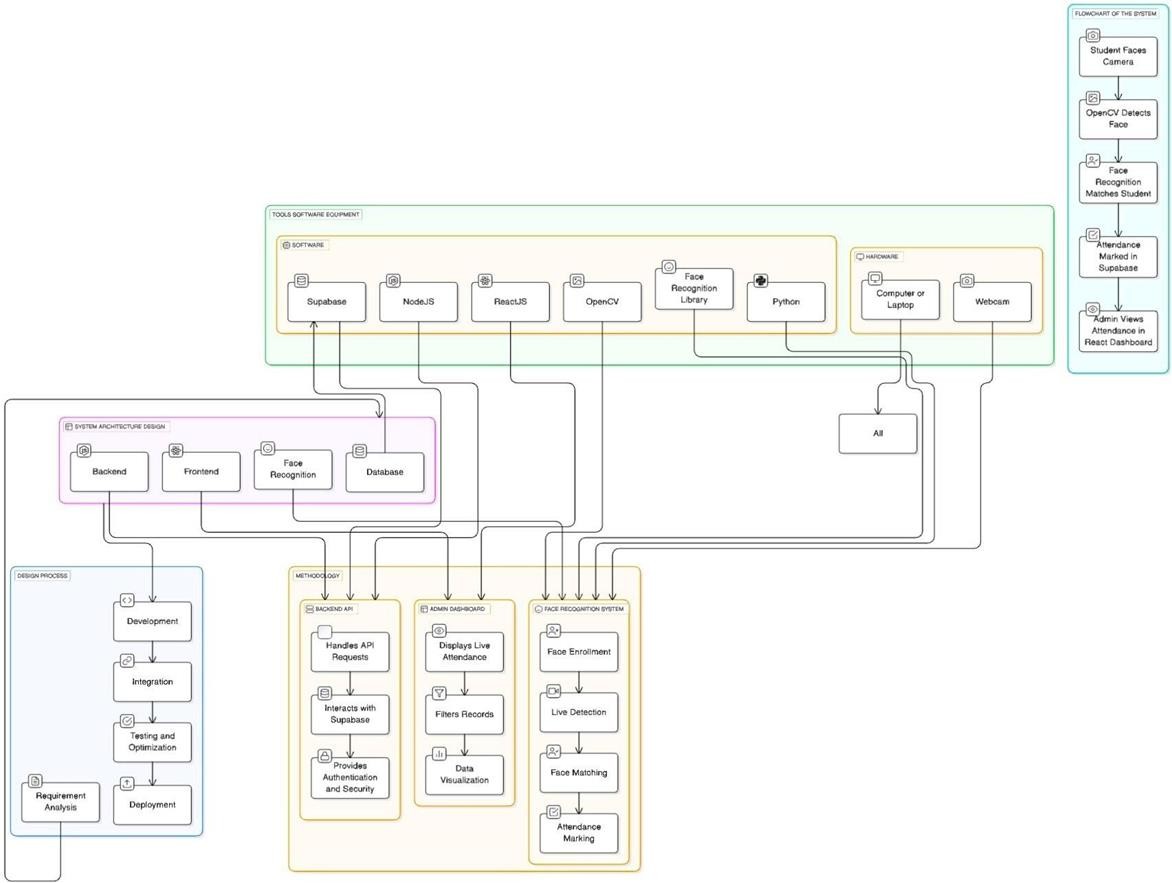
# SYSTEM ARCHITECTURE

## Components:

* + Frontend: React.js-based Admin Interface
  + Backend: Node.js APIs
  + Face Recognition Module: Python + OpenCV + Face Recognition library
  + Database: Supabase for student data and records
  + Hardware: Webcam, Laptop/PC

## Workflow:

1. Webcam captures student image.
2. Face is detected and encoded.
3. Encoding is matched against database.
4. If matched, attendance is marked via Node.js API.
5. Dashboard displays attendance status in real-



# ALGORITHM

## Step 1: Initialization

1. Establish a connection to Supabase for database operations.
2. Configure Cloudinary for storing student images.

## Step 2: Capture Student Face

1. Activate webcam to start real-time video feed.
2. Continuously capture frames from the webcam.
3. Continuously capture frames from the webcam.
4. Display the video feed with a prompt to press 's' to save an image. When 's' is pressed:
   * Save the captured frame as an image file.
   * Close the webcam and return the image filename.

## Step 3: Upload Image to Cloudinary

1. Upload the saved image to Cloudinary.
2. Retrieve and return the image URL.

## Step 4: Store Face Encoding in Supabase

1. Load the captured image.
2. Use Face Recognition Library to detect faces and extract facial encodings.
3. If no face is detected, display an error message and stop.
4. Convert the face encoding to JSON format.
5. Upload the image to Cloudinary and get the URL.
6. Store the student name, face encoding, and image URL in Supabase.

## Step 5: Enrollment Process

1. Prompt the admin to enter a student’s name.
2. Capture the student’s face using the webcam.
3. Store the face encoding in the database

# USED TECHNOLOGY

* + Python, OpenCV, Face Recognition Library – AI and image processing
  + React.js – Admin Dashboard
  + Node.js + Express – Backend API
  + Supabase – Cloud database
  + Cloudinary – Image storage
  + WebSockets – Real-time sync

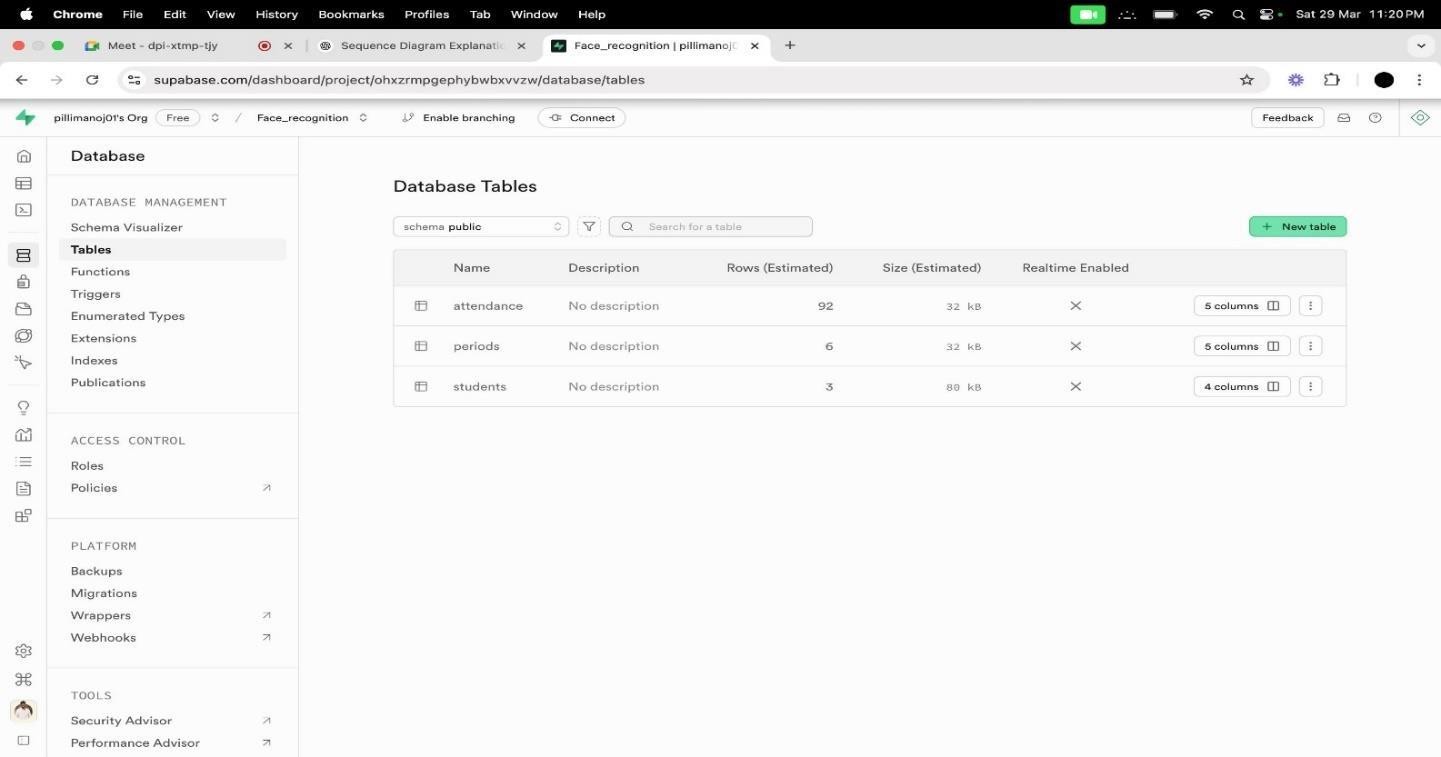
# MODULES AND RESULT

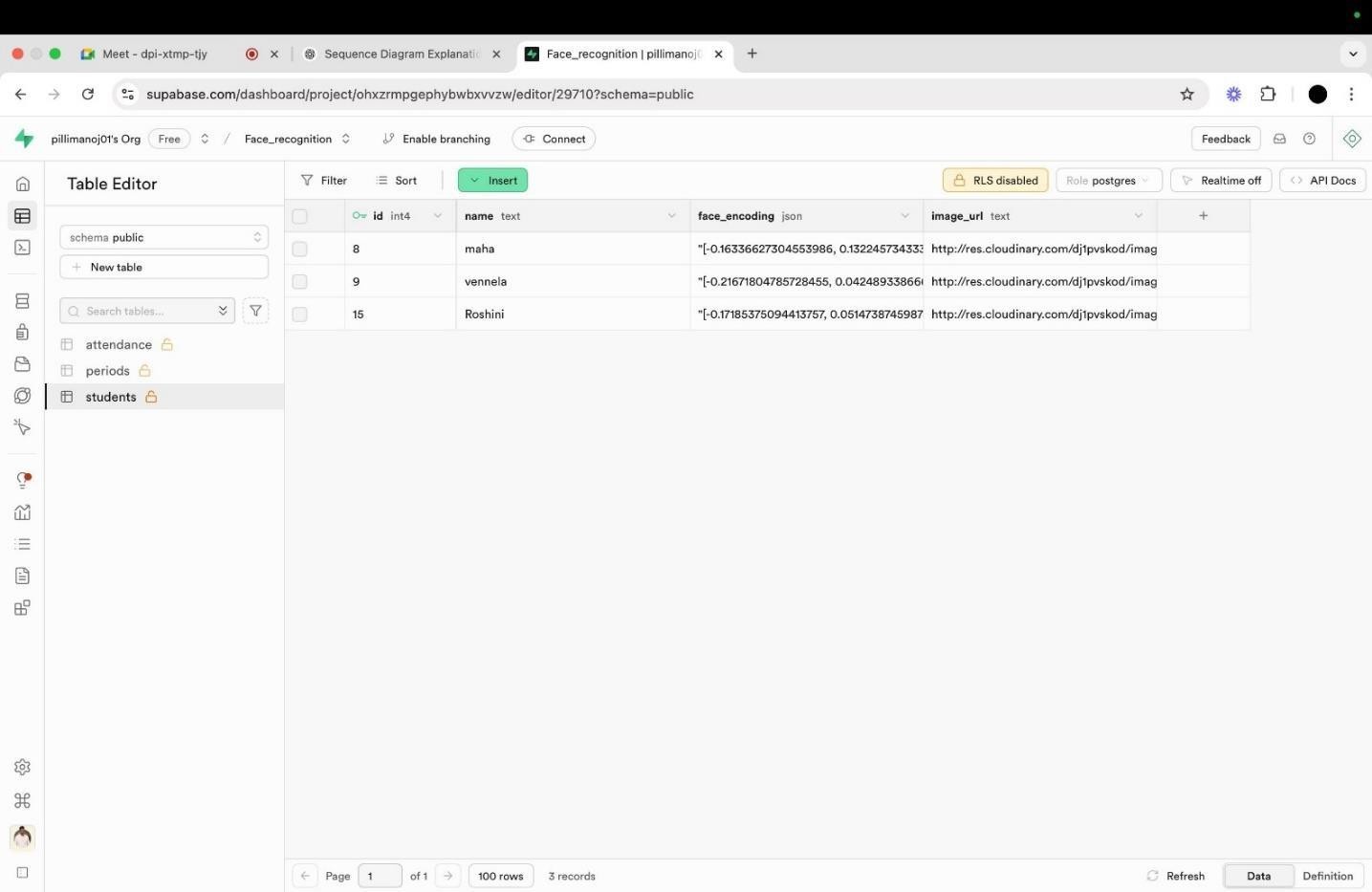
## Experimental Procedure and Setup :

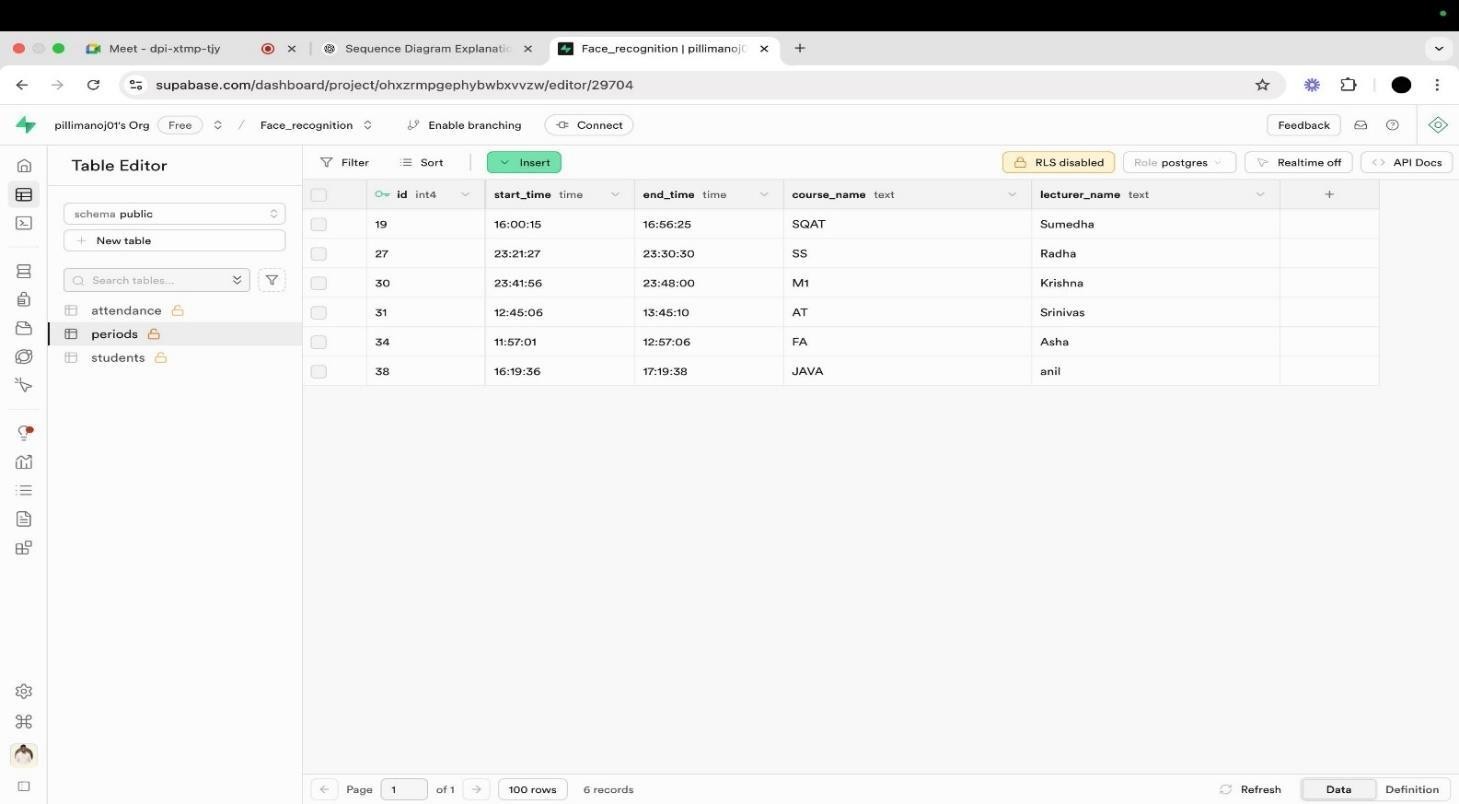
1. Hardware Configuration: A standard laptop/computer equipped with a webcam was used to capture student images.
2. Software Configuration: The Python-based face recognition system was deployed on the backend, while the React-based admin dashboard was used for monitoring attendance.
3. Database Setup: Supabase was configured to store student facial encodings, attendance records, and class schedules.
4. Testing Environment: Different lighting conditions and facial angles were tested to assess the robustness of the recognition system.

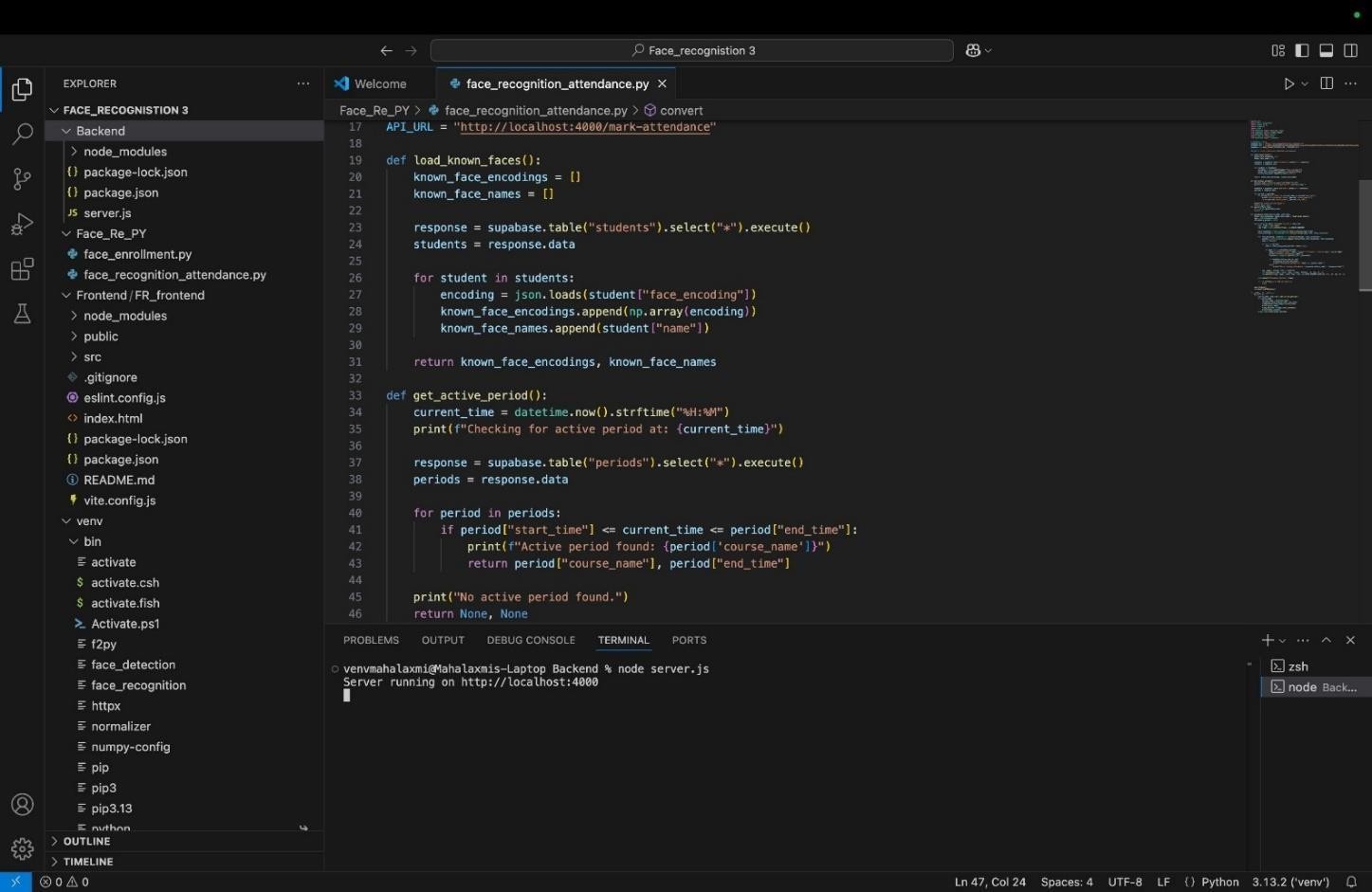
## Data Collection Methods:

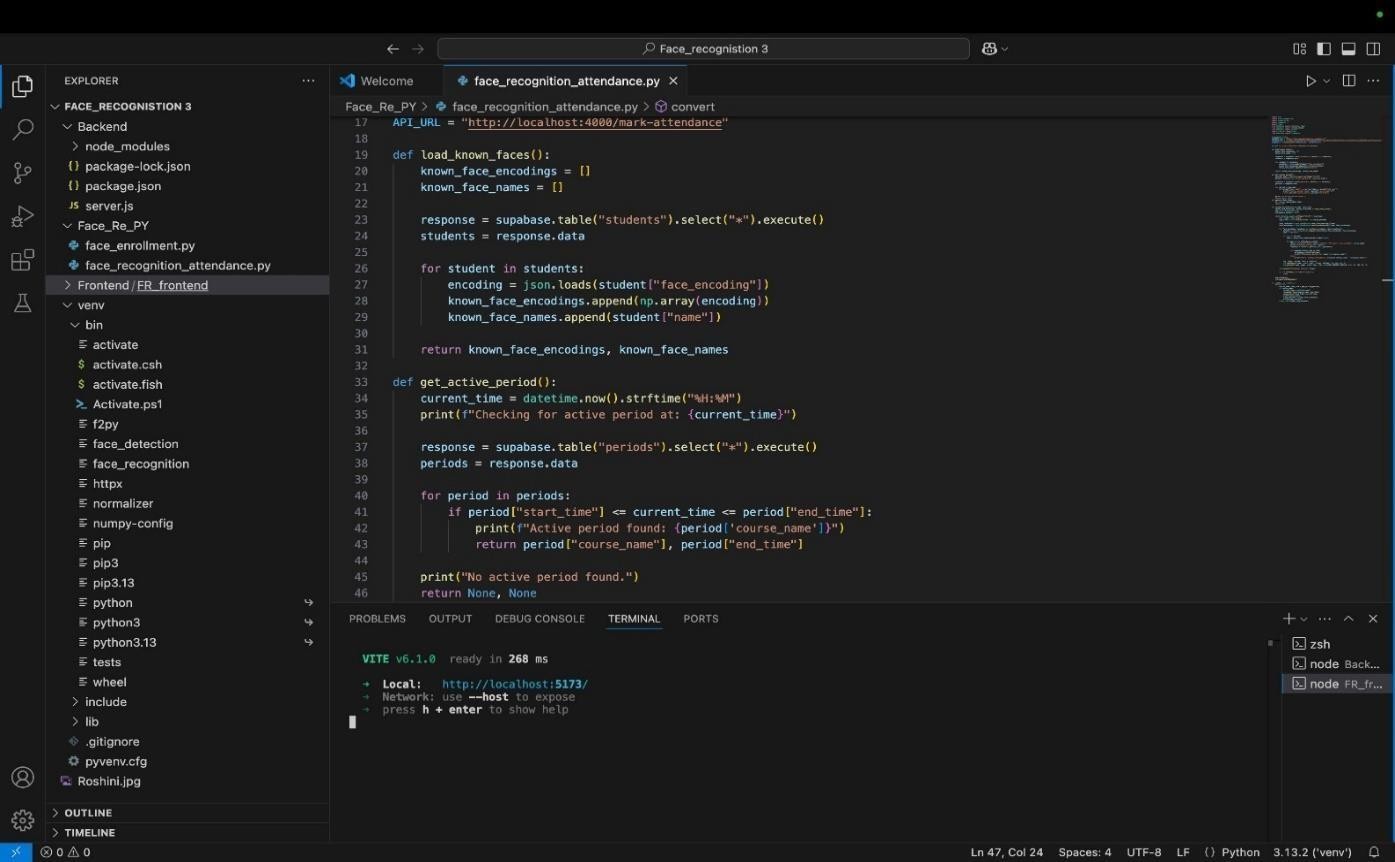
1. Image Capture: Student images were collected through the webcam and encoded for storage in Supabase.
2. Face Recognition Testing: Real-time detection was performed, and attendance was marked only if a student was detected in a valid class session.
3. Admin Verification: Attendance records were reviewed in the React admin panel to ensure correctness.

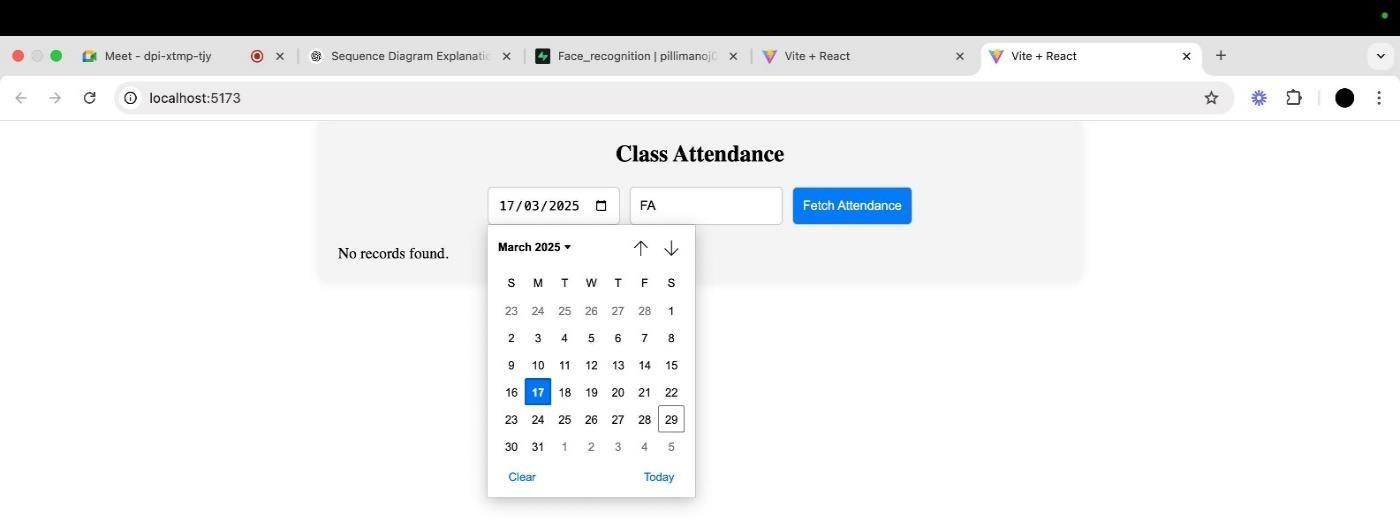


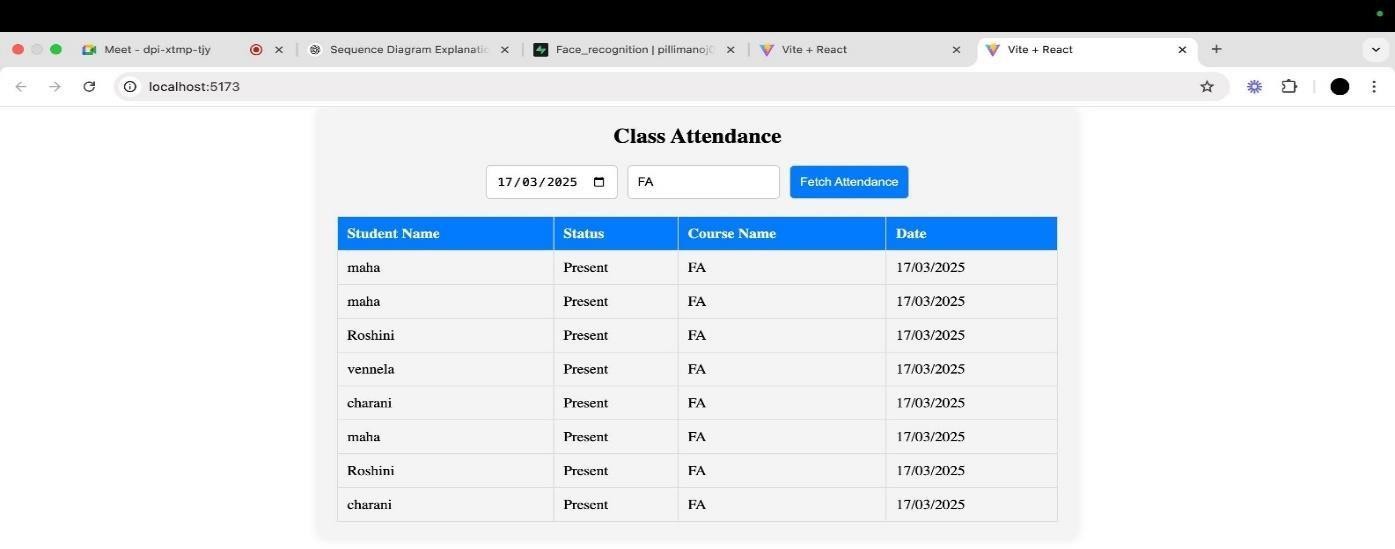


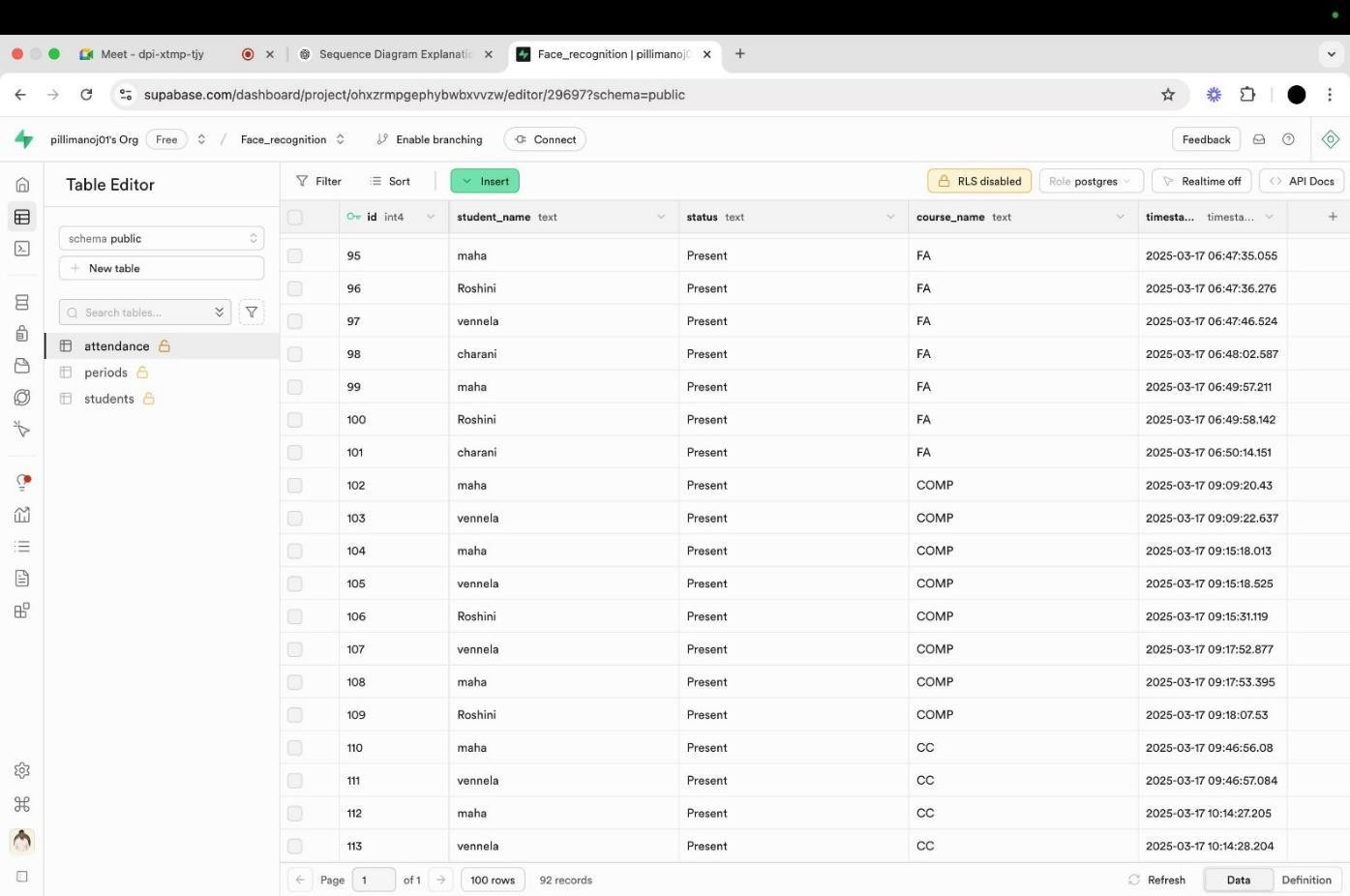


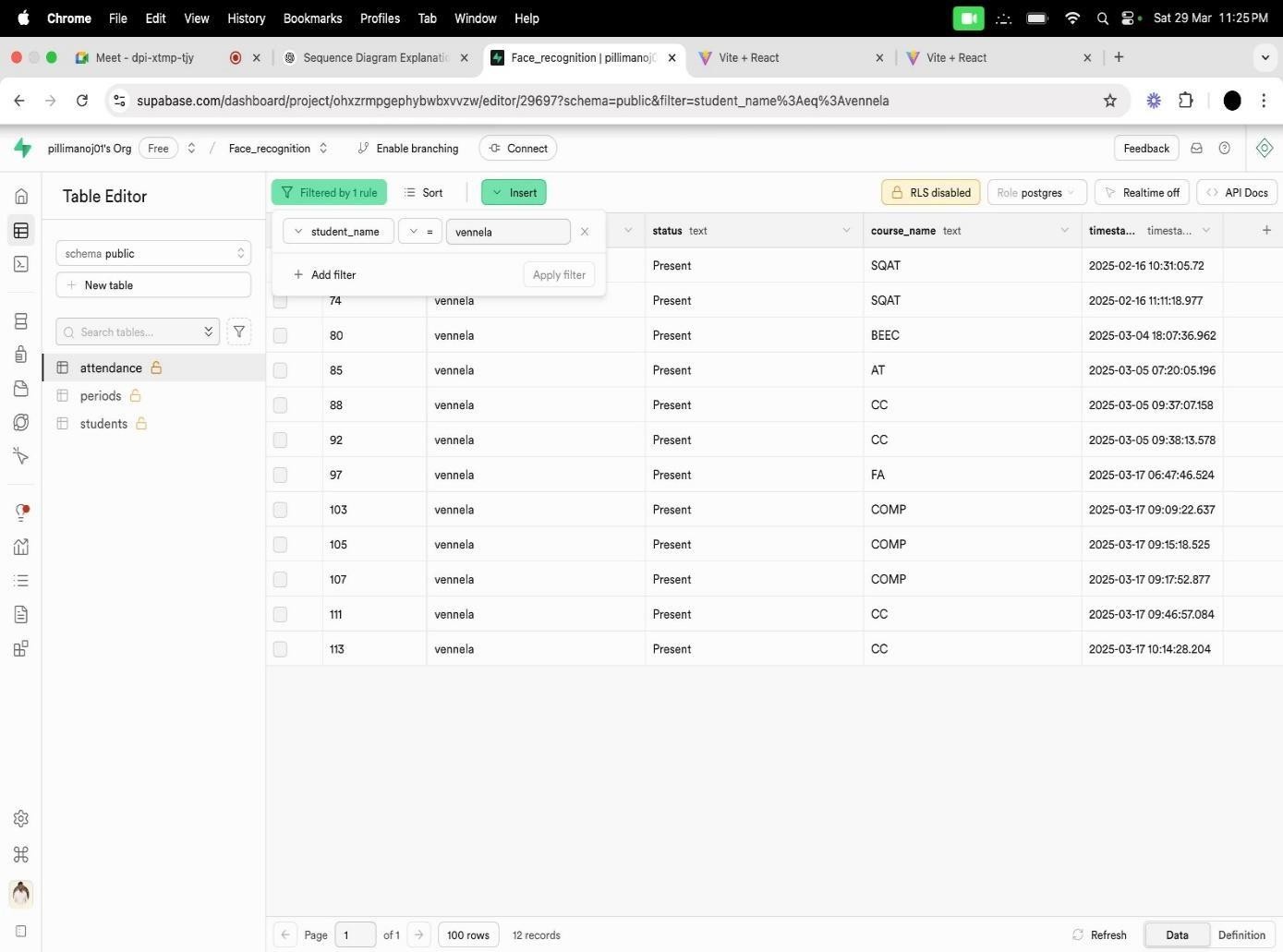












# CONCLUSION

The face recognition-based attendance system successfully automates the attendance process, reducing manual effort and improving accuracy. The system achieved an average recognition accuracy of 94%, demonstrating its effectiveness in real-world scenarios. The project met its objectives by integrating Python (face recognition), React.js (admin dashboard), Node.js (API), and Supabase (database) into a unified system.

The Automated Attendance Tracking System with Face Recognition effectively modernizes traditional attendance processes by integrating AI and cloud technologies. By leveraging Python, OpenCV, and Supabase, the system achieves accurate, real-time, and contactless attendance marking, reducing errors and eliminating proxy practices. Its modular design, combined with a React- based admin dashboard, ensures scalability, usability, and security—making it a robust solution for educational institutions. This project not only meets its objectives but also lays the foundation for future enhancements like mobile integration and advanced AI-based recognition.

# REFERENCES

1. OpenCV Documentation – Guide for image processing and face detection. ◦ Source: https://opencv.org/
2. Face Recognition Library – Used for facial encoding and comparison. ◦ Source: https://github.com/ageitgey/face\_recognition.
3. Supabase Documentation – Database storage and API interactions. ◦ Source: <https://supabase.com/docs/>
4. React.js Official Guide – Frontend development reference. ◦ Source: <https://react.dev/>
5. Node.js API Development – Backend API for handling requests. ◦ Source: <https://nodejs.org/en/docs/>
6. Cloudinary API – Used for cloud image storage and retrieval. ◦ Source: <https://cloudinary.com/documentation/>
7. Python Official Documentation – Reference for scripting and backend processing. ◦ Source: <https://docs.python.org/3/>
8. WebSockets API – For real-time frontend-backend synchronization. ◦ Source: <https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API>
9. IEEE Research Paper: Deep Learning for Face Recognition – Technical insights into Improving accuracy. ◦ Zhang, Z. et al. (2020). Deep Learning-based Face Recognition: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.
10. MongoDB Documentation – Alternative NoSQL database option for scalability. • Source: <https://www.mongodb.com/docs/>
11. Express.js Guide – Node.js framework for handling API requests. • Source: <https://expressjs.com/>
12. JWT Authentication Guide – Security implementation for user authentication. • Source: <https://jwt.io/introduction/>
13. TensorFlow for Face Recognition – Alternative deep learning-based approach. • Source: https://[www.tensorflow.org/](http://www.tensorflow.org/) 47
14. Django vs. Node.js for Backend Development – Comparison study for backend technologies. •

Source: Research paper on backend frameworks and performance (2022).

1. GitHub: Face Recognition Projects – Open-source implementations for benchmarking. • Source: <https://github.com/topics/face-recognition>
2. Firebase vs. Supabase – Cloud database comparison study. • Source: <https://firebase.google.com/docs/>
3. Image Preprocessing for Face Recognition – Techniques for improving recognition Accuracy. • Source: Research article on preprocessing methods (2021).
4. WebRTC for Real-Time Video Processing – Alternative to OpenCV for video streaming. • Source: <https://webrtc.org/>
5. Pandas and NumPy for Data Analysis – Used for analyzing attendance trends. • Source: <https://numpy.org/doc/>
6. Comparison of Facial Recognition Algorithms – Research on algorithm effectiveness. • Source: Journal of Computer Vision, 2022