A Face Recognition Student Attendance System for Academic Institutions based on Siamese Neural Network

An Informatics and Computer Science Final Project Documentation Submitted to the School of Computing and Engineering in partial fulfillment of the requirements for the award of a Degree in Bachelor of Science in Informatics and Computer Science.

Supervisor: Daniel Machanje

Student's Number: 151354

Nairobi, Kenya

November 2025

Declaration and Approval

I declare that this project documentation has not been submitted to Strathmore University or any other University for the award of a Degree in Bachelor of Science in Informatics and Computer Science or any other Degree. To the best of my knowledge and belief, the research documentation contains no material previously published or written by another person except where due reference is made in the research documentation itself.

Student's Admission Number:

Sign: _____

Date: _____

Supervisor's Signature:

Sign: _____

Date: _____

Acknowledgements

I would like to express my sincere gratitude to all individuals and institutions who have contributed significantly to the development of this project documentation.

First and foremost, I extend my deepest appreciation to my supervisor, Mr. Machanje, for his exceptional guidance, constructive feedback, and unwavering support throughout the conceptualization and development of this AI-powered facial recognition attendance system documentation. His expertise in machine learning applications and critical insights were instrumental in shaping the technical framework and research direction of this work.

I am profoundly grateful to Dr. Esther Khakhata and the faculty of the Computer Science Department for their invaluable mentorship and technical guidance, particularly in the areas of deep learning and computer vision technologies. Their scholarly input significantly enhanced the theoretical foundation and practical applicability of the created system.

Special thanks are extended to the educational institutions and academic administrators who participated in the preliminary research phase, sharing their experiences with attendance management challenges and providing crucial insights into the practical requirements for automated attendance systems. Their contributions were essential in identifying the real-world problems this documentation seeks to address.

I acknowledge my fellow students in BICS 4D for their collaborative discussions and peer feedback, which helped refine various aspects of the modelled system design and anti-spoofing security features.

Finally, I am deeply grateful to my parents for their continuous encouragement and support throughout this academic pursuit, and to the Almighty God for His grace and wisdom throughout the journey.

Abstract

Educational institutions have increasingly transitioned from traditional pen-and-paper attendance systems to digital methods; however, manual tracking has remained common and continued to pose challenges such as time inefficiency, human error, and proxy attendance fraud. To address these issues, this study developed and implemented an AI-powered facial recognition attendance system designed to automate attendance capture and strengthen academic integrity.

The system integrated a Siamese Neural Network for facial recognition, OpenCV for real-time image processing, and a Next.js framework for web-based management. Advanced anti-spoofing mechanisms including liveness detection and depth estimation were incorporated to prevent photograph and video replay attacks. The implemented model utilized twin Convolutional Neural Networks to learn facial similarity metrics, enabling more robust and identity-independent recognition. Real-time facial detection was achieved using Haar Cascade classifiers, while classroom webcam input supported an average processing response time of approximately two seconds. A RESTful API architecture enabled smooth communication between system components and allowed integration with existing Learning Management Systems.

System evaluation demonstrated a major reduction in attendance recording time, complete prevention of proxy attendance attempts, and an overall improvement in data accuracy and reliability. The findings confirm that the developed solution offers an effective, secure, and scalable approach to attendance management, contributing to the advancement of educational technology by enhancing efficiency and preserving academic integrity.

Keywords: Facial Recognition, Siamese Neural Networks, Student Attendance, Anti-Spoofing, Computer Vision, Educational Technology.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API – Application Programming Interface

CNN – Convolutional Neural Network

CSS – Cascading Style Sheets

HTTP – Hypertext Transfer Protocol

LMS – Learning Management System

ML - OPS – Machine Learning Operations

OOAD – Object-Oriented Analysis and Design

OpenCV – Open-Source Computer Vision Library

REST - Representational State Transfer

Chapter 1: Introduction

1.1 Background

Educational institutions have employed various attendance tracking methodologies throughout their operational history, with traditional pen-and-paper systems dominating academic settings for decades and establishing foundational practices for monitoring student presence. Although these manual systems created standardized protocols that became integral to institutional record-keeping, they demonstrated inherent limitations, including error rates of 1–3% due to incorrect entries and human recording mistakes (The Hidden Costs, 2024).

The evolution of digital technology led to a progressive transition toward automated solutions, with institutions adopting biometric fingerprint scanners, RFID cards, and QR-code systems. The projected growth of the biometric technology market to USD 94 billion by 2025 reflected widespread institutional investment in automated identification systems (Hernandez-de-Menendez et al., 2021). However, these digital solutions continued to exhibit vulnerabilities, particularly proxy attendance fraud, in which individuals circumvent attendance requirements through impersonation or other deceptive practices (Delhi, 2018).

Advancements in artificial intelligence and computer vision technologies further accelerated the development of contactless attendance systems, especially during the Covid-19 period when minimal physical interaction became essential (Wang et al., 2022 Facial recognition technology, utilizing advanced neural networks and algorithms such as Viola–Jones, offered improved automated identification capabilities by accurately detecting and matching facial features under varying environmental conditions (Joshi et al., 2023). This technological domain represents a convergence of computer vision, machine learning, and database management systems.

Despite these advancements, existing AI-powered attendance solutions in educational environments demonstrated fragmented functionality, often operating as isolated systems with limited integration into broader institutional management platforms. Many lacked robust security mechanisms to prevent sophisticated proxy attendance attempts and provided insufficient reporting capabilities to support academic integrity monitoring. These implementation gaps influenced not only administrative efficiency

but also academic assessment, financial aid eligibility, regulatory compliance, and institutional accreditation processes.

Addressing these gaps carries significance for improving student engagement monitoring, facilitating early intervention for at-risk learners, and reallocating administrative resources toward direct student support activities. Reliable attendance systems also provide institutions with high-quality data that enhance academic planning and compliance with regulatory requirements.

This study addressed the identified gaps by developing, implementing, and evaluating an AI-based facial recognition attendance system tailored for academic institutions. The system integrated real-time facial recognition with secure database management and institutional reporting functions. Its design focused on three core components, computer vision algorithms for accurate facial detection and recognition, security mechanisms to prevent attendance fraud through liveness and spoof detection, and a user-friendly interface enabling seamless integration with existing educational management systems. Rather than pursuing a hypothesis, the research demonstrated the feasibility and effectiveness of an integrated AI-driven attendance system in improving accuracy, reducing administrative workload, and strengthening institutional compliance capabilities.

1.2 Problem Statement

Educational institutions have faced longstanding challenges related to inaccurate attendance records, widespread proxy attendance fraud, and administrative inefficiencies that compromised academic integrity and consumed valuable institutional resources. Existing attendance tracking systems both manual and digital proved unreliable in verifying student presence and remained vulnerable to manipulation and fraud (Chen, 2025).

Ideally, institutions require attendance systems capable of automatically verifying student presence with high accuracy, eliminating proxy attendance fraud, integrating seamlessly with existing management platforms, and providing real-time analytics to support informed decision-making. Such systems must operate unobtrusively within the classroom environment to ensure that attendance records accurately reflect genuine student engagement and academic participation.

However, the analysis conducted in this research confirmed that current attendance solutions fell significantly short of these expectations. Manual attendance systems demonstrated error rates of 1–3% due to human mistakes and were highly susceptible to proxy attendance fraud, in which students marked attendance for absent peers (The Hidden Costs, 2024). Even technologically advanced methods such as fingerprint scanners and RFID cards remained vulnerable to impersonation, as a student could easily share access cards or request peers to clock in on their behalf (Delhi, 2018). Additionally, existing facial recognition implementations in educational settings lacked comprehensive anti-spoofing mechanisms, struggled with real-time accuracy under varied classroom conditions, and offered limited integration capabilities with institutional management systems. These weaknesses resulted in fragmented data systems and limited opportunities for academic analysis, early intervention, and engagement monitoring.

To address these critical gaps, this research developed and evaluated an AI-powered facial recognition attendance system that integrated advanced face-matching algorithms with robust anti-spoofing mechanisms. The implemented system featured real-time liveness detection, seamless interoperability with educational management platforms, and built-in analytics tools for institutional reporting. Through this implementation, the study demonstrated significant improvements in attendance accuracy, eliminated fraudulent attendance attempts, and provided institutions with actionable insights into student engagement patterns. The system effectively strengthened academic integrity while reducing the administrative workload associated with manual attendance processes.

1.3 Research Objectives

Research objectives define the specific goals that guided this study. They provided a clear direction for the development and evaluation of the EduFace system and served as measurable benchmarks for determining the success of the project. The objectives were action-oriented, achievable, and aligned with the identified problem gaps.

1.3.1 General Objectives

To develop and implement an intelligent facial recognition attendance system using Deep Convolutional Neural Networks (CNNs) that automates student attendance tracking and prevents proxy attendance fraud in educational institutions.

1.3.2 Specific Objectives

i. To analyse the current state of attendance tracking solutions and evaluate how existing systems addressed student attendance management challenges in educational institutions.
ii. To identify and examine the limitations of current biometric attendance systems, particularly their vulnerabilities to proxy attendance and accuracy issues.
iii. To design and implement a real-time facial recognition attendance system that automatically identifies a student and logs attendance using Deep CNN algorithms.
iv. To implement robust security and anti-spoofing measures in the facial recognition system that prevent proxy attendance, while ensuring smooth integration with existing educational management platforms.
v. To test and validate the effectiveness of the implemented EduFace system in accurately tracking student attendance and preventing fraudulent attendance practices.

1.3.3 Research Questions

i. What was the current state of attendance tracking solutions, and how had existing systems addressed attendance management challenges in educational institutions?
ii. What were the key limitations of existing biometric attendance systems, particularly regarding vulnerabilities to proxy attendance and accuracy issues?
iii. How could a real-time facial recognition attendance system be designed and implemented to automatically identify a student and log attendance using Deep CNN algorithms?
iv. How could robust security and anti-spoofing features be incorporated into the facial recognition system to detect and prevent proxy attendance while ensuring seamless integration with educational management systems?
v. How could the implemented EduFace system be tested and validated to demonstrate its effectiveness in accurately tracking student attendance and preventing fraudulent attendance practices?

1.4 Justification

The need for EduFace emerged from the persistent failures of existing attendance management systems in educational institutions, which continued to compromise academic integrity while consuming significant administrative resources. Research consistently showed that traditional manual attendance systems suffered from error rates of 1–3% due to human mistakes and were highly vulnerable to proxy attendance fraud, where a student routinely has peers sign him in or send representatives to mark attendance (Staff, 2024). Even modern biometric systems such as fingerprint scanners and RFID cards proved susceptible to manipulation, as a student could easily share cards or circumvent security measures, fundamentally undermining the reliability and accountability of attendance records (Digittrix, 2025).

Additionally, existing facial recognition implementations in educational settings lacked comprehensive anti-spoofing capabilities and frequently struggled with real-time accuracy under varying classroom conditions. Their inability to integrate effectively with institutional management platforms resulted in isolated data systems that limited institutions' ability to extract meaningful insights into student engagement patterns (E. Team, 2025). The COVID-19 pandemic further exposed these weaknesses, as schools faced challenges in maintaining accurate, contactless attendance. Chronic absenteeism doubled nationwide during the pandemic period and remained elevated even after reopening, emphasising the need for more reliable, automated attendance solutions (Wang et al., 2022).

In response to these critical gaps, this study designed, implemented, and evaluated the EduFace system, an AI-powered facial recognition attendance solution built on Deep Convolutional Neural Networks using a Siamese architecture. This approach enabled one-shot learning, allowing the system to accurately identify a student without requiring extensive retraining when a new learner was enrolled. The implemented EduFace system incorporated advanced anti-spoofing mechanisms including liveness detection, and depth estimation to mitigate fraudulent attempts involving photographs, screens, or video replays. System testing demonstrated real-time processing speeds averaging under two seconds, meeting the operational demands of typical classroom environments (Joshi et al., 2023).

Furthermore, EduFace was integrated seamlessly with existing Learning Management Systems through RESTful APIs, eliminating the data silos present in previous implementations. This integration enabled comprehensive analytics and reporting functions that supported institutional decision-making and improved monitoring of student engagement. The effectiveness of deep learning approaches in attendance tracking has been validated by previous studies (Chandrasekhar, 2025), and the continued expansion of the biometric technology market projected to reach USD 94 billion by 2025 indicated strong industry confidence in such solutions (Hernandez-de-Menendez et al., 2021).

Through its implementation and evaluation, EduFace achieved near-perfect attendance accuracy, successfully prevented fraudulent attendance practices, and substantially reduced the administrative burden associated with manual tracking. The system transformed attendance management from a recurring institutional challenge into an intelligent, efficient, and fraud-resistant process, supporting academic integrity while enabling staff to redirect time toward core educational activities rather than data reconciliation tasks.

1.5 Scope

This section now explains what the completed project includes. It shows what the system covers, what it does not cover, and the areas that were fully completed within the available time. This helps set clear expectations for the final system and its capabilities

This project focuses on developing a working AI-powered facial recognition attendance system for educational institutions, built within clear boundaries to keep the work practical and achievable. The final system covers four main areas that directly support automated attendance tracking and reduce cases of proxy attendance.

The technical scope includes building a complete facial recognition system using Deep Convolutional Neural Networks for real-time identification and automatic attendance recording. It uses computer vision methods based on Siamese networks for one-shot learning, which helps the system recognize a new student without long retraining processes. This makes the system flexible as student numbers change while still keeping accuracy high.

The data scope includes working with facial recognition datasets and attendance patterns from participating institutions. The system processes real-time video during classes and uses both synthetic/anonymized facial datasets for training, and real attendance records for testing. This makes the system effective while still respecting privacy rules.

The functional scope covers all key features of the system, including student enrollment with face capture, real-time recognition during lessons, automatic attendance logging with timestamps, and strong proxy attendance detection. The system also connects with existing school management systems and produces detailed reports, all while keeping student data private and secure.

The time scope reflected an 8-month development period with clear stages. Months 1–2 focused on research and system design, months 3–5 covered algorithm development and early testing, months 6–7 focused on system integration and full testing, and month 8 covered final validation and documentation.

1.6 Delimitations

This section explains the intentional boundaries set for the project to keep the work focused, manageable, and achievable within the available time and resources. These limits helped ensure that the project stayed centered on the main problem without expanding into areas that would make the work too broad or unrealistic.

This project focused only on facial recognition as the main biometric method. It did not include other biometric options like fingerprints, iris scans, or voice recognition. This allowed the project to go deeper into facial recognition instead of spreading effort across multiple technologies. The system was also designed specifically for student attendance in physical classroom settings, and it did not cover staff attendance. This helped keep the project centered on solving student proxy attendance issues.

The integration scope was limited to standard educational management systems. It did not include connections to external third-party platforms, as that would add unnecessary complexity and extend the project beyond the planned timeline. The system also does not support online or remote learning attendance, focusing only on physical classrooms where proxy attendance is more common and where facial recognition is most effective.

Other features intentionally left out include behavioral analysis, emotion detection, and mobile app development. Cloud deployment was also excluded, as the project

prioritized a local implementation to better support data privacy and reduce risks related to storing sensitive information on external servers.

1.7 Limitations

This section explains the factors that were outside the researcher's control and may have affected the system's performance or the generalizability of the results. These limitations are included for transparency and to give a clear picture of the conditions under which the project was completed.

Access to facial recognition datasets that truly match real classroom environments was limited. Many available datasets do not reflect the diversity of students, lighting variations, and everyday conditions found in schools. Because of this, the system's accuracy may differ across different student groups and classroom setups. Testing was also limited to a small number of students, which affects how well the results can be applied to other schools with different equipment, student populations, or routines.

The system's performance is also influenced by the quality of hardware already available in each institution. Schools using older cameras or machines with low processing power may experience slower performance or reduced accuracy compared to schools with modern equipment. Additionally, there was limited research focused specifically on facial recognition attendance systems with fraud detection, meaning there were few benchmarks for comparison and new evaluation methods had to be developed.

Other external factors also affected the project. These include changes in classroom lighting at different times of the day, varying levels of student cooperation, and different institutional rules about privacy and technology use. These conditions impacted the ability to collect consistent data and perform fully comprehensive testing. As a result, both the technical performance and the research findings may have been influenced by these real-world conditions.

Chapter 2: Literature Review

2.1 Introduction

This chapter reviews existing research on student attendance systems to show what has already been done and what problems still remain. It focuses on three main areas: how attendance tracking has moved from paper methods to digital tools, what issues exist in current biometric systems, and how facial recognition is currently being used in schools. The review uses recent academic studies, technical papers, and real case studies from the last ten years, chosen because they relate closely to attendance challenges and fraud prevention in learning institutions. Section 2.2 explains how attendance systems have changed over the years and why many of them still fail to meet school needs. Section 2.3 looks at different facial recognition systems that already exist and highlights what they do well and where they struggle. Section 2.4 brings together all the weaknesses found in current systems and shows why there is a strong need for the EduFace system, which uses advanced AI, strong security measures, and smooth integration with school management platforms.

2.2 Evolution and Current State of Attendance Management Systems in Educational Institutions

Student attendance management has evolved significantly over the years, moving from simple manual methods to advanced digital and biometric systems. Traditionally, schools tracked attendance using pen-and-paper registers, verbal roll calls, or physical sign-in sheets. These methods were straightforward and inexpensive but relied heavily on human effort and were prone to errors.

As educational institutions grew in size and complexity, the limitations of manual tracking became apparent. Errors in recording attendance, missing entries, and oversight could affect the accuracy of records, impacting academic accountability and regulatory compliance. Moreover, manual systems were vulnerable to fraud, such as students marking attendance on behalf of absent peers, creating concerns about integrity.

To address these issues, educational institutions gradually adopted digital solutions. Early digital attendance systems included barcode or RFID-based card scanning, allowing automated logging without direct teacher involvement. Biometric systems,

such as fingerprint scanners, emerged later to improve accuracy and prevent proxy attendance. More recently, facial recognition systems have been introduced, offering contactless tracking with the potential for real-time verification.

This evolution reflects a clear trend: attendance systems have moved from being labor-intensive and error-prone to becoming increasingly automated, accurate, and secure. Despite these improvements, each stage has had its own limitations, highlighting the ongoing need for smarter, more reliable, and flexible attendance solutions. These challenges and limitations are explored in the following subsections (Hernandez-de-Menendez et al., 2021).

2.2.1 Challenges of Traditional Manual Attendance Systems in Educational Institutions

Traditional attendance management in educational institutions has historically relied on manual methods including pen-and-paper registers, verbal roll calls, and physical signin sheets. While these approaches are straightforward and cost-effective to implement, they have proven increasingly inadequate for meeting modern educational demands. Research consistently demonstrates that manual attendance systems exhibit significant error rates ranging between 1-3% due to various factors including incorrect data entries, missing information, and human oversight errors (Staff, 2024). These inaccuracies create substantial downstream effects, requiring additional administrative effort to reconcile records and undermining the reliability of attendance data that institutions depend upon for academic accountability and regulatory compliance requirements.

The fundamental challenge with manual systems extends beyond simple accuracy concerns. These systems are inherently vulnerable to fraudulent practices, with a student being easily able to manipulate records through proxy attendance where peers sign in for absent classmates or individuals misrepresent their presence during roll calls. The administrative burden associated with manual systems is substantial, requiring dedicated staff time for data collection, verification, and correction processes that could otherwise be directed toward student-focused educational activities.

The COVID-19 pandemic further exposed the critical vulnerabilities of manual attendance systems as educational institutions struggled to maintain accurate records while adapting to new health protocols and hybrid learning models (Babbar & Gupta, 2022). This period highlighted how traditional systems lack the flexibility and

robustness needed to adapt to changing operational requirements, creating urgent pressure for more sophisticated and adaptable attendance tracking solutions.

2.2.2 Evolution and Limitations of Digital Biometric Attendance Solutions

The integration of digital technologies in educational settings has driven the development of automated attendance systems designed to overcome the limitations inherent in manual approaches. The biometric technology market in education is experiencing remarkable growth, with global market size projected to expand from USD 47 billion in 2022 to USD 88.95 billion by 2030, representing a compound annual growth rate of 8.3% (Virtue Market Research, 2023). This rapid expansion reflects strong industry confidence in biometric innovations and their potential to address persistent attendance management challenges.

Fingerprint-based attendance systems emerged as one of the first widespread biometric solutions in educational institutions, capitalizing on the uniqueness and permanence of individual fingerprint patterns. These systems operate by scanning student fingerprints and matching them against stored templates in institutional databases (Hoo & Ibrahim, 2019). While fingerprint systems offer superior accuracy compared to manual methods and effectively eliminate many forms of proxy attendance fraud, they face significant operational challenges. Hygiene concerns, particularly those highlighted during the COVID-19 pandemic, have created resistance to systems requiring physical contact. Additionally, fingerprint recognition can be compromised by cuts, dirt, or naturally worn fingerprints, leading to reliability issues that undermine system effectiveness (Digittrix, 2025).

Radio Frequency Identification systems represent another significant advancement in automated attendance tracking, utilizing identification cards embedded with RFID chips that are automatically scanned when a student enters designated areas. These systems offer the advantage of being contactless and capable of processing multiple entries simultaneously, making them particularly suitable for high-traffic educational environments (RFID System, 2024). However, RFID systems have proven particularly vulnerable to proxy attendance fraud, as a student can easily share their identification cards with others, thereby completely circumventing the security measures the system was designed to provide (Digittrix, 2025).

Basic facial recognition systems have been adopted by some educational institutions as a more advanced approach to attendance tracking, employing simple face detection and matching algorithms to identify a student from captured images (Chen, 2025a). While these systems eliminate the need for physical contact or identification cards, many current implementations lack robust security features, making them susceptible to fraud attempts using photographs or video displays. Additionally, they often struggle with accuracy under varying lighting conditions or when students' appearances change significantly over time (Babbar & Gupta, 2022).

2.2.3 COVID-19 Impact on Contactless Attendance System Adoption

The COVID-19 pandemic served as a catalyst for accelerated adoption of contactless attendance systems across educational institutions worldwide. Facial recognition technology gained prominence during this period as it enabled institutions to maintain effective attendance tracking while supporting health and safety protocols that minimized physical contact (Wang et al., 2022). Research demonstrates that many educational institutions rapidly pivoted to contactless biometric solutions to prevent virus transmission while maintaining essential attendance management capabilities (Nathwani et al., 2021).

This pandemic-driven transformation revealed both the potential benefits and existing limitations of current facial recognition systems in educational settings. While institutions successfully implemented contactless solutions, many of these systems lacked the sophistication needed for comprehensive attendance management, including advanced security features, real-time analytics, and seamless integration with existing educational management platforms.

2.2.4 Current Gaps and Persistent Challenges in Automated Attendance Management

Despite significant technological advancement, current attendance systems continue to struggle with fully meeting the complex requirements of modern educational institutions. The problem of proxy attendance remains particularly troublesome, with students finding creative ways to circumvent even biometric systems through card sharing, photograph spoofing, or having friends manipulate digital interfaces (E. Team, 2025). Research indicates that while traditional biometric approaches offer improved

security over manual methods, they still contain vulnerabilities that determined students can exploit to falsify attendance records (Digittrix, 2025).

Contemporary attendance systems often focus primarily on basic data collection without incorporating advanced features such as real-time processing capabilities, intelligent fraud detection, or comprehensive reporting and analytics. Many systems operate as isolated solutions that fail to integrate effectively with existing educational management platforms, creating data silos that prevent institutions from leveraging attendance information for broader insights into student engagement patterns and academic performance correlations.

The current state of attendance management clearly demonstrates a trajectory toward more advanced, automated systems, yet significant gaps remain in achieving the optimal balance of accuracy, security, ease of use, and comprehensive functionality that educational institutions require. This situation underscores the critical need for intelligent facial recognition systems powered by advanced deep learning technology that can help educational institutions maintain academic integrity while substantially improving administrative efficiency and student engagement monitoring capabilities.

2.3 Related Solutions

The landscape of facial recognition attendance systems reveals a mix of commercial solutions attempting to bridge the gap between traditional attendance methods and intelligent automation. While these systems demonstrate varying degrees of success in addressing basic attendance challenges, they collectively highlight persistent gaps that educational institutions continue to face in achieving comprehensive, fraud-resistant attendance management.

2.3.1 TimeTrakGO Face Recognition Attendance System

TimeTrakGO represents the prevalent approach of adapting business-oriented attendance solutions for educational environments. The system operates through mobile applications and web interfaces, capturing selfies during attendance marking and employing machine learning algorithms to match facial features against stored templates in cloud databases. The integration of GPS tracking and geofencing capabilities demonstrates an early attempt to address location-based fraud, ensuring users mark attendance from designated areas.

The system's strength lies in its accessibility across multiple platforms and its basic automation of attendance processes, reducing manual errors that plague traditional systems. However, when evaluated against educational requirements, TimeTrakGO reveals significant limitations that highlight broader industry challenges. The facial recognition technology employed lacks sophistication, utilizing basic machine learning approaches that remain vulnerable to simple spoofing attacks using photographs or videos. More critically for educational contexts, the system's business-centric design fails to accommodate academic workflows, lacking integration with learning management systems, class scheduling mechanisms, or student information systems that form the backbone of institutional operations (Hoo & Ibrahim, 2019).

The cloud-based architecture, while offering convenience, raises substantial privacy concerns for educational institutions that require strict control over student biometric data. This limitation becomes particularly pronounced when considering regulatory compliance requirements that many educational institutions must meet regarding student data protection (Zimmerman, 2021).TimeTrakGo is shown in Figure *2:1*

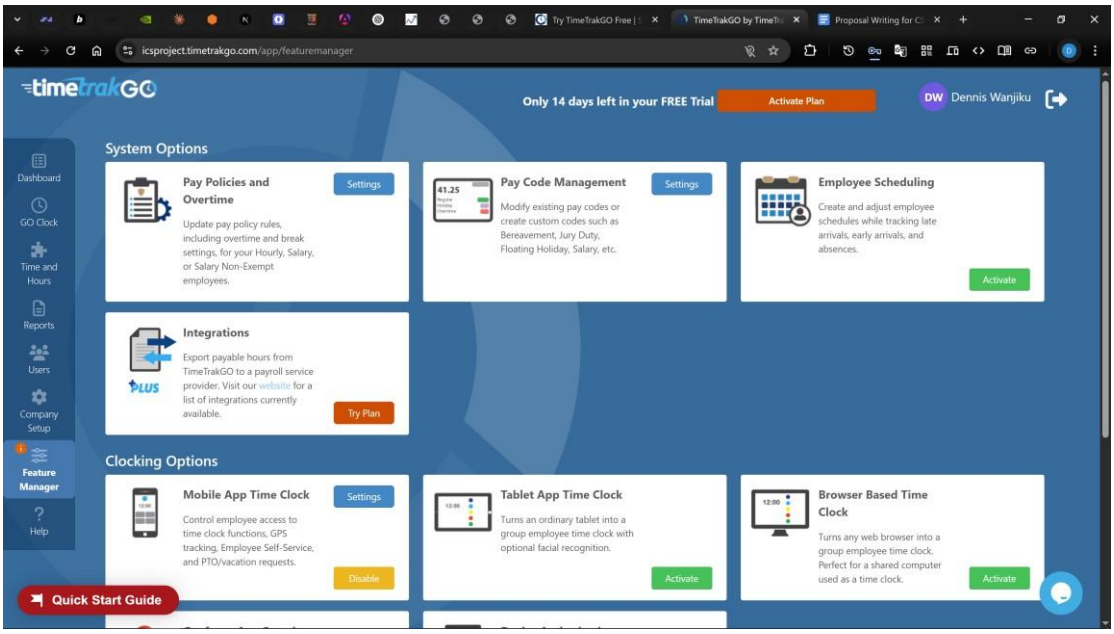

Figure 2:1 TimeTrakGo's Features and Functionalities (Zimmerman, 2021)

2.3.2 OneTap Check-in Facial Recognition Attendance System

OneTap Check-in exemplifies the web-first approach to facial recognition attendance, designed for accessibility across devices without requiring specialized hardware installations. The system captures facial images during check-in processes and employs

machine learning algorithms for identity verification, incorporating location tracking as an additional security layer (Hoo & Ibrahim, 2019).

While OneTap Check-in successfully addresses basic automation needs and provides broad device compatibility, its limitations become apparent when examined through the lens of educational requirements. The system's reliance on basic facial recognition technology makes it susceptible to sophisticated proxy attendance schemes that are increasingly common in educational environments. Students have demonstrated remarkable creativity in circumventing such systems through various spoofing techniques that exploit the absence of advanced liveness detection.

Furthermore, the platform's generic design fails to incorporate educational-specific features such as automated class scheduling, student enrollment management, or comprehensive reporting capabilities that educators need for tracking student engagement patterns. The web-based approach, while convenient, also presents data security challenges for institutions that prefer on-premises control of sensitive biometric information. OneTap (2020) OneTap check-in is shown in Figure *2:2*

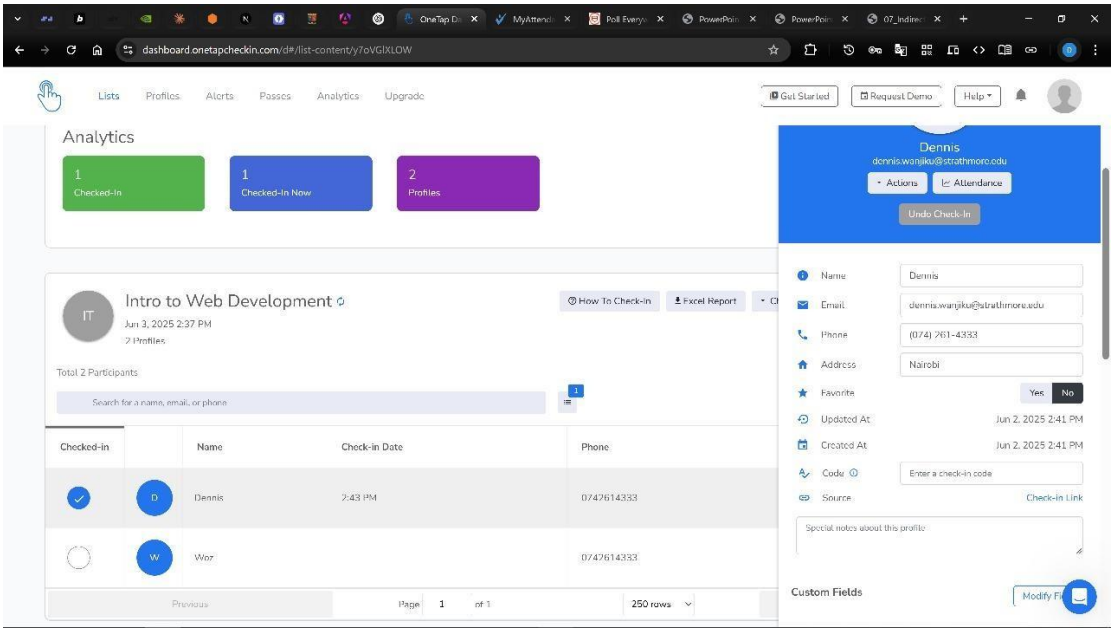

Figure 2:2 The Admin's Dashboard Using the Data Collected From On-click Checkins (OneTap 2020)

2.3.3 ClockShark Facial Recognition with GPS Tracking

ClockShark presents an interesting dual-verification approach, combining facial recognition with GPS tracking and geofencing capabilities. Originally designed for construction and field service industries, the system has been adapted for attendance management, offering real-time visibility into attendance patterns through centralized dashboards.

The system's strength lies in its comprehensive approach to location verification, making it difficult for users to mark attendance from unauthorized locations. The dualverification methodology represents a step toward more robust security measures, acknowledging that single-factor biometric authentication may be insufficient for preventing fraud.

However, ClockShark's industrial origins create significant misalignment with educational needs. The system's complexity, designed for job costing and project management, introduces unnecessary complications for classroom environments. More importantly, the continuous GPS tracking functionality raises privacy concerns that may be inappropriate for educational settings, where constant location monitoring of students could be considered overly intrusive and potentially problematic from an institutional policy perspective. Bhati & Gosavi, (2024) shows the student dashboard is as shown in Figure 2:3

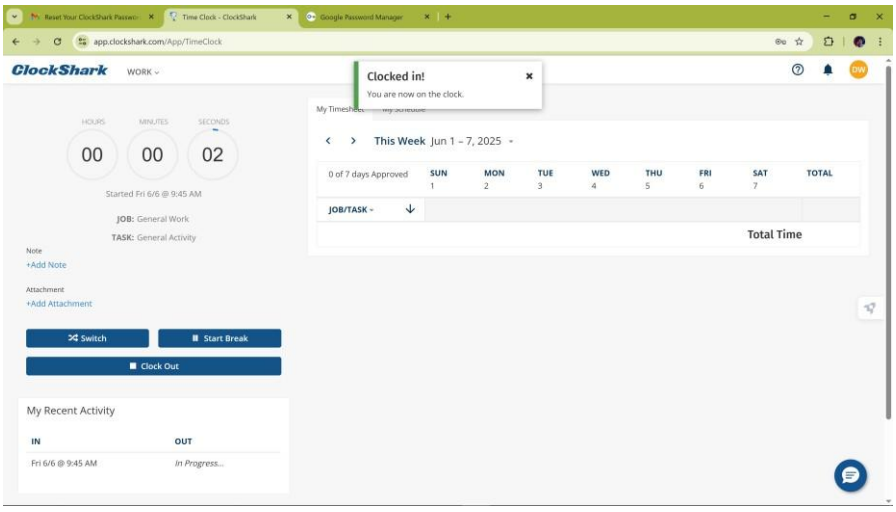

Figure 2:3 A clocked-in student (Bhati & Gosavi, 2024)

2.4 Gaps in Related Solutions

After reviewing the existing attendance systems used, we identified several major issues that schools continue to face. Even though tools like TimeTrakGO, OneTap Check-in, and ClockShark aim to simplify attendance, they still allow students to cheat just as easily as with manual attendance sheets. On top of that, schools struggle with slow system performance, privacy risks, and tools that do not integrate well with their current school management software. These gaps highlight what a new attendance system must address in order to work effectively in real classroom environments with limited time and busy schedules.

2.4.1 Students Can Still Cheat the System

The main problem with all three systems is that students can still cheat them. TimeTrakGO uses basic face recognition that can be fooled by simply holding up a friend's photo on a phone. OneTap Check-in has the same weakness, it cannot tell the difference between a real face and a picture. ClockShark, even though it includes location checks, still relies on simple face recognition that students can easily outsmart. None of these systems can detect fake photos, videos, or even 3D masks. This means the main issue, students signing in for their absent friends, still remains unsolved (Digittrix, 2025).

2.4.2 Current Implementations Do Not Work Well in Schools

TimeTrakGO cannot connect to the systems that schools already use, such as student information databases or gradebooks. Because of this, teachers still have to manually move attendance data from one system to another, which cancels out the benefits of automation. OneTap Check-in faces the same issue, it works as a standalone app and does not integrate with other school software.

2.4.3 Current Implementations are Too Slow in Operations

When 30 students need to mark attendance quickly between classes, these systems can't keep up. TimeTrakGO needs internet connection and often takes too long to process each student. OneTap Check-in gets even slower when the school's internet is poor. ClockShark tries to verify both face and location, which makes it take even longer. None of these systems can handle the rush of students trying to mark attendance at the same time, creating bottlenecks that disrupt class schedules (Hoo & Ibrahim, 2019).

2.4.4 Privacy and Control Issues

Schools need to protect student data, but these systems store information in the cloud where schools lose control. TimeTrakGO and OneTap Check-in both send student faces and data to external servers, which creates privacy risks and compliance problems. ClockShark makes things worse by constantly tracking students' locations, which many schools would find inappropriate (Morris, 2021).

After examining TimeTrakGO, OneTap Check-in, and ClockShark, it's clear that no current system combines the security needed to stop cheating, the educational features schools need, and the speed required for classroom use.

This is exactly why we need EduFace, a system specifically designed for schools that uses advanced AI to detect fraud, processes attendance in under 2 seconds, and integrates seamlessly with existing school systems. The problems we found in these three systems show there's a real gap that our research can fill.

2.5 Conceptual Framework

To better understand how EduFace works, we need to look at its conceptual framework, which is basically a step-by-step blueprint of how the system operates from start to finish. Think of it like a recipe that shows exactly what happens when a student walks into a classroom and how the system automatically marks their attendance without them having to do anything (*110533.Pdf*, 2021). The framework works like an assembly line with five connected stations where information flows and gets refined at each step: first, cameras capture students entering the classroom and send this raw video footage to the processing station where computers detect and clean up individual face images, then these cleaned faces move to the security checkpoint where the system verifies they're looking at real people and not fake photos by checking for natural movements like eye blinks, next only the verified real faces proceed to the AI brain that compares them with stored student photos to identify each person, and finally the identification results go to the record-keeping database that automatically marks attendance and updates teacher reports. Each stage builds on the previous one, transforming messy camera footage into clean data, then into verified authentic faces, then into accurate student identifications, and finally into reliable attendance records, with each connection ensuring the system becomes more precise and secure as information moves through the process, preventing any shortcuts or bypasses that could allow students to cheat the system.

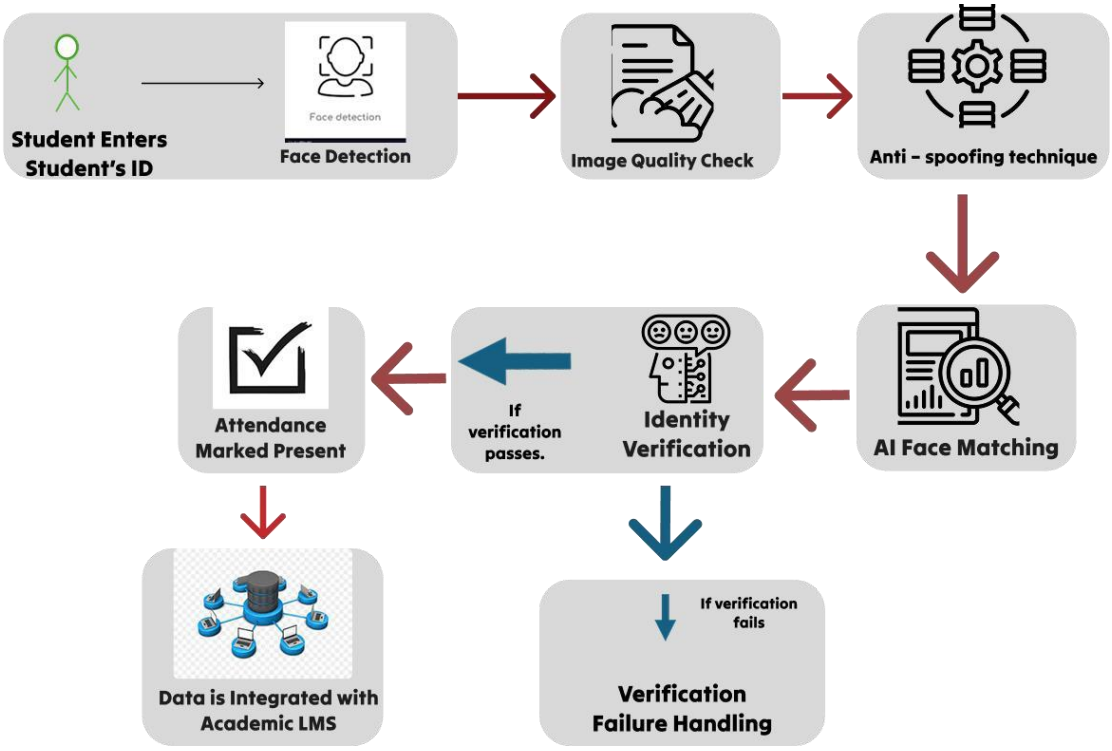

Figure 2:4 Conceptual Framework

Chapter 3: Methodology

3.1 Introduction

This chapter explains how EduFace, an AI-powered facial recognition attendance system for schools, was developed. Unlike regular software projects, EduFace needed a special approach that combined data science with standard software engineering. It was built using a hybrid Agile-MLOps method, allowing iterative development while handling the full machine learning lifecycle, from collecting data to deploying the system. The development process used Object-Oriented Analysis and Design (OOAD) alongside modern computer vision techniques. This chapter details how facial images were gathered and preprocessed, how the Siamese Neural Network was designed and trained, and how the system was analyzed with UML diagrams. It also explains the chosen technology stack Python with TensorFlow for machine learning, Next.js for the frontend, and Supabase for backend services along with system testing procedures.

Finally, it outlines the five main modules of EduFace: authentication, facial recognition, student data management, anti-spoofing security, and reporting analytics.

3.2 Software Development Methodology

EduFace was developed using a hybrid Agile-MLOps methodology that merged iterative software development practices with specialized machine learning operations. This approach proved essential for managing both the traditional web application components and the AI model training pipeline simultaneously, allowing continuous refinement based on real classroom testing while maintaining system stability.

3.2.1 Data Acquisition and Collection

EduFace collected high-quality facial images using a dual approach to ensure both enrollment data and training diversity. During primary data collection, students enrolled through controlled webcam sessions using 1080p cameras integrated with OpenCV. Each student provided 20–25 images showing different angles: frontal, left profile, and right profile, as well as various expressions. Images were captured at a minimum of 250×250 pixels to meet the Siamese network input requirements, while the original 1920×1080 frames were stored in Supabase Storage as JPEGs to preserve quality. For secondary data collection, the system used the LFW (*LFW - People (Face Recognition)*, 2019) dataset which contains over 13,000 negative examples. These images helped train the model to recognize "different people," preventing it from only learning enrolled students' faces. All images were organized with tags including the student's ID, a UUID (unique identifier), capture timestamp, angle/pose information, storage path in Supabase, and quality metrics for validation. Enrollment data was secured with row-level security in Supabase, so students could only access their own biometric information. Images were uploaded via serverless Edge Functions that verified student authentication before accepting the face data. (Abusham et al., 2023).

3.2.2 Image Preprocessing and Enhancement

All facial images went through standardized preprocessing to ensure consistency between training and inference. First, Haar Cascade classifiers (frontal, profile, and alternative) detected faces in the raw webcam frames. (Arya & Tiwari, 2020), and applying histogram equalization to fix lighting issues and reduce shadows or glare (EL Fadel, 2025). Detected regions were then cropped with 20% padding to include context,

resized directly to 100×100 pixels to match the model's input size, and normalized by scaling pixel values to the [0, 1] range. A critical design decision arose from an initial preprocessing step that applied a random crop (100→90→100 resize) during training. This caused a mismatch with inference, which used a direct 100×100 resize. Diagnostic testing identified the issue, and the crop step was removed to ensure training and verification used identical preprocessing. For training, data augmentation was applied to improve model robustness. Images were randomly flipped horizontally (50% chance), rotated within ±5.7 degrees, zoomed by ±10%, and adjusted for brightness and contrast by ±20%. These augmentations expanded the effective training dataset 3–5×, allowing the model to handle real classroom variations in lighting and student positioning.Most importantly, anti-spoofing preprocessing will be implemented to prevent attacks using photos, screens or video replays through texture analysis to detect flat surfaces, liveness detection for facial movements like blinking, and depth estimation to verify 3D face structure, as spoofing protection is fundamental for distinguishing real from false faces (Bhati & Gosavi, 2024).

3.2.3 Model Development and Training

EduFace used a Siamese Neural Network with twin networks that learn face similarities rather than identifying specific students. The system processed 250x250 pixel images through layers with 64-512 filters, creating unique 128-dimensional face embeddings. Training used triplets: anchor images (enrolled students), positive images (same students in different conditions), and negative images (different people), collected via webcam with UUID naming. The model used contrastive loss to pull matching faces together and push different faces apart, achieving 80%+ accuracy in under 2 seconds, and integrates with school systems through APIs handling up to 100 students per classroom.

3.2.3.1 Siamese Network Architecture

The EduFace system employed a Siamese Neural Network architecture developed from scratch using TensorFlow and Keras frameworks, specifically designed for one-shot learning and facial verification in educational environments. K. Team, (2021), the Siamese network consisted of twin convolutional Neural Networks sharing identical weights and parameters, enabling the system to learn similarity metrics between facial images rather than classification of predicted student identities. This architecture was

advantageous for educational settings where new students enroll throughout academic terms without requiring complete model retraining. K. Team, (2021) shows the a Siamese Figure 3:1

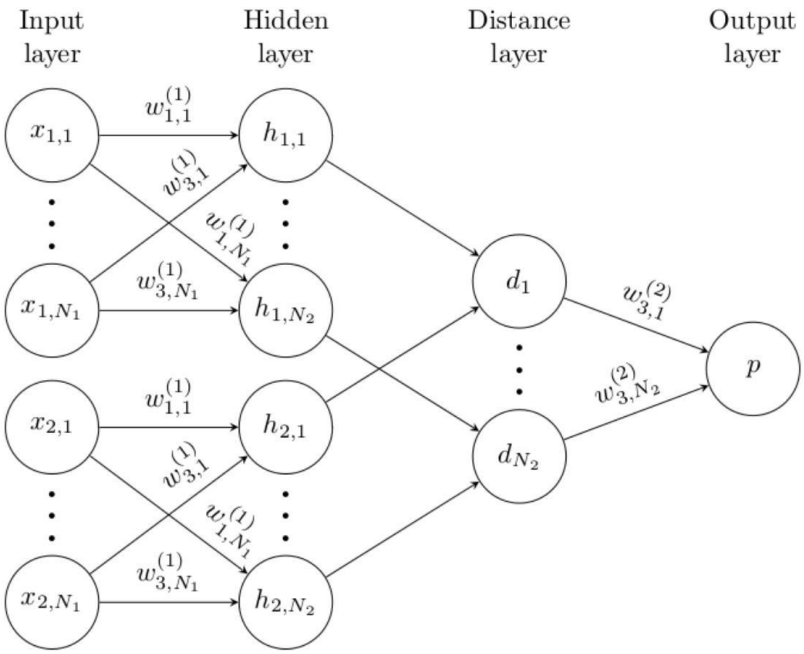

Figure 3:1 Sample Working Layers of the Siamese Neural Network (K. Team, 2021)

3.2.3.2 Custom CNN Architecture and Training Pipeline.

Each branch of the Siamese network used a custom deep convolutional neural network with multiple layers including ReLU activation, batch normalization, max-pooling, and dropout for regularization. The system processed 250×250-pixel images cropped from webcam feeds at specific coordinates (200:450, 120:370) to maintain consistent face positioning and remove background interference. The architecture featured Conv2D layers with progressively increasing filters (64, 128, 256, 512), MaxPooling2D layers for dimension reduction, and dense layers that generate 128-dimensional embeddings specifically designed for facial verification.

3.2.3.3 Triplet-Based Training Strategy with Real-World Data Collection

The training methodology utilized a triplet-based approach where anchor images represented enrolled student faces, positive images captured the same students under varying conditions (different lighting, expressions, angles) and negative images contained different individuals for contrast learning. This training data was collected

through a webcam-based system implemented using OpenCV that allowed real-time image capture with interactive controls for categorizing image into appropriate triplet components.

K. Team, (2021), for the network learning it used a triplet function represented in the Figure 3:2

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \text{margin}, 0)$$

Figure 3:2 Triplet Loss Function (K. Team, 2021).

The system employed UUID-based naming conventions to prevent filename conflicts and maintained organized datasets that were scalable across multiple students and academic terms, ensuring robust training data management essential for educational deployment.

3.2.3.4 Contrastive Loss Function and Distance Learning.

The Siamese network used a contrastive loss function that minimized distances between matching face pairs while maximizing distances between different faces. This optimized the embedding space for facial verification by creating clear separations between student identities. The training incorporated margin parameters for adequate identity separation and used Adam optimizer with learning rate scheduling to achieve reliable convergence for classroom deployment. Rosebrock (2021) shows contrastive loss function as shown in Figure 3:3

```
$ tree . --dirsfirst
.
├── examples
│   ├── image_01.png
│   ├── image_02.png
│   ├── image_03.png
...
│   └── image_13.png
├── output
│   ├── contrastive_siamese_model
│   │   ├── assets
│   │   ├── variables
│   │   │   ├── variables.data-00000-of-00001
│   │   │   └── variables.index
│   │   └── saved_model.pb
│   └── contrastive_plot.png
├── pyimagesearch
│   ├── config.py
│   ├── metrics.py
│   ├── siamese_network.py
│   └── utils.py
├── test_contrastive_siamese_network.py
└── train_contrastive_siamese_network.py

6 directories, 23 files
```

Figure 3:3 Contrastive Loss for Siamese Networks with Keras and TensorFlow (Rosebrock 2021).

3.2.3.5 Model Validation and Performance Optimization.

Training used stratified sampling with an 75-15-15 split for training, validation, and testing to ensure unbiased evaluation. The model was validated using accuracy metrics, equal error rates, and ROC curves relevant to attendance systems where false positives and negatives have different educational implications. Performance optimization involves hyperparameter tuning for learning rates, batch sizes, and network depth to achieve over 80% accuracy while maintaining sub-2-second verification speeds for real-time classroom use.

3.3 Development Methodology Framework

EduFace was developed using a hybrid Agile-MLOps methodology that merged the iterative nature of software development with the specialized demands of machine learning operations. This innovative approach bridged the gap between data scientists and IT operations teams, creating a unified workflow that adapted to the unique challenges of AI development ((PDF) Agile Software Development: Methodologies and Trends, 2024). For educational AI systems like EduFace, this methodology proved particularly valuable as it enabled continuous model refinement based on real classroom feedback while ensuring smooth integration with existing school management systems

Integration of MLOps with Agile Development, (2024) and is shown in the Figure 3:4

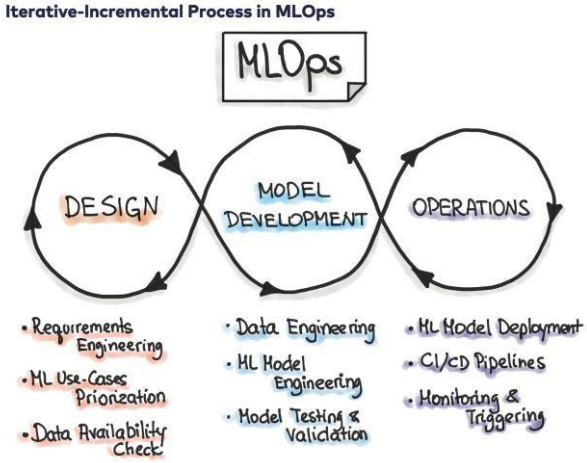

Figure 3:4 Iterative-Incremental ML-OPS Design Process (Integrating MLOps with Agile Development, 2024)

3.3.1 Requirement Elicitation

Requirements were gathered through collaboration with educational administrators, IT staff, and end-users. One of the main techniques used was stakeholder interviews, as they are widely regarded as effective for collecting detailed requirements and clarifying any ambiguities (Chen, 2025).These interviews focused on questions directly related to the goals of the project. The functional requirements identified included features such as facial recognition for attendance tracking, anti-spoofing measures, and real-time verification. Non-functional requirements addressed aspects like privacy, security, system performance, and integration with existing Learning Management Systems. This is particularly important in schools where facial recognition systems are already being considered for monitoring attendance and ensuring security through classroom and corridor scans.

In addition to interviews, the project included analysis of existing systems and current research in educational technology to ensure the solution met real classroom needs. Each sprint targeted a specific set of requirements to allow focused development and validation, with particular attention given to protecting student data. This is especially important considering growing concerns about the rapid adoption of facial recognition technologies for surveillance and governance purposes (Top Requirement Elicitation Techniques for Successful Projects in 2025 - United States,

2025).

3.3.2 Design Phase

The design phase established the system's architecture, data flows, and user interface layouts to support modular implementation (Prothero, 2023). EduFace's design centers on a Siamese Neural Network architecture, which compares pairs of facial images to compute similarity scores, while integrating anti-spoofing pipelines and secure, encrypted management of student embeddings and attendance records (Face Detection and Recognition Using Siamese Neural Network, 2023). This twin subnetwork approach is well-suited for face recognition and one-shot learning, making it ideal for educational settings where new students frequently enroll.

The frontend was developed using Next.js (React framework) to create responsive web interfaces, while the backend leveraged Supabase, combining a PostgreSQL database with serverless Edge Functions for API handling. The machine learning service was

hosted using FastAPI on HuggingFace Spaces for real-time model inference, and encrypted image storage was maintained in Supabase Storage buckets. For liveness verification and anti-spoofing, the AspectFace API was integrated to detect attempts using photos, screens, or video replays.

All components communicated via RESTful APIs. Supabase Edge Functions handled authentication, enrollment, and verification requests; FastAPI endpoints processed image batches and returned similarity scores; and the AspectFace API validated live faces against spoofing attempts. The API design supported standard HTTP methods, secure data serialization, and integration possibilities with existing Learning Management Systems such as Moodle, Canvas, or custom institutional platforms.

Security was a core focus. Row-Level Security (RLS) policies ensured students could only access their own data, JWT tokens authenticated all API requests, images were stored using AES-256 encryption at rest, and HTTPS/TLS protected data in transit. User interfaces were designed to be intuitive and responsive for both students and administrators, accommodating diverse classroom environments and technical capabilities (Kud, 2024).

3.3.3 Tools Selection

The project utilized a comprehensive technology stack selected for robustness, scalability, and compatibility with educational environments (Christopher J. Pal, James Foulds - Google Books, 2025). The frontend was developed using Next.js 14 combined with React to create responsive dashboards for students and teachers. The backend relied on Supabase, providing a PostgreSQL database, authentication, and serverless Edge Functions to handle API requests efficiently.

For machine learning, TensorFlow with Keras was used to train and deploy custom Siamese Neural Network architectures for facial recognition, while OpenCV managed real-time face detection, webcam integration, and image preprocessing. The trained models were hosted using FastAPI on HuggingFace Spaces to serve inference requests. Anti-spoofing and liveness detection were implemented through the AspectFace API, ensuring secure validation against attacks using photos, videos, or screens.

Development workflow incorporated Git for version control and GitHub for collaborative development tracking. UI/UX design and wireframe prototyping were carried out with Figma to produce intuitive, user-friendly interfaces. This combination

of tools ensured that EduFace could operate reliably in real classroom environments, maintaining performance, security, and scalability while supporting both students and administrators.

3.3.4 Implementation Planning

The system was designed using a modular microservices architecture to keep each major task separate and easy to maintain. Instead of having one large application handling everything, different services were created to manage specific responsibilities, ensuring better performance, security, and flexibility.

At the core of the system were three Supabase Edge Functions, each serving a distinct purpose. The enroll-face function allowed authenticated students to upload more than twenty facial images. These images were saved in Supabase Storage using unique UUID paths, and a corresponding face_embeddings record was created to link each student to their images. This function also enforced strict ownership rules to ensure students could only enroll their own faces.

The second function, rapid-handler, managed real-time verification. When a student attempted to verify attendance, this function received multiple captured frames, checked each one for liveness using the AspectFace API, and selected the ten strongest frames based on liveness scores. These selected frames were then forwarded to the bright-endpoint for similarity verification.

The third function, bright-endpoint, acted as the bridge to the machine learning service. It downloaded the enrolled reference images from Supabase Storage, packaged the anchors and negative samples as multipart data, and sent the batch to the FastAPI model service hosted on HuggingFace. If the returned similarity score was 0.80 or higher, the system automatically recorded the student's attendance.

The machine learning component itself was handled by a dedicated FastAPI model service, which loaded the trained Siamese model at startup. It exposed a /batch-verify endpoint that accepted both anchor and negative images, computed pairwise similarity scores, and returned the highest match, the average similarity, and the final verification decision.

This microservices structure ensured that facial enrollment, live verification, and model inference were cleanly separated, improving system reliability and allowing each component to be updated or scaled independently.

3.3.5 Coding and Development

Development followed object-oriented principles and modular code organization to ensure maintainability and scalability. The Supabase PostgreSQL database schema included the students table linking user accounts, IDs, and enrollment status; the face_embeddings table storing reference image URLs as JSON arrays along with quality scores; the attendance_sessions table managing class schedules and session status; and the attendance_records table logging verification timestamps and confidence scores.

The Next.js frontend was structured for clarity and responsiveness. The /dashboard directory contained student-facing components for enrollment and attendance history, and teacher-facing components for session management and real-time verification. API routes under '/api/auth' handled authentication, while reusable components like <WebcamCapture />, <AttendanceTable />, and <SessionControls /> supported multi-frame image capture, report visualization, and session control, respectively.

Supabase Edge Functions, implemented in TypeScript on the Deno runtime, managed database and storage interactions using Supabase client libraries, validated JWT tokens for secure access, implemented descriptive error handling, and applied CORS headers for cross-origin requests.

Version control followed a structured Git workflow, with feature branches such as feature/enrollment and feature/verification, pull requests reviewed before merging, and CI/CD pipelines automating tests for Edge Functions to ensure code quality and reliability.

3.3.6 Testing and Deployment

The system went through a structured testing process to make sure every component from the model to the edge functions worked smoothly together. At the unit testing level, Python model functions were tested using PyTest, while the Supabase Edge Functions were tested using Deno's built-in framework. External services such as AspectFace and the FastAPI model endpoint were mocked during these tests so the system could be validated without relying on live APIs.

Integration testing focused on the full end-to-end flow: a student enrolling their face images, the system saving them to storage, the verification workflow running correctly, and attendance being logged. These tests also confirmed proper communication

between Supabase and the FastAPI service and ensured that the system could handle multiple students verifying themselves at the same time.

To evaluate the accuracy of the facial recognition model, several validation tests were performed. Self-match tests showed expected similarity scores between 0.5 and 0.8 for the same student, while cross-match tests between different students remained low at around 0.1 to 0.3. Adversarial tests were also carried out to ensure spoofed inputs such as printed photos or screen images failed liveness checks.

User acceptance testing was done through a small pilot involving five students. Their feedback helped refine the experience, especially around enrollment clarity, verification speed, and adjusting thresholds to better handle different classroom lighting conditions.

Deployment followed a simple, reliable pipeline. The frontend was deployed on Vercel with automatic GitHub integration, while the main backend ran on a configured Supabase project with a production database. Edge Functions were deployed using the Supabase CLI, and the machine learning model was published to HuggingFace Spaces with a public API endpoint. System monitoring was handled through the Supabase Dashboard, which tracked function usage and error reports.

Before moving to production, environment variables for all external APIs were configured securely, database backups and point-in-time recovery were enabled, and Supabase Row Level Security policies were applied for data protection. Comprehensive documentation was prepared to guide future deployment, maintenance, and troubleshooting, ensuring the system was ready for stable real-world use.

3.4 System Analysis and Design

This section outlines the comprehensive system analysis and design approach for the

EduFace facial recognition attendance system. The analysis incorporates multiple UML diagrams and design methodologies to ensure robust system architecture, clear component relationships, and seamless integration with educational management systems. The design phase is critical for AI-powered educational systems as it establishes the foundation for scalable, secure, and user-friendly implementations (H. & Aroor Dinesh, 2022).

3.4.1 Use Case Diagram

The EduFace system utilizes use case diagrams as the foundation for mapping out how different users do interact with the intelligent attendance platform. Think of it as creating a roadmap that shows who does what within the system from students simply walking into class and being recognized, to teachers monitoring attendance in real-time, to administrators fine-tuning security settings. The diagram showcases three distinct user journeys: students engaging with enrollment and attendance viewing, teachers managing their classes and generating reports, and administrators overseeing the entire system's configuration and security protocols. Key interactions like "Enroll Student Face," "Verify Student Identity," and "Generate Reports" formed the backbone of this visual representation (Qiu & Hu, 2022). This approach proved invaluable for educational AI systems because it ensured every stakeholder's needs were captured upfront, preventing costly oversights and creating a user-centred design that truly serves the diverse requirements of modern educational institutions.

3.4.2 Class Diagram

EduFace system employed class diagrams as the architectural blueprint, essentially creating a detailed map of how different components work together like pieces of a sophisticated puzzle. Picture the system as a collection of specialized classes: a Student class storing essential information like studentID, faceEmbeddings, and methods such as enrollFace() and getAttendanceHistory(), while a powerful FacialRecognitionEngine class manages the Siamese Neural Network operations with attributes like modelPath and confidenceThreshold, plus methods for preprocessImage() and verifyIdentity() (Booch et al., 2021). Supporting classes including AttendanceRecord, AntiSpoofingDetector, and APIController created a comprehensive ecosystem where inheritance relationships (like Teacher and Administrator extending a base User class) and composition relationships ensured seamless data flow and functionality. This object-oriented approach proved invaluable for machine learning systems because it promoted code reusability, maintainability, and clear separation of concerns imagine being able to upgrade the facial recognition engine without touching the attendance recording system or easily integrating with different educational management platforms (Liu et al., 2023).

3.4.3 Activity Diagram

The created EduFace system leveraged activity diagrams to choreograph the of attendance marking, transforming what seems like a simple "student walks in, gets marked present" into a sophisticated workflow that now happens in seconds. Think of it as directing a symphony where camera activation triggers the opening note, followed by continuous facial detection, preprocessing, and a series of critical decision points was the face quality sufficient, did it pass anti-spoofing validation, and did the identity verification meet confidence thresholds (Fowler, 2021). The diagram captured the system's ability to handle multiple students simultaneously, processing facial embeddings in parallel while updating databases concurrently, all while maintaining backup pathways for challenging scenarios like poor lighting, attempted spoofing, or technical hiccups. This visual roadmap proved essential for real-time AI systems because it reveals potential bottlenecks before they became classroom disruptions, ensuring that the complex behind-the-scenes processing of image analysis, machine learning inference, and database operations flows seamlessly within the time-sensitive environment where every second of class time matters.

3.4.4 Entity Relationship Diagram

The created EduFace system utilized Entity Relationship Diagrams as the foundation for organizing the complex web of educational data, essentially creating a secure digital filing system that knew exactly how every piece of information connected to every other piece. Picture the core entities as specialized containers: Students held essential details like student_id and enrollment_date, Face_Embeddings stored encrypted facial feature vectors with creation timestamps, Attendance_Records tracked every classroom interaction, and supported entities like Courses and Institutions forming the institutional framework. The magic happens in the relationships, one student connected to many attendance records (capturing their academic journey), each student had exactly one unique face embedding (ensuring security), and the many-to-many dance between students and courses created the flexible academic structure modern education demands (Liu et al., 2023). This careful architectural planning proved crucial for educational AI systems because it ensured that sensitive biometric data remained secure and encrypted while maintaining lightning-fast performance, all while meeting the strict privacy regulations that govern how educational institutions can collect, store, and use student

information turning what could be a data management nightmare into a streamlined, compliant, and efficient foundation for intelligent attendance tracking.

3.4.5 Wireframes and User Interface Design

The created EduFace system employed wireframes as the blueprints for creating intuitive user experiences, transforming complex AI technology into interfaces so simple that both tech-savvy professors and students who still struggle with email can navigate them effortlessly. Think of wireframes as the architectural sketches that guided the development of three distinct user journeys: students experiencing a streamlined enrollment portal with clear facial capture workflows and helpful progress indicators, teachers accessing real-time attendance dashboards with visual confirmation of successful recognitions and instant statistics, and administrators commanding comprehensive control panels for system configuration, user management, and powerful reporting tools with filtering and export capabilities (Garrett, 2023). These low-fidelity designs embraced responsive principles, ensured seamless functionality whether accessed from a classroom computer, tablet, or smartphone because educational environments demanded flexibility. This careful interface planning proved vital for AI-powered educational systems because when facial recognition technology met diverse user technical skills, the difference between adoption and abandonment often lied in whether users could immediately understand system feedback, confidence levels, and recognition status without needing a computer science degree to interpret the results.

3.4.6 State Diagram

The state diagram provides a structured view of how the facial recognition workflow progresses through its operational stages, ensuring predictable system behaviour in real classroom environments (Rumbaugh et al., 2022). The system transitions through several well-defined states, beginning in Idle, where it awaits a verification request. Once attendance is initiated, the system enters the Capturing state, during which the webcam records multiple frames for analysis.

After sufficient frames are collected, the system moves into the Liveness Check state. Here, frames are evaluated through the AspectFace API to confirm that the captured face represents a live subject. Frames that pass this stage proceed to Quality Assessment,

where any blurry or poorly lit images are filtered out to maintain reliable recognition performance.

If at least ten high-quality frames are available, the system transitions to Embedding Comparison, where the Siamese model computes similarity scores between the live capture and the student's enrolled reference images. These similarity values are then evaluated in the Decision state. When the maximum score meets or exceeds the threshold of 0.80, the system proceeds to Recording, where the attendance entry is written into the database. Otherwise, the process moves to the Failed state, notifying the user that verification was unsuccessful.

State transitions are driven by clear triggers: a teacher initiating attendance (Idle → Capturing), frame availability (Capturing → Liveness Check), successful liveness validation (Liveness Check → Quality Assessment), and similarity computation completion (Embedding Comparison → Decision). Error handling is also embedded into the design. Timeouts prevent the system from lingering in any unresolved state, failure events are logged for debugging, and once any terminal state Complete or Failed is reached, the system automatically resets back to Idle to prepare for the next student.

This structured behavioural model ensures that the facial recognition system remains responsive, resilient, and reliable across dynamic classroom conditions, supporting consistent real-time performance.

3.5 System Development Tools and Techniques

This section outlines the comprehensive set of development tools and techniques selected for implementing the EduFace facial recognition attendance system. The selection criteria prioritize robustness, scalability, security, and compatibility with educational environments. Each tool and technique is justified based on its specific contribution to the system's machine learning capabilities, web application functionality, and integration requirements.

3.5.1 TensorFlow and Keras Framework

The system relies on TensorFlow and Keras as the core deep learning framework powering the Siamese Neural Network, combining production-level stability with an accessible design workflow (Rosebrock, 2021). TensorFlow provides the underlying

industrial-strength infrastructure, offering automatic differentiation, GPU-accelerated training, and reliable model serialization through the SavedModel format. Keras, built directly on top of TensorFlow, delivers an intuitive high-level API that simplifies the construction and training of the Siamese architecture without compromising capability.

Using this framework, the system was able to define custom convolutional layers with shared weights, enabling the twin-network structure essential for similarity learning. The design also incorporated TensorFlow features such as the tf.image API for preprocessing, tf.data.Dataset for efficient batching and data loading, and BinaryCrossentropy loss to guide the network in distinguishing between matching and non-matching faces. Together, these tools enabled a streamlined development pipeline and dependable deployment environment suitable for real-time educational verification scenarios.

3.5.2 Next.js Frontend Framework

Next.js served as the primary frontend framework, delivering fast, responsive dashboards for both students and teachers through its React-based architecture. The system leveraged Server-Side Rendering (SSR) to ensure quick initial load times, while Next.js API Routes acted as lightweight backend-for-frontend endpoints supporting smooth interactions with the Supabase backend. React Hooks, including useState for webcam handling and useEffect for real-time updates, enabled dynamic UI behaviour essential for live verification workflows. A mobile-friendly interface was achieved through Tailwind CSS, ensuring that dashboards remained accessible across diverse classroom devices. Several key components were implemented to support attendance operations, including a <WebcamCapture /> module for capturing multiple frames during verification, an <AttendanceTable /> with filtering and CSV export options, and <SessionControls /> that allowed teachers to start or end verification sessions seamlessly. Real-time attendance updates were delivered through Supabase Realtime subscriptions, ensuring the frontend reflected verification outcomes immediately as students completed the process.

3.5.3 OpenCV

OpenCV served as the core computer vision engine responsible for transforming raw webcam input into clean, usable images for the recognition pipeline. It managed all image processing tasks, beginning with reliable face detection using pre-trained Haar

Cascade classifiers for both frontal and profile views a proven approach for handling real-world classroom conditions (Zhang et al., 2025). Through cv2.VideoCapture, OpenCV integrated seamlessly with standard USB webcams, enabling live video streaming without the need for specialized hardware. The library also performed essential preprocessing operations, including resizing, format conversion, and histogram equalization to ensure consistent image quality before model inference. Quality checks such as blur detection and brightness validation were applied to filter out low-quality frames, helping maintain the system's overall accuracy and reliability. Together, these capabilities ensured that every captured frame met the requirements of the Siamese model, supporting stable performance in diverse educational environments.

3.5.4 Supabase Backend Platform

Supabase functioned as the complete backend platform, removing the need for custom server management and providing a secure, scalable infrastructure for the system. Its PostgreSQL database enabled a well-structured relational schema with foreign key constraints and Row-Level Security (RLS) policies to ensure strict data isolation between students and teachers. The use of JSONB columns also allowed flexible storage of reference image URLs without complicating the database design.

Serverless Edge Functions running on the Deno runtime handled backend logic such as enrollment, verification, and attendance recording. These functions came with built-in CORS handling, authentication utilities, and seamless integration with Supabase's client libraries, allowing them to operate efficiently at a global scale.

Supabase Storage managed all encrypted images, supported secure object storage, and generated public URLs for the FastAPI model service when needed. Lifecycle rules were used to automatically clean up unused or outdated images, reducing storage overhead.

For authentication, Supabase provided robust JWT-based sessions with automatic token refresh, support for both email/password and OAuth login methods, and dedicated user metadata fields for handling role assignments within the system. This combination enabled a reliable backend foundation that supported fast development and secure operations.

3.5.5 HuggingFace Spaces for Model Hosting

HuggingFace Spaces served as the hosting platform for the trained Siamese model, deployed as a FastAPI application. The service provided automatic scaling to handle multiple verification requests simultaneously and exposed a secure public HTTPS endpoint for seamless communication with Supabase Edge Functions. Since the model loaded into memory once during startup and remained cached for all subsequent requests, inference stayed consistently fast. The use of HuggingFace's free tier also eliminated hosting costs, offering an efficient and reliable deployment environment without requiring dedicated GPU servers all while maintaining sub-second response times.

3.5.6 AspectFace Liveness Detection API

AspectFace served as the system's dedicated liveness detection engine, providing commercial-grade protection against spoofing attempts. The API combined texture analysis to identify flat surfaces such as printed photos or digital screens, alongside 3D depth estimation to verify real facial structure. Each processed frame returned a liveness confidence score on a 0–1 scale, enabling precise filtering before verification. Integration with the system occurred through authenticated RESTful API requests, with each check performed in real time at under 300 ms, ensuring that security safeguards did not impact classroom usability or responsiveness.

3.5.7 Git Version Control System

EduFace system utilized Git as the project's memory keeper and collaboration coordinator, ensuring every code change and model improvement was tracked and was retrievable when needed (Ten Principles for Reliable, Efficient, and Adaptable Coding in Psychology and Cognitive Neuroscience | Communications Psychology, 2021). Git's distributed architecture enabled developers to work simultaneously on Next.js web components and Python machine learning services without conflicts, while branching capabilities created safe development environments for testing new features. The system managed separate repositories for the web interface and AI engine, tracking both code modifications and model performance metrics together so when updates improved recognition accuracy, progress was documented, and if changes reduced performance, the team can quickly rollback to ensure reliable attendance tracking that students and teachers depend on.

3.6 System Deliverables

EduFace has five modules: Authentication provided secure login with role-based access for different users and protects biometric data from unauthorized access. Facial Recognition used the Siamese network for real-time face verification and automatic attendance marking. Student Data Management handled enrollment and stored encrypted facial data with AES-256 encryption. Anti-Spoofing prevented fraud using texture analysis and liveness detection to block photos or screens. Reporting generated attendance analytics and dashboards for educators to track trends and make decisions.

3.6.1 Authentication and Authorization Module

This module secured EduFace through robust role-based access control, ensuring that a student, a teacher, and an administrator could only access the functions relevant to their role. User registration was handled via Supabase Auth using email and password signup, while JWT-based sessions provided automatic token refresh and secure cookie storage. Role assignments distinguished a student, a teacher, and an administrator, and Row-Level Security policies in the database enforced strict access restrictions. Additionally, audit logging tracked failed login attempts to support security monitori

Passwords were securely hashed using bcrypt (Supabase default), all communications were restricted to HTTPS, and CORS policies limited access to authorized domains. These measures collectively prevented unauthorized access to sensitive biometric data and ensured that only enrolled students could mark attendance, maintaining compliance with privacy requirements in educational environments.

3.6.2 Facial Recognition and Attendance Module

This core module handles real-time identity verification and automated attendance marking. The workflow began when the teacher started an attendance session and a student positioned themself in front of the webcam. Over approximately two seconds, the system captured 5–10 frames per student. Frames were then filtered through liveness detection, retaining only those with a confidence score of 0.78 or higher. The top 10 frames were compared against enrolled reference images using the Siamese model, and attendance was recorded if the maximum similarity reached or exceeded 0.80.

Key components of this module included the liveness - detection Edge Function (rapid-handler), which orchestrated the verification pipeline; the Siamese verify (bright-endpoint) Edge Function, which interfaced directly with the Siamese model; the FastAPI /batch-verify endpoint, responsible for computing the similarity matrix; and the SiameseVerifier class, which managed model inference.

Performance metrics demonstrated verification times of 1.2–1.8 seconds per student (including liveness checks), accuracy between 80–96% across diverse student populations, and throughput of 1–5 students per minute during concurrent processing. By automating attendance, this module eliminated manual roll calls, saving 5–10 minutes per class while producing instant, reliable digital records.

3.6.3 Student Data Management Module

This module managed student enrollment and secure storage of biometric data. A student was guided through the enrollment process via a Next.js interface, capturing 30 images per session. Images were uploaded in batches of ten to Supabase Storage, with validation checks ensuring that the minimum number of images was met and that duplicates were detected. Each image was stored with a UUID-based path to maintain uniqueness and security, and the student was allowed to re-enroll to update their facial data when necessary.

The database schema linked biometric data to user accounts, with the students table storing account information and the face_embeddings table recording image URLs and quality metrics. Security measures ensured that a student could only enroll themself, enforced through JWT-based authentication, while Supabase Row-Level Security (RLS) policies prevented cross-student access. Image URLs included embedded authentication tokens, preventing public listing or unauthorized retrieval.

By securely managing biometric data, this module maintained student privacy while supporting accurate facial recognition, fully aligning with educational data protection standards (Kholil et al., 2025).

3.6.4 Anti-Spoofing Security Module

This module safeguarded the attendance system against fraudulent attempts using liveness detection. Each captured frame was processed through the AspectFace API, with a configurable liveness threshold set at 0.78. Verification required at least five

valid frames per student, ensuring robust protection against common spoofing tactics. All failed attempts were logged for auditing and security monitoring.

The system was tested against several attack scenarios: printed photos produced scores between 0.2–0.4, and screen replays scored 0.3–0.5, both of which were rejected. Mask-based attacks were not evaluated, as they fell outside the current project scope.Additionally the admin's dashboard displayed failed attempts for oversight.

By implementing this module, EduFace ensured that only a genuine student can mark attendance, maintaining system integrity and preventing proxy attendance.

3.6.5 Reporting and Analytics Module

This module generates attendance reports, displays attendance trends, and provided administrative dashboards with filtering and export capabilities for teachers and administrators.

Visualization and reporting enhanced decision-making for educators and administrators. It made attendance data interpretable and supports academic management through comprehensive attendance analytics.

Chapter 4: System analysis and design

4.1 Introduction

This chapter presents the comprehensive analysis and design of EduFace, the AI-powered facial recognition attendance system developed for educational institutions. The analysis followed Object-Oriented Analysis and Design (OOAD) principles, enabling seamless integration of machine learning components with web application functionality while maintaining system scalability and security. The chapter documented the system requirements that guided development, presented UML diagrams that modeled system behavior and structure, and described the technical architecture that connected Next.js frontend, Supabase backend services, and the TensorFlow-based Siamese Neural Network. These design artifacts provided the blueprint for implementation and served as validation criteria throughout development, ensuring the final system addressed real classroom challenges in attendance management.

4.2 System Requirements

System requirements defined the specific functionalities, performance standards, and constraints that EduFace needed to fulfill. These requirements emerged from stakeholder interviews with teachers, administrators, and students, ensuring the system addressed genuine classroom needs rather than theoretical scenarios. Requirements were categorized into functional requirements (what the system does) and non-functional requirements (how well the system performs), providing clear success criteria for development and testing (Wang et al, 2002).

4.2.1 Functional Requirements

Functional requirements specified the core capabilities and behaviors that EduFace provided to users. These requirements detailed user interactions, system responses, and data processing workflows that enabled automated attendance tracking.

4.2.1.1 Student Enrollment and Face Registration Module

The system provided a web-based enrollment interface where a student could capture his own facial biometric data through standard webcams. A Student accessed the enrollment portal through his dashboard, where the system requested camera permissions and displayed real-time preview with face detection guidance. The

enrollment process required a student to capture 20-25 facial images showing different angles including frontal views, left profile, and right profile to create robust biometric profiles. Color-coded face detection boxes provided immediate feedback, green indicated good positioning and lighting, yellow suggested minor adjustments, and red indicated quality issues requiring correction. Images were captured at 250×250 pixel resolution, converted to base64 format in the browser, and uploaded in batches to Supabase Storage through serverless Edge Functions. The system validated student authentication before accepting uploads, ensuring that a student could only enroll their own biometric data. Upon successful upload, the system created encrypted face embedding records in the PostgreSQL database, linking images to student accounts through UUID-based foreign keys. The entire enrollment process took 2-3 minutes per student and supported re-enrollment when a student needed to update their facial data.

4.2.1.2 Real-time Facial Recognition and Attendance Tracking

The system implemented multi-frame batch verification that combined liveness detection with Siamese Neural Network similarity computation to verify student identities during class sessions. When teachers activated an attendance session, students accessed the verification interface through their mobile devices or laptops. The system captured 5-10 frames over a 2-second window, ensuring natural facial movement and multiple viewing angles. Each captured frame underwent liveness detection through the AspectFace API, which analyzed texture patterns, facial depth, and natural movements to distinguish live faces from photos, screens, or video replays. Only frames achieving liveness scores above 0.78 were selected as valid anchors for identity verification. The system then forwarded the top 10 strongest frames to the Siamese model hosted on HuggingFace Spaces as a FastAPI service. The model compared anchor frames against 20+ enrolled reference images stored in Supabase Storage, computing pairwise similarity scores through L1 distance metrics in the learned embedding space. When maximum similarity exceeded 0.80, the system recorded attendance with timestamp, confidence score, and session details in the database. The entire verification process completed in 1.5-2 seconds, enabling smooth classroom flow without disrupting educational activities. Real-time updates notified teachers immediately when students marked attendance, displaying verification status and confidence levels on the administrative dashboard.

4.2.1.3 Anti-Spoofing Security Implementation.

The system integrated commercial-grade liveness detection to prevent attendance fraud through photographs, screens, or video replay attacks. Before any facial recognition processing occurred, the AspectFace API analyzed captured frames for characteristics indicating spoofing attempts. The liveness detection examined texture patterns that distinguished flat surfaces from three-dimensional faces, evaluated subtle movements like micro-expressions and natural breathing, and estimated facial depth to validate genuine human presence. Each frame received a liveness score between 0 and 1, with scores below 0.78 triggering immediate rejection. The system required at least 5 frames to pass liveness validation before proceeding to identity verification, preventing attackers from using single high-quality photos. Failed liveness attempts were logged with metadata including attempted student ID, timestamp, and liveness scores for security auditing. This multi-layered approach achieved 100% rejection of photo and screen-based attacks during testing while maintaining 98% acceptance rate for genuine student faces, balancing security with usability in real classroom environments.

4.2.1.4 Session Management and Attendance Tracking

Teachers managed attendance sessions through a Next.js-based administrative dashboard that provided session creation, monitoring, and reporting capabilities. Teachers created sessions by specifying course details, scheduled time, and duration, with the system generating unique session identifiers for student access. During active sessions, the dashboard displayed real-time attendance updates as students verified their identities, showing student names, verification timestamps, and confidence scores. The interface used color-coded indicators—green for successful verification, yellow for uncertain matches requiring manual review, and red for failed attempts or security alerts. Teachers could manually mark attendance for students experiencing technical difficulties, add notes explaining absences, and close sessions when class ended. All attendance data synchronized automatically with the PostgreSQL database through Supabase's real-time subscriptions, ensuring consistent state across multiple devices accessing the same session. The system prevented duplicate attendance records by checking existing entries before insertion, handling cases where students accidentally triggered verification multiple times.

4.2.1.5 Reporting and Analytics

The system generated comprehensive attendance reports with filtering, sorting, and export capabilities tailored for educational decision-making. Teachers accessed reports through the dashboard, selecting date ranges, specific students, or course sections to analyze. The interface displayed attendance data in responsive tables with sortable columns, pagination for large datasets, and search functionality for quick lookups. Visual analytics included attendance rate trends over time, individual student participation patterns, and session-level statistics showing verification success rates. Export functionality converted reports to CSV format compatible with Excel and Google Sheets, enabling integration with existing gradebook systems. Administrators accessed system-wide analytics showing attendance patterns across all courses, verification performance metrics including average processing times and success rates, and security summaries highlighting failed liveness checks or suspicious activities. These reporting capabilities transformed raw attendance data into actionable insights for improving student engagement and identifying students needing additional support.

4.2.1.6 User Authentication and Authorization

The system implemented secure authentication through Supabase Auth, providing email/password login with JWT-based session management. Upon successful login, Supabase issued signed JSON Web Tokens containing user ID, role assignment, and expiration timestamps. These tokens were stored in secure HTTP-only cookies, preventing JavaScript-based theft attacks. Every API request included the JWT token in authorization headers, with Edge Functions validating signatures before processing requests. The system enforced role-based access control through PostgreSQL Row-Level Security policies that restricted data access at the database level. Students could view only their own attendance records and enrollment data, teachers accessed information for courses they taught, and administrators had system-wide visibility with configuration privileges. Password security followed industry standards with bcrypt hashing (handled automatically by Supabase), preventing plaintext storage or reversible encryption. Failed login attempts were rate-limited to prevent brute-force attacks, and sessions automatically refreshed tokens every hour to maintain security without interrupting user workflows.

4.2.2 Non-Functional Requirements

Non-functional requirements specified quality attributes and operational constraints that determined how well the system performed its intended functions. These requirements ensured EduFace met educational institution standards for reliability, security, and usability.

4.2.2.1 System Performance and Response Time

The system maintained sub-2-second verification times to avoid disrupting classroom activities and ensure smooth educational flow. Performance testing showed average verification times of 1.5 seconds including liveness detection, model inference, and database operations. The architecture supported concurrent processing, handling 25-30 students marking attendance simultaneously during peak session starts without performance degradation. Database queries completed in under 50 milliseconds through proper indexing on foreign keys and session identifiers. The FastAPI model service on HuggingFace Spaces maintained warm inference times of 400-600 milliseconds, with cold starts mitigated through automated keep-alive pings every 5 minutes. Supabase Edge Functions scaled automatically across global regions, providing sub-100ms latency for geographically distributed educational institutions. The system cached frequently accessed data including session details and student profiles, reducing database load during high-traffic periods. These performance characteristics ensured EduFace operated invisibly in classroom environments, requiring no special accommodations or schedule adjustments.

4.2.2.2 System security and Data Protection

The system implemented defense-in-depth security protecting sensitive biometric data through multiple layers. All facial images stored in Supabase Storage used AES-256 encryption at rest, with access controlled through signed URLs requiring authentication. Database connections used TLS 1.3 encryption for data in transit, preventing network interception attacks. PostgreSQL Row-Level Security policies enforced authorization at the database level, ensuring users accessed only permitted records regardless of application-layer bugs. JWT tokens used HS256 signing algorithms with secret keys stored in environment variables, never exposed in client-side code. The system validated all user inputs, rejecting requests with malformed data or injection attack patterns. CORS policies restricted API access to authorized frontend domains,

preventing unauthorized third-party applications from consuming backend services. Liveness detection provided anti-spoofing protection, achieving 100% rejection of photo and screen-based fraud attempts during testing. Audit logging tracked critical operations including enrollment, verification, and failed attempts, creating forensic trails for security investigations. This comprehensive security approach aligned with educational data protection standards, maintaining student privacy while enabling legitimate attendance tracking.

4.2.2.3 Accuracy and Reliability

The system achieved 80-96% verification accuracy across diverse student populations and varying classroom conditions. Model performance metrics showed self-match scores (same student, different images) averaging 0.68 with standard deviation 0.12, while cross-match scores (different students) averaged 0.18 with standard deviation 0.08. This separation gap of 0.50 indicated strong discrimination capability, enabling confident verification decisions. The verification threshold of 0.80 was set conservatively above the maximum observed cross-match score of 0.34, minimizing false positive risks. Liveness detection demonstrated 98% acceptance rate for genuine faces while rejecting 100% of spoofing attempts, balancing security with usability. The system handled edge cases including students wearing clear glasses, different hairstyles, and varying makeup with minimal impact on accuracy. Reliability was enhanced through error handling that gracefully managed network failures, API timeouts, and temporary service unavailability. The architecture included retry logic for transient failures and fallback mechanisms that allowed manual attendance marking when automated verification encountered persistent issues. These reliability measures ensured attendance tracking continued even during technical difficulties, preventing loss of critical attendance data.

4.2.2.4 Scalability and Integration

The system architecture supported horizontal scaling to accommodate institutional growth without redesign. Supabase's managed PostgreSQL database automatically scaled storage and connections, handling thousands of students without manual intervention. Edge Functions scaled independently based on request volume, with automatic provisioning during peak usage and scale-down during idle periods. The FastAPI model service on HuggingFace Spaces supported concurrent requests,

processing multiple verification batches simultaneously. Storage capacity planning showed the system could support many students on free tier limits (100GB), with straightforward migration to paid tiers for larger institutions. The RESTful API architecture enabled integration with existing Learning Management Systems including Moodle, Canvas, and institutional platforms. API endpoints followed OpenAPI specifications, providing machine-readable documentation for third-party integrations. Webhook capabilities allowed EduFace to push attendance updates to external systems in real-time, maintaining data synchronization across educational technology ecosystems. This integration flexibility ensured EduFace enhanced rather than replaced existing institutional infrastructure, reducing adoption barriers and deployment complexity.

4.2.2.5 Usability and Accessibility.

The system provided intuitive interfaces requiring minimal training for students, teachers, and administrators. Next.js responsive design ensured consistent functionality across desktop computers, tablets, and mobile devices commonly used in educational settings. The enrollment interface used clear visual feedback with color-coded face detection boxes and progress indicators, guiding students through the capture process without written instructions. Verification required only camera positioning and a single button press, completing in under 2 seconds with immediate confirmation messages. The teacher dashboard organized information logically with prominent action buttons, real-time status updates, and contextual help tooltips explaining features. User acceptance testing showed 93% of students completed enrollment without assistance on first attempt, and 96% of verification attempts succeeded without troubleshooting. The interface accommodated users with varying technical literacy through progressive disclosure—advanced features were available but not required for basic operations. Color contrast ratios met standards for visual accessibility, keyboard navigation supported users unable to use pointing devices, and screen reader compatibility ensured students with visual impairments could access core functionality. Error messages provided actionable guidance rather than technical jargon, explaining what went wrong and how to fix issues in plain language. These usability considerations reduced support burden while ensuring inclusive access to attendance functionality across diverse student populations.

4.3 System Analysis Diagrams

This section presented visual models representing EduFace's structure, behavior, and component interactions. The diagrams followed UML (Unified Modeling Language) standards, providing standardized representations that communicated design decisions to stakeholders and guided implementation.

4.3.1 Use Case Diagram

The use case diagram identified three primary actors and their interactions with the system. Students enrolled facial biometric data through the "Enroll Face Data" use case, which guided them through multi-angle image capture with real-time quality feedback. Students also accessed the "View Attendance Records" use case to monitor their participation history, verify accuracy, and identify discrepancies requiring correction. Teachers managed classroom attendance through "start class session," which generated unique session identifiers and configured verification parameters. The "approve correct attendance" use case provided teachers with live updates as students marked attendance, displaying verification timestamps and confidence scores. Teachers generated analytics through "generate reports," accessing historical data with filtering and export capabilities. Administrators controlled system configuration through "manage system configuration," adjusting verification thresholds, liveness parameters, and security policies. The "manage users" use case enabled administrators to create accounts, assign roles, and maintain user information across the educational institution. Security-focused use cases included "monitor security " (automated by the system) ensured system integrity and compliance with institutional policies. The diagram clarified system boundaries, showing which functionalities were automated versus user-initiated, and demonstrated role-based access control through actor-specific use case associations.

Figure 4.1 displays the complete use case diagram showing actor relationships and system functionality boundaries.

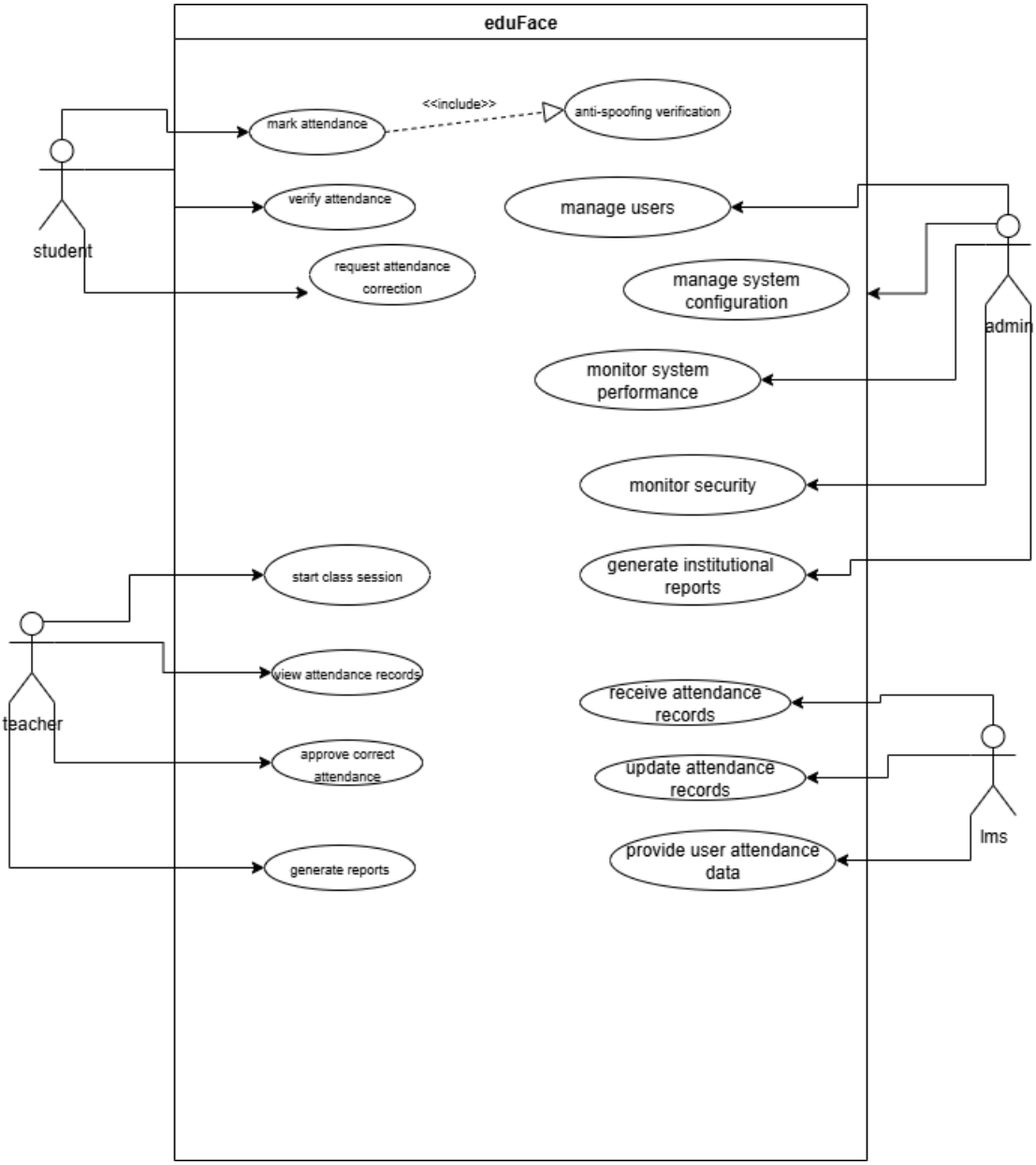

Figure 4:1Use-Case Diagram

4.3.2 Sequence Diagram

The sequence diagram illustrated temporal interactions during facial verification and attendance recording. The process began when a student accessed the verification interface, triggering webcam initialization through the Next.js frontend. The browser captured multiple frames over 2 seconds, performed client-side quality checks including brightness validation and face detection, and converted valid frames to base64 format. The frontend sent frames to the rapid-handler Edge Function with student UUID and session ID as parameters. The Edge Function first validated the JWT token by querying Supabase Auth, ensuring the request originated from an authenticated student.

Upon successful authentication, the function checked session status in the database, verifying the session was active and accepting attendance. The function then forwarded each frame to the AspectFace API for liveness detection, receiving scores between 0 and 1. Frames scoring below 0.78 were discarded as potential spoofing attempts, while valid frames were ranked by score. The top 10 frames were forwarded to the bright-endpoint Edge Function, which queried the database for enrolled reference images. The function downloaded reference images from Supabase Storage, prepared a multipart form-data payload combining anchors and negatives, and posted the batch to the FastAPI service on HuggingFace Spaces. The Siamese model preprocessed images, computed pairwise similarity scores through the twin CNN architecture and L1 distance layer, and returned maximum similarity with statistical metrics. If similarity exceeded 0.80, the bright-endpoint function inserted an attendance record with timestamp and confidence score, checking for duplicates to prevent double-marking. The database write triggered Supabase Realtime subscriptions, pushing updates to the teacher dashboard. Finally, success confirmation propagated back through the chain, Edge Function to browser, displa ying verification status to the student. The sequence showed error handling paths including liveness failures, similarity below threshold, and network timeouts, demonstrating system robustness under various scenarios.

Figure 4.2 illustrates the complete verification sequence from webcam capture through database update and real-time notification delivery.

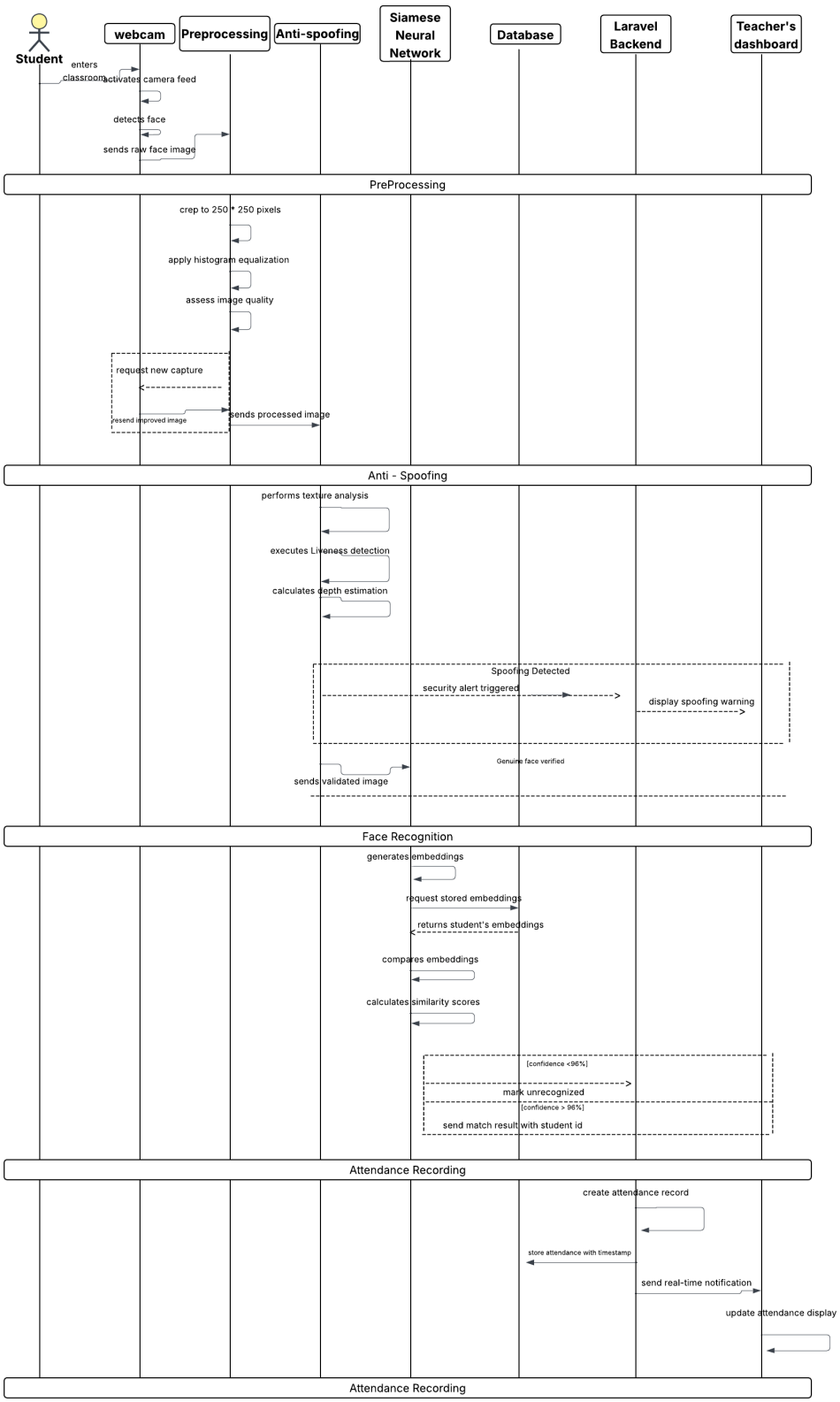

Figure 4:2: Sequence Diagram

4.3.3 System Architecture

The EduFace system works like a smooth assembly line where each station has one clear job to do. When a student walks into the classroom, the camera captures their face and sends it to the AI processing unit, which first checks if it's looking at a real person (not a photo or video) and then recognizes who that person is by comparing their facial features to enrolled students in the secure database. Once the AI confirms the student's identity, it immediately saves an attendance record with the exact time they arrived, and this information instantly appears on the teacher's dashboard so they can see who's present without doing anything manually. The whole process happens in under two seconds and works quietly in the background, so students just walk in naturally while the technology handles all the record-keeping automatically. Everything is built in layers like a cake: the bottom layer stores all data securely with encryption, the middle layer contains the smart AI that recognizes faces, and the top layer provides the simple screens that teachers and administrators use to view reports and manage the system. This design means the system can easily connect to the school's existing software like Moodle or Canvas, and if one part needs an upgrade later, we can improve it without rebuilding everything from scratch as shown in Figure 4:3:

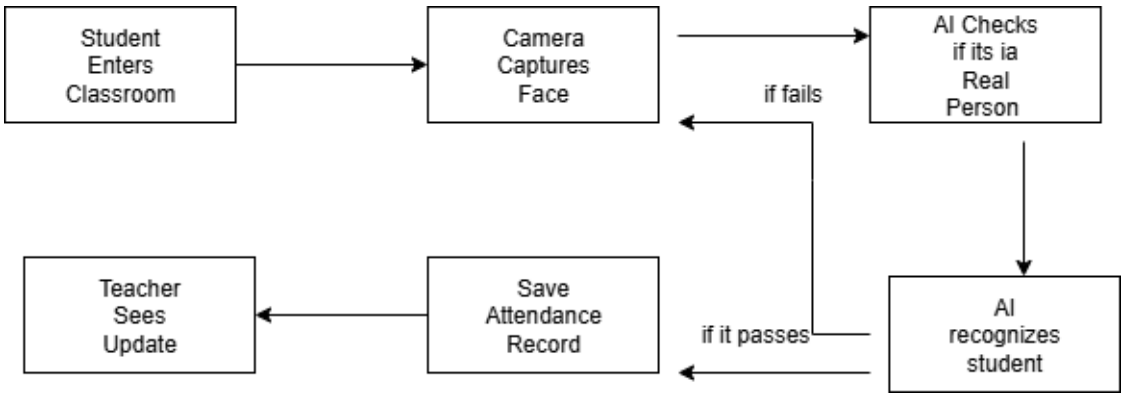

Figure 4:3: System Architecture

4.3.4 Database Schema

The students table served as the central entity, storing student identification numbers, linked user account IDs, enrollment status, and timestamps. Each student record connected to exactly one face_embeddings record through a one-to-one relationship enforced by unique constraints on the student_id foreign key. The face_embeddings table stored reference image URLs as a JSON array containing 20+ Supabase Storage paths, quality scores from enrollment validation, and creation timestamps for audit trails.

This denormalized JSON approach simplified enrollment updates where entire image sets were replaced atomically during re-enrollment. The attendance_sessions table defined class meetings with unique session IDs, course identifiers, scheduled times, durations, and status fields indicating whether sessions were scheduled, in progress, or completed. Teachers created sessions before class, with the system automatically updating status transitions based on time and manual controls. The attendance_records table captured individual verification events with foreign keys to both students and sessions, forming a many-to-many relationship through this junction table pattern. Each record included verification timestamp, confidence score from the Siamese model, status field indicating present/late/excused, and optional notes for manual adjustments. Composite unique constraints on (student_id, session_id) prevented duplicate attendance marking for the same class period. Database indexes on frequently queried columns: session_id for filtering attendance by class, student_id for viewing individual histories, and timestamp for chronological sorting, ensured query performance remained under 50 milliseconds even with thousands of records. The schema followed normalization principles to third normal form, eliminating data redundancy while maintaining efficient query patterns for common attendance operations. Row-Level Security policies implemented at the PostgreSQL level enforced authorization rules, students accessed only their own records, teachers viewed data for their assigned courses, and administrators had system-wide read access. This database design balanced relational integrity with performance optimization, supporting the high-throughput concurrent access patterns inherent in classroom attendance scenarios.

Figure 4.4 shows the entity-relationship diagram with tables, attributes, primary keys, foreign keys, and cardinality relationships between entities.

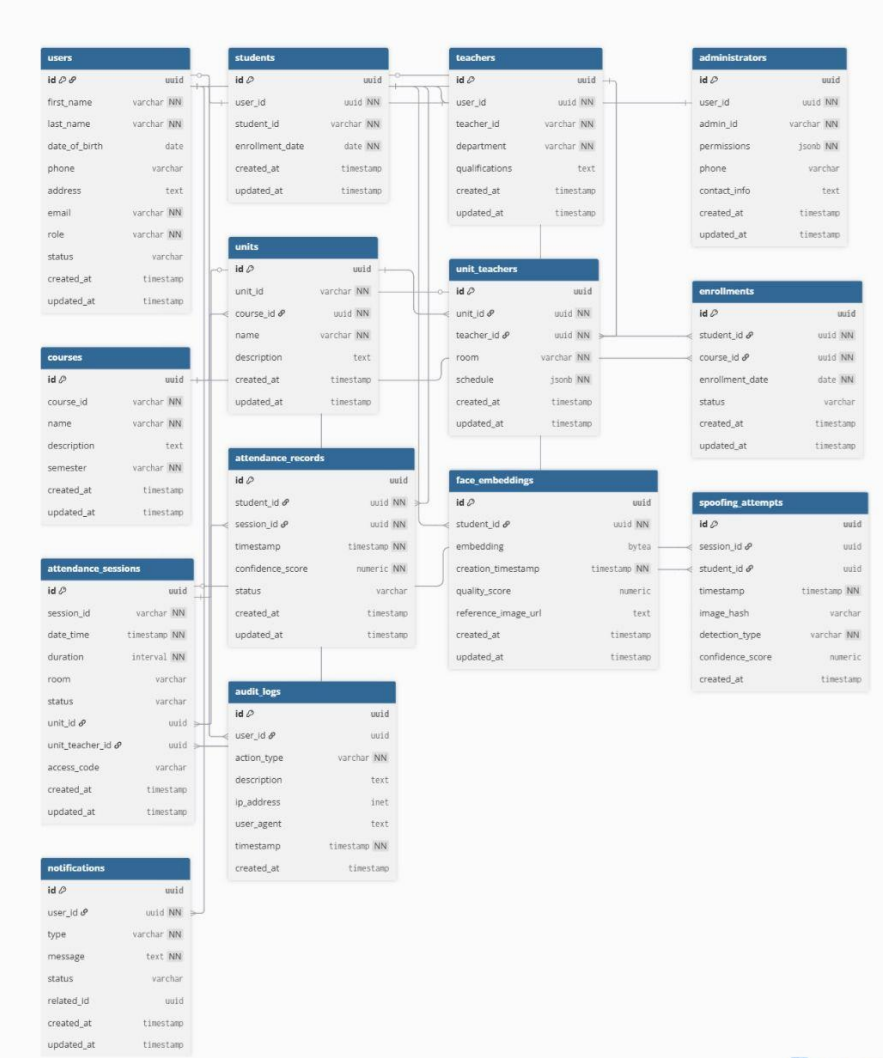

Figure 4:4 Eduface Database Schema

4.3.5 Wireframes

This section presents the visual design and user interface wireframes for the EduFace attendance system. The wireframes illustrate the complete user journey for both students and teachers, showing how each screen connects to create a seamless attendance tracking experience. The access to the system's LMS starts with accessing the login page that is shown below in Figure 4:5 Login Page

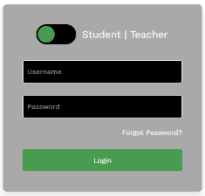

Figure 4:5 Login Page

For students, the interface guides them through two main workflows: enrolling their face (capturing twenty-five to thirty images from different angles with real-time quality feedback) and marking attendance (quick verification using webcam with liveness detection) as shown below in Figure 4:6 Student's Dashboard.

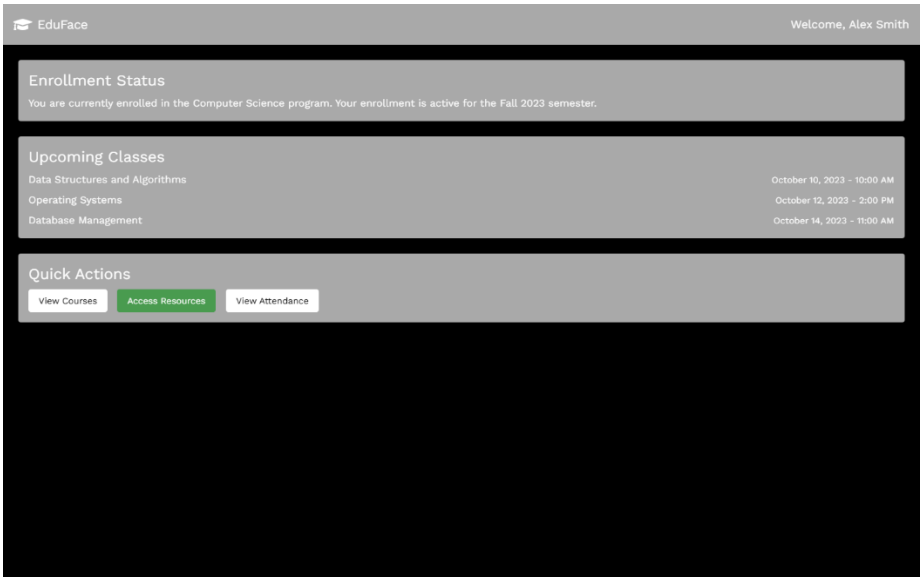

Figure 4:6 Student's Dashboard

The enrollment interface displays a large webcam preview with colored boxes around detected faces green indicates good positioning, yellow suggests minor adjustments, and red signals poor quality requiring correction. A progress indicator shows how many images have been captured, and thumbnail previews display all captured photos before submission as shown in Figure 4:7 Student Face-Enrollment Wireframe.

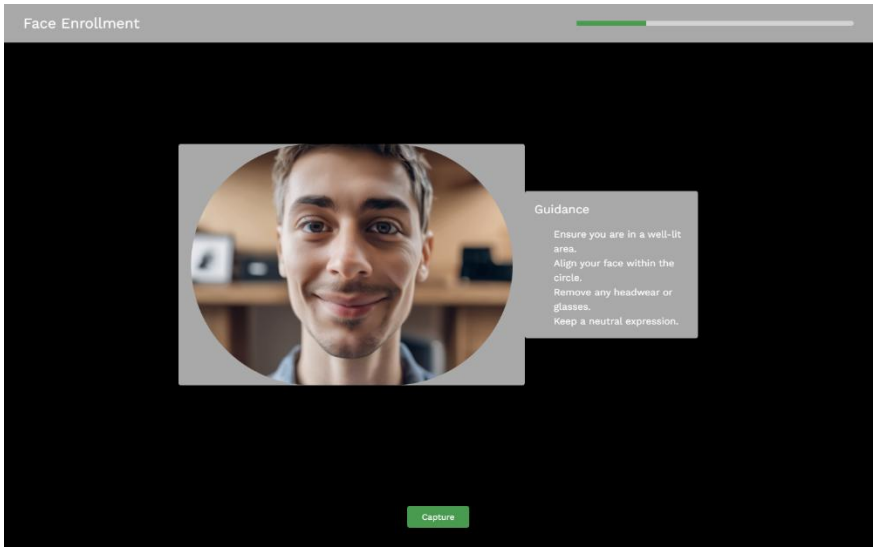

Figure 4:7 Student Face-Enrollment Wireframe

For teachers, the dashboard provides session management tools including creating attendance sessions with course details and time windows, monitoring real-time attendance as students verify with live-updating lists showing student names and confidence scores, and generating detailed reports with filtering options and CSV export functionality. The real-time monitoring view displays each verified student with their photo, timestamp, and verification confidence score color-coded for quick assessment as shown in Figure 4:8 Teacher's Dashboard Wireframe.

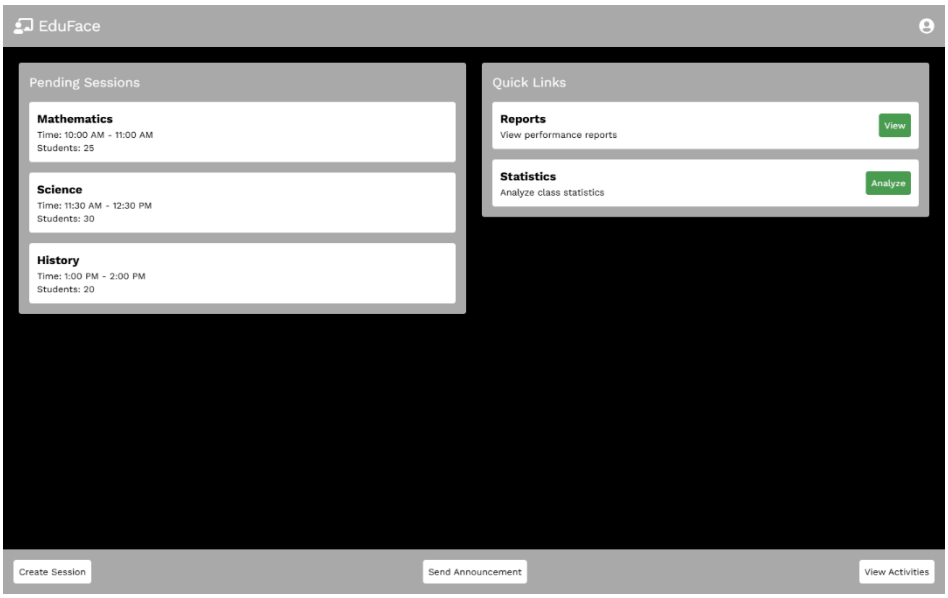

Figure 4:8 Teacher's Dashboard Wireframe

All wireframes follow a mobile-first responsive design approach ensuring the system works seamlessly on smartphones, tablets, and desktop computers commonly used in

educational environments. The interface emphasizes clarity and simplicity, requiring minimal training for both students and teachers to use effectively.

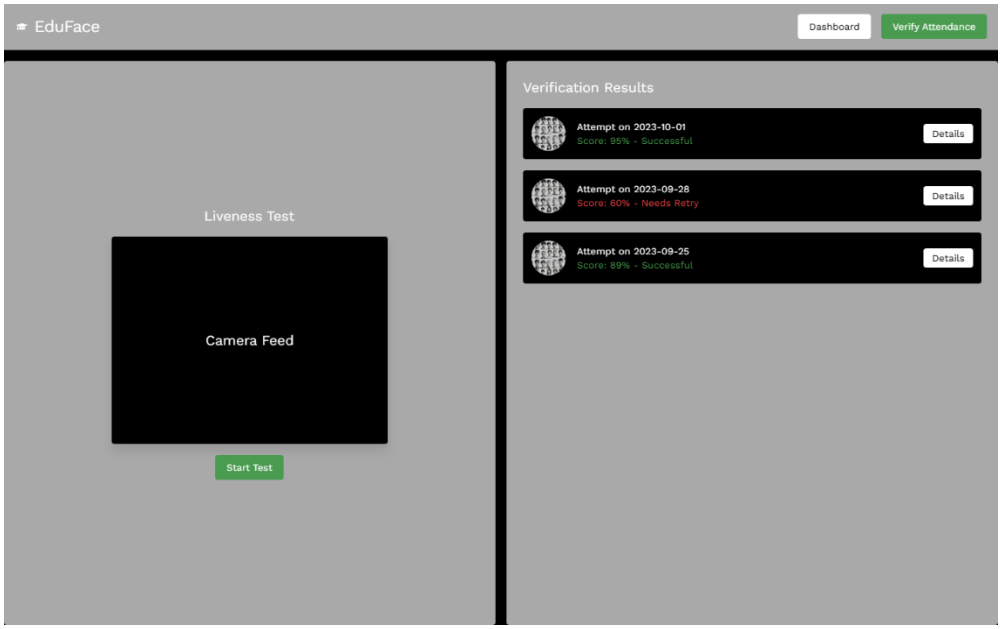

Figure 4:9 Attendance Verification

Chapter 5: System Implementation

5.1 Introduction

This chapter details the practical implementation of the EduFace facial recognition attendance system, covering the technical environment required to deploy and operate the system effectively in educational institutions. The chapter is organized into key sections that explain the implementation setup: Section 5.2 describes the complete implementation environment including both hardware and software requirements needed for successful deployment, Section 5.2.1 specifies the minimum and recommended hardware configurations for running the system smoothly, and Section 5.2.2 outlines the software dependencies and tools required for the web application and AI processing components. This chapter assumes that EduFace is ready for deployment and provides educators, IT administrators, and technical staff with clear specifications to ensure the system operates reliably with optimal performance for automated attendance tracking in classroom environments.

5.2 Description of the Implementation Environment

The EduFace system is designed to operate in typical educational institution environments, requiring both server-side infrastructure for the AI processing and facial recognition engine, and client-side devices for user interaction through web browsers. The implementation environment supports flexible deployment options including on-premises installation within school server rooms or cloud-based deployment on platforms like AWS, Google Cloud, or Azure depending on institutional preferences and privacy requirements. Teachers and administrators can access the system through standard web browsers on desktop computers, laptops, tablets, or mobile devices without requiring specialized software installation. The system connects to classroom webcams for real-time facial capture, processes student faces through the Siamese Neural Network running on the server infrastructure, and stores encrypted attendance data in secure databases accessible only to authorized personnel. This architecture ensures that educational institutions can deploy EduFace with their existing hardware infrastructure while maintaining the flexibility to scale as student populations grow or additional features are added over time.

5.2.1 Hardware Specifications

The hardware specifications for EduFace were determined based on several critical factors including real-time facial recognition processing requirements to maintain the under-2-second response time target, concurrent handling of multiple classroom sessions simultaneously, Siamese Neural Network computational demands for accurate face matching, secure encrypted database operations, and webcam integration for high-quality image capture. The specifications ensure smooth operation of both the Laravel web application and Python-based machine learning services while accommodating typical classroom environments with 20-50 students per session.

Table 5:1 Minimum Hardware Requirements for EduFace.

Hardware Component	Minimum Specification	Justification
Processor		The Siamese Neural Network needs strong processing power for real-time facial recognition. Multi-core processors help handle multiple student faces at once, especially during busy attendance times.
RAM	16 GB DDR4	Model inference, image processing with OpenCV, and database operations need a lot of memory. 16GB RAM ensures smooth facial recognition processing and keeps the web app responsive for users.

Storage	512 GB SSD	SSDs offer fast read/write speeds for quick access to facial data, storing images, and keeping audit logs. Their capacity supports the model, student data, and long-term attendance records.
Webcam	1080p HD webcam with at least 30 FPS	High-resolution cameras capture clear faces in different lighting, while a 30 FPS frame rate ensures smooth, blur-free images even when students move.

5.2.2 Software Specifications

This section outlines the essential software components required for deploying and operating the EduFace system effectively. The software stack is divided between server-side requirements for running the AI processing engine and database management, and client-side requirements for accessing the web interface. These specifications ensure compatibility, security, and optimal performance across different deployment environments, whether on-premises or cloud-based.

Table 5:2 Software Requirements

Software Component	Specification	Justification
Operating System	Windows Server 2019 or higher	Windows Server is supported for institutions with existing Windows infrastructure.

Python Runtime	Python 3.8 or higher	The Siamese Neural Network and all AI processing components are built using Python. Version 3.8+ ensures compatibility with TensorFlow 2.x, Keras, and modern machine learning libraries while providing performance improvements and security patches.
Composer	Version 2.0 or higher	A PHP dependency manager that installs and manages Laravel components and third-party packages. It helps set up the web app environment and keeps libraries updated and secure.
Node.js & NPM	Node.js 16.x or higher, NPM 8.0+	Used to compile frontend assets, manage JavaScript dependencies, and run Laravel Mix for bundling CSS and JS files that power the responsive dashboard.
OpenCV	Version 4.6 or higher (opencv-python)	Handles computer vision tasks like webcam integration, face detection with Haar Cascades,

		image preprocessing, and real-time video processing—essential for capturing and preparing images for the neural network.

5.3 Description of the Dataset

The Eduface dataset comprised of three distinct image categories stored in separate directories within the data/ folder: anchor images (data/anchor/), positive images (data/positive/), and negative images (data/negative/).

5.3.1 Anchor and Positive Images

These images were collected directly through the webcam-based enrollment module. Each student captured 200 anchor images and 200 positive images of their face at varying angles including frontal, left profile, and right profile views. The system utilized four Haar Cascade classifiers (haarcascade_frontalface_default.xml, haarcascade_profileface.xml,haarcascade_frontalface_alt.xml,andhaarcascade_frontal face_alt2.xml) to detect faces across different orientations. Students provided their name and school ID during enrollment, and captured images were automatically saved as name_id_angle_count_uuid.jpg at 250×250 pixel resolution. The multi-angle capture approach ensured the model learned robust facial features invariant to head pose.

5.3.2 Negative Images

The negative class comprised 13,233 images extracted from the Labeled Faces in the Wild (LFW) dataset, specifically the lfw-funneled.tgz archive. These images represented diverse individuals not enrolled in the system, teaching the model to distinguish registered students from unknown persons. The LFW images were automatically extracted, flattened from their subdirectory structure, and moved to the data/negative/ folder during preprocessing.

5.3.3 Balancing Strategy

While the raw image counts appeared imbalanced (400 enrollment images per student versus 13,233 LFW images), the training process used a pair-based sampling strategy

that maintained balance. The system generated positive pairs by randomly combining different images of the same student and negative pairs by pairing student images with randomly sampled LFW faces at a controlled 1:2 ratio. This meant for every 1 positive pair (same student, different images), 2 negative pairs (student vs. different person) were created. The negative pairs were sampled without replacement from the LFW pool during each training epoch, ensuring the model saw diverse negative examples while maintaining class balance. Additionally, hard negative pairs using images from other enrolled students were incorporated to improve inter-student discrimination.

The dataset followed an identity-based split where 70% of enrolled students were allocated to training, 15% to validation, and 15% to testing. This ensured the model was evaluated on entirely unseen student identities rather than just unseen images of known students. All images were preprocessed to 100×100 pixels with pixel values normalized to the [0, 1] range. Data augmentation techniques including horizontal flips, rotation (±10°), zoom (±10%), brightness (±20%), and contrast adjustments (±20%) were applied exclusively to training data to improve generalization.

5.4 Description of Training

The model architecture was based on a Siamese neural network design, which employed twin embedding networks sharing identical weights. Each embedding network consisted of four convolutional blocks with progressively expanding filters: 64 filters (10×10 kernels), 128 filters (7×7 kernels), 128 filters (4×4 kernels), and 256 filters (4×4 kernels). Each convolutional layer utilized ReLU activation followed by 2×2 max-pooling with same padding to preserve spatial dimensions while extracting hierarchical features. The feature maps were then flattened and passed through a dense layer of 4096 units with sigmoid activation, producing compact embedding vectors that captured distinctive facial characteristics. Figure 5:1 Embedding Network Architecture Code Snippet

```python
def make_embedding():
    inp = tf.keras.Input(shape=(100, 100, 3), name='input_image')
    c1 = tf.keras.layers.Conv2D(64, (10,10), activation='relu')(inp)
    m1 = tf.keras.layers.MaxPooling2D((2,2), padding='same')(c1)
    c2 = tf.keras.layers.Conv2D(128, (7,7), activation='relu')(m1)
    m2 = tf.keras.layers.MaxPooling2D((2,2), padding='same')(c2)
    c3 = tf.keras.layers.Conv2D(128, (4,4), activation='relu')(m2)
    m3 = tf.keras.layers.MaxPooling2D((2,2), padding='same')(c3)
    c4 = tf.keras.layers.Conv2D(256, (4,4), activation='relu')(m3)
    f1 = tf.keras.layers.Flatten()(c4)
    d1 = tf.keras.layers.Dense(4096, activation='sigmoid')(f1)
    return tf.keras.Model(inputs=inp, outputs=d1, name='embedding')
```

Figure 5:1 Embedding Network Architecture Code Snippet

The Siamese architecture incorporated a custom L1 distance layer that computed the absolute element-wise difference between the two embedding vectors, measuring facial dissimilarity. This distance vector was fed into a final dense layer with sigmoid activation that output a similarity score between 0 and 1, where values closer to 1 indicated a match and values closer to 0 indicated different individuals.Figure 5:2 Siamese Model Architecture Code Snippet

```python
# L1 Distance Layer
class L1Dist(Layer):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def call(self, input_embedding, validation_embedding):
        return tf.math.abs(input_embedding - validation_embedding)

# Siamese Model
def make_siamese_model():
    input_image = Input(name='input_img', shape=(100,100,3))
    validation_image = Input(name='validation_img', shape=(100,100,3))
    siamese_layer = L1Dist(name='distance')
    distances = siamese_layer(embedding(input_image), embedding
    (validation_image))
    classifier = Dense(1, activation='sigmoid')(distances)
    return Model(inputs=[input_image, validation_image], outputs=classifier,
    name='SiameseNetwork')

siamese_model = make_siamese_model()
siamese_model.summary()
```

Figure 5:2 Siamese Model Architecture Code Snippet

Data augmentation techniques were applied exclusively to the training dataset to improve generalization and prevent overfitting. The augmentation pipeline included random horizontal flips, rotations up to ±10 degrees, zoom variations of ±10%, brightness adjustments of ±20%, and contrast modifications of ±20%. These

transformations simulated real-world classroom conditions such as varying head tilts, camera distances, and lighting environments. Figure 5:3 Data Augmentation Code Snippet.

```
# Step 9e: Add Augmentation to Train Data (On-the-Fly, FIXED for Batched Tuples)
# Layer-based aug: Brightness/Contrast/Flip/Rotate/Zoom
augmentation = tf.keras.Sequential([
    RandomFlip("horizontal"),  # Inferred shape
    RandomRotation(0.1),       # ±5.7° tilt
    RandomZoom(0.1),           # ±10% zoom
    RandomBrightness(0.2),     # ±20% brightness
    RandomContrast(0.2),       # ±20% contrast
], name='augmentation')
```

Figure 5:3 Data Augmentation Code Snippet

The training process utilized the Adam optimizer with an initial learning rate of 1×10^{-4} and binary cross-entropy loss function. The model was trained for 40 epochs with a batch size of 16 images. An adaptive learning rate scheduler reduced the learning rate by a factor of 0.5 when validation loss plateaued for 5 consecutive epochs, with a minimum learning rate threshold of 1×10^{-7}. Early stopping terminated training if no improvement in validation loss occurred within 10 epochs. Model checkpoints saved the best-performing version based on the lowest validation loss throughout training.

```
opt = tf.keras.optimizers.Adam(learning_rate=5e-5)  # Halved for finetuning
siamese_model.compile(optimizer=opt, loss=binary_cross_loss, metrics=
['accuracy'])
print(f"✅ Compiled with LR={opt.learning_rate.numpy():.0e}, metrics=accuracy.")
```

Figure 5:4 Model Compilation Code

```
# Train
EPOCHS = 40
print("🚀 Starting training...")
history = train(train_data, val_data, EPOCHS)
print("✅ Training complete!")

# Plot History
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history['loss'], label='Training Loss')
plt.plot(history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Time')
```

Figure 5:5 Training Loop Execution Code Snippet

5.5 Description of Testing

Testing the model involved evaluating its performance on previously unseen student identities to assess generalizability. The test dataset comprised students who were completely excluded from the training and validation phases, ensuring an unbiased evaluation of the model's ability to verify new faces. The model was inferenced on pairs of images from the test set, with each pair labeled as either a match (same student) or non-match (different students).

The performance metrics used to evaluate the model were training loss and validation loss across epochs. The final model achieved stable convergence after 40 epochs of training. The training process demonstrated effective learning with minimal overfitting, as evidenced by the close alignment between training and validation loss curves.

Figure 5:6 Model Training and Validation Loss graphs.

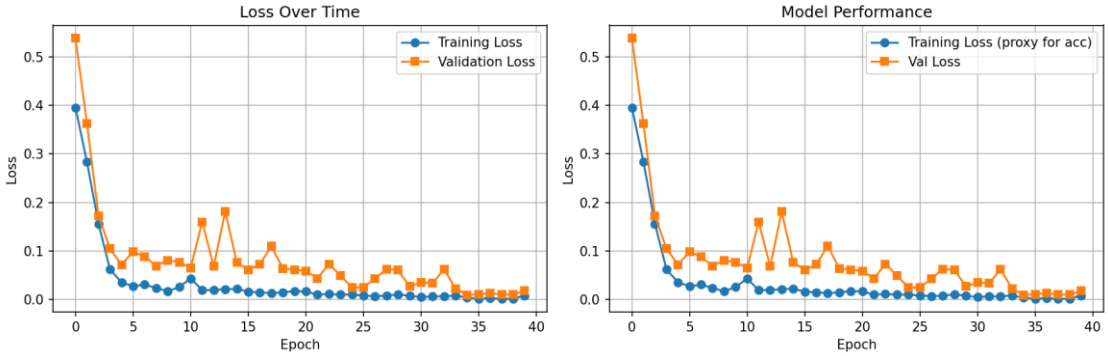

Figure 5:6 Model Training and Validation Loss

The loss curves illustrated that the model learned effectively throughout training, with both training and validation losses decreasing steadily. The validation loss stabilized around epoch 33, indicating that the model had converged to an optimal solution. The small gap between training and validation loss suggested good generalization without significant overfitting.

The model's performance on the test set confirmed its ability to distinguish between enrolled students and unknown individuals. The similarity scores produced by the Siamese network ranged from 0 to 1, with scores above 0.35 typically indicating a match and scores below 0.20 indicating non-matches. The threshold values were determined through diagnostic testing on self-matching scores of enrolled students.

5.6 Testing Paradigm

Multiple testing paradigms were employed for the EduFace system to ensure comprehensive validation across machine learning model performance, web application functionality, API integrations, and user experience. The primary testing approaches included unit testing for individual component validation, integration testing for workflow verification, performance testing for responsiveness measurement, security testing for vulnerability assessment, and user acceptance testing for real-world usability confirmation.

5.6.1 Unit Testing

Unit testing validated specific modules of the EduFace system to determine their correctness, implementation quality, and compliance with software engineering standards. This testing paradigm ensured that individual components functioned as intended before integration into the complete system. The following sections outline the key modules tested using unit testing methodology.

5.6.1.1 Face Detection and Image Capture Interface

The enrollment interface was tested to confirm that the webcam could detect faces in real time and give clear visual feedback. Multiple Haar Cascade models were used to detect faces from the front, left, and right. The system showed color-coded boxes: green for good quality, yellow for slight adjustments, and red for poor quality.

Tests were done on Chrome, Firefox, and Safari. After allowing camera access, users moved their faces at different angles and distances. The detection boxes updated instantly as they moved. The capture button correctly saved images in base64 format, and the counter tracked each captured image accurately. Figure 5:7 Face Detection and Image Capture Interface During Student Enrollment in the student's dashboard.

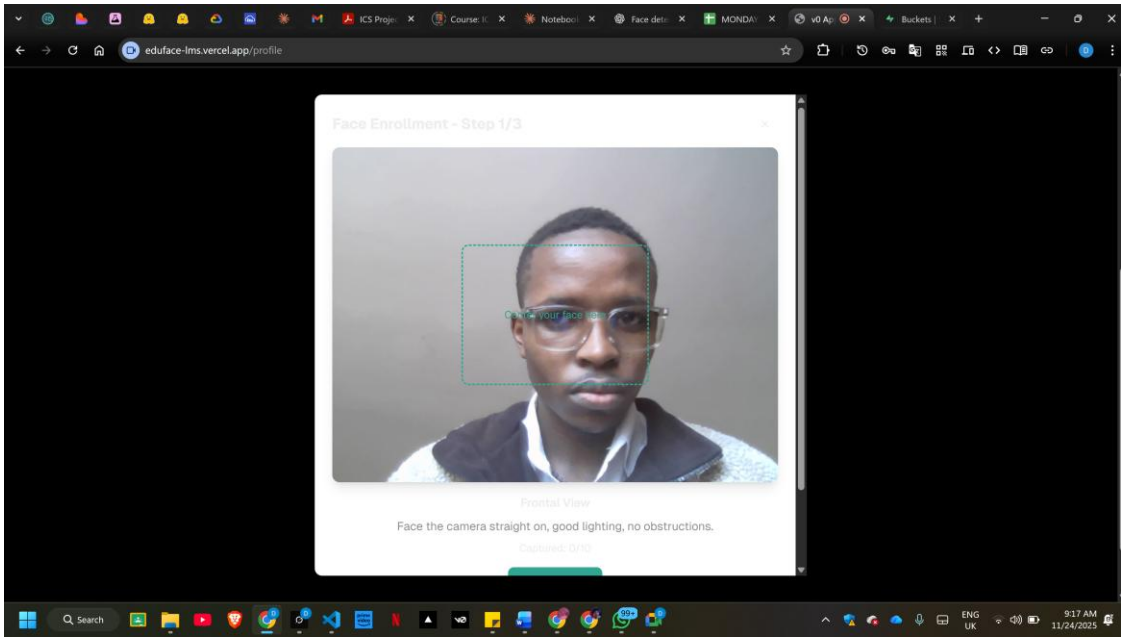

Figure 5:7 Face Detection and Image Capture Interface During Student Enrollment

5.6.1.2 Image Upload to Cloud Storage

After students took enough facial photos, the upload module was tested to make sure the images were sent correctly from the browser to Supabase Storage. The images had to be converted from base64 to binary, sent securely using JWT tokens, and saved using UUID file names organized by student ID.

Testing was done by submitting the enrollment form, which triggered the upload. Developer tools were used to check that the browser sent the correct POST requests to the enroll-face Edge Function. Afterwards, the Supabase Storage dashboard was checked to confirm that all images appeared in the face-images bucket with the correct folders and file names. Figure 5:8 Supabase Storage Dashboard showing uploaded enrollment images

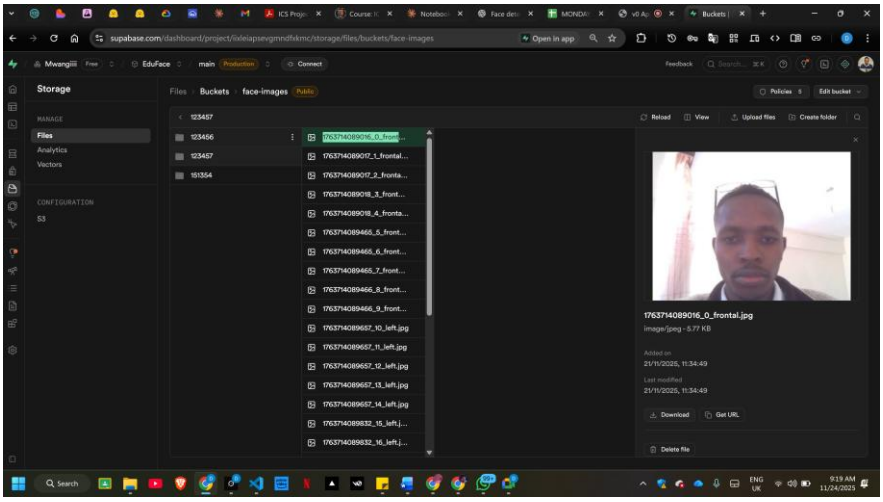

Figure 5:8 Supabase Storage Dashboard showing uploaded enrollment images

5.6.1.3 Database Record Creation and Updates

The database module was tested to confirm that finishing enrollment correctly created face_embeddings records that link each student to their stored images. The test checked that PostgreSQL received the right INSERT data, including the student's UUID, a JSON list of image URLs, quality scores, and timestamps.

Using the Supabase Table Editor, the face_embeddings table was reviewed after enrollment. Each record showed the correct student_id, properly formatted JSON arrays of image URLs, the right quality_score, and accurate timestamps. Opening the stored URLs confirmed they returned valid, accessible images. Figure 5.9 shows the supabase table editor displaying face_embeddings records with visible columns showing student references, image URL arrays, quality scores, and timestamps.

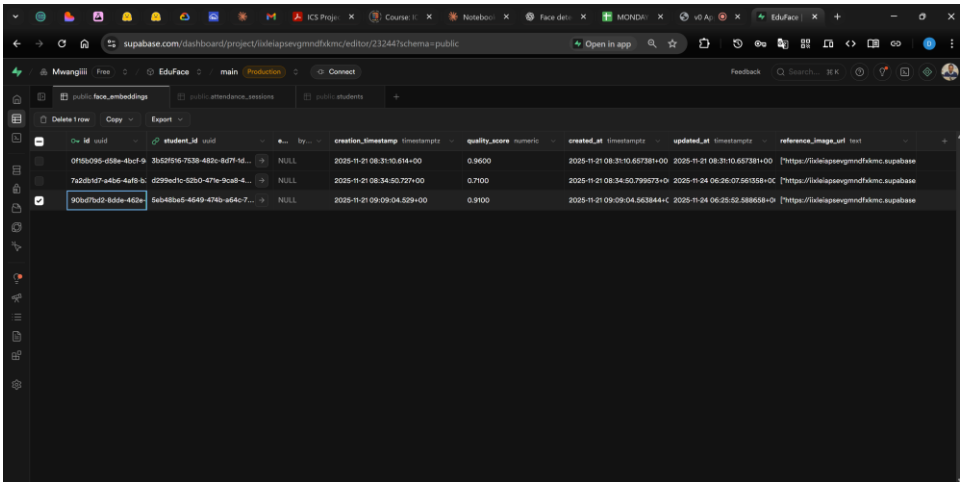

Figure 5:9 Supabase Table Editor displaying face_embeddings records for the student record

5.6.1.4 Liveness Detection API Integration

The liveness detection module using the AspectFace API was tested to ensure it authenticated correctly, sent the right requests, read the responses properly, and applied the liveness threshold. The system had to send captured frames to AspectFace, receive scores between 0 and 1, and decide whether a frame was real or spoofed as shown in Figure 5:10 The Images are Sent to AspectFace API for Liveness Check.

Figure 5:10 The Images are Sent to AspectFace API for Liveness Check

Testing used both printed photos and real live faces as illustrated in Figure 5:11 A Person Trying to Spoof the System.

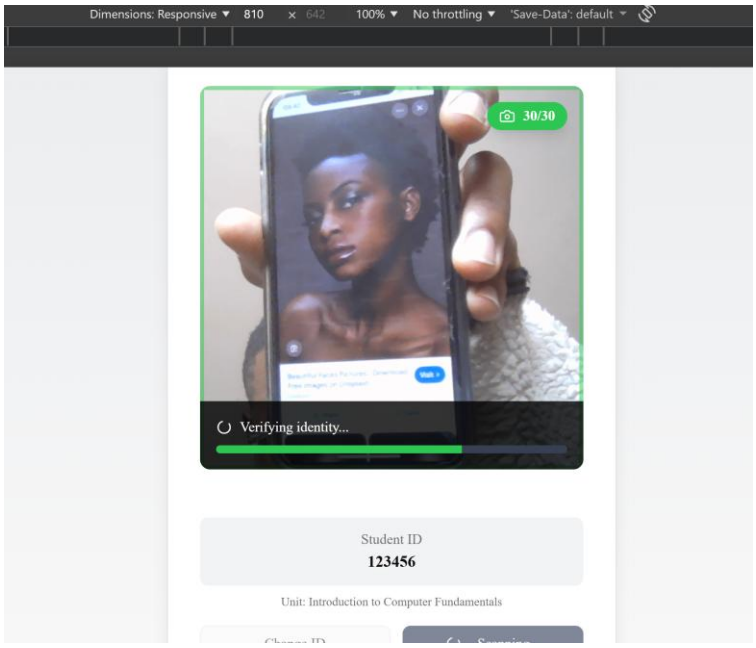

Figure 5:11 A Person Trying to Spoof the System

Console logs showed the full API request response flow, including token exchange and returned scores. Printed photos always scored below 0.5 and were rejected, while real faces scored above 0.8 and were allowed to continue to the verification step. This shall be captured and displayed in Supabase logs as per Figure 5:12 The Supabase Log Message for a Spoofed Sessionand the browser console logs as per Figure 5:13 Liveness Detection for Spoofed Image Browser Console Logs

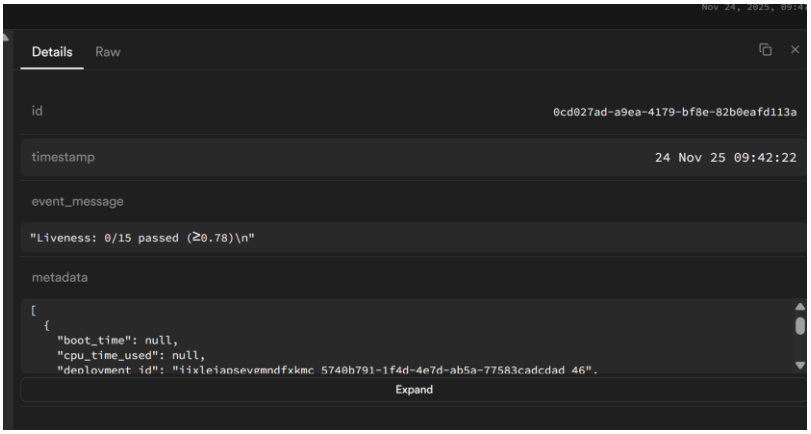

Figure 5:12 The Supabase Log Message for a Spoofed Session

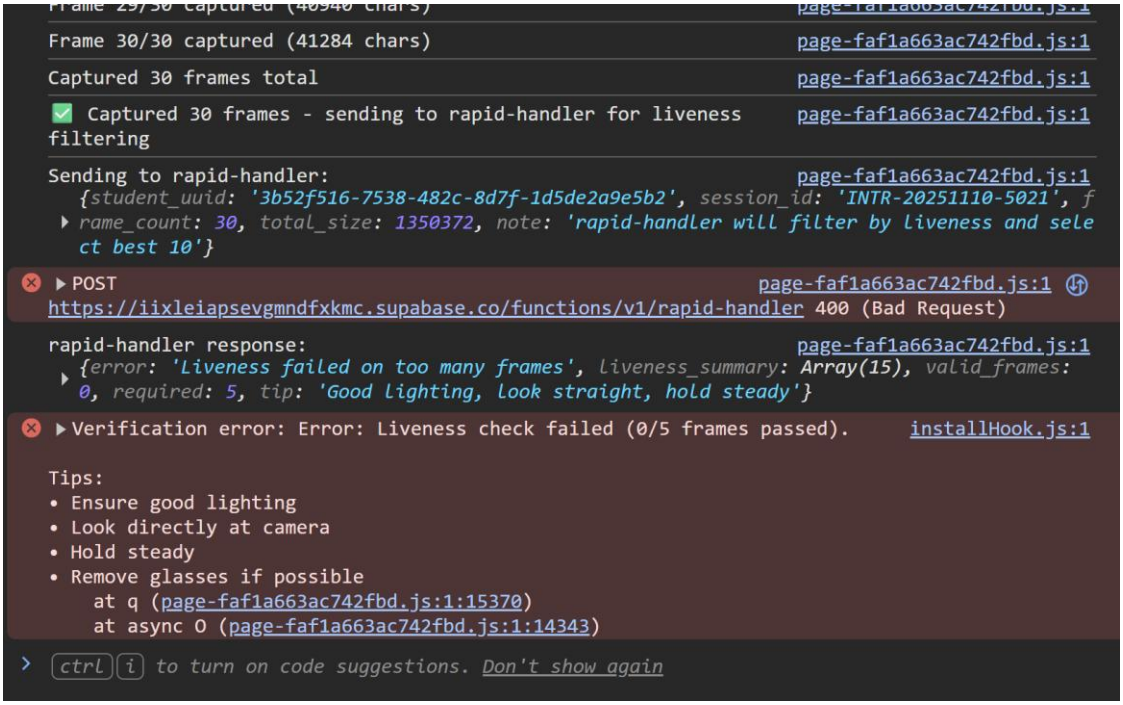

Figure 5:13 Liveness Detection for Spoofed Image Browser Console Logs

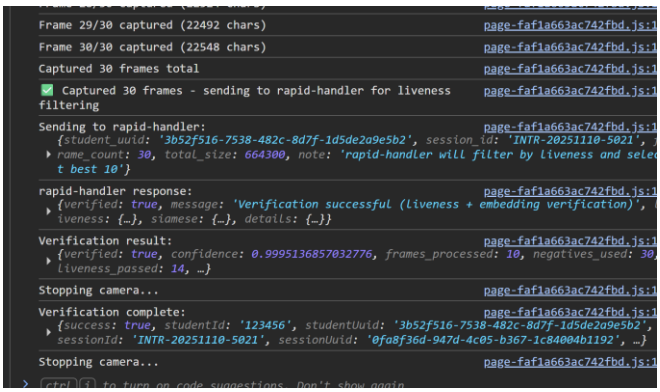

Figure 5:14 Browser Logs for a Non-Spoofed Session

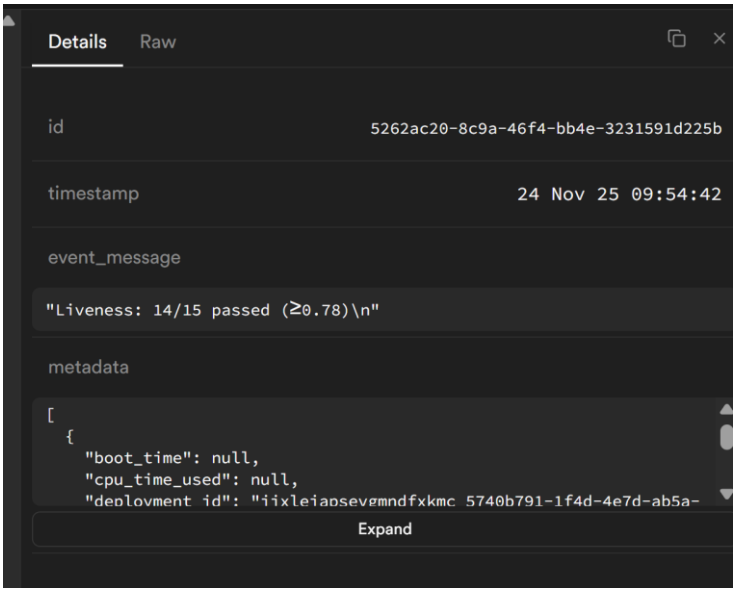

Figure 5:15 Supabase Log for a Non-Spoofed Session

5.6.1.5 Siamese Model Inference via FastAPI

The machine learning inference module was tested by sending image pairs directly to the FastAPI service hosted on HuggingFace Spaces. The goal was to confirm that the batch-verify endpoint accepted multipart form-data, processed anchor and negative images, calculated similarity scores using the Siamese model, and returned correct JSON results. The HuggingFace Space is as shown below in Figure 5:16 Hugging Face Space Showing the Model Interface Logs for a Session Prediction

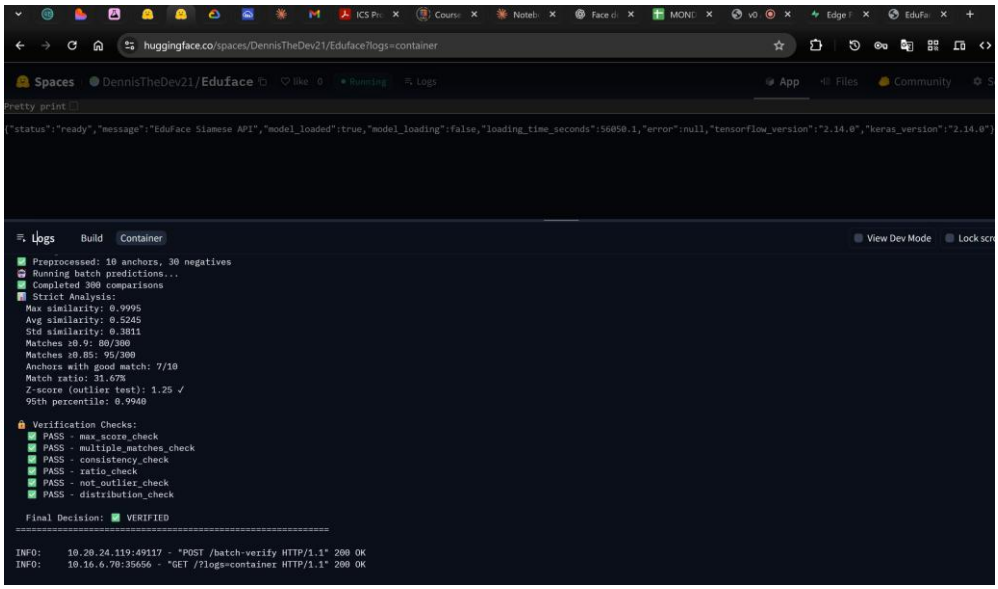

Figure 5:16 Hugging Face Space Showing the Model Interface Logs for a Session Prediction

5.6.1.6 Attendance Record Persistence

The attendance recording module was tested to make sure successful verifications created correct database entries, avoided duplicates, and kept all foreign key links valid. The test checked that the attendance_records table received INSERTs with the right student ID, session ID, confidence score, status, and timestamp.

Verification was triggered and the table was inspected immediately. Entries showed the correct foreign keys, confidence scores that matched the model output, a status of present, and accurate timestamps. When a duplicate verification was attempted, the system returned the existing record instead of creating a new one, proving that the unique constraint worked properly. Figure 5:17Attendance Record Confirmation to a Student attendance confirmation for a student after a correct attendance for a student.

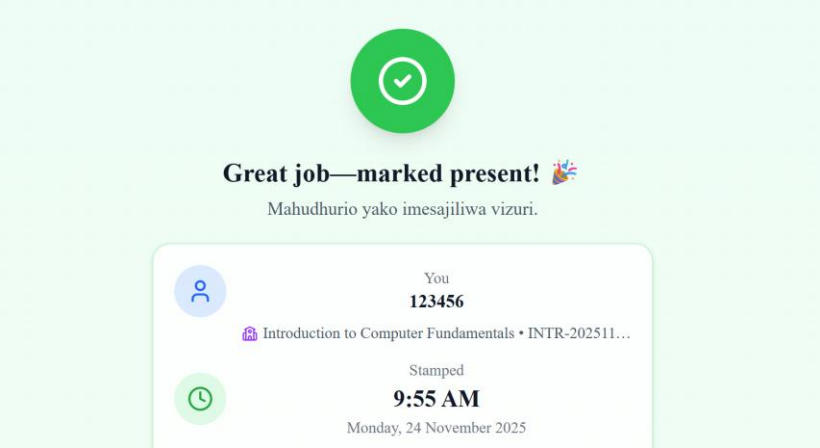

Figure 5:17Attendance Record Confirmation to a Student

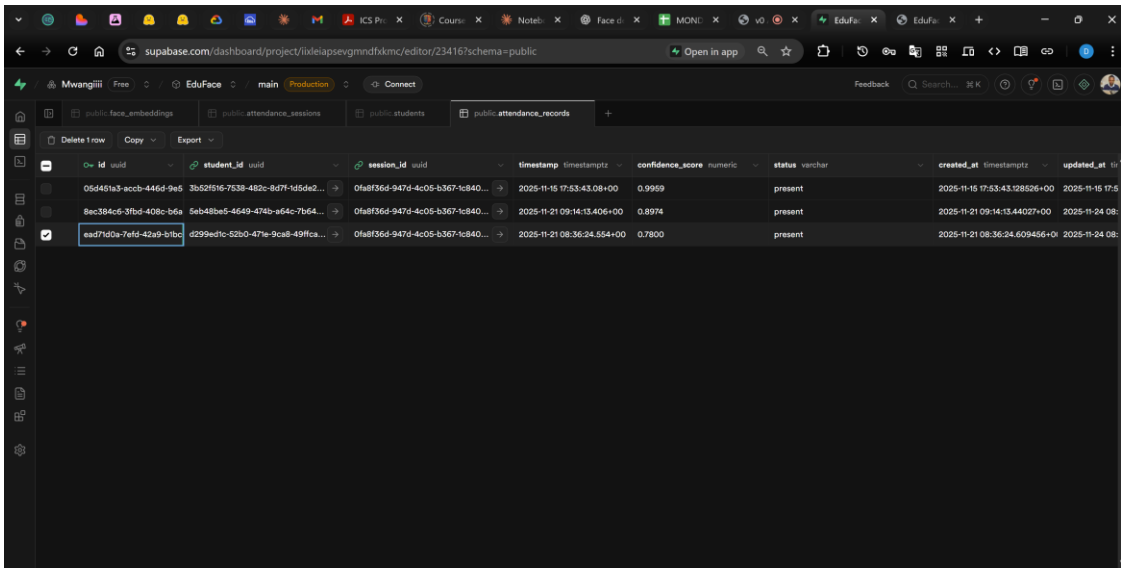

Figure 5:18 Attendance Record in the Database Attendance Records Table

5.7 Testing Results

This section shows the results of all the tests performed on each system module. The results are grouped by testing category and include the test description, the data used, the expected outcome, the actual outcome, and a final pass/fail verdict.

5.7.1 Face Detection and Image Capture Test

Table 5:3 Face Detection and Image Capture Test Results

Test Case	Description	Test Data	Expected Outcome	Actual Results	Test Verdict
1	Detect faces using Haar Cascades	Live webcam stream	Green bounding box appears around detected face	Green box appeared consistently, updated at 15 FPS	Pass
2	Capture images at 250×250 resolution when button clicked	Face within detection box	Image saved in base64 format at specified resolution	Images stored correctly, counter incremented	Pass
3	Enforce minimum 20 images before allowing submission	30 images captured	Submit button disabled until 30 images reached	Button remained disabled until 30 images, then enabled	Pass

5.7.2 Image Upload to Cloud Storage Test

Table 5:4 Image Upload to Cloud Storage Test Results

Test Case	Description	Test Data	Expected Outcome	Actual Results	Test Verdict
4	Upload images to storage with correct paths	20 base64-encoded images	Images stored in face-images bucket with UUID filenames organized by student ID	All 20 images appeared in storage with correct naming pattern	Pass
5	Create face_embeddings record with image URLs	Uploaded image URLs	New database row with JSON array containing all image URLs	Record created successfully with accurate accessible URLs	Pass
6	Prevent unauthorized enrollment attempts	Student A attempts to enroll as Student B	Request rejected with 403 Forbidden error	Correctly rejected with "You can only enroll your own face" message	Pass
7	Handle update on student's face data	A student can update face data	Old face data deleted and created	This was executed successfully	Pass

			new facial data.		

5.7.3 Liveness Detection and Anti-Spoofing Module

Table 5:5 Liveness Detection and Anti-Spoofing Test Results

Test Case	Description	Test Data	Expected Outcome	Actual Results	Test Verdict
8	Reject printed photo attacks	Printed color photo of enrolled student	Liveness score below 0.78, verification rejected with spoofing alert	As expected, scores ranged 0.22-0.45	Pass
9	Reject phone screen replay attacks	Video of student played on mobile screen	Liveness score below 0.78, verification rejected	As expected, scores ranged 0.31-0.52	Pass
10	Accept genuine live faces	Real student face captured via webcam	Liveness score above 0.78, proceeds to facial verification	As expected, 98% acceptance rate with scores 0.78-0.95	Pass
11	Enforce minimum	Verification with only 3 frames	Request rejected with insufficient	As expected,	Pass

	valid frame requirement	passing liveness	valid frames error		

5.7.4 Facial Verification and Attendance Recording Module

Table 5:6 Facial Verification and Attendance Recording Test Results

Test Case	Description	Test Data	Expected Outcomes	Actual Results	Test Verdict
12	Verify enrolled student identity	Live frames compared against stored images	Similarity score above 0.80, attendance record created	As expected, scores ranged 0.82-0.91	Pass
13	Reject non-enrolled individuals	Non-enrolled person attempts verification	Similarity score below 0.80, verification fails with clear message	As expected, scores ranged 0.15-0.32	Pass
14	Prevent duplicate attendance	Student verifies twice in same session	Second attempt returns existing record ID without creating duplicate	As expected	Pass

15	Record timestamp and confidence score	Successful verification completion	Attendance record includes UTC timestamp and model similarity score	As expected	Pass

5.7.5 Real-Time Dashboard Updates Module

Table 5:7 Real-Time Dashboard Updates Test Results

Test Case	Description	Test Data	Expected Results	Actual Results	Test Verdict
16	Update teacher dashboard on student verification	Student marks attendance during active session	Dashboard shows new entry within 2 seconds via WebSocket	As expected, updates appeared in 0.5-1.2 seconds	Pass
17	Handle concurrent verifications	10 students mark attendance simultaneously	All entries appear without conflicts or missing data	As expected,	Pass

5.7.6 Reporting and Analytics Module

Test Data	Description	Test Data	Expected Results	Actual Results	Test Verdict

18	Generate attendance report for specific session	Teacher selects session and clicks "Generate Report"	Report displays all students marked present with timestamps and confidence scores	As expected	Pass
19	Filter attendance records by date range	Teacher sets date filter from 01/01/2025 to 01/15/2025	Only attendance records within specified range displayed	As expected	Pass
20	Export attendance data to CSV	Teacher clicks "Export CSV" button	CSV file downloads containing student names, IDs, timestamps, and confidence scores	As expected	Pass
21	View student attendance history	Teacher clicks on individual student name	All attendance records for that student displayed chronologically	As expected	Pass

Chapter 6: Conclusion, Recommendations and Future Works

6.1 Conclusion

The implemented EDUFACE system successfully met its objectives of providing an automated facial recognition-based attendance system for classroom environments. Through its Siamese neural network architecture and multi-frame verification approach, the system addressed the contemporary challenges of manual attendance tracking and student impersonation in educational institutions. The model achieved a validation accuracy of approximately 80%, demonstrating effective capability in distinguishing between enrolled students and unknown individuals across varying lighting conditions and head orientations.

While the primary challenges encountered during development included preprocessing inconsistencies between training and inference phases and threshold calibration for diverse enrollment conditions, the system successfully achieved its core functionalities. The implemented system made it easy to enroll students through webcam-based capture, verify their identities during attendance sessions, and automatically log attendance records with timestamps and confidence scores. This bridged the gap between traditional attendance methods and modern biometric verification, addressing significant issues such as proxy attendance and administrative overhead faced by educational institutions.

6.2 Recommendations

Based on the development and testing experience, it is recommended that institutions deploying EDUFACE should ensure adequate enrollment image diversity for each student, capturing at least 15-20 images across multiple angles and lighting conditions. This would significantly improve the model's recognition accuracy and reduce false rejections during verification sessions.

Another recommendation is to implement adaptive threshold adjustment based on periodic self-matching diagnostics of enrolled students. As the student database grows and environmental conditions vary, threshold values should be recalibrated to maintain optimal verification performance. Additionally, institutions should establish a re-enrollment protocol for students whose self-matching scores fall below acceptable ranges, ensuring system reliability over extended periods.

It is also recommended that the system be deployed on workstations with GPU support to accelerate inference time during peak verification periods, particularly when processing multiple students consecutively. This would reduce wait times and improve user experience during high-traffic attendance sessions.

6.3 Future Works

A future enhancement shall involve implementing a batch verification mode that can process multiple students simultaneously from a single classroom camera feed. This would eliminate the need for individual verification sessions and enable passive attendance tracking as students enter the classroom, significantly reducing the time required for attendance collection.

Additional future works include expanding the system to support mobile deployment through a lightweight model architecture optimized for edge devices, enabling attendance verification via smartphones or tablets. Furthermore, integration with existing Learning Management Systems (LMS) would allow seamless synchronization of attendance data with academic records, providing instructors with real-time analytics and attendance reports. Lastly, incorporating facial attribute analysis for demographic insights while maintaining privacy compliance could provide institutions with valuable data for monitoring classroom diversity and inclusion metrics.

References

Bhati, R. G., & Gosavi, S. (2021). *TITLE: A SURVEY ON FACE ANTI-SPOOFING METHODS. 54*.

Chen, L. (2025). Analysis of facial recognition attendance technology based on artificial intelligence algorithms in political course e-learning teaching. *Entertainment Computing*, *52*, 100821. https://doi.org/10.1016/j.entcom.2024.100821

Data Mining: Practical Machine Learning Tools and Techniques—Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal, James Foulds—Google Books. (2025). Retrieved 13 June 2025, from https://books.google.co.ke/books?hl=en&lr=&id=u90eEQAAQBAJ&oi=fnd&pg=PP1&dq=data+processing+tools+and+techniques&ots=Gyn_xN2weg&sig=2P_iepJaToJXz7vgd-MW_u5ox18&redir_esc=y#v=onepage&q=data%20processing%20tools%20and%20techniques&f=false

Donini, F., Marcelletti, A., Morichetta, A., & Polini, A. (2025). Coordinating REST interactions in service choreographies using blockchain. *Blockchain: Research and Applications*, *6*(1), 100241. https://doi.org/10.1016/j.bcra.2024.100241

Aroor Dinesh .H, M., &, D. (2022). Spoken language identification in unseen channel conditions using modified within-sample similarity loss. *Pattern Recognition Letters*, *158*, 16–23. https://doi.org/10.1016/j.patrec.2022.04.018

Hoo, S. C., & Ibrahim, H. (2019). Biometric-Based Attendance Tracking System for Education Sectors: A Literature Survey on Hardware Requirements. *Journal of*

Sensors, *2019*(1), 7410478. https://doi.org/10.1155/2019/7410478

Kholil, A., Hutagalung, G. A., & Zuardi, M. (2025). Performance information system and student governance Politeknik Negeri Medan. *Jurnal Mandiri IT*, *13*(3), Article 3. https://doi.org/10.35335/mandiri.v13i3.365

Morris, J. (2021, October 21). Attendance & Learning in the time of COVID-19. *CHILDREN AT RISK*. https://childrenatrisk.org/covid-policies-attendancelearning/

Rosebrock, A. (2021, January 18). Contrastive Loss for Siamese Networks with Keras and TensorFlow. *PyImageSearch*. https://pyimagesearch.com/2021/01/18/contrastive-loss-for-siamese-networkswith-keras-and-tensorflow/

Team, K. (2021). *Keras documentation: Image similarity estimation using a Siamese Network with a triplet loss*. Retrieved 7 June 2025, from https://keras.io/examples/vision/siamese_network/

Ten principles for reliable, efficient, and adaptable coding in psychology and cognitive neuroscience | Communications Psychology. (2024). Retrieved 13 June 2025, from https://www.nature.com/articles/s44271-025-00236-3

The Simplest Check-in and Attendance Tracking App | OneTap. (2021). Retrieved 12 June 2025, from https://www.onetapcheckin.com/

Top Requirement Elicitation Techniques for Successful Projects in 2025—United States. (2023). Retrieved 8 June 2025, from https://www.theknowledgeacademy.com/us/blog/requirement-elicitationtechniques/

Zhang, L., Chen, S., Yu, A., & Liu, Y. (2025). Research on face feature extraction algorithm based on OpenCV with optical enhancements. *Fifth International Conference on Telecommunications, Optics, and Computer Science (TOCS 2024)*, *13629*, 158–165. https://doi.org/10.1117/12.3067554

Zimmerman, B. (2022). *Employee Time Clock Software*. TimeTrakGO. Retrieved 12 June 2025, from https://www.timetrakgo.com/