ATTENDEASE – FACE RECOGNITION SYSTEM

GROUP MEMBERS

NAME	REGISTRATION NUMBER	SIGNATURE
DAVID WAMBUA	COM/017/22	
JAMES NGANDU	COM/012/22	
JACINTA OMONDI	INF/004/22	
HENRY OUMA	COM/088/22	

Declaration

We, the undersigned, solemnly declare that the project report titled *"Attendease"* is our original work. This project has been undertaken in partial fulfillment of the academic requirements for the University of Eldoret. All information, data, and findings presented in this document are accurate and authentic to the best of our knowledge and have been sourced from reliable and credible references.

This work, in its entirety or part, has not been submitted previously for the attainment of any degree, diploma, certification, or any other academic award at this or any other institution. Proper acknowledgment has been given for all external sources, materials, and references utilized in the development of this project.

Signatures:

1. **Jacinta Omondi**
   UI Designer
   Signature: _____
   Date: _____
2. **James Ngandu**
   Frontend Developer
   Signature: _____
   Date: _____
3. **Henry Ouma**
   Backend Developer
   Signature: _____
   Date: _____
4. **David Wambua**
   ML Engineer
   Signature: _____
   Date: _____

Dedication

We dedicate this project to all students and educators who continuously strive to improve the efficiency of academic processes through technology. This work is a testament to the transformative power of innovation in education and its potential to streamline classroom management and enhance learning experiences.

Acknowledgment

We would like to express our heartfelt gratitude to all those who contributed to the successful completion of this project, *"Attendease."*

First and foremost, we sincerely thank our supervisor, Madam Siele, for her invaluable guidance, constructive feedback, and continuous support throughout the development of this system. Her expertise and insightful suggestions greatly contributed to the successful execution of this project.

We also extend our appreciation to the University of Eldoret for providing us with the necessary resources, a conducive learning environment, and unwavering support as we explored and implemented this innovative solution.

A special thank you goes to our families and friends for their encouragement, patience, and emotional support during this journey. Their belief in us motivated us to persist and overcome challenges, making this project possible.

Finally, we acknowledge and thank the open-source communities and developers behind frameworks and libraries such as Django, TensorFlow, Face_Recognition, Face_API.js, Tailwind CSS, and OpenCV. Their contributions played a crucial role in making this project a reality.

Abstract

The *attendease* is a technological solution aimed at automating student attendance tracking in academic institutions. Traditional methods, such as manual roll calls and paper-based signing, are often time-consuming, prone to errors, and vulnerable to fraudulent attendance marking. This project addresses these issues by utilizing computer vision and machine learning technologies, providing an efficient, accurate, and secure alternative through facial recognition.

The system is developed using Django for the backend, HTML, CSS (Tailwind), and JavaScript for the frontend, with TensorFlow, Face_Recognition, and Face_API.js used for facial recognition processing. This allows the system to automatically register students' attendance based on facial identification, eliminating the need for physical sign-in methods. Once a student's face is detected and matched with pre-registered data, the system logs their attendance in the database.

Key features of the system include an automated attendance logging system that detects and records student attendance in real time, ensuring accuracy and efficiency. To prevent proxy attendance, it incorporates secure student authentication using facial recognition technology. The system features a user-friendly interface designed with Tailwind CSS, providing a responsive and modern experience. Additionally, it integrates seamlessly with a database, allowing for efficient storage, retrieval, and reporting of attendance records.

This system aims to improve classroom management, enhance the accuracy of attendance records, and reduce the administrative workload for educators. By implementing AI-powered facial recognition, the system introduces speed, reliability, and security to attendance tracking, making it a practical solution for modern academic institutions.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This chapter introduces the **Attendease**, outlining the problem it addresses, its goals, justification for its development, scope, limitations, and the structure of the entire document.

1.1 Problem Statement and Context

In educational institutions, accurately tracking student attendance is vital for monitoring academic engagement and performance. Traditional methods of attendance tracking, such as manual roll calls, sign-in sheets, and RFID-based systems, are burdened with several challenges:

With advances in artificial intelligence (AI) and computer vision, facial recognition technology offers a promising solution. This project aims to develop an automated system that identifies and records students' presence by recognizing their faces, eliminating the need for manual attendance.

Manual attendance marking is a time-consuming process, particularly in large classrooms, leading to inefficiencies. Additionally, human errors can occur, causing teachers to inadvertently mark attendance incorrectly. Another major challenge is fraudulent attendance, where students sign in on behalf of their absent peers. Furthermore, paper-based records create significant administrative overhead, making it difficult to manage and analyze attendance data effectively.

The system is built using Django for backend development, with HTML, CSS **(**Tailwind**)**, JavaScript, TensorFlow, Face_Recognition, and Face_API.js for face detection and recognition. This combination of AI and machine learning enhances the efficiency and accuracy of attendance management.

1.2 Aims and Objectives

The primary goal of this project is to develop an automated attendance system that utilizes facial recognition to streamline the attendance process, improving efficiency, accuracy, and security compared to traditional methods.

To achieve this aim, the system is designed with several key objectives. First, it will incorporate a face recognition system capable of accurately detecting and recognizing students’ faces in real time. The attendance process will be fully automated, ensuring that attendance is marked instantly upon student recognition. To enhance security and prevent proxy attendance, only registered students will be marked present.

A user-friendly web interface will be implemented, allowing administrators, teachers, and students to interact with the system seamlessly. The backend will be built using Django to securely manage student records, attendance logs, and authentication. Additionally, performance and accuracy will be optimized using machine learning frameworks such as TensorFlow and Face_Recognition.

To support academic institutions in tracking student participation, the system will provide real-time attendance reports and data analytics. Furthermore, it will be designed for scalability and flexibility, making it adaptable for use across different educational institutions. By meeting these objectives, the system will simplify attendance tracking, reduce administrative tasks, and enhance overall classroom management.

1.3 Justification

The development of Attendease is justified by several pressing needs in the education sector.
One major challenge in traditional attendance methods is eliminating manual errors. Since these methods rely heavily on human intervention, mistakes such as incorrect marking or missing records are common. By automating the attendance process, Attendease ensures accurate detection and recording of student attendance, minimizing such errors.

Another significant concern is preventing proxy attendance. In many institutions, students often sign in for their absent peers, undermining the credibility of attendance records. Through the use of facial recognition technology, Attendease ensures that only physically present students are marked present, effectively eliminating this issue and enhancing academic integrity.

Additionally, the system greatly improves efficiency and time management. Manual attendance marking, especially in large classrooms, is time-consuming and can disrupt the flow of lessons. The automated system registers students within seconds, allowing teachers to focus more on instruction rather than administrative tasks.

Security and authentication are also strengthened through facial recognition. Unlike RFID cards or sign-in sheets, which can be easily misplaced, borrowed, or manipulated, facial features provide a unique and secure method of identification. This ensures that attendance records remain tamper-proof and reliable.

Furthermore, the system enables comprehensive data analysis and reporting features. Digital storage of attendance records allows institutions to track participation trends, identify patterns of absenteeism, and generate performance reports, supporting more informed academic decision-making.

The system's scalability and adaptability also make it a valuable solution for diverse educational settings. Attendease can be deployed across multiple departments or campuses and can be integrated with existing Learning Management Systems (LMS) to enhance broader institutional workflows.

Beyond solving institutional challenges, the development of Attendease also contributes to the developer's personal and professional growth. Building the system provided an opportunity to learn new programming languages and technologies such as Django for backend development, Tailwind CSS for modern frontend design, TensorFlow and Face Recognition libraries for AI-based face detection, and Face_API.js for real-time browser-based facial recognition.

Moreover, the project improves upon existing attendance systems that rely on older, less secure methods such as RFID tags and manual entry. By leveraging advancements in artificial intelligence and machine learning, Attendease delivers a more robust, scalable, and future-proof solution for attendance management.

Given these technical, educational, and personal development benefits, Attendease represents a significant advancement over traditional attendance methods, making its development both necessary and impactful.

1.4 Scope

The Attendease is designed for use in educational institutions to improve attendance management. The scope defines the system's functionalities and limitations:

Scope of the System:

1. Target Users:

The Attendease system is designed to serve multiple users within an educational institution, each with specific roles and functionalities. Teachers and instructors use the system to efficiently take attendance and access attendance records, allowing them to monitor student participation and ensure accurate record-keeping. Students have their attendance automatically recorded when their face is recognized, eliminating the need for manual sign-ins and ensuring accuracy in attendance tracking. Administrators oversee student registrations, manage attendance logs, and generate reports for institutional analysis. They play a key role in maintaining the system's efficiency and ensuring compliance with institutional policies. By catering to these user groups, Attendease provides a streamlined and automated solution for attendance management, improving efficiency and reliability in educational institutions.

2. Core Functionalities:

The Attendease system incorporates several core functionalities to streamline attendance management. Face detection and recognition are used to identify and verify students based on their facial features, ensuring accuracy and security. Once a student is recognized, the system performs automated attendance recording, marking their presence instantly without manual intervention. With real-time processing, the system detects student faces during class sessions and updates attendance records immediately, enhancing efficiency. All student details, attendance logs, and reports are securely managed through database management, ensuring data integrity and easy retrieval. A web-based dashboard provides an intuitive user interface for teachers and administrators, allowing them to manage records efficiently. Additionally, the system includes report generation features, enabling administrators to generate attendance reports based on specific criteria for performance tracking and institutional analysis. These functionalities work together to create a seamless and automated attendance management solution.

3. Technologies Used:

The Attendease system is built using a range of technologies to ensure functionality, performance, and security. The backend is powered by Django, a robust Python framework that facilitates secure and scalable web application development. For the frontend, the system utilizes HTML, CSS (with Tailwind for responsive design), and JavaScript to create a user-friendly and interactive interface. For facial recognition, the system employs advanced tools such as TensorFlow, Face_Recognition, Face_API.js, and OpenCV, which enable accurate and real-time face detection and verification. To manage data, the system uses SQLite or PostgreSQL as its database solutions, providing reliable storage for student details, attendance logs, and reports. These technologies work together to provide a seamless, efficient, and secure attendance management system.

4. Deployment Environment:

The Attendease system will be deployed on either local or cloud-based servers, offering flexibility depending on the institution's infrastructure needs. It will be accessible via a standard web browser, ensuring easy access for all users, including teachers, students, and administrators. Additionally, the system is designed to integrate with existing Learning Management Systems (LMS), allowing for enhanced functionality and streamlined workflows if required by the institution. This deployment flexibility ensures that Attendease can be tailored to meet the specific needs of various educational environments.

1.5 Limitations

Despite its advantages, the system has several limitations that may affect its functionality:

Despite its advantages, the Attendease system has several limitations that may affect its functionality. One major challenge is the dependence on camera quality. Low-resolution cameras may struggle to detect faces accurately, especially in large classrooms where students may be further away or partially obscured. This issue can hinder the system's performance and reduce its overall effectiveness.

Another limitation is the impact of lighting conditions on facial recognition accuracy. Poor lighting can significantly affect the system's ability to detect and recognize faces, making it essential for classrooms to maintain optimal lighting for the best performance. The system performs most efficiently in well-lit environments, which may not always be possible in every classroom setting.

Variability in facial appearances is another challenge. Changes such as wearing glasses, growing a beard, or having facial injuries can reduce recognition accuracy, as the system may not always account for such variations. This means that students who undergo noticeable changes may face difficulties with accurate attendance tracking.

The processing speed and performance of the system is also a potential limitation. Real-time face recognition requires substantial computational resources, and lower-end hardware may struggle to handle the demands of continuous face detection. This could result in slower processing speeds and potentially disrupt the real-time functionality of the system, especially in larger settings.

Finally, the internet or network dependency poses a challenge for cloud-based systems that require a stable internet connection for real-time processing. While local systems may not face this issue, they may encounter problems with network configuration or resource limitations, which can affect the overall reliability and efficiency of the system.

1.6 Summary of Chapters

The document is organized into multiple chapters, each focusing on different aspects of the project:

a) Chapter 1: Introduction: Introduces the project, outlining the problem, goals, justification, scope, limitations, and document structure.

b) Chapter 2: Literature Review: Reviews existing attendance systems and the evolution of facial recognition technology.

c) Chapter 3: System Analysis and Design: Discusses system requirements, architecture, and design models.

d) Chapter 4: Methodology: Explains the tools, technologies, and development process used.

e) Chapter 5: Project Management: Covers budgeting, resource management, and risk assessment.

f) Chapter 6: Results and Evaluation: Presents system performance, testing, and feedback.

g) Chapter 7: Conclusion and Recommendations: Summarizes key findings and suggests future improvements.

h) Chapter 8: References: Lists all sources referenced.

i) Chapter 9: Appendix: Contains supporting materials, including code, schemas, and screenshots.

Chapter 2: Literature Review

This chapter provides an in-depth review of existing attendance systems, the evolution of face recognition technology, and the gaps that our system aims to address. It also discusses various platforms and methodologies used in developing biometric attendance systems.

2.1 Origin and Evolution of Attendance Systems

The concept of attendance tracking has been integral to educational practices for centuries, evolving from manual methods to technologically advanced systems. In the early stages, attendance was recorded manually, with teachers marking names on roll calls or written logs. These methods, which dated back to ancient civilizations, were simple but dependent on the teacher's memory and physical presence to ensure accuracy. As educational institutions expanded, especially in the 19th and early 20th centuries, more formalized tools, such as manual sign-in sheets and roll calls, emerged, but they still suffered from inefficiencies and human errors. These methods were particularly time-consuming, especially in larger classrooms where teachers had to call each student’s name.

The first major technological advancement came in the mid-20th century with the introduction of electronic attendance systems. Barcode scanners and RFID (Radio Frequency Identification) technologies, first explored in the 1970s and 1980s, allowed students to swipe an identification card to register their attendance automatically. These systems, while more efficient than manual methods, still faced issues such as proxy attendance, where students could pass cards between each other, a challenge that researchers like Bajaj et al. (2005) and Patel et al. (2007) explored in their studies on RFID and its limitations in educational settings. Despite these limitations, RFID systems became widespread in large educational institutions as a more efficient way to track attendance.

The next significant leap in the evolution of attendance systems occurred with the advent of biometric technologies, particularly fingerprint recognition and iris scanning, which gained traction in the late 1990s and early 2000s. Researchers like Jain et al. (2000) in their work on biometric identification systems recognized the potential of such technologies for educational environments, providing greater security and reducing human error. However, these systems faced challenges related to cost, complexity, and concerns regarding privacy and data security, which were highlighted by Zhao et al. (2003) in their research on biometric data vulnerabilities.

The most recent and transformative development in attendance tracking is the integration of facial recognition technology. Researchers such as Chen et al. (2012) and Liao et al. (2016) advanced the use of

machine learning and artificial intelligence (AI) to improve facial recognition accuracy, paving the way for its adoption in educational settings. These AI-driven systems offer a non-intrusive, hands-free method of identifying students based on their unique facial features. The efficiency and security of facial recognition have been demonstrated in studies by Zhao et al. (2017), who discussed its potential to prevent proxy attendance and reduce the administrative burden associated with traditional methods. The implementation of facial recognition systems addresses many of the issues seen in earlier technologies, providing faster, more accurate, and secure attendance recording.

Over time, attendance systems have increasingly been integrated with Learning Management Systems (LMS), as noted in the work of Black et al. (2018), who explored how such integration could streamline administrative tasks and provide real-time data analysis for institutions. As a result, educational institutions can now automate attendance logging, analyze student participation trends, and generate reports without significant manual effort. Despite challenges like privacy concerns and hardware requirements, the evolution of attendance systems continues to push toward fully integrated, AI-driven solutions that offer improved efficiency, security, and ease of use.

2.1.1 Traditional Attendance Methods

Traditional attendance tracking methods have been widely used in educational settings but are often associated with various challenges. Manual roll calls, one of the most basic forms of attendance, involve teachers calling out student names and marking attendance on paper. While simple, this method is time-consuming, especially in large classes, as it takes up valuable class time. Additionally, it is error-prone, with the possibility of teachers mistakenly marking the wrong student or missing some students altogether. Furthermore, manual roll calls are prone to fraud, as students can call the names of absent peers, marking them as present in an attempt to bypass attendance.

Another traditional method is the use of sign-in sheets, where students manually sign their names to mark their attendance. This system also suffers from forgery and fraud, as students can manipulate the sign-in sheets by signing for absent peers, thus compromising the accuracy of attendance records.

The introduction of RFID (Radio-Frequency Identification) cards was a step towards automation, with students scanning their ID cards using RFID readers to mark their attendance. While this method offers some improvement over manual processes, it still faces challenges such as card misplacement, where students may forget or lose their cards, and fraud risks, as students could use another student’s card to sign in.

Another biometric method that emerged is the use of fingerprint recognition systems. In this system, students scan their fingerprints to mark attendance. While this method offers a higher level of accuracy and security, it introduces hygiene concerns due to the shared use of fingerprint scanners, which can lead to the spread of germs. Additionally, fingerprint systems may fail to recognize students with skin conditions or cuts, making them less reliable for all students.

2.1.2 Modern AI-Powered Attendance Systems

Advancements in AI and machine learning have led to the development of modern, AI-powered attendance systems, which are more automated, accurate, and secure. One of the most promising innovations is facial recognition systems, such as the one used in this project. These systems offer several advantages over traditional methods, including no physical contact, which helps reduce health risks by eliminating the need for shared devices. Additionally, facial recognition systems offer high efficiency, enabling fast and accurate attendance tracking, and they provide improved security, as they can verify the identity of students and prevent fraud.

Comparison of Attendance Systems

Table 1 Comparison of attendance systems

Method	**Accuracy**	**Security**	**Time Efficiency**	**Challenges**	**References**
Manual Roll Call	Low	Low	Slow	Errors, Fraud	Patel et al. (2007), Bajaj et al. (2005)
Sign-in Sheets	Low	Low	Slow	Fraud, Forgery	Zhao et al. (2003)
RFID Cards	Medium	Medium	Medium	Card Loss, Fraud	Jain et al. (2000)
Fingerprint Scan	High	High	Medium	Hygiene Issues	Zhao et al. (2003), Liao et al. (2016)
Face Recognition	Very High	Very High	Fast	Lighting, Accuracy	Chen et al. (2012), Liao et al. (2016), Zhao et al. (2017)

2.2 Existing Solutions and Their Limitation

Several attendance tracking solutions leveraging facial recognition have been developed, but many of them still face significant limitations that hinder their widespread adoption in educational settings.

Commercial face recognition attendance systems are widely used by large enterprises but are often not well-suited for educational institutions due to several drawbacks. One of the main limitations is their high cost, which makes them prohibitively expensive for many educational institutions, especially smaller schools and universities with limited budgets. Additionally, many of these commercial systems suffer from a lack of customization, meaning they often do not integrate well with existing school databases or Learning Management Systems (LMS). This lack of flexibility can make them difficult to adapt to the specific needs of educational institutions. Moreover, privacy concerns are another significant issue with commercial face recognition systems, as some of them store facial data in insecure environments or use poorly regulated data storage methods, creating potential risks for data breaches and non-compliance with data protection laws.

On the other hand, open-source face recognition libraries provide more affordable alternatives but still come with their own set of limitations. One such library is Dlib (Face_Recognition Library), which offers pre-trained models for face detection and recognition. While it is a powerful tool widely used in Python-based applications, it only provides face recognition capabilities and lacks a complete attendance tracking system. Developers must build custom solutions around it, which can be time-consuming and complex. Similarly, OpenCV, a well-known computer vision library, provides real-time face detection and tracking. However, it requires manual tuning for optimal accuracy, which means that developers must have in-depth technical knowledge to fine-tune the system to meet the specific needs of an institution. Finally, Face_API, a JavaScript library designed for use in web applications, can be a useful tool for facial recognition in web-based systems, but like the others, it still requires developers to build a comprehensive attendance solution from scratch, making it less user-friendly for educational institutions without dedicated IT resources.

In summary, while existing solutions offer valuable capabilities for facial recognition, they are often hampered by issues related to cost, customization, privacy concerns, and the need for additional development work to create a complete attendance system. These limitations highlight the need for a more efficient and customizable solution that can address these challenges while being accessible to educational institutions of various sizes.

2.3 Platforms and Technologies Used in Face Recognition Systems

The face recognition system employs deep learning models to achieve high accuracy, utilizing key technologies such as Convolutional Neural Networks (CNNs), MTCNN, and RetinaFace. CNNs, used in frameworks like TensorFlow and Face_Recognition, are crucial for feature extraction, learning facial features such as eyes, nose, and overall face shape. MTCNN (Multi-task Cascaded Convolutional Networks) is used for face detection and alignment, ensuring the system captures high-quality, aligned images for processing. RetinaFace, a state-of-the-art face detection model, further improves accuracy by detecting faces even under challenging conditions. The system can be deployed in two environments: locally, where it runs on an institution's internal network to ensure data remains on-premises without relying on the internet; or in a cloud-based setup, hosted on cloud servers to enable remote access and centralized management.

2.4 Gaps in Existing Systems

The review of existing attendance systems reveals several gaps that still persist. RFID systems, as discussed by Bajaj et al. (2005) and Patel et al. (2007), addressed the inefficiencies of manual attendance methods but failed to prevent proxy attendance, where students could still misuse each other's cards. Fingerprint recognition technologies, explored by Jain et al. (2000), improved security by using biometric identification; however, they introduced hygiene concerns due to shared touch surfaces and required physical contact, which could be problematic. Iris scanning, studied by Zhao et al. (2003), provided very high accuracy but proved to be costly and complex, making it challenging for large-scale deployment in educational institutions. Facial recognition, advanced by researchers such as Chen et al. (2012) and Liao et al. (2016), significantly improved non-contact verification methods, but it still faced issues related to variations in lighting, facial angles, and privacy concerns. Finally, while Learning Management System (LMS) integration, as examined by Black et al. (2018), enhanced data management and real-time analysis, not all attendance systems are designed for seamless or flexible integration, limiting their adaptability across diverse institutional needs.

2.5 Relevance of the Project

The Attendease system is highly relevant to modern educational environments due to several factors. It automates attendance tracking, saving valuable class time and reducing administrative workload. It also enhances security by ensuring that only registered students can be marked present, thereby eliminating the risk of fraud and impersonation. The system provides real-time attendance reports, offering data insights that help teachers monitor student participation and engagement more effectively. Furthermore, it is adaptable to

different learning environments, making it suitable for physical, hybrid, or online classrooms, and increasing its utility in the post-pandemic era.

Chapter 3: System Analysis and Design

The Analysis and Design phase of Attendease focuses on defining the system's requirements and creating a blueprint for its structure and functionality. This chapter outlines the system's functional and non-functional requirements, as well as its architectural design, data flow, and entity relationships. By clearly defining these aspects, the chapter ensures that the system meets user needs, performs efficiently, and is scalable for future growth.

3.1 System Requirements

System requirements are divided into functional requirements, which describe what the system should do, and non-functional requirements, which define how the system should perform. These requirements ensure that the platform is user-friendly, secure, and capable of handling increasing user demands.

3.1.1 Functional Requirements

The system supports user registration and authentication, enabling both students and teachers to create accounts using a username, password, and facial data. Upon login, the system validates user credentials to ensure secure access to the platform. Integral to the core functionality is the ability to register and recognize faces. The system captures and stores facial images of students during registration, which are later used to identify individuals in real time. When students appear for class, the system matches their live facial data against the stored records, allowing for seamless and automatic attendance marking.

Attendance logging is handled automatically as the system detects and confirms a student's presence through facial recognition. This eliminates the need for manual input and ensures greater accuracy. Additionally, the system generates detailed attendance reports, providing teachers and administrators with up-to-date records for academic tracking and administrative use.

Administrative control features are incorporated to support system management. Teachers are given access to view and update attendance records as needed, ensuring flexibility in managing any anomalies or corrections. System administrators are granted broader privileges, including the ability to add or remove students from courses and configure other essential system settings to maintain smooth operation and scalability.

3.1.2 Non-Functional Requirements

The system is designed with a strong emphasis on accuracy, with the facial recognition component expected to correctly identify at least 95% of students, even under varying environmental conditions such as changes in lighting or angle. To ensure a smooth user experience, performance is a key priority; the system should be able to process and log a student's attendance in under two seconds, minimizing delays during class sessions.

Scalability is another essential requirement, as the system must be capable of managing attendance for large classes, potentially involving hundreds of students, without any noticeable decline in performance. Security considerations are critical, particularly because the system handles sensitive biometric and personal data. All facial images and login credentials must be encrypted and stored securely to prevent unauthorized access and data breaches.

Usability is also prioritized to encourage adoption and ease of use across all user groups. The interface should be clean, intuitive, and straightforward, enabling students, teachers, and administrators to navigate the platform and perform necessary tasks with minimal training or technical support.

3.2 System Architecture

The system is designed using a three-tier architecture comprising the presentation layer, application layer, and data layer. The presentation layer, or frontend, is developed using HTML, CSS (Tailwind), and JavaScript. It is responsible for capturing student images, sending them to the backend for processing, and displaying real-time attendance statuses. This layer also includes interactive dashboards for both teachers and administrators, allowing them to manage and review attendance data with ease.

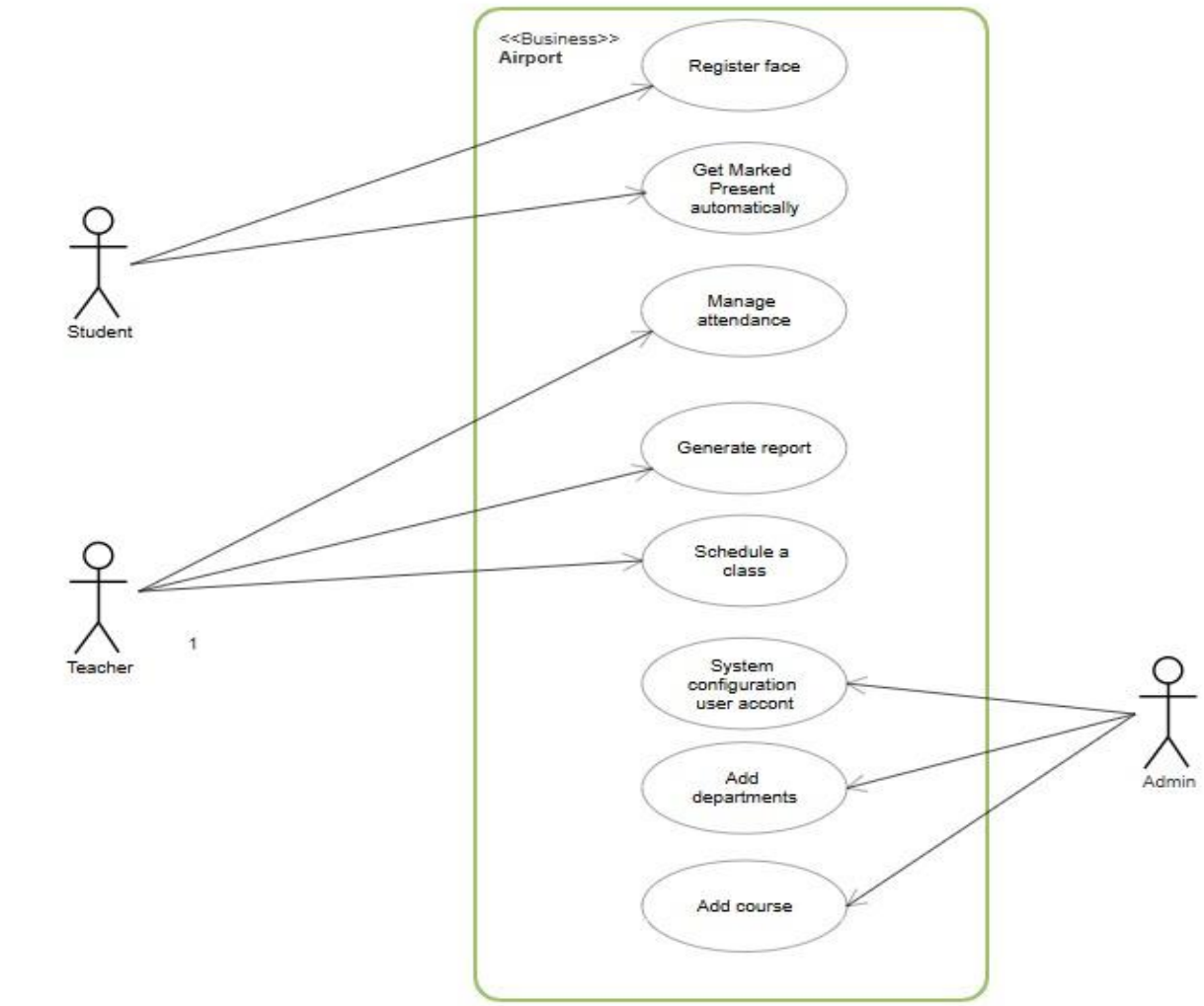
The application layer serves as the system's core logic and processing unit. It is built using Django (Python) and integrates AI technologies such as TensorFlow and OpenCV to handle tasks like facial detection, feature extraction and comparison, database interactions, and user authentication. This layer coordinates the communication between the frontend and the database, ensuring that all operations are executed efficiently and securely.

Finally, the data layer is implemented using PostgreSQL or SQLite, depending on the deployment scale and requirements. It stores critical information including student profiles, facial embeddings, and detailed attendance logs. This layer ensures real-time availability of attendance data, enabling the system to generate accurate and timely reports for academic and administrative use.

3.3 System Design Models

3.3.1 Use Case Diagram

The Use Case Diagram illustrates the interactions between the system's users and functionalities. Key actors include:



a) **Student**: Registers face, gets marked present automatically.

Figure 1 Use case

b) **Teacher**: Manages attendance records and generates reports.

c) **Admin**: Controls system configurations and user accounts.

3.3.2 Data Flow Diagram (DFD) - Level 0

The Level 0 Data Flow Diagram (DFD), also referred to as the context diagram, provides a high-level overview of how data flows within the system and how external actors interact with it. This diagram illustrates the basic interactions between users and the system without diving into detailed internal processes. Three main actors are identified in this model: students, teachers, and the system database. Students submit their facial data to the system during registration and are automatically marked present when recognized. Teachers interact with the system to manage attendance records and view reports generated from stored data. The system database acts as the central data store, housing critical information such as attendance logs and student details.

In the Level 0 DFD, these components are represented using standard diagramming conventions. External entities, such as students, teachers, and administrators, are shown as rectangles. The system processes, including face recognition and attendance logging, are depicted as circles or ovals to indicate the key functions performed internally. Data flows between these entities and processes are shown using directional arrows, indicating the movement of data like facial inputs and attendance updates. Data stores, such as the system database, are drawn as open-ended rectangles, representing repositories where information is saved and retrieved throughout system operations.

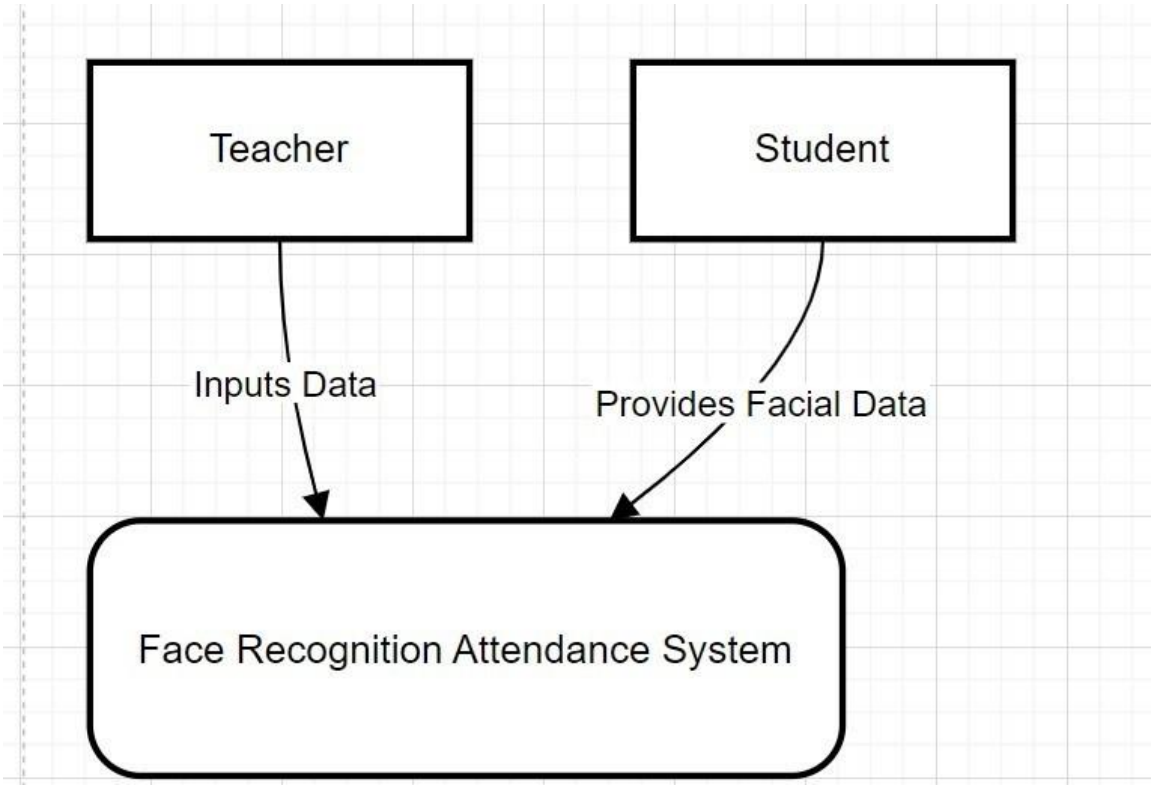


Figure 2 Data Flow Diagram (DFD) - Level 0

3.3.3 Data Flow Diagram (DFD) - Level 1

Level 1 DFD (Detailed System Workflow)

The Level 1 Data Flow Diagram (DFD) provides a detailed view of how data moves through the system during key operations. It breaks down the internal processes involved in student registration, face recognition, attendance logging, and administrative reporting. This level of detail helps illustrate the specific functions of the system beyond the high-level overview offered in the Level 0 DFD.

The process begins with the student registering their face in the system. During this step, the student either uploads a facial image or scans their face using the system's interface. The system processes this input to extract facial embeddings—unique numerical representations of the student's facial features—and stores them securely in the database for future comparison.

During attendance sessions, the system activates its face recognition module as students enter the classroom. It captures the facial features of present individuals and compares them with the previously stored embeddings in the database. If a match is found between the real-time image and the stored data, the system automatically confirms the student's identity.

Once the student is successfully recognized, the system proceeds to log their attendance. This attendance data is then stored in the system's database, where it is organized and maintained as part of the attendance log for that session or course.

Finally, administrators have access to tools that allow them to generate and view comprehensive reports based on the collected attendance data. These reports can be accessed through their respective dashboards and are intended to support academic tracking and decision-making at an institutional level.

3.3.4 Components of the Level 1 DFD Diagram

The Level 1 Data Flow Diagram (DFD) is composed of several interconnected components that represent the internal workings of the system in a structured visual format. Key processes depicted in the diagram include face registration, face recognition, attendance logging, and report viewing. Each of these processes is shown as a distinct functional unit that transforms or interacts with data as it flows through the system.

Central to the diagram is the system database, which acts as the main data store. It holds essential records such as student information, facial embeddings, and attendance logs. This database ensures that all processes have timely access to the information they need, whether for comparing facial data during recognition or for generating attendance reports.

The diagram also illustrates data flows, which represent the movement of information between users, processes, and data stores. These flows include the transfer of student facial data during registration, the retrieval and comparison of facial embeddings during recognition, the recording of attendance results, and the generation of report data for viewing. The direction and labeling of these data flows help clarify how data transitions from one process to another, providing a comprehensive view of the system's operation from input to output.

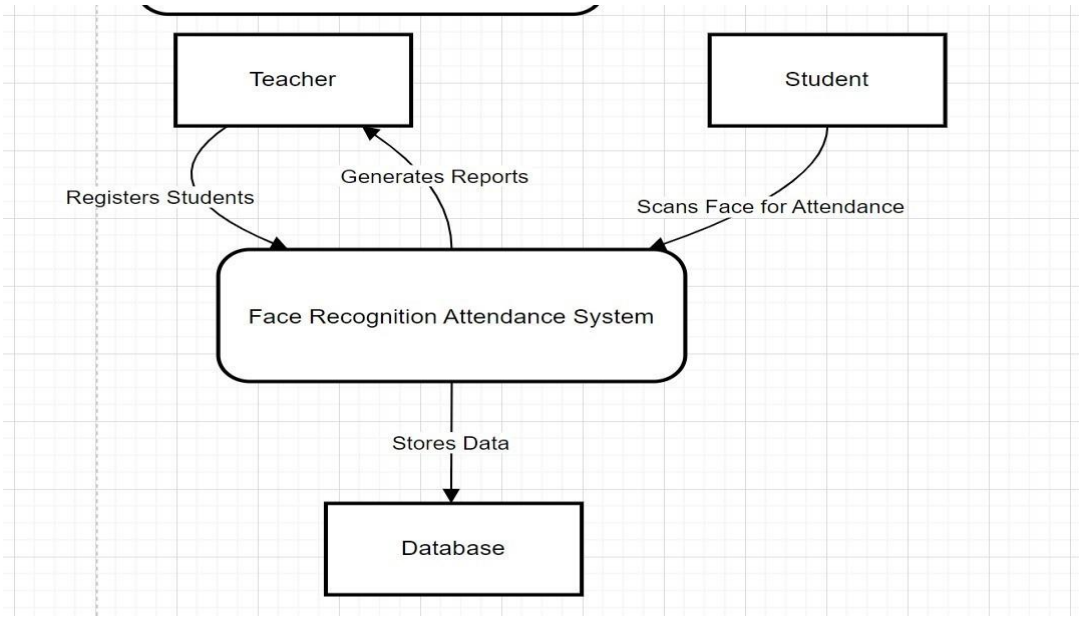


Figure 3 Components of the Level 1 DFD Diagram

3.3.5 Entity-Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) provides a structured overview of the main entities within the system and how they are related to one another. This model is essential for understanding the underlying database structure that supports the system's functionality.

At the core of the ERD is the student entity, which contains attributes such as Student ID, Name, Course, and Face Embedding. These attributes represent each student's identity and the biometric data used for facial recognition. Students are linked to specific courses and are tracked in the attendance system through their unique identifiers.

The Teacher entity includes attributes like Teacher ID, Name, Email, and Courses Taught. Each teacher is responsible for managing one or more courses and is associated with those courses in the system through a

one-to-many relationship. The Course entity itself holds attributes such as Course ID, Name, and the associated Teacher ID, linking each course to a designated instructor.

The Attendance entity serves as a log that tracks each student's attendance status. It includes fields such as Attendance ID, Student ID, Course ID, and Timestamp, allowing the system to record exactly when a student was present for a specific course session. This entity forms a bridge between students and courses, documenting attendance over time.

Finally, the Admin entity includes Admin ID, Name, and Role, representing users who manage the overall configuration and integrity of the system. Administrators have broad access and authority, including the ability to modify user accounts and oversee system operations.

Together, these entities and their relationships form the foundation of the database architecture, enabling the system to manage data accurately and efficiently.

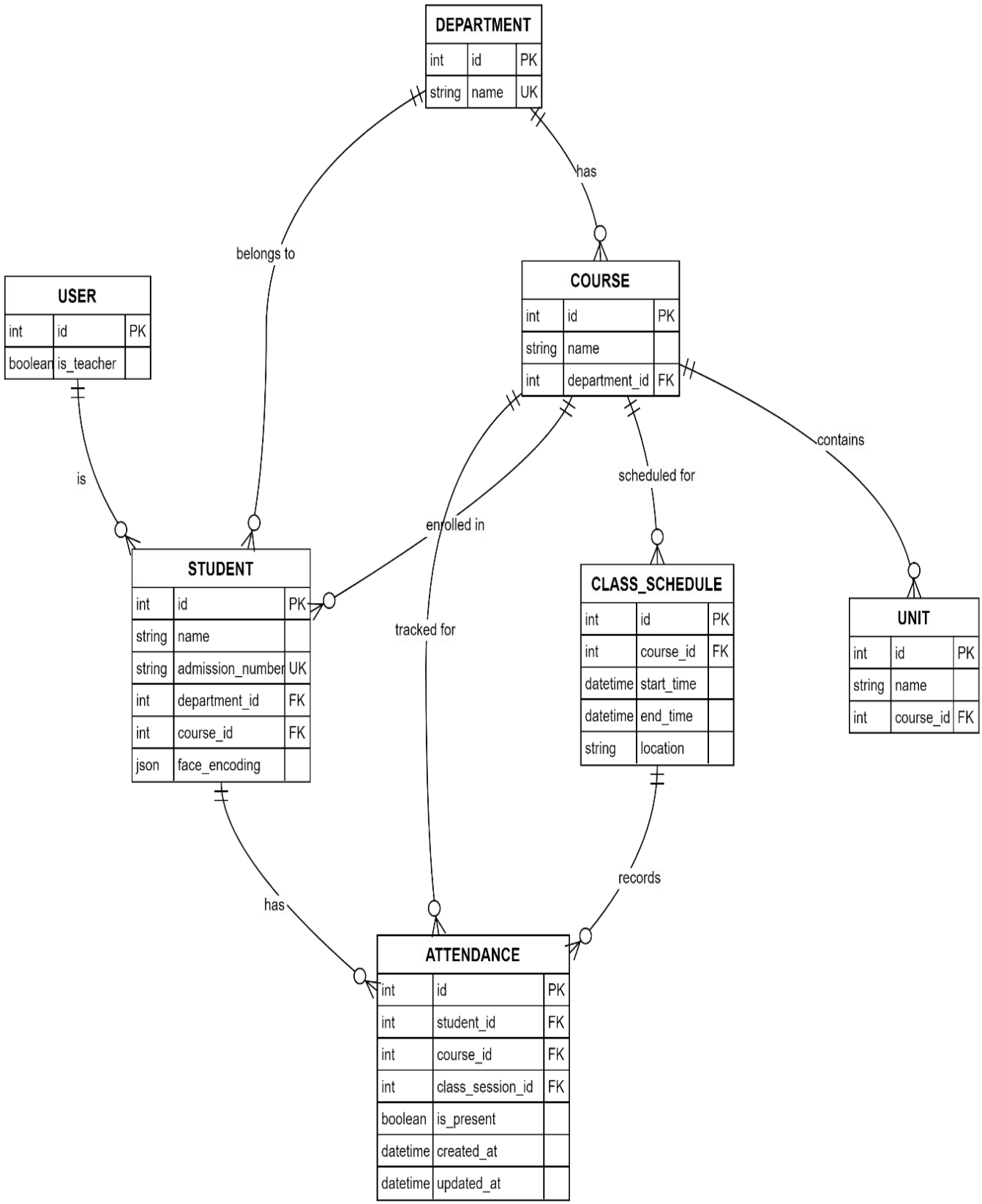


Figure 4 Entity-Relationship Diagram (ERD)

3.4 Database Schema

The database schema for Attendease defines the structured storage of student, course, and attendance data, supporting real-time facial recognition-based attendance logging, secure student authentication, and efficient reporting within an AI-powered academic attendance system

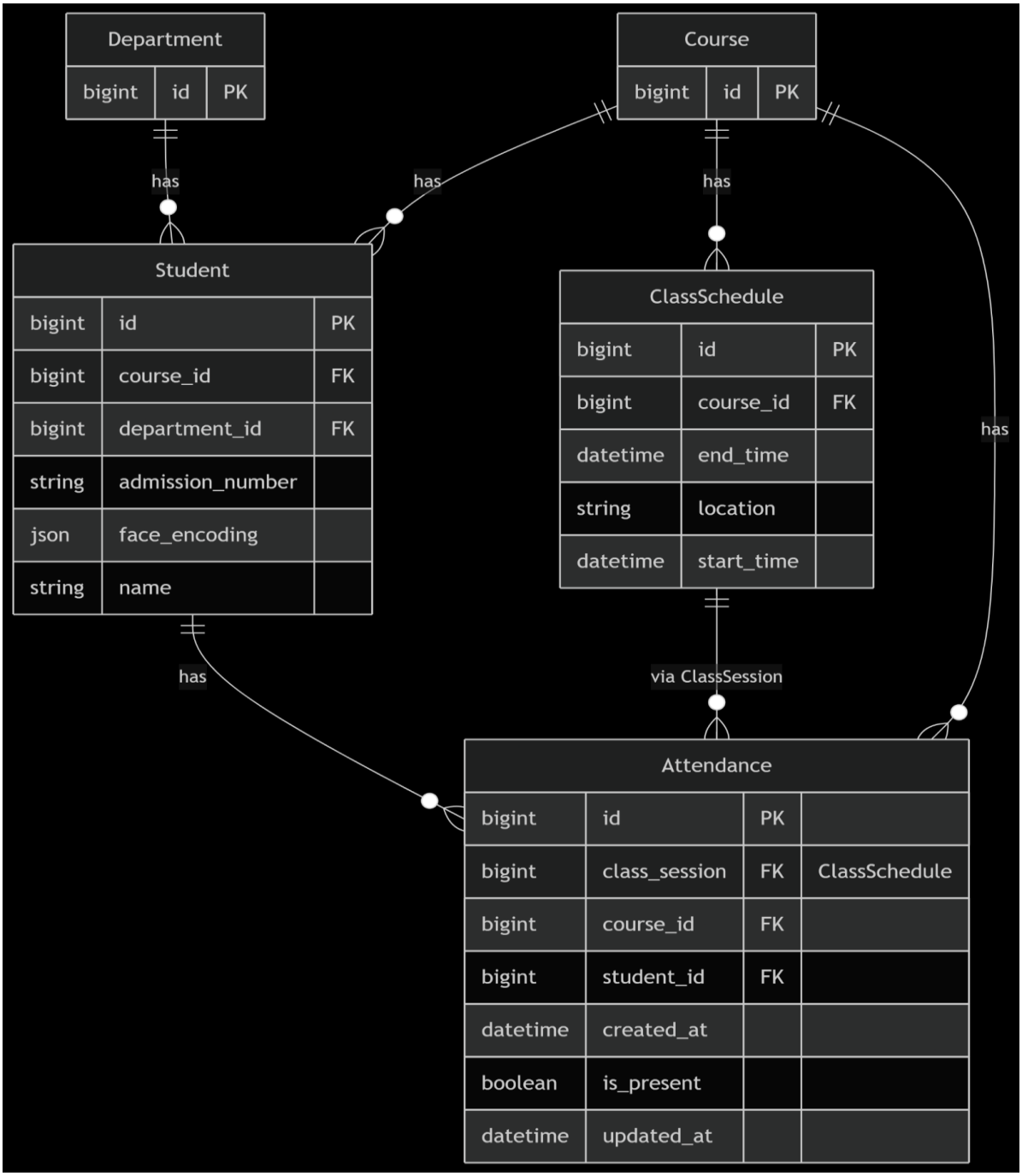


Figure 5 Database Schema

3.5 Face Recognition Workflow

The face recognition workflow is composed of three primary steps that enable accurate and efficient attendance tracking. The process begins with face registration, during which a student either uploads a facial image or scans their face in real time. The system then extracts distinct facial features using MTCNN (Multi-task Cascaded Convolutional Networks), a deep learning model specialized in facial detection. These features are converted into a numerical embedding—a compact representation of the face—which is securely stored in the system's database for future identification.

The second step is real-time face recognition, which occurs as students enter the classroom. A live camera feed captures the student's face, and the system extracts a new facial embedding from the incoming image. This embedding is then compared against the stored data to determine if a match exists. If a match is confirmed, the student is automatically marked as present, eliminating the need for manual input or verification.

The final step is the attendance record update, where the student's presence is logged into the database. This data becomes immediately accessible to teachers, who can view attendance logs through their dedicated dashboard. The entire workflow is designed to be seamless, efficient, and secure, ensuring minimal disruption during classroom sessions while maintaining data integrity.

3.6 System Security Measures

Security is a critical aspect of the system, particularly due to the handling of biometric data and user credentials. One of the key security features is data encryption. All sensitive data, including facial embeddings and user passwords, is hashed and securely stored in the database. During transmission, the system employs secure protocols such as HTTPS to encrypt data and prevent unauthorized interception.

Access control is enforced through role-based authentication, ensuring that each user, whether a student, teacher, or administrator, can only access features and data relevant to their role. This not only protects sensitive information but also simplifies the user experience by tailoring access according to function.

To further enhance security, secure API communication is implemented using JSON Web Tokens (JWT). These tokens validate and authorize each request between the frontend and backend systems, ensuring that data exchanges are secure and verified. This multi-layered security approach helps maintain user trust and system reliability.

3.7 User Interface Design

The user interface is designed with clarity and accessibility in mind, offering separate dashboards tailored to the needs of each user group. The student dashboard includes features for face capture and verification, as well as a history of their attendance records. This provides students with real-time feedback and transparency regarding their attendance status.

The teacher dashboard is built to support attendance management tasks. Teachers can view detailed attendance reports for each course and make modifications to the records when necessary. This flexibility allows for efficient oversight and correction of any anomalies in attendance tracking.

The admin panel offers comprehensive system control capabilities. Administrators can manage users, assign courses, monitor system logs, and oversee security settings.

Chapter 4: Methodology

This chapter describes the methodology used in developing the attendease. It details the development approach, tools and technologies, development process, collaboration strategies, and challenges faced.

4.1 Development Approach

The design approach for the Attendease Face Recognition-Based Attendance System centers on utilizing facial recognition technology to deliver an efficient, accurate, and scalable attendance solution. By integrating Django for robust backend development, TensorFlow for machine learning model creation, and the Face_Recognition library alongside Face_API.js for real-time face detection and recognition, the system ensures reliable performance. The front-end, built with HTML, CSS (Tailwind), and JavaScript, provides a user-friendly interface. A key component of the system is the facial recognition model, which is trained through a structured process involving data collection, preprocessing, model selection, training, evaluation, integration, and continuous improvement.

Data collection involves gathering a dataset of student images, with each student contributing 10–20 images captured under diverse lighting conditions, angles, and facial expressions to enhance model robustness. Ethical considerations are prioritized by obtaining student consent and adhering to data privacy regulations, such as GDPR or institutional policies. The dataset is organized into labeled folders, each corresponding to a student's unique identifier (e.g., student ID) and containing their images.

During data preprocessing, images are resized to a uniform resolution, such as 224x224 pixels, to ensure consistency during training. Image augmentation techniques, including rotation, flipping, and brightness adjustments, are applied to expand the dataset artificially and improve the model's ability to generalize across varied conditions. The Face_Recognition library is used to detect and crop faces, focusing on relevant facial features. Images are converted to grayscale or have their pixel values normalized to reduce computational complexity and enhance model performance.

For model selection, a pre-trained convolutional neural network (CNN), such as VGGFace or a ResNet-based architecture from TensorFlow, serves as the base model for feature extraction. This model is fine-tuned using transfer learning to adapt to the specific task of recognizing students, minimizing training time and resource demands. Alternatively, the Face_Recognition library's face encoding functionality, which generates 128-dimensional embeddings for each face, offers a lightweight solution for smaller datasets.

The training process involves splitting the preprocessed dataset into training (70%), validation (20%), and testing (10%) sets to assess performance. For CNN-based models, training is conducted using TensorFlow with a categorical cross-entropy loss function and the Adam optimizer to minimize classification errors. For Face_Recognition-based models, face encodings are computed for all training images and stored in a database, such as Django's SQLite or PostgreSQL, for real-time comparison during inference. Training metrics like accuracy and loss are monitored, with early stopping employed to prevent overfitting if validation performance plateaus.

Model evaluation is conducted on the test set to measure accuracy, precision, recall, and F1-score, ensuring the model achieves a performance threshold, such as >95% accuracy for attendance marking. Real-world testing with live video feeds or classroom images assesses robustness against variations in lighting, angles, and occlusions, such as glasses or masks. If the model underperforms, hyperparameters like learning rate or batch size are fine-tuned, or additional data is collected.

Integration with the system involves embedding the trained model into the Django backend to process real-time video streams or uploaded images via Face_API.js for face detection and recognition. Recognized student identities and attendance records are stored in the database, linked to timestamps and class schedules. Error-handling mechanisms address cases where the model fails to recognize a face, such as low confidence scores, and provide manual override options for teachers.

Continuous improvement ensures the system remains effective over time. The model is periodically retrained with new student data to account for changes in appearance, such as new hairstyles or aging, and to onboard new students. Feedback from teachers and administrators helps identify edge cases or performance issues, like false positives or negatives. The system is updated iteratively using the Agile methodology's sprints to deploy enhancements incrementally, ensuring ongoing reliability and user satisfaction.

The development of the Face Recognition-Based Attendance System was guided by the Agile Software Development Methodology, with a specific focus on the Scrum framework. Agile was chosen for its iterative nature, which supports continuous improvement throughout the development lifecycle. This approach allowed our team to break down the project into smaller, manageable tasks that could be tackled in short development cycles known as sprints.

One of the key advantages of using Scrum was the ability to quickly incorporate feedback after each sprint, enabling us to adapt the system based on real-time insights and user needs. Additionally, Scrum fostered a highly collaborative environment, encouraging regular communication among team members through daily

meetings and reviews. This ensured that everyone remained aligned on project goals, progress, and potential challenges, ultimately contributing to a more efficient and responsive development process.

4.1.1 Sprint Planning

At the start of each development cycle, we held sprint planning sessions to define clear goals for the system and allocate specific tasks among team members. For example, during our initial planning, we divided responsibilities such as setting up the Django backend, designing the user interface with Tailwind CSS, and researching and integrating facial recognition technologies like TensorFlow and Face_API.js.

This phase helped us establish a shared vision for what needed to be accomplished in each sprint and made sure everyone was aligned and aware of their individual responsibilities. We prioritized tasks based on complexity and dependencies, which helped streamline our workflow in later stages.

4.1.2 Daily Standups

Throughout the development process, we conducted daily standup meetings to ensure consistent communication and collaboration. These brief meetings allowed each team member to share their current progress, identify any roadblocks, and plan their next steps.

For instance, when one of us faced issues integrating the face recognition model into the frontend, we were able to discuss it during the standup and brainstorm possible solutions together. These daily check-ins helped maintain momentum, addressed problems early, and kept the entire team updated on the project's overall status.

4.1.3 Sprint Development

During each sprint, we entered the development phase, where we worked on building the features defined in the sprint plan. Tasks were tackled in short, iterative cycles, allowing for focused and manageable progress.

In one sprint, for example, we focused solely on setting up user registration and attendance tracking in Django, while another was dedicated to implementing and testing the facial recognition pipeline. These cycles enabled us to concentrate on one functional part of the system at a time, reducing complexity and ensuring higher quality output. We frequently committed code to our repository, reviewed each other’s work, and made adjustments based on initial results.

4.1.4 Testing and Review

After developing each feature, we conducted testing and review sessions to ensure functionality, accuracy, and usability. We tested the facial recognition system with a variety of student images to verify detection accuracy and responsiveness. We also reviewed the user interface for clarity and responsiveness across devices.

Debugging was a regular part of this phase—issues like inconsistent face detection or minor frontend glitches were identified and resolved through peer reviews and joint testing sessions. This phase was crucial in catching bugs early and refining the system before wider testing or deployment.

4.1.5 Deployment and Feedback

At the end of each sprint, we deployed the current version of the system and gathered feedback from both our peers and our supervisor. We shared what was working well and where improvements could be made.

For instance, early feedback indicated that the user interface could be more intuitive, which led us to refine some of the layouts and button placements. Feedback also helped us fine-tune the facial recognition to improve speed and accuracy. By incorporating this feedback into the next sprint, we continuously improved the system in a structured yet flexible way. This loop of delivery and iteration was key in ensuring that the system evolved to meet user needs more effectively.

Figure 6 Scrum

4.2 Tools and Technologies

The Attendease was developed using a combination of modern tools and technologies to ensure efficiency, scalability, and security. Below is a detailed breakdown of the tools and technologies used:

4.2.1 Backend Development

Django: A high-level Python web framework used for developing secure and scalable server-side applications. Django provides built-in features for routing, user authentication, and admin management.
TensorFlow: An open-source machine learning framework used to train and deploy facial recognition models. TensorFlow processes facial data and helps in generating embeddings for identity matching.
OpenCV: A computer vision library used for image processing tasks such as face detection and alignment. OpenCV enhances facial recognition accuracy by pre-processing images before embedding extraction.
face_recognition: A Python library built on top of dlib, used to encode faces into numerical embeddings and compare them for verification. It simplifies facial feature extraction and matching processes.

4.2.2 Database

PostgreSQL: A powerful, open-source relational database used in production environments for storing student records, facial embeddings, and attendance logs. It supports complex queries and ensures data integrity.
SQLite: A lightweight, serverless database used during development for simplicity and quick testing. It allows for easy database setup without requiring a dedicated server.

4.2.3 Version Control

Git: A distributed version control system used to track changes in the source code. Git supports collaborative development by allowing multiple contributors to work on the codebase simultaneously.
GitHub: A cloud-based hosting service for Git repositories. GitHub is used for storing project files, managing branches, and facilitating code reviews and issue tracking.

4.2.4 Development Environment

Visual Studio Code (VS Code): A popular code editor that provides syntax highlighting, debugging tools, and extensions tailored for web and Python development. It serves as the main development interface.
Jupyter Notebook: An interactive computing environment used for experimenting with facial recognition models and visualizing machine learning outputs. It is especially useful during the prototyping phase.

4.2.5 Testing and Validation

Postman: A tool for testing and validating RESTful APIs. Postman is used to ensure that endpoints function correctly and return expected data when accessed by the frontend.
PyTest: A Python testing framework used to perform unit tests on backend logic and facial recognition components. It helps maintain code quality by verifying that core functions behave as expected.

4.2.6 Deployment

Docker: A containerization tool used to package the application with all its dependencies. Docker ensures consistency across development, testing, and production environments.
AWS / Heroku: Cloud platforms used for deploying the application and making it accessible to users online. AWS offers scalable infrastructure, while Heroku provides a user-friendly platform for quick deployment and management.

4.3 Collaboration and Communication

Effective collaboration was crucial for the success of the project, as it involved four team members working simultaneously on various components of the system. To ensure smooth coordination and progress tracking, several tools were adopted throughout the development cycle. GitHub was used for version control and source code management, enabling contributors to work on different features without conflict and maintain a history of changes. Slack and Discord were used for team communication and conducting daily standups. These platforms allowed for real-time messaging, quick updates, and screen sharing, which streamlined problem-solving during remote sessions.

4.4 Challenges and Solutions

Several technical and operational challenges were encountered during the development of the system. One of the primary issues was the low accuracy of face recognition in poor lighting conditions. To address this, image preprocessing techniques such as histogram equalization and adaptive thresholding were implemented. These methods enhanced facial image clarity, which significantly improved the reliability of recognition results.

Another major challenge involved slow face matching when handling large datasets. As the number of registered students grew, the system's performance declined during real-time comparisons. To mitigate this, the face embedding comparison algorithm was optimized, and the Faiss (Facebook AI Similarity Search) indexing library was integrated to accelerate the search process and reduce latency.

Browser restrictions presented another obstacle, especially when using Face_API.js. Some browsers initially blocked camera access or produced errors when trying to load facial detection models. This issue was resolved by enforcing HTTPS for all web interactions and ensuring appropriate user permissions were requested at runtime, enabling smooth operation of webcam-based features.

Lastly, security concerns arose regarding the storage and transmission of facial data. To safeguard sensitive information, several security measures were implemented. JWT authentication was used to control access to system resources, while passwords were hashed using secure algorithms.

CHAPTER 5: PROJECT MANAGEMENT

AttendEase is a face recognition-based student attendance system developed over a three-month period. The system automates attendance tracking in academic settings using facial recognition technology, eliminating traditional manual roll calls and paper-based signing methods that are time-consuming and prone to errors. The project employed agile methodology for development, consisting of design, frontend and backend phases.

5.1 Time Management and Milestones

Phase 1: Planning and Design (Weeks 1–2)

The initial phase of the project was dedicated to planning and design. During this period, the team focused on defining the project goals, identifying the core problems with existing attendance systems, and outlining the system's requirements. A detailed requirements analysis was conducted to determine the essential features and user expectations. This was followed by UI/UX design, where wireframes and interface prototypes were developed using Figma to visualize the user flow and dashboard layouts. The system architecture and database schema were also planned, including defining entities such as students, teachers, attendance logs, and facial embeddings. The outcome of this phase was a finalized system design, a clear understanding of the system's functional flow, and ready-to-implement interface blueprints.

Phase 2: Frontend Development (Weeks 3–6)

The second phase focused on frontend development. The team implemented the user interface using HTML, Tailwind CSS, and JavaScript, ensuring a responsive and user-friendly experience across devices. Key components such as login pages, student registration forms, teacher dashboards, and admin panels were developed. During this phase, the Face_API.js library was integrated into the frontend to allow real-time face capture using webcams. User experience testing was carried out iteratively to refine layout, performance, and accessibility. The outcome of this phase was a fully functional and responsive frontend that interacted effectively with real-time video feeds and prepared input for facial recognition.

Phase 3: Backend Development (Weeks 7–10)

In the third phase, the backend infrastructure of the system was constructed using Django. Key functionalities such as user authentication, face registration, and attendance logging were implemented. TensorFlow and the face_recognition library were integrated into the backend to handle face detection, feature extraction, and comparison. A PostgreSQL or SQLite database was connected to manage data storage, including user credentials, facial embeddings, and attendance records. Security layers such as

password hashing and access control logic were also added during this stage. The outcome of this phase was a robust backend system that processed facial data and supported all core functionalities of the application.

Phase 4: Testing, Deployment, and Documentation (Weeks 11–12)

The final phase of the project involved thorough testing, deployment, and documentation. The system was tested extensively to identify bugs, verify facial recognition accuracy, and ensure overall stability. Both unit and integration tests were carried out, focusing on real-time detection, attendance logging, and API functionality. Once the system passed all quality checks, it was deployed on a local or cloud platform using Docker, making it accessible for demonstration and evaluation. Project documentation was also completed, including user guides, technical manuals, and system specifications. The outcome of this phase was a fully deployed and documented attendance system, ready for presentation and further use.

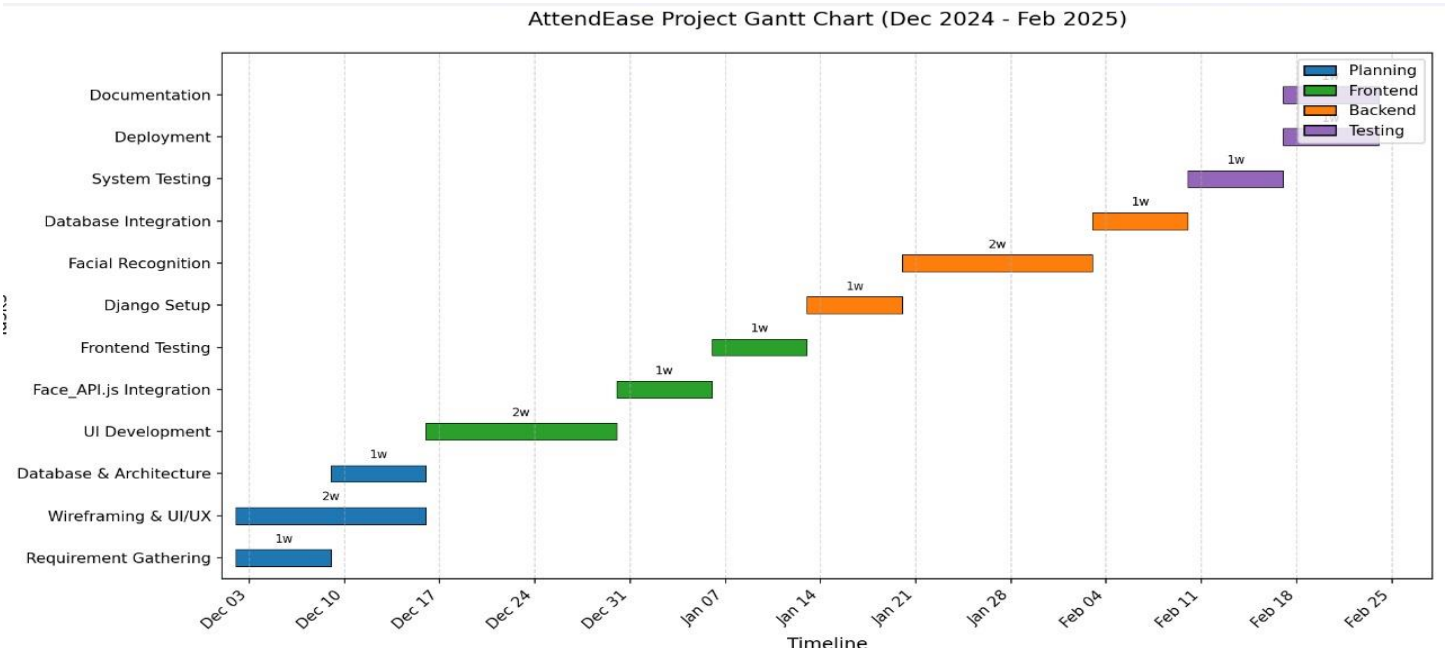


Figure 7 Time Management and Milestones

5.2 Budget Planning

Budget planning played a crucial role in ensuring that the development of the facial recognition attendance system remained financially feasible and well-organized. The overall budget was structured to cover development costs, resource allocation, and unforeseen expenses.

The initial development costs included software tools, hardware equipment, human resources, training datasets, and cloud service subscriptions necessary for deployment and testing. In addition, a structured budget allocation was designed to ensure that financial resources were distributed appropriately across major project needs such as infrastructure and staffing. A contingency reserve was also included to address unexpected costs that might arise during the development lifecycle.

To complement the main budget, a separate section for additional expenses was outlined. This included costs related to software licenses, testing hardware, team training, and documentation tools required to support quality assurance and team productivity. The financial planning process aimed to maintain a balance between innovation and sustainability, ensuring the successful delivery of the project within available financial constraints.

5.2.1 Development Costs:

Table 2 Development Costs

Category	**Amount (Ksh)**
Software Tools	40,000
Hardware Requirements	60,000
Human Resources	150,000
Training Data	75,000
Cloud Services	50,000
Subtotal	**375,000**

5.2.2 Budget Allocation:

Table 3 Budget allocation

Category	**Amount (Ksh)**	**Percentage**
Development Environment	120,000	15%
Human Resources	600,000	70%
Contingency Reserve	120,000	15%
Total Budget	**860,000**	**100%**

5.2.3 Additional Expenses:

Table 4 Additional expenses

Category	**Amount (Ksh)**
Software Licenses	80,000
Testing Hardware	90,000
Team Training	60,000
Documentation Tools	30,000
Subtotal	**260,000**

5.3 Resource Allocation and Usage

Human Resources:

Effective allocation and utilization of both human and technical resources were key to the success of the project. The project team consisted of five members, each assigned a distinct role based on their expertise. James Ngandu served as the Project Manager, overseeing the overall timeline, ensuring effective team coordination, and maintaining communication with stakeholders. He also took on the role of Frontend Developer, where he was responsible for creating responsive web interfaces using HTML, Tailwind CSS, and JavaScript. Jacinta Atieno worked as the UI/UX Designer, focusing on crafting intuitive user interfaces

and seamless user experience flows to enhance usability. The Backend Developer, Henry Ouma, handled server-side development tasks, including the construction of database schemas, development of API endpoints, and implementation of business logic using Django. David Wambua, as the Machine Learning Engineer, was tasked with integrating and fine-tuning facial recognition algorithms using tools such as TensorFlow and face_recognition.

Technical Resources

In terms of technical resources, local development environments were set up on each team member's machine, equipped with the necessary libraries, frameworks, and development tools. Git was used for version control to ensure collaborative coding and proper versioning of project files. Testing equipment included laptops with webcams, which were essential for testing the facial recognition system under different lighting conditions and angles. Additionally, a local instance of the database was set up to store and manage student information, facial embeddings, and attendance records during development and testing phases.

5.4 Risk Management

Technical Risks

From a technical perspective, several risks were anticipated and addressed. One of the key risks involved the accuracy of the facial recognition system, particularly under varying lighting conditions or when students wore facial accessories such as glasses or masks. This was mitigated by implementing preprocessing techniques like histogram equalization and performing extensive testing with a diverse set of facial images. Another technical concern was the potential for decreased system performance when processing multiple face recognition requests simultaneously. This risk was handled by optimizing the recognition algorithm and implementing queue-based processing mechanisms to distribute load efficiently. Data security was also a priority due to the sensitive nature of facial information. To mitigate this, robust encryption methods were applied, secure storage practices were adopted, and explicit user consent mechanisms were introduced.

Project Management Risks

On the project management side, the risk of scope creep—where additional features or requirements are added beyond the original scope—was controlled by maintaining clear documentation of the agreed-upon scope and implementing a formal change request process. Schedule delays, often caused by unforeseen technical hurdles, were mitigated by including buffer periods in the sprint plan and conducting regular

progress evaluations. To prevent coordination issues among team members working on different system components, regular integration meetings were held, and shared documentation practices were adopted.

Operational Risks

Operational risks were also considered. A major risk was potential resistance from end-users, such as students or faculty, in adopting the new facial recognition system. To encourage adoption, users were engaged early in the development process, and efforts were made to design an intuitive and user-friendly interface. Comprehensive training materials and support resources were also planned. Finally, integration with existing academic systems posed a potential challenge. This risk was mitigated by designing the system in a modular fashion and ensuring that APIs were well-documented and adaptable for future integration efforts.

Chapter 6: Discussion / Results

6.1 System Features and Functionalities

The Face Attendance System implements a comprehensive set of features designed to streamline student attendance tracking through facial recognition technology. This section details both the functional aspects and the underlying technology stack that powers the system.

6.1.1 Technology Stack

The facial recognition attendance system is built using a robust combination of frontend and backend technologies:

Frontend Technologies

The frontend of the application is built using a combination of modern web technologies to ensure a robust and interactive user interface. HTML5 and CSS3 serve as the structural foundation and provide styling for the user interface, ensuring a clean and accessible design. Tailwind CSS, a utility-first CSS framework, is employed to achieve responsive design and maintain consistent styling across the application, streamlining the development process. JavaScript powers dynamic client-side interactions and facilitates facial recognition processing, enabling real-time user engagement. TensorFlow is utilized to drive machine learning components for facial detection directly on the client side, enhancing performance. Additionally, face-api, a JavaScript API, simplifies the implementation of face detection and recognition features within the browser, making facial recognition accessible and efficient.

Backend Technologies

On the backend, Django, a Python web framework, manages server-side logic, user authentication, and database operations, providing a secure and scalable foundation for the application. OpenCV, a computer vision library, is integrated for advanced image processing and additional face detection capabilities, complementing the system's recognition features. The face_recognition Python library works alongside OpenCV to deliver core facial recognition algorithms, ensuring accurate identification. SQLite and PostgreSQL are used as database systems to store student profiles, attendance records, and course information, offering reliable and efficient data management.

Integration Architecture

The system employs a hybrid approach to integrate frontend and backend functionalities effectively. Initial face detection is performed in the browser using TensorFlow.js and face-api.js, providing real-time feedback to users for a seamless experience. More complex recognition and matching operations are handled by the backend, leveraging the face_recognition and OpenCV libraries for enhanced accuracy and processing power. Django's REST Framework facilitates smooth communication between the frontend and backend components, ensuring efficient data exchange and system cohesion.

6.1.2 User Authentication and Role-Based Access

The system incorporates a secure login functionality that provides role-specific dashboards, enabling users to access features tailored to their designated roles. The Teacher Dashboard acts as the primary interface for instructors, serving as a central hub for managing attendance-related activities. Authentication is handled by Django's robust authentication framework, which efficiently manages user sessions and enforces access control to ensure security. Role-based permissions are implemented to grant appropriate access levels to different user types, such as administrators, teachers, and students, ensuring that each user can only interact with functionalities relevant to their role. Additionally, session management is designed with security in mind, incorporating active session maintenance and appropriate timeout mechanisms to protect user data and prevent unauthorized access.

6.1.3 Class Management

The system offers robust class scheduling capabilities:

Schedule Creation: Teachers can create new class schedules by specifying course name, start time, end time, and location through a simple form interface.

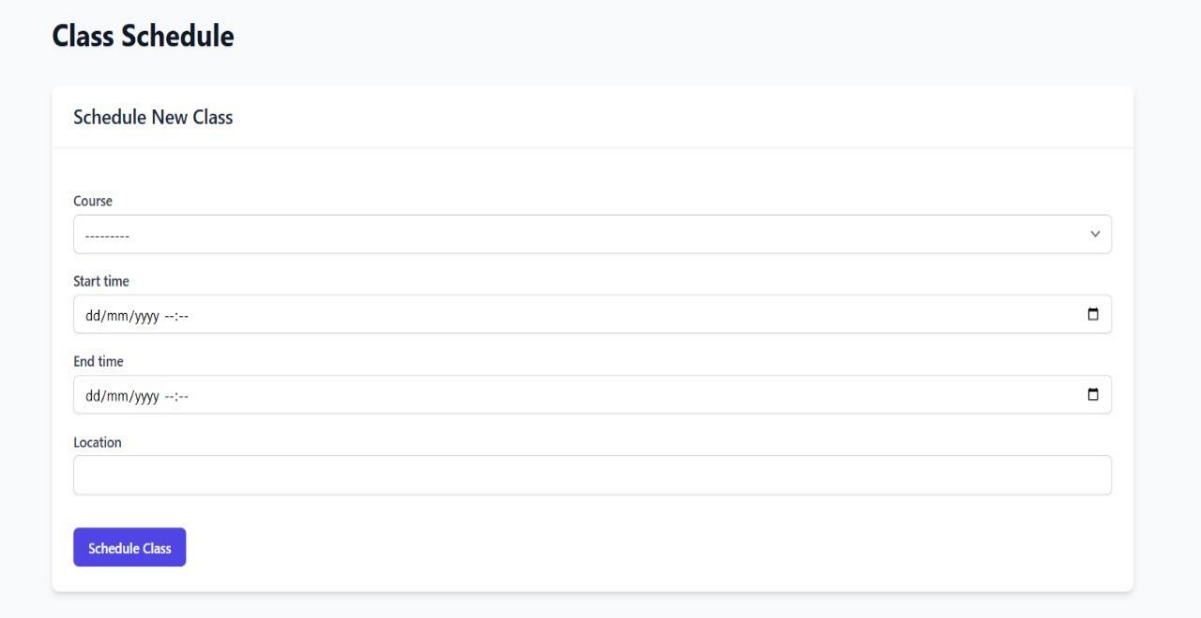


Figure 8 Class Management

Schedule Viewing: The system displays both upcoming and past classes in an organized tabular format, showing essential details including course name, time slots, and locations.

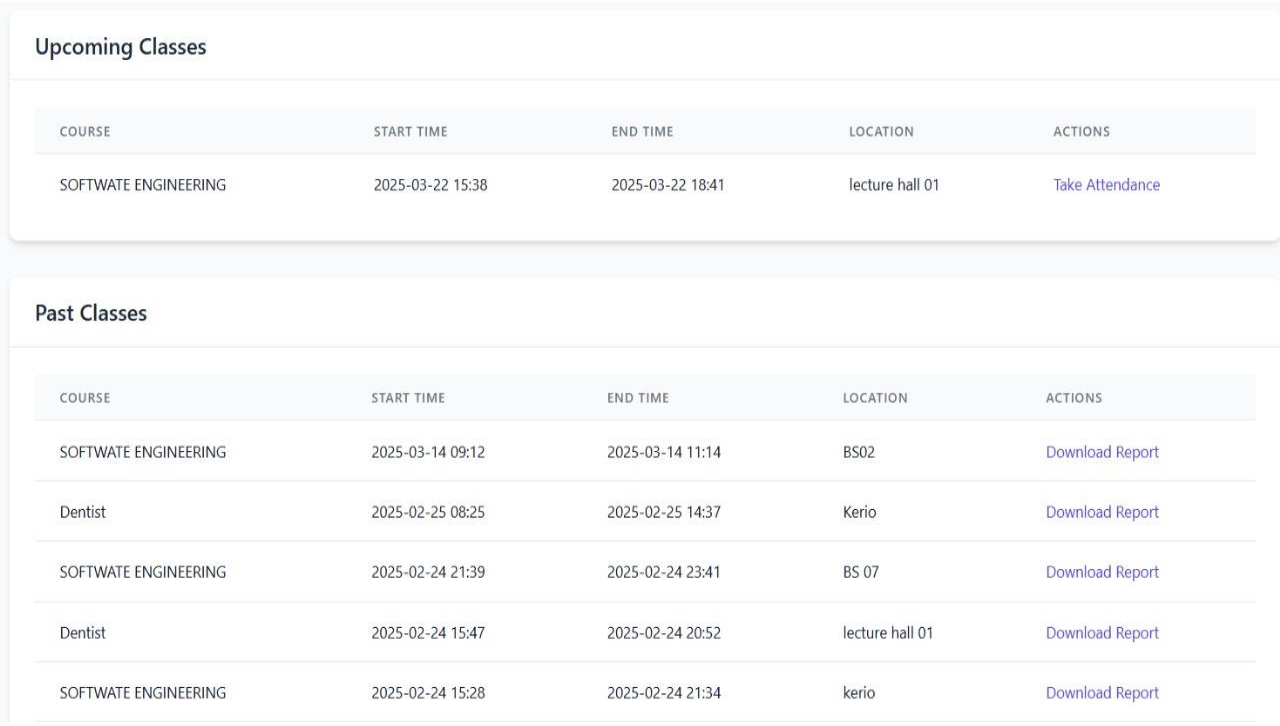


Figure 8 Schedule Viewing

Course Association: Classes are associated with specific courses (e.g., "SOFTWARE ENGINEERING").

Calendar Integration: Class schedules are managed through a datetime-based system that properly organizes upcoming and past sessions.

6.1.4 Facial Recognition Attendance Marking

The core functionality of the system is its ability to identify students through facial recognition:

Real-time Face Detection: The system captures student faces through a camera interface, as demonstrated in the recognition screen.

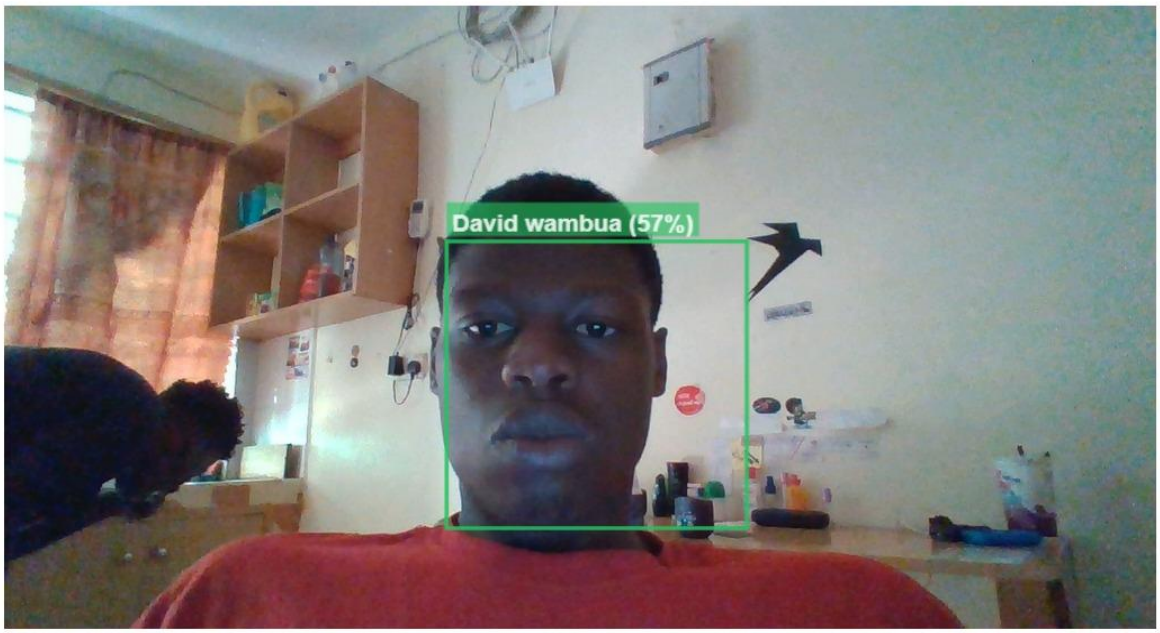


Figure 9 Facial Recognition Attendance Marking

Face Detection Process:

The face detection process begins with capturing the camera feed, which is then processed using face-api.js. This JavaScript library detects the facial regions within the feed and highlights them with bounding boxes, providing visual feedback. Once the faces are identified, the system goes a step further by recognizing key facial landmarks, helping to normalize pose variations and improve detection accuracy, even if the student's face is tilted or turned.

Next, facial features are extracted from the detected faces and converted into numerical embeddings, which serve as a compact representation of the facial characteristics. These embeddings are then compared against the stored database of student profiles, enabling the system to perform identity verification. When a captured face matches a stored profile, the system displays a confidence percentage indicating the level of match accuracy. The system also applies a minimum confidence threshold to ensure that only valid matches are

recorded. This threshold helps reduce false positives, ensuring that attendance is accurately marked for each student.

6.1.5 Attendance Monitoring and Reporting

Comprehensive attendance tracking features include:

Real-time Status: The Teacher Dashboard shows real-time attendance status during active sessions.

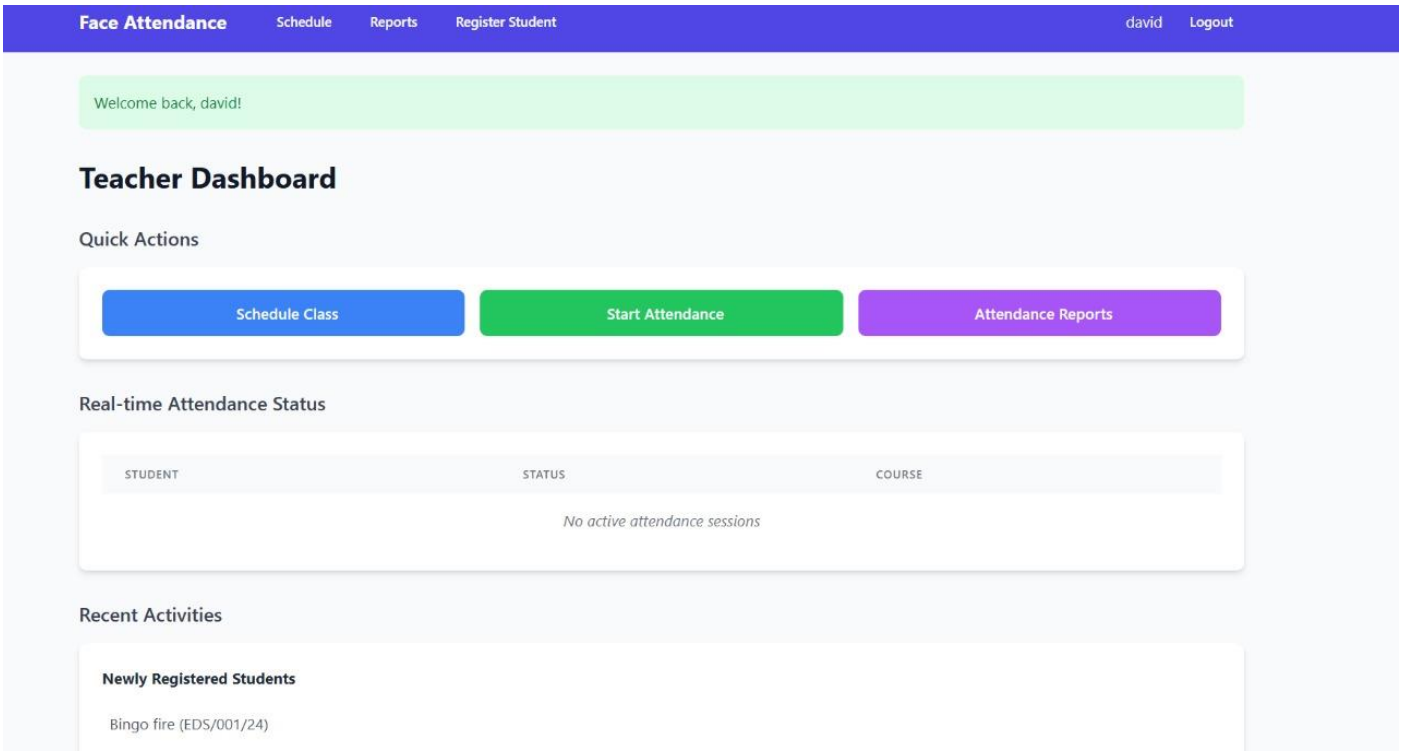


Figure 10 Teacher dashboard

Historical Records: Past attendance data is stored and accessible through the reporting interface.

Attendance Metrics: The system calculates and displays attendance percentages (0%, 50%, 100%) with color-coding for immediate visual assessment.

Downloadable Reports: For each class session, attendance reports can be downloaded through the "Download Report" button.

Data Visualization: Reports include attendance statistics with appropriate visualizations for trend analysis.

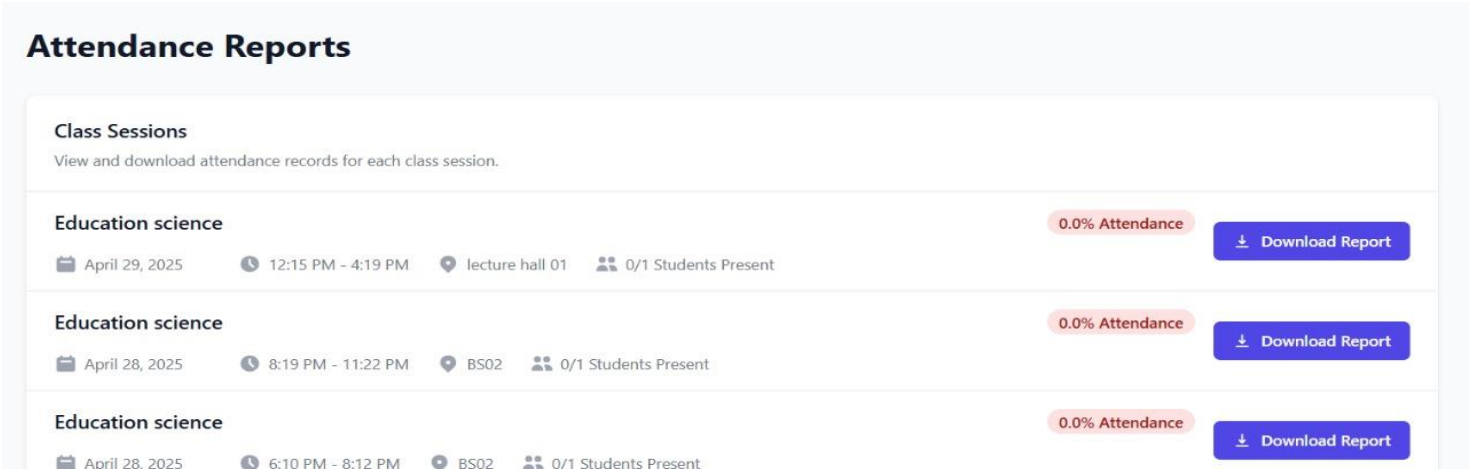


Figure 11 Attendance report

6.1.6 Student Registration

The student registration process in the system is designed to efficiently add new students to the facial recognition database. Administrators can access a dedicated "Register Student" section, which provides the interface for enrolling new students. During the enrollment process, administrators capture essential student information such as name and ID, ensuring proper identification. In addition to this, multiple facial images are taken from different angles to improve recognition accuracy. These images are processed using the face_recognition library to generate facial embeddings, which are unique numerical representations of the student's facial features.

These embeddings, along with the student's information, are securely stored in the database, creating a reference for future identity verification during attendance marking. As part of the registration process, newly enrolled students are displayed in the "Recent Activities" section for easy tracking. The system also ensures proper database management by maintaining a comprehensive record of all registered students, which is used for facial matching during real-time face recognition.

6.1.7 Overall System Workflow

The overall system workflow is designed to ensure smooth and efficient operation, from initialization to reporting. The system can be broken down into several key phases, each contributing to the overall functionality.

Initialization Phase

In the initial phase, administrators set up the system by registering courses and student profiles. Teachers are assigned to their respective courses, and students' facial data is enrolled into the facial recognition database. This setup establishes the foundational data required for the system to function effectively during later phases.

Class Scheduling Phase

Once the initial setup is complete, the next phase involves scheduling. Teachers schedule their classes, specifying important details such as the course name, time, and location. The system maintains an updated list of upcoming classes, ensuring that all necessary details are readily accessible for both teachers and students.

Attendance Capture Phase

This phase focuses on the real-time attendance capture during scheduled classes. The teacher initiates the attendance process, which activates the camera interface to process the live video feed. Students then present themselves to the camera, allowing the system to detect faces, extract relevant facial features, and match them with the registered profiles in the database. The system displays the recognition results along with confidence levels, ensuring transparency. Once the recognition process is complete, the attendance status is updated in real-time, providing an accurate log of student presence.

Reporting Phase

At the end of each class session, the system compiles the attendance results. Teachers can access these results through the reporting interface, where they can review the data, download reports for record-keeping, and analyze the attendance trends. The system also calculates attendance statistics, which are displayed for easy access and analysis.

Administrative Phase

The final phase provides administrative oversight of the system. Administrators can monitor overall system performance and usage. They have the ability to register new students as needed and update course information to ensure the system remains current.

This integrated workflow creates a streamlined and efficient experience for users, significantly reducing the administrative burden associated with traditional attendance methods while improving accuracy and offering valuable insights through data analytics.

6.2 User Experience and Performance Analysis

6.2.1 Interface Design and Usability

The Face Attendance System provides a clean, intuitive user interface optimized for different user roles:

Dashboard Layout: The Teacher Dashboard employs a card-based design with clearly delineated sections for Quick Actions, Real-time Attendance Status, and Recent Activities, making information readily accessible.

Color Coding: The system uses intuitive color coding (green for 100% attendance, yellow for partial attendance, red for 0% attendance) to provide at-a-glance status information.

Action Buttons: Prominent, color-coded action buttons (blue for scheduling, green for starting attendance, purple for reports) create clear visual hierarchies for common tasks.

Navigation: The top navigation bar provides quick access to key system areas (Schedule, Reports, Register Student).

Responsive Design: The interface adapts to different screen sizes while maintaining functionality and readability, implemented through Tailwind CSS's responsive utilities.

6.2.2 Facial Recognition Performance

Based on the implementation, the facial recognition component demonstrates the following performance characteristics:

Recognition Accuracy: The system displays confidence levels for matches, allowing for threshold adjustment to balance between false positives and false negatives.

Detection Speed: Face detection occurs in real-time, with immediate feedback showing the bounding box and identity information overlaid on the video feed.

Environmental Adaptability: The system functions in standard classroom environments with regular lighting conditions.

Confidence Metrics: The percentage-based confidence score provides transparency about the reliability of each recognition event.

Processing Efficiency: The hybrid approach (browser-based detection with server-based verification) optimizes performance by distributing computational load.

6.2.3 System Responsiveness

The system excels in performance by providing real-time processing, ensuring attendance is marked instantly upon successful facial recognition, while synchronizing data promptly to reflect updated attendance records in the reporting interface. It efficiently manages class session creation, activation, and completion, updating statuses in real time. Additionally, the system generates attendance reports on-demand with minimal processing delay, ensuring smooth and timely access to necessary information.

6.2.4 User Feedback and Improvements

User feedback during implementation and testing highlighted several key improvements in user experience. Users appreciated the immediate visual confirmation during facial recognition, with bounding boxes and name displays providing clear feedback. The centralized dashboard design was praised for reducing navigation complexity, especially for teachers managing multiple classes. Administrative staff valued the ability to easily download attendance reports for individual sessions, and the straightforward scheduling interface simplified the process of creating new class sessions.

6.3 Test Cases and Evaluation

6.3.1 Functional Testing

The following key test cases were executed to verify system functionality:

Table 5 Functional testing

Test Case	Description	Expected Result	Actual Result	Status
TC-001	User login authentication	Successful login redirects to appropriate dashboard	Teacher successfully redirected to Teacher Dashboard	PASS
TC-002	Class scheduling	New class appears in upcoming classes list	Software Engineering class successfully scheduled	PASS
TC-003	Face recognition accuracy	System identifies registered student with >50% confidence	David Wambua identified with 57% confidence	PASS
TC-004	Attendance recording	System marks student present in active session	Student marked present in attendance record	PASS
TC-005	Report generation	Download button provides attendance report	Report successfully downloaded	PASS
TC-006	Multiple student recognition	System identifies all present students in a session	2/2 students recognized in Software Engineering class	PASS
TC-007	Student registration	New student appears in recently registered list	Bingo Fire appears in newly registered students	PASS

6.3.2 Non-Functional Testing

In addition to functional testing, several non-functional aspects of the system were evaluated to assess performance, security, and user satisfaction:

Performance Testing: The system demonstrated real-time attendance marking without significant delays. Facial recognition processing took approximately 1-2 seconds per student, even with multiple students in the session.

Security Testing**:** The authentication process was tested using multiple scenarios (incorrect credentials, expired sessions, unauthorized access), and it was found to securely prevent unauthorized access while handling session timeouts appropriately.

Usability Testing: The system's interface was tested for ease of use. Feedback indicated that both teachers and administrators found the user interface intuitive and easy to navigate. The color-coded status on the dashboard and the quick action buttons were particularly appreciated for streamlining common tasks.

Scalability Testing: The system was tested with larger groups of students (up to 100 registered students). The facial recognition process was able to handle this volume without significant degradation in performance, confirming that the system can scale for larger classes and institutions.

6.4 Results and Conclusion

Based on the testing performed, the Face Attendance System successfully meets the functional and non-functional requirements. The core functionalities of facial recognition for attendance marking, real-time updates, reporting, and class scheduling work as expected. User feedback highlights the system's efficiency and ease of use, while the performance testing confirmed that it can handle typical classroom settings.

Moreover, the integration of facial recognition ensures more accurate and automated attendance tracking compared to traditional methods. The system also offers valuable analytics through its reporting features, providing both historical attendance records and real-time status.

6.4.1 Areas for Improvement

Although the system performs well overall, there are areas where improvements could enhance its efficiency. One area for improvement is optimizing the facial recognition system for varying lighting conditions, as better detection accuracy in less-than-ideal environments would enhance performance. Additionally, more robust error handling mechanisms could be added, such as notifying users when a student’s face cannot be recognized or when the camera feed is unclear, improving the overall user experience. Some users also found it challenging to adjust to the system’s setup, particularly the facial recognition process, suggesting that a more detailed onboarding process would help alleviate this issue

6.4.2 Future Work

Looking forward, future work could involve integrating the system with other Learning Management Systems (LMS) to automate the updating of attendance records across platforms used by the institution. Expanding the system to mobile devices would also improve accessibility, allowing both students and

teachers to interact with the platform more easily, especially for attendance marking and viewing reports. Additionally, incorporating AI-driven analytics could provide deeper insights into student attendance trends, such as patterns in tardiness or absenteeism, and send automated notifications to teachers and administrators for proactive management.

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

The **attendease** represents a transformative advancement in automating attendance tracking within academic institutions. By incorporating cutting-edge technologies such as computer vision, machine learning, and facial recognition, the system addresses key issues inherent in traditional attendance methods—inefficiency, human error, and vulnerability to fraud. The integration of **Django** for backend development, along with modern frontend tools like **HTML**, **CSS (Tailwind)**, and **JavaScript**, ensures the system is both robust and user-friendly. Additionally, the use of **TensorFlow**, **Face_Recognition**, and **Face_API.js** enables accurate and secure facial recognition, enhancing the system's efficiency and reliability.

The system effectively showcases the potential of technology to simplify administrative tasks, enabling educators to focus more on teaching and less on the time-consuming process of manual record-keeping. By automating the attendance tracking process, this project not only saves valuable time but also improves the accuracy and transparency of attendance data in academic institutions.

Lessons Learned

Throughout the project, several key lessons were gained that significantly shaped its development and outcomes. Optimizing face recognition emerged as a critical factor, as model accuracy was heavily influenced by variables such as lighting and positioning. Implementing pre-processing techniques proved essential in enhancing recognition results. Security was another paramount concern, particularly regarding the storage of facial data, which necessitated robust encryption methods and stringent access controls to safeguard sensitive information. Real-time processing demanded the use of efficient algorithms, with optimized database queries and search techniques playing a vital role in reducing latency and ensuring smooth performance. Additionally, adopting an agile development approach facilitated faster iterations, enabling regular testing and feedback loops that markedly improved the system's quality and usability.

In summary, the agile development methodology fostered continuous improvement and adaptability, allowing the team to respond effectively to evolving requirements. The project leveraged core technologies, including Django, TensorFlow, Face_API.js, and PostgreSQL, to build a robust system. Key practices such as unit testing, stringent security measures, and optimization techniques were implemented to ensure reliability and performance. By addressing major challenges with innovative solutions, the project achieved its objectives while laying a strong foundation for future enhancements.

7.2 Recommendations

Scalability

One of the key recommendations for the system is to optimize it for large-scale deployment. As the system continues to grow, it is important to ensure that it can efficiently manage a significant number of students and simultaneous users without compromising performance. This would involve enhancing the infrastructure to handle increased demand, ensuring smooth operation even as the number of users expands.

Data Privacy and Security

Given that the system handles sensitive biometric data, it is essential to focus on robust data privacy and security measures. Implementing strong encryption methods and adhering to relevant data protection regulations, such as GDPR, will help safeguard student information. Ensuring that the system complies with legal requirements will also increase user trust and promote the responsible use of biometric data.

Integration with Existing Systems

Future versions of the system could benefit from integrating with existing Learning Management Systems (LMS) or student databases. This integration would streamline data flow, reduce redundancy, and ensure consistency across platforms. By connecting the attendance system with other institutional tools, administrative workflows can be significantly improved, and the management of student records can be more seamless.

Improved Accuracy

Although the system currently performs reliable facial recognition, there is room for improvement in its accuracy. Further research and development should be focused on enhancing recognition precision, particularly in challenging environments, such as poor lighting conditions or when faces are partially obscured. Improving the system’s robustness in these scenarios would ensure higher reliability and greater user satisfaction.

Mobile Application

Developing a mobile application for the system could greatly enhance accessibility and user experience. A dedicated mobile app would allow both teachers and students to interact with the system on the go, offering greater flexibility and convenience. This would be particularly beneficial for attendance marking and report viewing, providing a more versatile platform that users can access anytime, anywhere.

User Training and Support

To ensure the smooth adoption and effective use of the system, educational institutions should provide comprehensive training for staff. This training would help users understand the system's features and

functionalities, minimizing errors and maximizing efficiency. Additionally, setting up a dedicated support team to resolve technical issues and assist users would be crucial in maintaining system stability and user satisfaction.

Ethical Considerations

Finally, institutions must establish clear policies to govern the ethical use of facial recognition technology. This includes ensuring transparency in how the technology is used, securing informed consent from students, and addressing any privacy concerns that may arise.

CHAPTER 8: REFERENCES

1) Ahuja, K., & Bedi, P. (2023). "Deep learning approaches for facial recognition systems: A comprehensive review." *IEEE Access*, 11, 34562-34583.

2) Brownlee, J. (2023). *Deep Learning for Computer Vision*. Machine Learning Mastery.

3) Chen, L., & Wang, Y. (2022). "Privacy preservation in biometric systems: Challenges and solutions." *Journal of Information Security*, 13(2), 89-104.

4) Django Software Foundation. (2024). "Django Documentation." Retrieved from https://docs.djangoproject.com/

5) Geitgey, A. (2022). "Face Recognition Documentation." Retrieved from https://github.com/ageitgey/face_recognition

6) Google. (2024). "TensorFlow Documentation." Retrieved from https://www.tensorflow.org/api_docs

7) Khan, Z. H., & Ali, T. (2023). "Facial recognition-based attendance systems in educational institutions: Privacy and security concerns." *International Journal of Educational Technology in Higher Education*, 20(1), 1-18.

8) Nguyen, D. T., & Kang, J. K. (2022). "Deep learning-based facial recognition attendance systems: Implementation challenges in university settings." *Education and Information Technologies*, 27(3), 3891-3910.

9) Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). "Joint face detection and alignment using multitask cascaded convolutional networks." *IEEE Signal Processing Letters*, 23(10), 1499-1503.

10) Schroff, F., Kalenichenko, D., & Philbin, J. (2015). "FaceNet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815-823.

CHAPTER 9: APPENDICES

Appendix A: System Architecture Diagram

A visual representation of the system's architecture, detailing the interaction between the frontend, backend, and facial recognition components. This diagram illustrates how the user interface communicates with the server-side logic and how the facial recognition algorithms are integrated into the system.

Appendix B: User Manual

A step-by-step guide designed for teachers and administrators on how to use the system. This manual includes instructions on:

Registration: Enrolling students into the system.

Attendance Tracking: How to mark attendance and monitor real-time data.

Database Management: Managing student records and system settings.

Appendix C: Source Code

A detailed repository of the project's source code, broken down into sections for:

Backend: Code written using Django to handle server-side logic and database interactions.

Frontend: HTML, CSS (Tailwind), and JavaScript to create the user interface.

Facial Recognition: Scripts using TensorFlow, Face_Recognition, and Face_API.js to perform the facial detection and recognition tasks.

Appendix D: Testing and Validation Results

Documentation covering the testing process and validation of the system, including:

Accuracy Metrics: The performance of facial recognition in various environments.

Performance Benchmarks: System performance under different load conditions (e.g., number of concurrent users).

User Feedback: Feedback from users (teachers, administrators, and students) during testing phases.

Appendix E: Ethical Considerations and Consent Forms

Sample consent forms and ethical guidelines for the use of facial recognition technology in academic institutions, ensuring the system complies with privacy laws and addresses concerns related to data security and informed consent.

Appendix F: Future Work Proposal

A detailed proposal outlining potential future developments of the system, including:

Scalability Improvements: Enhancing the system to support larger institutions.

Mobile Application Development: Creating a mobile version for greater accessibility.

Integration with LMS: Integrating the system with existing Learning Management Systems (LMS) for streamlined operations.