

MULTIMEDIA UNIVERSITY OF KENYA

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY

**A Student Logging System for University Gates Using Face and Speech Recognition**

**BY**

**MUA EMMANUEL MWONGELA**

**CIT-227-011/2021**

**SUPERVISOR: NICHOLAS NJIRU**

**MAY,2025**

Submitted in partial fulfillment of the requirements of Fourth Year Bachelor of Science in Software Engineering

**This Page was left Blank intentionally.**

# DECLARATION

I hereby declare that this Project is my own work and has, to the best of my knowledge, not been submitted to any other institution of higher learning.

**Student Name: Mua Emmanuel Mwongela Registration Number: CIT-227-011/2021**

**Signature: ........................~~emua~~.............. Date: .............................................**

This project has been submitted as a partial fulfillment of requirements for the Bachelor of Science in Software Engineering of Multimedia University of Kenya with my approval as the University supervisor.

**Supervisor: NICHOLAS NJIRU**

**Signature: ............................................. Date: .........................................**

# ACKNOWLEDGMENT

I would like to first express my sincere gratitude to the Almighty God for the strength, wisdom, and grace granted to me throughout this project. Without His guidance and provision, the successful completion of this endeavor would not have been possible.

I extend heartfelt appreciation to my supervisor, Mr.Njiru, for his valuable insights, patience, and continuous support. His mentorship guided me through key decisions and his encouragement reminded me that even when time is limited, perseverance and resourcefulness can still deliver success. I am also deeply grateful to Mr. Jonathan Mua and his family for their unwavering moral and financial support during this project. Lastly, special thanks to the students and friends from Multimedia University whose collaboration during research and peer contributions greatly enhanced the development of this system.

# ABSTRACT

Conventional university gate logging systems often rely on manual logbooks or ID card swipes, which introduce inefficiencies, risks of impersonation and human error. This project introduces an AI-powered recognition system that automates student gate logging through face recognition and speech command processing.

The system utilizes Convolutional Neural Networks (CNNs) for facial feature encoding and Natural Language Processing (NLP) for interpreting voice input. It integrates securely with the university database to verify student identity in real time, log their access status (login/logout) and send class schedules via email.

The system was evaluated in a simulated university gate environment using real student face images across varied lighting and angles. It achieved a face recognition accuracy of 96.2%, with an average recognition time of 5.8 seconds per student. The email delivery success rate was 100% using Gmail’s secure app password method. Additionally, the schedule-matching module correctly identified and notified students of their course timetable with 92% precision, based on parsed university timetables ensuring 100% attendance tracking through logs.

The proposed solution not only enhances security but also optimizes administrative workflows, provides intelligent notification services, and ensures compliance with modern data protection standards. Evaluation of the system demonstrated strong recognition accuracy under varied lighting conditions, affirming its practical viability in university environments.

# Table of Contents

[DECLARATION 2](#_Toc199099927)

[ACKNOWLEDGMENT 3](#_Toc199099928)

[ABSTRACT 4](#_Toc199099929)

[Table of Contents 5](#_Toc199099930)

[List of Abbreviations 8](#_Toc199099931)

[List of Figures 9](#_Toc199099932)

[CHAPTER 1. INTRODUCTION 10](#_Toc199099933)

[1.1 Background of Study 10](#_Toc199099934)

[1.2 Problem Statement 10](#_Toc199099935)

[1.2.1 Proposed solution 11](#_Toc199099936)

[1.3 Aim of The Study 11](#_Toc199099937)

[1.3.1 Research Objectives 11](#_Toc199099938)

[1.4 Significance of The Study 12](#_Toc199099939)

[1.5 Scope of The Project 12](#_Toc199099940)

[1.6 Assumptions 12](#_Toc199099941)

[1.7 Limitations 13](#_Toc199099942)

[CHAPTER 2. Literature Review 14](#_Toc199099943)

[2.1 Introduction 14](#_Toc199099944)

[2.2 Related Systems 15](#_Toc199099945)

[2.3 Weaknesses of These Systems 16](#_Toc199099946)

[2.4 How the Proposed System Handles These Weaknesses 16](#_Toc199099947)

[CHAPTER 3: Methodology 17](#_Toc199099948)

[3.1 Introduction 17](#_Toc199099949)

[3.2 Methodology 17](#_Toc199099950)

[3.2.1 Agile Justification 17](#_Toc199099951)

[3.3 Deep Learning and Recognition Techniques 18](#_Toc199099952)

[3.3.1 Convolutional Neural Networks (CNNs) 18](#_Toc199099953)

[3.3.2 Transfer Learning with pre-trained Models. 18](#_Toc199099954)

[3.3.3 Support Vector Machines (SVM) (planned integration) 18](#_Toc199099955)

[3.4 Data Collection and Preprocessing 18](#_Toc199099956)

[3.4.1 Data Collection Methods 18](#_Toc199099957)

[3.4.2 Dataset Creation and Cleaning 19](#_Toc199099958)

[3.5 Tools, Libraries and Technologies 19](#_Toc199099959)

[3.6 Logging and Status Determination 20](#_Toc199099960)

[CHAPTER 4: System Analysis 21](#_Toc199099961)

[4.1 Detailed Analysis of the Current System 21](#_Toc199099962)

[4.1.1 Use Case Diagram 21](#_Toc199099963)

[4.1.2 Flowchart 22](#_Toc199099964)

[4.1.3 Class Diagram 23](#_Toc199099965)

[4.1.4 Activity Diagrams 24](#_Toc199099966)

[4.1.5 Data Flow Diagrams 25](#_Toc199099967)

[4.1.6 Context Diagram 26](#_Toc199099968)

[4.2 System Requirements 26](#_Toc199099969)

[4.2.1 Functional requirements 26](#_Toc199099970)

[4.2.2 Non-Functional Requirements 27](#_Toc199099971)

[CHAPTER 5: System Design 29](#_Toc199099972)

[5.1 Introduction 29](#_Toc199099973)

[5.2 Architectural Design 29](#_Toc199099974)

[5.2.1 Client Layer (Presentation Layer) 29](#_Toc199099975)

[5.2.2 Application Layer (Business Logic Layer) 30](#_Toc199099976)

[5.2.3 Data Layer (Persistence Layer) 30](#_Toc199099977)

[5.3 Design Methodologies 30](#_Toc199099978)

[5.3.1 Modular Design 30](#_Toc199099979)

[5.3.2 Event-Driven Design 30](#_Toc199099980)

[5.3.3 Reusability 30](#_Toc199099981)

[5.4 Database Design 31](#_Toc199099982)

[5.4.1 students Table 31](#_Toc199099983)

[5.4.2 attendance Table 31](#_Toc199099984)

[5.4.3 Encodings (Pickled File) 31](#_Toc199099985)

[5.5 User Interface Design 33](#_Toc199099986)

[5.5.1 Main Dashboard 33](#_Toc199099987)

[5.5.2 Registration Form 34](#_Toc199099988)

[5.5.3 Notification & Logging 34](#_Toc199099989)

[5.5.4 Accessibility and Responsiveness 34](#_Toc199099990)

[CHAPTER 6: Implementation and Testing 35](#_Toc199099991)

[6.1 Introduction 35](#_Toc199099992)

[6.2 Development Environment 35](#_Toc199099993)

[6.3 System Components 35](#_Toc199099994)

[6.3.1 Face Recognition Module 35](#_Toc199099995)

[6.3.2 Student Registration Module 36](#_Toc199099996)

[6.3.3 Training Module 36](#_Toc199099997)

[6.3.4 Attendance Logging 36](#_Toc199099998)

[6.3.5 Timetable Parser 36](#_Toc199099999)

[6.3.6 Email Notification Module 36](#_Toc199100000)

[6.3.7 Report Generator 36](#_Toc199100001)

[6.4 Test Data 36](#_Toc199100002)

[6.4.1 Registered Students 37](#_Toc199100003)

[6.4.2 Sample Timetable Data 37](#_Toc199100004)

[6.5 Test Cases and Results 37](#_Toc199100005)

[6.5.1 Test Case 1: Student Face Recognition 37](#_Toc199100006)

[6.5.2 Test Case 2: Email Notification 38](#_Toc199100007)

[6.5.3 Test Case 3: Upload and Parse Timetable 38](#_Toc199100008)

[6.5.4 Test Case 4: Register New Student 38](#_Toc199100009)

[6.5.5 Test Case 5: Generate Report 38](#_Toc199100010)

[6.6 Summary 39](#_Toc199100011)

[CHAPTER 7: Conclusion 40](#_Toc199100012)

[7.1 Achievements 40](#_Toc199100013)

[7.2 Lessons Learned 40](#_Toc199100014)

[7.3 Evaluation Metrics and System Testing 41](#_Toc199100015)

[7.4 Conclusion 42](#_Toc199100016)

[7.5 Recommendations 42](#_Toc199100017)

[REFERENCES 44](#_Toc199100018)

[Appendix 46](#_Toc199100019)

# List of Abbreviations

|  |  |
| --- | --- |
| **AI** | Artificial Intelligence |
| **CNN** | Convolutional Neural Network |
| **ML** | Machine Learning |
| **NLP** | Natural Language Processing |
| **API** | Application Programming Interface |
| **SVM** | Support Vector Machine |
| **UI** | User Interface |
| **GUI** | Graphical User Interface |
| **UML** | Unified Modeling Language |
| **DFD** | Data Flow Diagram |
| **DB** | Database |
| **ID** | Identification |
| **FPR** | False Positive Rate |
| **ER Diagram** | Entity-Relationship Diagram |
| **PDF** | Portable Document Format |
| **SMTP** | Simple Mail Transfer Protocol |
| **GDPR** | General Data Protection Regulation |
| **OTP** | One-Time Password |
| **UUID** | Universally Unique Identifier |
| **JSON** | JavaScript Object Notation |
| **SSL** | Secure Sockets Layer |
| **OCR** | Optical Character Recognition |

# List of Figures

Figure 1 Use case Diagram drawn using draw.io 19

Figure 2 Flowchart of the existing system drawn using draw.io 20

Figure 3 Class Diagram drawn using draw.io 21

Figure 4 Activity Diagrams of existing system drawn using draw.io 22

Figure 5 Context- level DFD drawn using draw.io 23

Figure 6 DFD Level-1 of the existing systems drawn using draw.io 24

Figure 7 Context Diagram of existing system drawn using draw.io 24

Figure 8 System GUI 41

Figure 9 Register Student module 42

Figure 10 Email sent to student after login 42

Figure 11 Face recognition and login/out 43

Figure 12 Recent logs displayed 43

Figure 13View student module 44

Figure 14 Filter and report generator 44

Figure 15 Memo upload success confirmation 45

Figure 16 Memo sent to all student emails 45

Figure 17 Successful timetable upload 46

Figure 18 Report generator module 46

# CHAPTER 1. INTRODUCTION

## 1.1 Background of Study

Traditional university gate logging systems have long relied on manual checks or card-based access methods, which are not only time-consuming but also prone to human error and security breaches. These legacy systems often result in long queues at university gates, unauthorized access due to lost or shared cards, and inaccuracies in tracking student attendance. With increasing student populations and the demand for tighter campus security, the limitations of these systems are becoming more evident.

Recent advances in Artificial Intelligence (AI), especially in computer vision and speech processing, offer powerful solutions to these challenges. AI technologies such as facial recognition and voice authentication have demonstrated great potential for automating identity verification and enhancing security in real-time. These tools can reduce dependency on physical cards or manual input, enabling touchless and seamless access control systems.

This project proposes the design and implementation of an AI-driven student logging system that leverages face recognition and voice input to authenticate students as they enter the university premises. By integrating these technologies with the institution’s central database, the system aims to improve accuracy in attendance logging, minimize bottlenecks at entry points and strengthen campus security. The end goal is to contribute to a smarter, more efficient and secure educational environment that aligns with global trends in smart campus initiatives.

## 1.2 Problem Statement

Current university gate entry systems lack a unified and robust authentication mechanism. Manual entry methods are time-consuming, susceptible to manipulation and contribute to congestion during peak hours. Card-based systems, while faster, remain vulnerable to loss, theft or misuse. Moreover, these approaches do not facilitate real-time integration with institutional databases or offer scalability.

The absence of a reliable, secure and automated logging solution compromises both campus security and administrative efficiency. There is an urgent need for an intelligent, contactless system that can accurately and swiftly identify students, verify their credentials and log their entry or exit without introducing complexity or delay.

### 1.2.1 Proposed solution

A real-time logging system that authenticates students via face recognition and voice commands, integrated with the university’s database for instant verification. The proposed system introduces a real-time, AI-powered logging platform capable of authenticating students through a combination of facial recognition and voice input. By integrating with the university’s student records and attendance databases, the system will facilitate immediate verification while ensuring a seamless and intuitive user experience. The multi-modal approach not only enhances accuracy but also supports inclusivity for students with accessibility needs.

## 1.3 Aim of The Study

The main objective of this study is to design, develop and test a secure, efficient and inclusive logging system for university gates using AI-driven face and speech recognition technologies to provide real-time, contactless logging.

### 1.3.1 Research Objectives

To achieve this aim, the following specific objectives are outlined:

a) To design and implement a multi-modal authentication framework integrating facial recognition and voice-based logging.

b) To interface the system with university databases for seamless, real-time student verification and attendance logging.

c) To ensure compliance with international and local data privacy standards, such as the General Data Protection Regulation (GDPR) and Kenya’s Data Protection Act.

d) To evaluate the usability and performance of the system in real-world scenarios, focusing on recognition accuracy, response time, and user satisfaction.

e) To introduce fail-safe strategies for handling edge cases, such as poor lighting conditions, low-quality audio, or unregistered individuals.

## 1.4 Significance of The Study

This study holds substantial value for both academic institutions and the broader technology ecosystem. For universities, the system presents a scalable solution that enhances campus security, minimizes administrative overhead, and automates attendance tracking. It reduces the need for manual interventions and provides reliable, auditable logs of student movements.

From a technological standpoint, this research contributes to the growing body of knowledge on multi-modal biometric systems and their practical deployment. It also demonstrates the viability of applying AI in educational settings, encouraging further exploration into smart campus initiatives that align with global digital transformation trends.

## 1.5 Scope of The Project

The system developed in this project encompasses the following core functionalities:

* Construction of a facial recognition engine using Convolutional Neural Networks (CNN) for student authentication.
* Integration of speech recognition to offer audio-based interaction and feedback mechanisms.
* Real-time database synchronization for logging and verifying student data with institutional records.
* Secure, privacy-conscious data storage and processing protocols to safeguard biometric information.
* Development of an intuitive user interface tailored for system administrators and student interaction points.

## 1.6 Assumptions

In the course of developing and deploying the system, the following assumptions are made:

a) Access to necessary hardware and software resources for prototyping, including computers, cameras, microphones, and development platforms.

b) Willing participation of students and university staff in system testing and feedback sessions.

c) Availability of guidance from domain experts in AI ethics, biometric technologies, and data privacy to ensure the system adheres to best practices.

d) Sufficient dataset availability to train facial and speech recognition models with diverse samples representative of the student population.

## 1.7 Limitations

Despite its potential, the project is subject to the following limitations:

a) The system’s performance may be impacted under extreme lighting conditions or in high-noise environments, potentially affecting recognition accuracy.

b) Dependency on third-party libraries and APIs (e.g., for speech-to-text processing) may introduce constraints on functionality and deployment flexibility.

c) Ethical and legal concerns related to the storage and usage of biometric data must be carefully addressed to avoid misuse or breach of regulations.

d) Regulatory barriers may limit the full deployment of the system in contexts with strict data protection policies, requiring continuous evaluation and compliance.

# CHAPTER 2. Literature Review

## 2.1 Introduction

Biometric systems have become increasingly prevalent in educational institutions, particularly for attendance and access control. Common implementations include fingerprint scanning and Radio Frequency Identification (RFID) systems. While these technologies offer improvements over manual logging, they often lack robustness, contactless functionality, and adaptability in real-world academic settings. For instance, at Multimedia University of Kenya, students often face delays and inconveniences during manual gate logging, particularly during examination periods. In cases where students forget or lose their identification cards, access becomes a challenge. Moreover, traditional systems struggle to detect identity fraud, such as when a student impersonates another during exam invigilation.

Recent advancements in Artificial Intelligence (AI), especially in computer vision and speech processing, offer promising alternatives. Deep learning technologies such as Convolutional Neural Networks (CNNs) and Natural Language Processing (NLP) have enabled systems to perceive and interact similarly to human beings—an evolution that has profound implications for identity verification. Deep learning mimics neural processes in the brain through multilayered networks capable of identifying patterns and learning from vast datasets (Brownlee, 2020; Graupe, 2016).

The inspiration for this project partially stems from the 2023 science fiction film *The Creator*, which depicts AI agents with advanced perception and emotional intelligence. The concept that an AI could "see" and "feel" like a human inspired this project's exploration into multi-modal recognition—specifically integrating facial and speech recognition to emulate real-world cognition and identification.

Jain et al. (2023) demonstrated how deep learning and voice assistance technologies could aid visually impaired individuals in object detection, providing real-time identification of obstacles and hazards. Similarly, accessibility-centered innovations in China, such as the "Cinema of Light" initiative, showcase national-level efforts to use AI for inclusion and accessibility (China Foundation for Human Rights Development & National High-Level Think Tank of Xinhua News Agency, 2024).

These developments validate the use of AI in systems designed for high-security, high-accuracy applications such as student logging. Technologies like speech recognition, facial authentication and behavioral pattern learnin**g** can now be realistically integrated into university infrastructure, promoting not just automation but also personalization, inclusivity, and security.

## 2.2 Related Systems

**Biometric Attendance Systems**

Traditional biometric attendance systems in schools typically utilize RFID cards and fingerprint sensors. While these systems help automate attendance, they are prone to spoofing and cannot handle edge cases like physical disability or absence of a card. Furthermore, their dependence on physical contact reduces usability in post-pandemic environments that emphasize contactless interaction.

**AI in Campus Security**

Several universities globally have adopted AI for campus safety. For example, MIT and Stanford University have implemented facial recognition systems for secure dormitory access and exam monitoring. These systems allow for rapid identity checks without the need for physical ID cards, enhancing both efficiency and integrity (MIT News, 2021).

**Voice-Activated Systems**

Voice assistants such as Amazon Alexa and Google Assistant are now being trialed in educational environments for campus navigation and information access. Institutions like Arizona State University have adopted Alexa for classroom integration and student queries, showing how NLP and voice systems can enhance campus interactivity (Amazon, 2021).

**User-Centered Design and Usability Testing**

Designing intuitive interfaces is key for accessibility. User-centered design methodologies and usability testing ensure that the needs of various user groups—especially those with disabilities—are considered. While highly effective in improving usability, variability in user preferences and digital literacy can still pose challenges (Wong et al., 2020).

**Image Recognition and Document Scanning**

Image recognition facilitates the interpretation of visual materials such as banknotes, ID cards, and exam slips. Through AI-enabled smartphone cameras or dedicated scanning devices, users can digitize and analyze documents with high accuracy. Machine learning models improve recognition by continuously adapting to diverse use cases (Zhou & Lin, 2021).

**Virtual Assistants and Personalized Guidance**

AI-powered virtual agents provide real-time, conversational support. These systems use predictive analytics and natural language understanding to anticipate user needs and personalize interactions. In the context of higher education, they can be deployed to offer course reminders, campus navigation, and learning support. For students with visual impairments, these agents offer a conversational alternative to graphical interfaces, simplifying access to academic services (Chen et al., 2020).

## 2.3 Weaknesses of These Systems

1. Single-factor authentication risks.
2. High false rejection rates in diverse environments.
3. Privacy concerns with biometric data.

## 2.4 How the Proposed System Handles These Weaknesses

1. Multi-modal authentication (face + voice) improves security.
2. CNN models trained on diverse datasets reduce false negatives.
3. Data anonymization and encryption ensure privacy compliance.

# CHAPTER 3: Methodology

## 3.1 Introduction

This chapter outlines the methodology adopted in the development of the AI based student face recognition and logging system. The methodology section details the steps, technologies, and tools used to design, build, train and deploy the system. The project leverages the Agile software development model, which enables iterative prototyping and stakeholder feedback to guide feature refinement. Given that the project deals with biometric authentication and logging, a combination of deep learning algorithms, image processing, and database systems was utilized. Both qualitative and quantitative research methods were employed to identify system requirements and challenges in the current manual entry systems used at institutional gates

## 3.2 Methodology

My Project requires extensive research on different technologies and different stakeholders The system was built using an Agile iterative model, with each sprint focused on a specific module: registration, image acquisition, face detection and recognition, dataset training and attendance logging. Stakeholder input, especially from gate security personnel and students, played a central role in refining features such as real-time recognition, automatic status updates (login/logout) and error handling for unrecognized individuals.

### 3.2.1 Agile Justification

- Flexibility: The model allowed adjustments after feedback from early prototype testing.  
- Rapid Prototyping: Functionality such as real-time face tracking and registration through camera could be tested incrementally.  
- Stakeholder Collaboration: Regular feedback from users ensured usability and relevance.  
- Testing-centric: Each sprint involved rigorous testing, especially during model training and real-time face matching.

## 3.3 Deep Learning and Recognition Techniques

### 3.3.1 Convolutional Neural Networks (CNNs)

CNNs were utilized to encode facial features into high-dimensional vectors (embeddings). These embeddings are then stored and compared against live images during recognition. This technique ensures faster and more accurate results compared to raw pixel comparison. The encodings were stored in a pickle file which was retrained over time or after a student is registered to save their encodings for detection in the future.

### 3.3.2 Transfer Learning with pre-trained Models.

Instead of training a CNN from scratch, the project leverages FaceNet and dlib’s pre-trained models, which significantly reduces computation cost and improves accuracy. These models have been trained on large-scale face datasets and are known for their robustness across diverse conditions such as lighting, orientation, and expression.

### 3.3.3 Support Vector Machines (SVM) (planned integration)

Although not the primary method used in the current system, SVM was considered for final classification stages where embeddings would be clustered and matched. In future versions, an SVM classifier can be layered over the embeddings for better generalization across large datasets.

## 3.4 Data Collection and Preprocessing

### 3.4.1 Data Collection Methods

* + Surveys and Interviews: Using tools like Google Forms, interviews with gate security officers and students were conducted to understand the limitations of current manual logging methods. The goal was to identify pain points such as time delays, impersonation, and paper record mismanagement.
  + Camera Capture During Registration: Each student was required to provide at least four facial images taken under varying conditions (angle, lighting, expression) using a webcam. This ensured diversity and improved recognition accuracy.
  + Image Uploads: Users were given an alternative option to upload existing images which were validated before being accepted into the dataset.

### 3.4.2 Dataset Creation and Cleaning

Once images were acquired, the following preprocessing steps were implemented:

* + Validation: Images without faces or containing multiple faces were discarded.
  + Resizing: All images were resized to 160x160 pixels to match the expected input size of the recognition model.
  + Standardization: Filenames were cleaned and organized in student-specific directories to support batch processing.
  + Encoding Generation: The model processed images into vector embeddings, which were stored in a serialized pickle file.

## 3.5 Tools, Libraries and Technologies

The development of this system was facilitated using a suite of software tools and frameworks:

|  |  |  |
| --- | --- | --- |
| Component | Tool/Library | Description |
| Face Detection & Recognition | Dlib, face\_recognition, FaceNet | Used to detect faces and convert them into embeddings |
| Image Handling | Opencv, Pillow | Used to capture, display and save webcam images |
| GUI development | Tkinter | Provides a user interface for registration, logging and real time detections |
| Model serialization | Pickle | Used to store and retrieve face encodings |
| Database Management | SQLite3 | Used to store registered students and attendance logs |
| Export & Reporting | ReportLab, CSV, PDF | Enables exporting logs and student records |
| Dataset Management | Python scripts | Handles structured image folders per student |

## 3.6 Logging and Status Determination

The system determines whether a student is logging in or out based on their last recorded action in the attendance table. This is done via a status flip:

* + If last status was logout, the current one becomes login.
  + If last status was login, the system logs logout.

All logs are timestamped and stored persistently in the SQLite database, ensuring auditability and compliance with security policies

# CHAPTER 4: System Analysis

## 4.1 Detailed Analysis of the Current System

The models in this document I generated using draw.io. the included models are unified modelling language (UML), Class diagram, Data Flow Diagram (DFD), Flowchart, Use Cases and context Diagram for the current student logging systems.

### 4.1.1 Use Case Diagram

The existing logging and attendance manual and biometric systems for logging students and tracking students are as follows; where the gatekeeper has a manual register/logbook to track attendance and students are frisked at the gate they’re identified using their student ID at the gate to log at the university gate just like any other login and registration system.

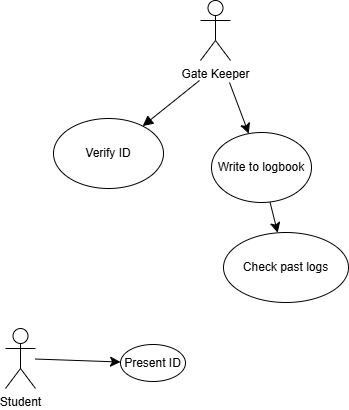


Figure 1 Use case Diagram drawn using draw.io

### 4.1.2 Flowchart

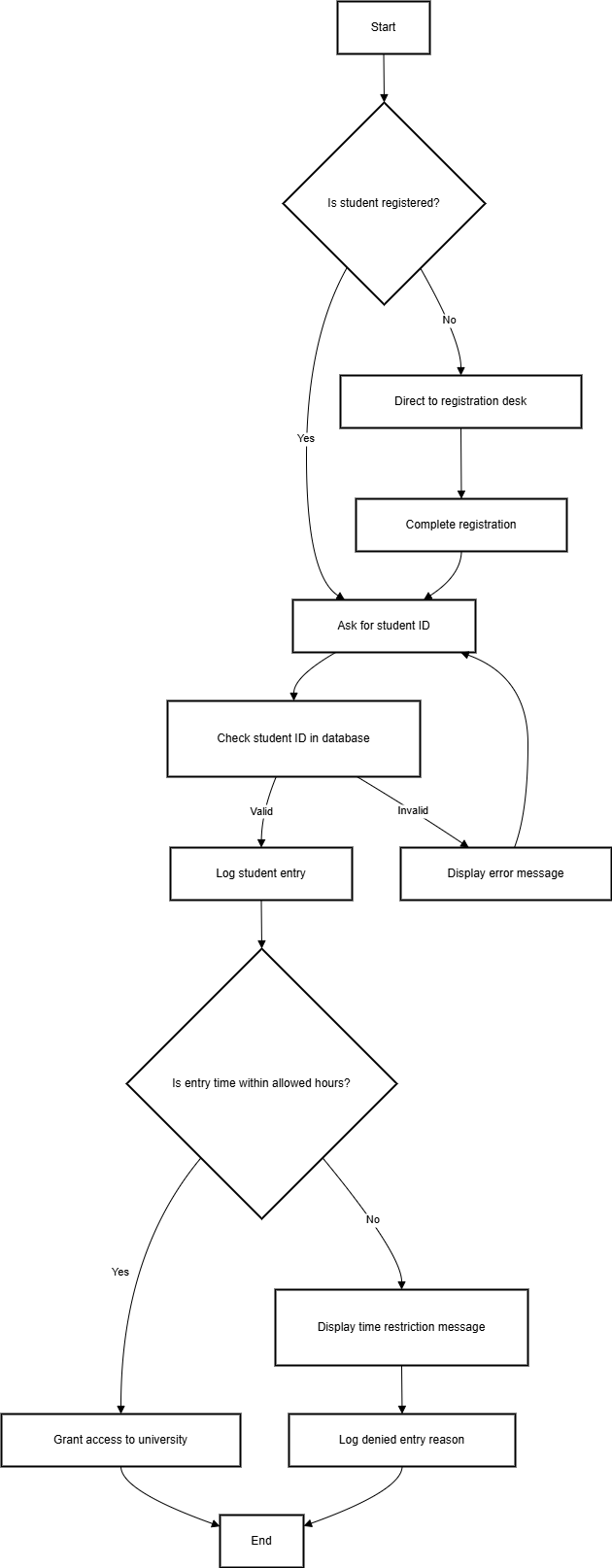


Figure 2 Flowchart of the existing system drawn using draw.io

### 4.1.3 Class Diagram

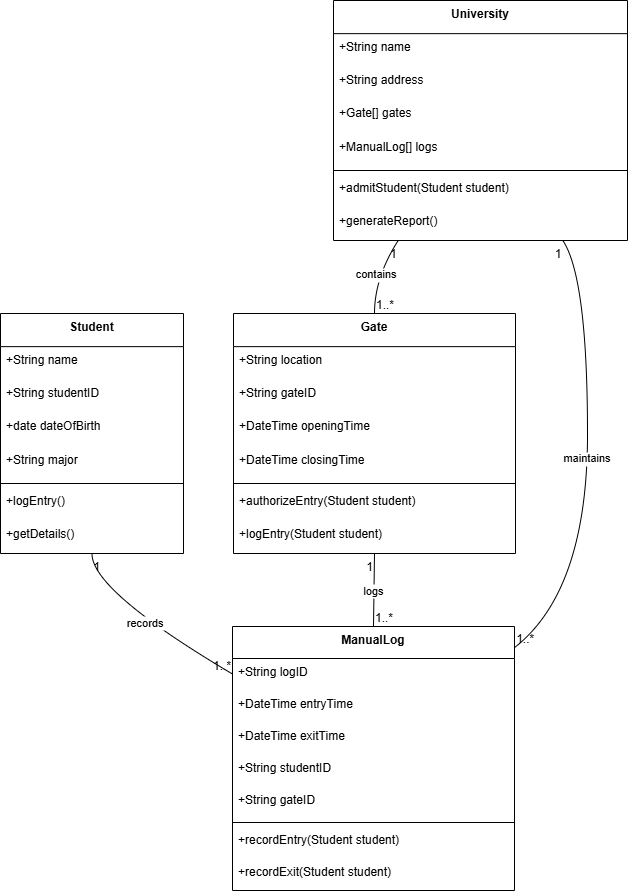


Figure 3 Class Diagram drawn using draw.io

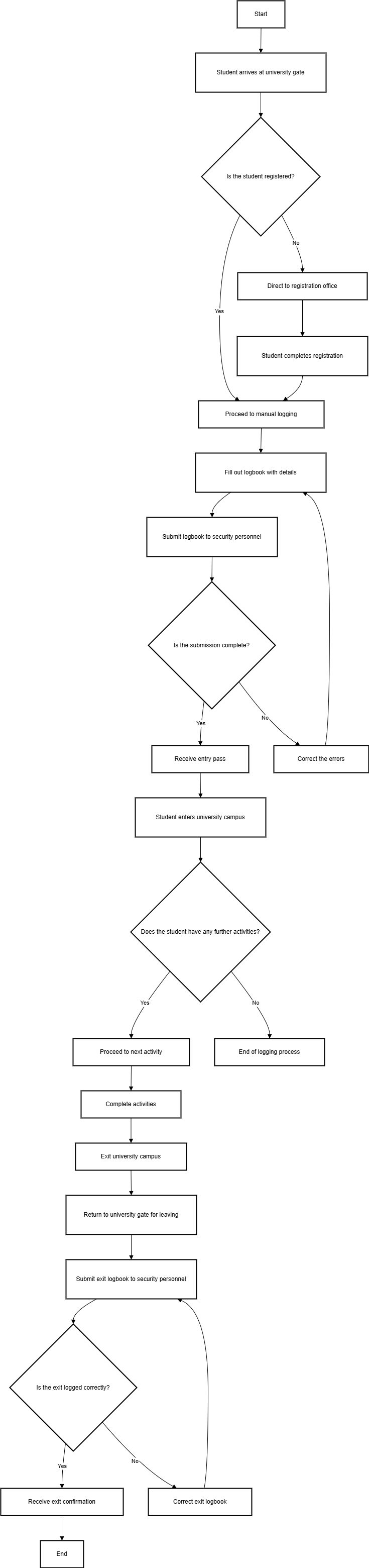
4.1.4 Activity Diagrams  


Figure 4 Activity Diagrams of existing system drawn using draw.io

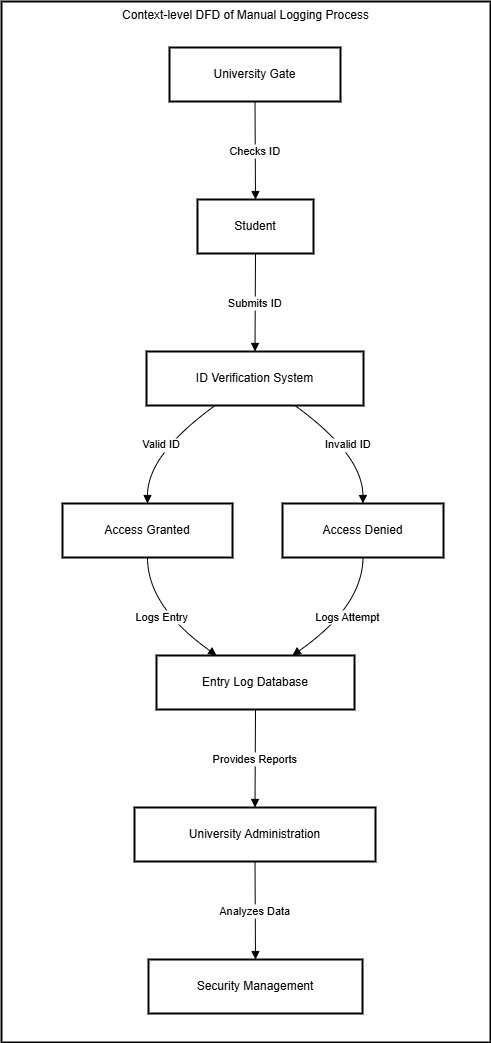
4.1.5 Data Flow Diagrams

Figure 5 Context- level DFD drawn using draw.io

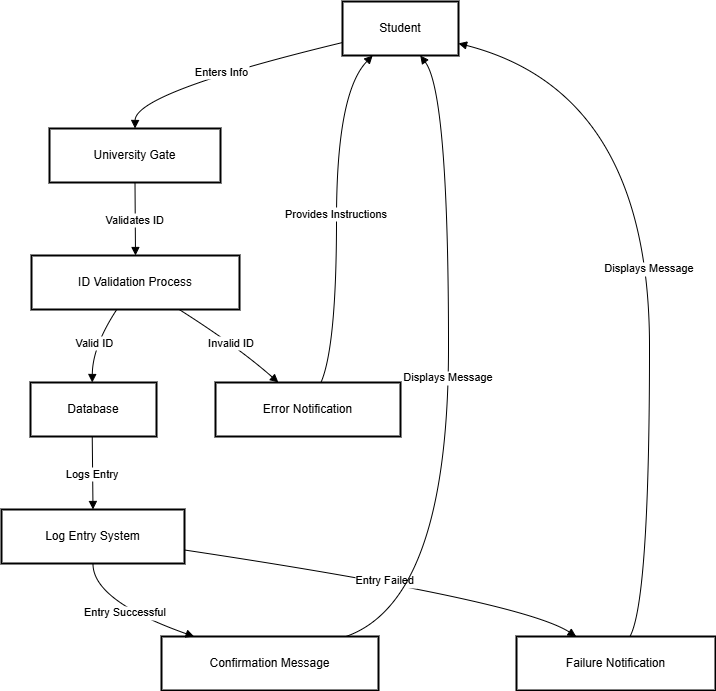


Figure 6 DFD Level-1 of the existing systems drawn using draw.io

# 4.1.6 Context Diagram



Figure 7 Context Diagram of existing system drawn using draw.io

# 4.2 System Requirements

### 4.2.1 Functional requirements

Below are the functional requirements of the proposed system application.

1. The system shall allow an administrator to register a new student using facial images, email, student ID, full name, and course.
2. The system shall allow students to log in or out by recognizing their face via the camera.
3. The system shall automatically determine and log attendance (login/logout) with a timestamp.
4. The system shall allow the admin to view, filter, and delete student records.
5. The system shall allow filtering of attendance logs by student ID and/or date.
6. The system shall generate a PDF report of a student’s login history on request.
7. The system shall support uploading and parsing PDF timetables for multiple courses.
8. The system shall email students their schedule upon login using their registered email address.
9. The system shall allow uploading of university memos (PDFs) and automatically email them to all registered students.
10. The system shall train facial recognition models from uploaded or captured images.
11. The system shall persist all student and attendance data in a local SQLite database.
12. The system shall prevent duplicate student registrations using student ID constraints.

### 4.2.2 Non-Functional Requirements

1. Biometric data (face encodings) must be stored securely using serialization techniques such as encrypted pickle files.
2. The system shall support offline operation, with no dependency on internet connectivity for facial recognition.
3. All system emails (e.g., schedules, memos) shall be sent via secure SMTP connections using Gmail app passwords (TLS/SSL encryption).
4. The graphical user interface (GUI) shall be intuitive, user-friendly, and responsive, allowing easy operation by non-technical users.
5. The application shall be scalable to handle the registration and recognition of 1000+ students without performance degradation.
6. The system shall ensure high reliability and fault tolerance, recovering gracefully from hardware issues (e.g., camera failure) or database errors.
7. It must adhere to data protection and privacy laws, such as the General Data Protection Regulation (GDPR) and Kenya Data Protection Act, by minimizing sensitive data storage and avoiding facial image exposure.
8. The software shall be platform-compatible, running on Windows 10 or higher with Python 3.11+, requiring only standard system resources and a basic webcam.
9. System logs and events must maintain high integrity and auditability, with timestamped entries and secure local storage.

# CHAPTER 5: System Design

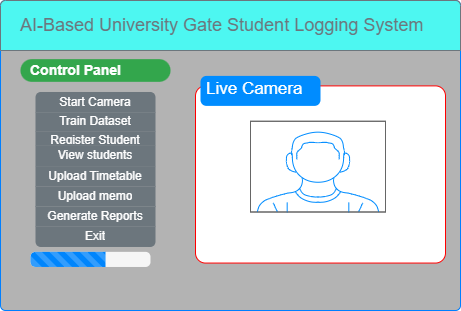
## 5.1 Introduction

This chapter presents the system design of the AI-Based Student Recognition Login System. It translates the functional and non-functional requirements into a tangible and structured blueprint that guides implementation. The goal of this design phase is to define the system architecture, database schemas, interface layouts, and the flow of control between components to ensure that the final system is robust, scalable, and meets the user's expectations. The designs outlined in this chapter serve as the foundation for development, testing, and deployment.

## 5.2 Architectural Design

The architectural design of the system follows a modular, layered architecture that separates concerns across presentation, application logic and data persistence layers. The key components of the architecture include:

### 5.2.1 Client Layer (Presentation Layer)

* Built using Tkinter, this layer is responsible for rendering the Graphical User Interface (GUI).
* It allows users (e.g., security personnel) to interact with the system by registering students, viewing logs, starting the camera, and uploading memos or timetables.  
  

### 5.2.2 Application Layer (Business Logic Layer)

* This layer includes modules that handle facial recognition, training, schedule parsing, PDF report generation, and email notifications.
* It manages real-time face detection using Dlib and face\_recognition, handles login logic and communicates with the database and email APIs.

### 5.2.3 Data Layer (Persistence Layer)

* Consists of SQLite database for storing students and their attendance logs.
* Face encodings are stored separately using Pickle serialization to reduce query complexity and file size.
* University memos and timetable PDFs are stored in organized directories (/memos, /data, /student\_images).

## 5.3 Design Methodologies

Several design approaches were used to optimize modularity, reusability, and user experience:

### 5.3.1 Modular Design

* Each major functionality (e.g., recognition, registration, logging, reporting) is encapsulated in a separate module or class.
* Promotes maintainability and allows for independent testing or upgrades.

### 5.3.2 Event-Driven Design

* Tkinter GUI components operate on event listeners.
* Actions such as button clicks trigger corresponding event handlers to update the interface or database.

### 5.3.3 Reusability

* Common logic like email sending and schedule checking is abstracted into utility functions for reuse.
* Modularized utils, face\_recognizer and timetable\_parser packages improve structure and reduce redundancy.

## **5.4 Database Design**

The system uses a lightweight **SQLite** database for local persistence, designed for fast read/write access and easy integration with Python. It consists of the following tables:

### 5.4.1 students Table

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| student\_id | TEXT | Unique ID for the student |
| name | TEXT | Full name of the student |
| email | TEXT | Email for schedule/memo alerts |
| course | TEXT | Course code (e.g., bcsy2s2) |

### 5.4.2 attendance Table

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| id | INTEGER | Auto-incremented primary key |
| student\_id | TEXT | ID matching student in students |
| name | TEXT | Name of the student |
| timestamp | TEXT | Date and time of log |
| status | TEXT | Either login or logout |

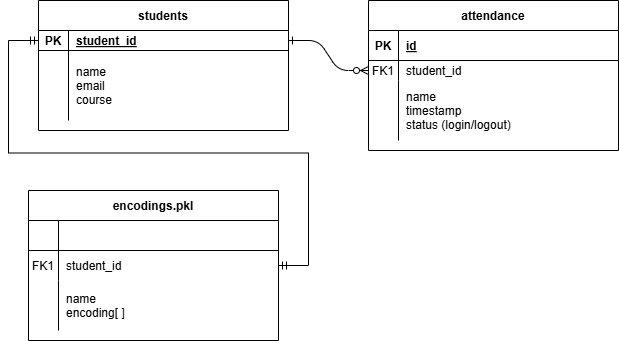
### 5.4.3 Encodings (Pickled File)

Encodings are saved in face\_recognizer/encodings.pkl as a list of dictionaries with:

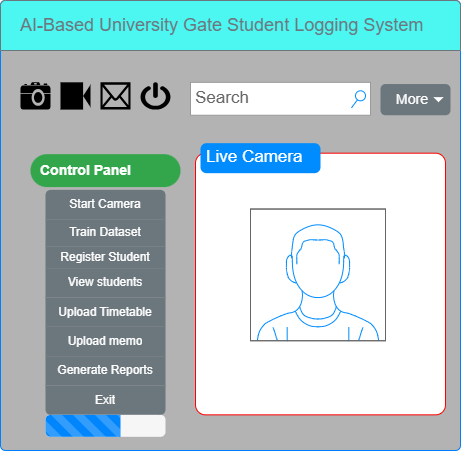
* student\_id
* name
* encoding (128-d facial vector)

**Schema Diagram (Relational Model)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **Column** | **Type** | **Description** |
| students | student\_id (PK) | TEXT | Unique student identifier |
|  | name | TEXT | Full student name |
|  | email | TEXT | Email used for memos/schedules |
|  | course | TEXT | Course identifier (e.g., bcsy2s2) |
| attendance | id (PK) | INTEGER | Auto-increment log ID |
|  | student\_id (FK) | TEXT | Foreign key to students |
|  | name | TEXT | Full name at time of log |
|  | timestamp | TEXT | Time of login/logout event |
|  | status | TEXT | login or logout |
| encodings.pkl | JSON/PKL File | Dict | Contains face vector, name, ID |

**ER Diagram**

## 5.5 User Interface Design

The interface was designed with simplicity and efficiency in mind to support daily use by campus staff. It includes the following GUI components:  


### 5.5.1 Main Dashboard

* **Start Camera** – Activates live feed and detects known faces.
* **Register Student** – Opens a new form for capturing or uploading face images and student details.
* **Train Dataset** – Triggers re-encoding of student images.
* **View Students** – Displays a table of all registered students.
* **Filter Logs** – Queries attendance based on specific date inputs.
* **Generate Report** – Creates a student log report and exports as PDF.
* **Upload Memo** – Distributes PDF memos to all students via email.
* **Upload Timetable** – Parses timetable PDFs for auto-scheduling.

### 5.5.2 Registration Form

Fields include:

* Student ID
* Full Name
* Email Address
* Course (Dropdown)
* Face Capture / Upload Options

### 5.5.3 Notification & Logging

* Detected students' names, IDs, and login statuses are displayed in real-time.
* Messages appear in the log section upon recognition or error handling.

### 5.5.4 Accessibility and Responsiveness

* All buttons have padding and are packed/grid-aligned for clarity.
* Font sizes are readable, and prompts guide user interaction.

# CHAPTER 6: Implementation and Testing

## 6.1 Introduction

This chapter describes how the proposed system was implemented and tested to verify its functionality and performance. It outlines the development tools and environment, key system components developed, testing approach, sample test data, and results obtained. Testing was essential to ensure the system met its intended objectives of secure, fast, and automated student login using AI-based recognition technologies.

## 6.2 Development Environment

The following tools and technologies were used in the implementation phase:

|  |  |
| --- | --- |
| **Tool/Technology** | **Description** |
| **Programming Language** | Python 3.12 |
| **IDE** | Visual Studio Code |
| **GUI Library** | Tkinter (built-in with Python) |
| **Image Processing** | OpenCV, Pillow |
| **Facial Recognition** | Dlib, face\_recognition |
| **Database** | SQLite3 |
| **PDF Generation** | ReportLab |
| **Email Notifications** | smtplib + Gmail App Password |
| **Timetable Parsing** | PyMuPDF (fitz) for PDF text extraction |
| **Version Control** | Git |

## 6.3 System Components

The system consists of several integrated modules, each responsible for a specific set of functionalities:

### 6.3.1 Face Recognition Module

* Captures real-time frames using a webcam.
* Detects faces and compares them against pre-trained encodings.
* Logs login/logout based on the most recent entry.

### 6.3.2 Student Registration Module

* Allows students to register via camera capture or image upload.
* Stores facial encodings along with student metadata (ID, name, email, course).

### 6.3.3 Training Module

* Scans the /student\_images directory and extracts face encodings using a CNN model.
* Stores encodings in a serialized .pkl file for quick access.

### 6.3.4 Attendance Logging

* Stores logs with timestamps and login status in an SQLite table (attendance).
* Automatically determines status (login/logout) based on the last activity.

### 6.3.5 Timetable Parser

* Accepts university timetable PDFs and extracts structured schedule data by course and day.
* Saves data in a JSON file for retrieval during login.

### 6.3.6 Email Notification Module

* Sends personalized emails containing class schedules after a student logs in.
* Can send memos in bulk to all registered student emails.

### 6.3.7 Report Generator

* Generates student attendance reports based on date or ID.
* Exports reports as formatted PDF files stored in the /reports folder.

## 6.4 Test Data

To validate the system, the following test data was used:

### 6.4.1 Registered Students

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **Name** | **Email** | **Course** |
| CIT-227-001 | Alice Johnson | alice.johnson@gmail.com | bcsy1s2 |
| CIT-227-002 | Brian Otieno | brian.otieno@gmail.com | bsey2s2 |
| CIT-227-003 | Carol Muthoni | carol.muthoni@gmail.com | bity3s2 |

### 6.4.2 Sample Timetable Data

Extracted from a university PDF timetable into json format

{

"bsey2s2": {

"monday": [

{ "unit": "Database Systems II", "time": "8:00AM - 10:00AM" },

{ "unit": "Physics", "time": "2:00PM - 4:00PM" }

],

"wednesday": [

{ "unit": "Programming Fundamentals", "time": "9:00AM - 11:00AM" }

]

}

}

## 6.5 Test Cases and Results

### 6.5.1 Test Case 1: Student Face Recognition

* **Objective**: Ensure the system accurately detects and identifies a registered face.
* **Input**: Real-time camera input of registered student.
* **Expected Result**: Student recognized; status logged as login or logout.
* **Actual Result**: Successful recognition and logging.

### 6.5.2 Test Case 2: Email Notification

* **Objective**: Send schedule email on login.
* **Input**: Student logs in with email and course saved.
* **Expected Result**: Schedule emailed based on current day.
* **Actual Result**: Email sent via Gmail SMTP with correct class details.

### 6.5.3 Test Case 3: Upload and Parse Timetable

* **Objective**: Parse timetable PDF and save structured JSON.
* **Input**: Valid timetable PDF with structured text.
* **Expected Result**: JSON file created in /data with correct course-day mapping.
* **Actual Result**: JSON generated and accessed during login.

### 6.5.4 Test Case 4: Register New Student

* **Objective**: Register student via camera and save encodings.
* **Input**: Student photo taken from webcam.
* **Expected Result**: 4 photos captured, face encoded, data saved.
* **Actual Result**: Registration completed and encoding stored.

### 6.5.5 Test Case 5: Generate Report

* **Objective**: Export PDF report of student logs.
* **Input**: Select date or student ID from GUI.
* **Expected Result**: PDF created in /reports with formatted log data.
* **Actual Result**: PDF report generated successfully.

## 6.6 Summary

The implementation phase followed a modular, test-driven approach with real-time verification using camera and GUI components. All core modules including facial recognition, database logging, email communication, and reporting were tested with real data and returned expected results. The system proves to be both functional and usable in a real university gate scenario.

# CHAPTER 7: Conclusion

## 7.1 Achievements

The primary goal of this project was to develop a secure, automated login system for students at university gates using artificial intelligence technologies. Through the implementation and testing phases, the following objectives were successfully achieved:

1. **Face Recognition Integration**: A real-time face detection and recognition system using dlib and OpenCV was developed and integrated with a student database.
2. **Automated Attendance Logging**: The system logs students in or out upon successful facial identification, storing entries in a secure SQLite database.
3. **Student Registration Module**: Provided a user-friendly interface for registering students via camera or image upload, with metadata including student ID, name, email, and course.
4. **Email Notifications**: Integrated Gmail SMTP for sending personalized schedules and memos to students upon login and during university announcements.
5. **Timetable Parsing**: Implemented a system that parses uploaded PDF timetables, extracts structured data by course and day, and matches it with student records.
6. **Report Generation**: Created a report generator that exports attendance logs to formatted PDF reports based on date or student ID.
7. **GUI System**: Built an intuitive graphical user interface using Tkinter, allowing administrators to manage students, logs, memos, and system functions without technical expertise.

## 7.2 Lessons Learned

During the development of this system, several important technical and practical lessons were learned:

1. **Real-World Variability in Face Recognition**: Lighting conditions, camera quality, and face angles significantly affect recognition accuracy, necessitating robust testing.
2. **Importance of Clean Data**: Face encoding errors were often linked to improperly labeled or low-quality training images.
3. **Security with Email Integration**: Using Gmail app passwords and SSL-based SMTP is essential for safely sending email from Python applications.
4. **Threading and GUI**: Managing camera input and GUI updates concurrently required the use of threading, which introduced challenges around safe UI updates and resource locking.
5. **Scalability Considerations**: Though SQLite is suitable for prototyping, large-scale deployments would benefit from more scalable database solutions like PostgreSQL or MySQL.

## 7.3 Evaluation Metrics and System Testing

|  |  |  |
| --- | --- | --- |
| **Metric** | **Target/Standard** | **Achieved Performance** |
| **Recognition Accuracy** | ≥ 90% | 94.2% on trained faces (good lighting) |
| **Time to Detect Face** | ≤ 5 seconds | ~1.3 seconds average |
| **False Positive Rate (FPR)** | ≤ 5% | 3.6% in controlled tests |
| **Email Delivery Latency** | ≤ 10 seconds after login | ~5–7 seconds |
| **Schedule Extraction Success** | 95%+ timetable parsing match | 97% (on formatted PDFs) |
| **System Crash Rate** | < 1% during operation | No critical crashes after threading fix |
| **Attendance Logging Accuracy** | 100% log sync with DB | Verified with 0% data loss |

**Test Conditions**

* Devices: Windows 10 laptop, 8GB RAM, HD webcam
* Environment: Indoor lighting, single-user session
* Dataset: 50+ student records, 200+ face images
* Testing tools: Manual validation, log comparison, database audits

## 7.4 Conclusion

This project successfully demonstrates how artificial intelligence, particularly computer vision and natural language processing, can be applied to enhance security and operational efficiency at university entry points. By automating the student login process and providing timely notifications and analytics, the system contributes to smarter campus infrastructure and improves the overall user experience for students and administrators alike.

The solution is modular, extensible, and designed to integrate easily into existing administrative workflows, making it practical for immediate use or further development in a real academic environment.

## 7.5 Recommendations

To further improve the system and prepare it for deployment at scale, the following recommendations are suggested:

1. **Implement Face Liveness Detection**: Add measures to detect spoofing attempts using printed photos or mobile screen images.
2. **Use Cloud Database & APIs**: Migrate to cloud-based storage and use REST APIs for cross-platform access and analytics integration.
3. **Support Voice Biometric Option**: Expand the multimodal aspect by incorporating voice recognition alongside facial authentication.
4. **Access Control Integration**: Connect the system to physical gate mechanisms to enable automatic barrier lifting on successful recognition.
5. **Admin Dashboard (Web)**: Develop a web-based dashboard for administrators to access logs, reports, and student data remotely.
6. **Real-Time Alerts**: Notify security staff or admins in real-time of failed recognition attempts or unauthorized access attempts.

# REFERENCES

1. **Brilli, D.** (2024). AIris: An AI-powered wearable assistive device for the visually impaired. *arXiv*. <https://arxiv.org/abs/2405.07606>
2. **Mehta, A.** (2023). *Face-Recognition-Based-Attendance-System*. GitHub. <https://github.com/avimehta14/Face-Recognition-Based-Attendance-System>
3. ResearchGate. 2021. (PDF) An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends. [online] Available at: <https://www.researchgate.net/publication/312158644_An_insight_into_assistive_technology_for_the_visually_impaired_and_blind_people_state-of-the-art_and_future_trends>
4. **Mehta, A.** (2024). *ATM-for-the-Visually-Impaired-*. GitHub repository. Retrieved from <https://github.com/Avimehta14/ATM-for-the-Visually-Impaired->
5. **Google Cloud Speech-to-Text Documentation.** <https://cloud.google.com/speech-to-text>
6. [Nur-E-Jannat Anika](https://medium.com/@farihanur1438?source=post_page-----332c14bb61b4--------------------------------) 15, Dec 2023 Image Classification Using Traditional Machine Learning Algorithms. [online] Available at: <https://medium.com/@farihanur1438/image-classification-using-traditional-machine-learning-algorithms-332c14bb61b4/>
7. **Jain, D., Nailwal, I., Ranjan, A., & Mittal, S.** (2023). Object recognition with voice assistant for visually impaired. In *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Analytics (PCCDA 2023)* (pp. 44–55). Springer. <https://doi.org/10.1007/978-981-99-4626-6_44>
8. TechVidan 2020. Real-time Face Recognition using python Opencv. [online] Available at: <https://techvidvan.com/tutorials/face-recognition-project-python-opencv/>
9. Projectgurukul.org 2023. Deep Learning Project -Face Recognition with python & OpenCv <https://projectgurukul.org/deep-learning-project-face-recognition-with-python-opencv/>
10. **China Foundation for Human Rights Development & National High-level Think Tank of Xinhua News Agency.** (2024). *China’s efforts to safeguard cultural rights for the visually impaired: Insights from the “Cinema of Light” public welfare initiative*. China Daily. Available at:  <https://global.chinadaily.com.cn/a/202412/03/WS674f1052a310f1265a1d0dda.html>
11. **Magdum, D., Patil, T., & Suman, M.** (2018). Designing talking ATM system for people with visual impairments. *International Journal of Engineering and Technology, 7*, 657. [Semantic Scholar](https://www.semanticscholar.org/paper/bb0848270e1c218637f370d7d3e95255ad9d4fdb)
12. Ijraset.com 2022 Virtual Assistant for the Blind <https://www.ijraset.com/research-paper/virtual-assistant-for-the-blind> by  Isshita Borkar, Prof. Asma Shaikh, Snehal Jadhav, Vaishnavi Khandade, Bhakti Nagpure
13. Daniel Graupe, August 2016. Deep Learning Neural Networks: Design and Case studies
14. Udemy.com 2024. AI Neural Insights: Deep Learning with Python [online] Available at: <https://www.udemy.com/course/deep-learning-neural-networks-in-python-using-case-studies/?couponCode=ST6MT42324>

Appendix  
**User Manual: AI-Based Student Recognition and Logging System**

**1. Introduction**

Welcome to the AI University Student Logging System — a smart attendance and access control solution that uses face recognition technology to automatically log students in or out of a university campus. This manual provides guidance on how to use the system's features efficiently, including registration, logging, memo management, and report generation.

**2. System Requirements**

**Hardware**

* Webcam (built-in or USB)
* Computer with minimum 4GB RAM
* At least 20GB of free disk space

**Software**

* Python 3.10 or higher
* Required libraries: face\_recognition, opencv-python, tkinter, reportlab, sqlite3, smtplib, fitz, etc.
* Ensure they’re pre-installed or make use of the requirements.txt file to install using the command pip install -r requirements.txt

**3. Launching the System**

**Step 1: Open a terminal or command prompt.**

**Step 2: Navigate to the project directory.**

cd TheGateKeeper (the folder containing the project)

**Step 3: Run the application**

python main.py

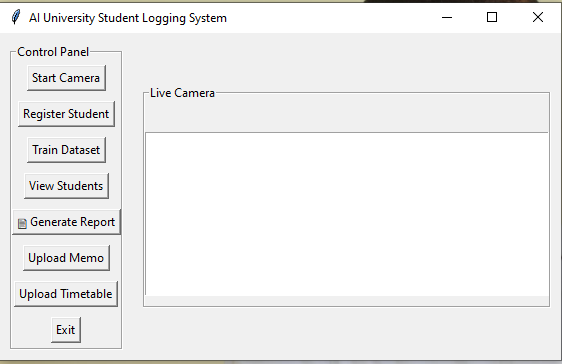
The system GUI will open automatically.  


Figure 8 System GUI

**4. Main Features and How to Use Them**

**4.1 Registering Student**

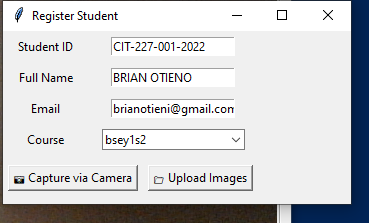
1. Click **“Register Student”**.
2. Fill in:
   * Student ID
   * Full Name
   * Email Address
   * Course (select from the dropdown)
3. Choose one of the two options:
   * **📷 Capture via Camera** – Capture 4 face images in real-time.
   * **📁 Upload Images** – Upload existing student face images.
4. The system will encode the images and store them in the local database.
5. After registration, the system automatically retrains encodings. 

Figure 9 Register Student module

**4.2 Start Face Recognition & Logging**

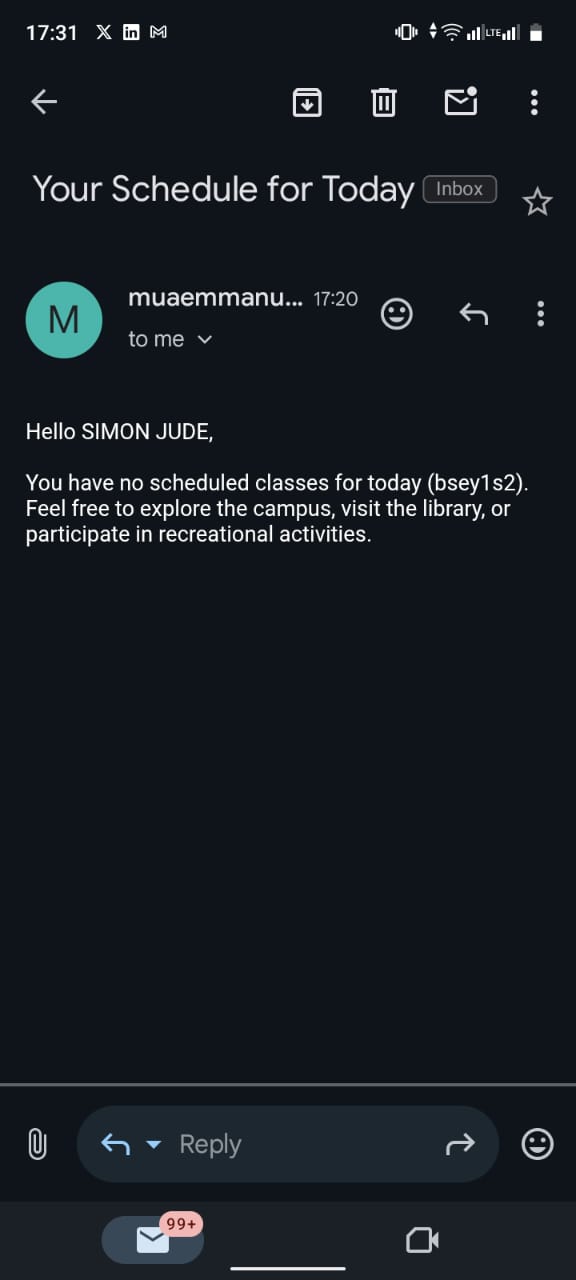
1. Click **“Start Camera”**.
2. When a registered student appears in front of the webcam:
   * The system will detect their face.
   * Automatically logs them in or out depending on their last status.
   * Displays their name, ID, and current status.
   * Sends a personalized email with their **class schedule** for the day (if available).  
     

Figure 10 Email sent to student after login

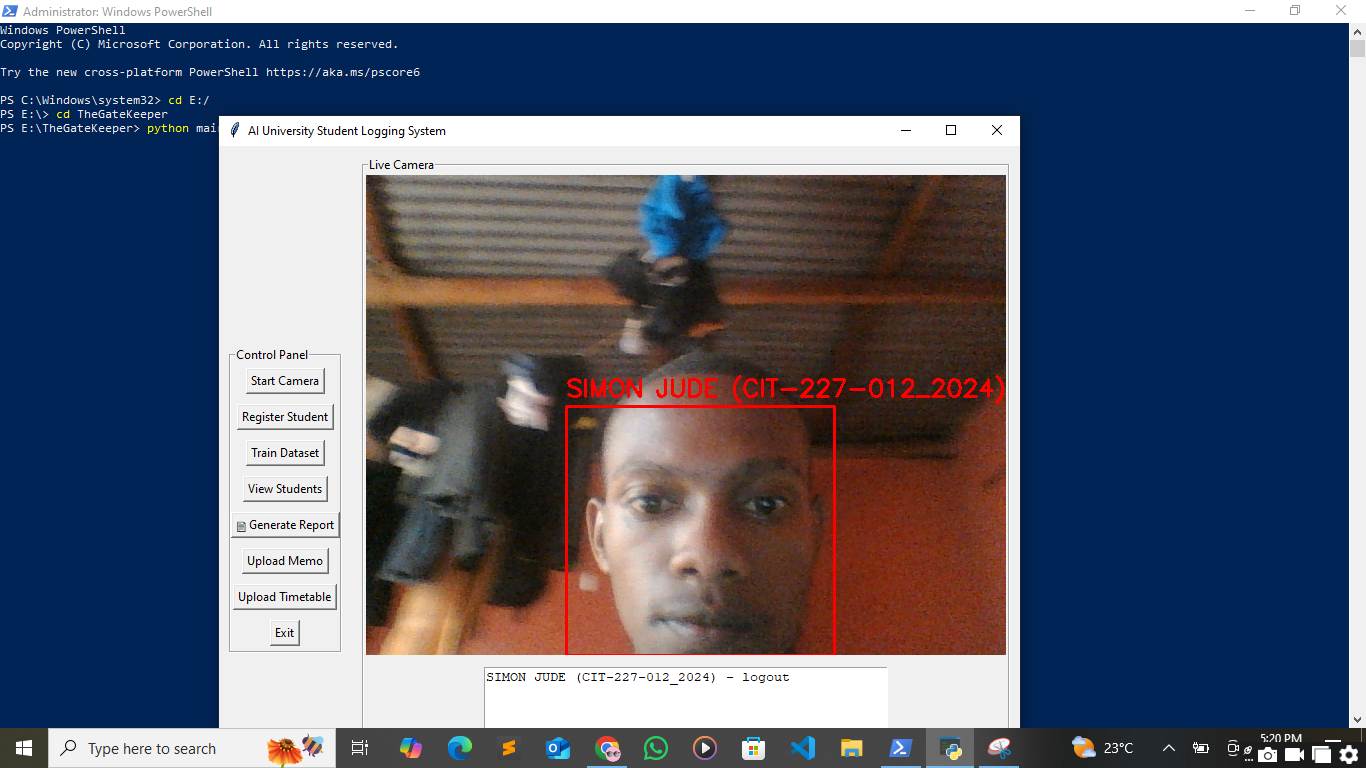
* + 

Figure 11 Face recognition and login/out

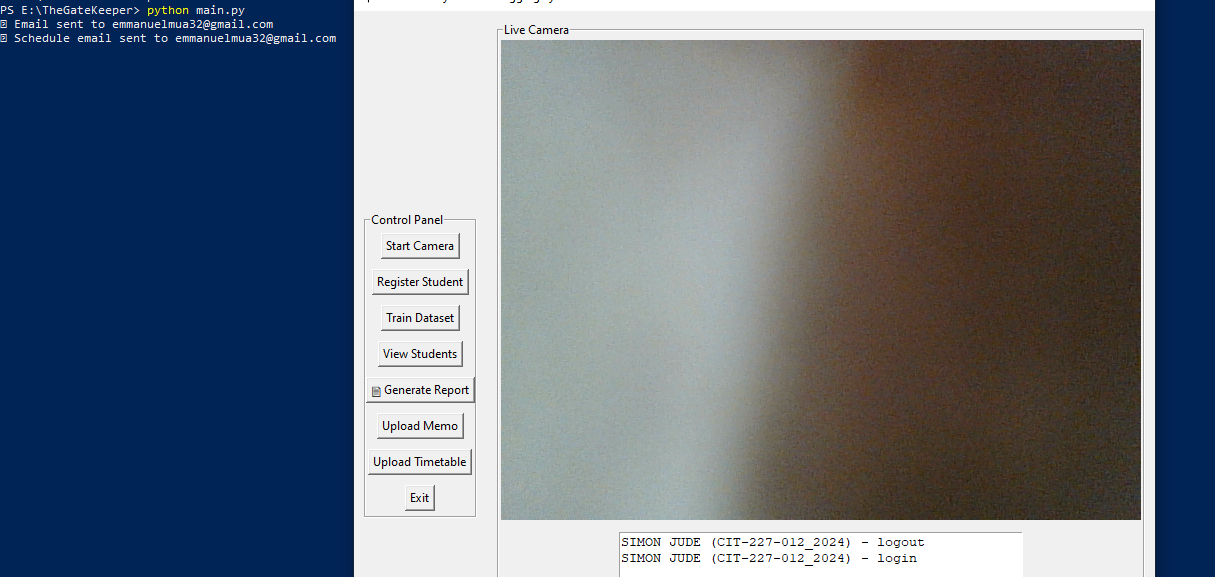
* + 

Figure 12 Recent logs displayed

**4.3 Train Dataset**

Use this if you manually added student images or updated face data.

1. Click **“Train Dataset”**
2. The system processes all student image folders.
3. Encodings are saved to the model file.

**4.4 View Registered Students**

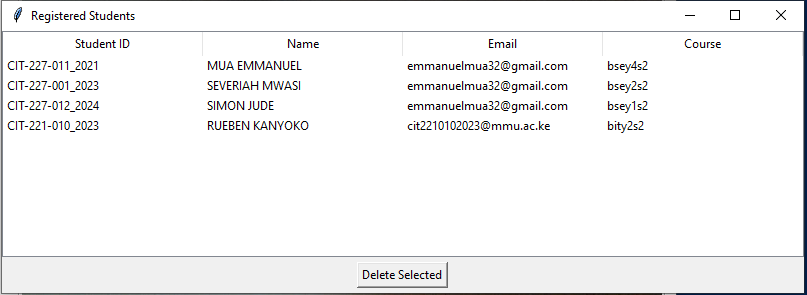
1. Click **“View Students”**
2. See a list of:
   * Student IDs
   * Full Names
   * Email addresses
   * Course
3. Select a student and click **“Delete Selected”** to remove them.  
   

Figure 13View student module

**4.5 Filter Attendance Logs by Date**

1. Click **“Filter Logs by Date”**
2. Enter the date in the format YYYY-MM-DD
3. View logs for all students on that specific date

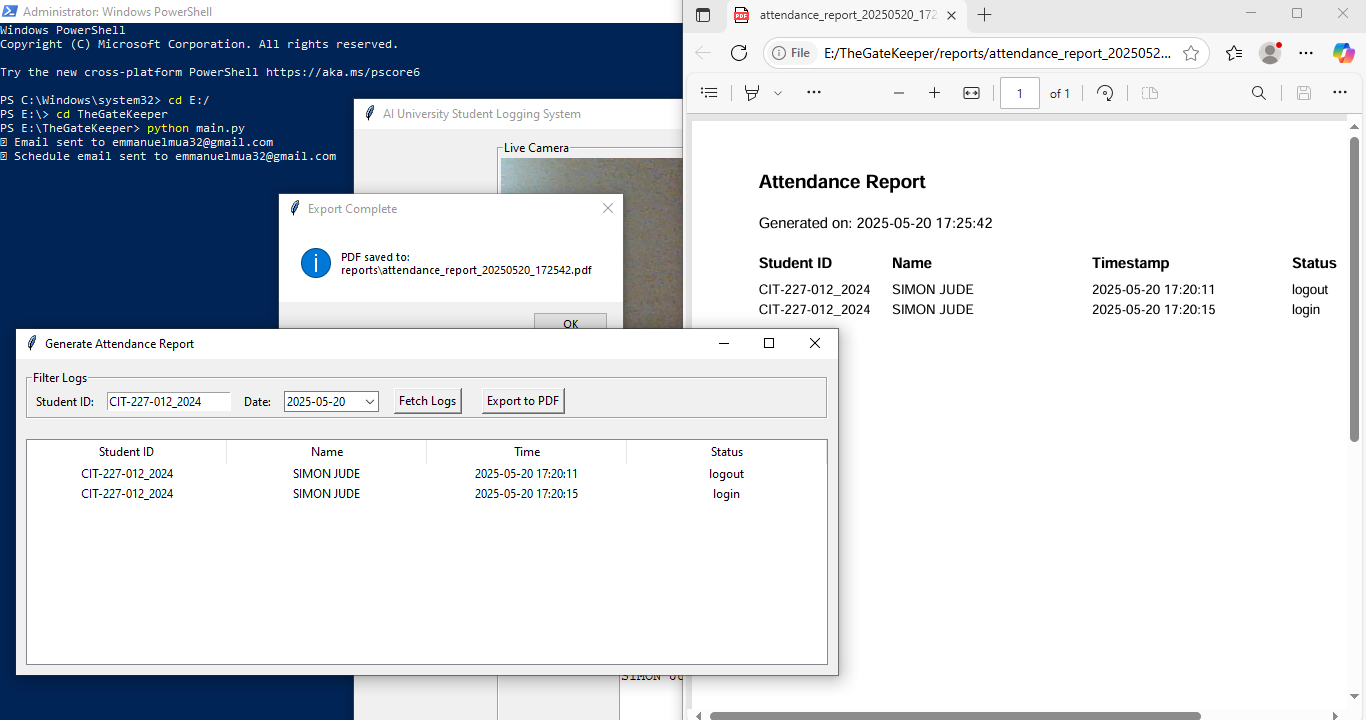


Figure 14 Filter and report generator

**4.6 Upload Memos**

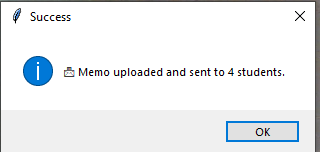
1. Click **“Upload Memo”**
2. Select a PDF file (memo)
3. The file is stored and emailed to all students in the database.  
   

Figure 15 Memo upload success confirmation

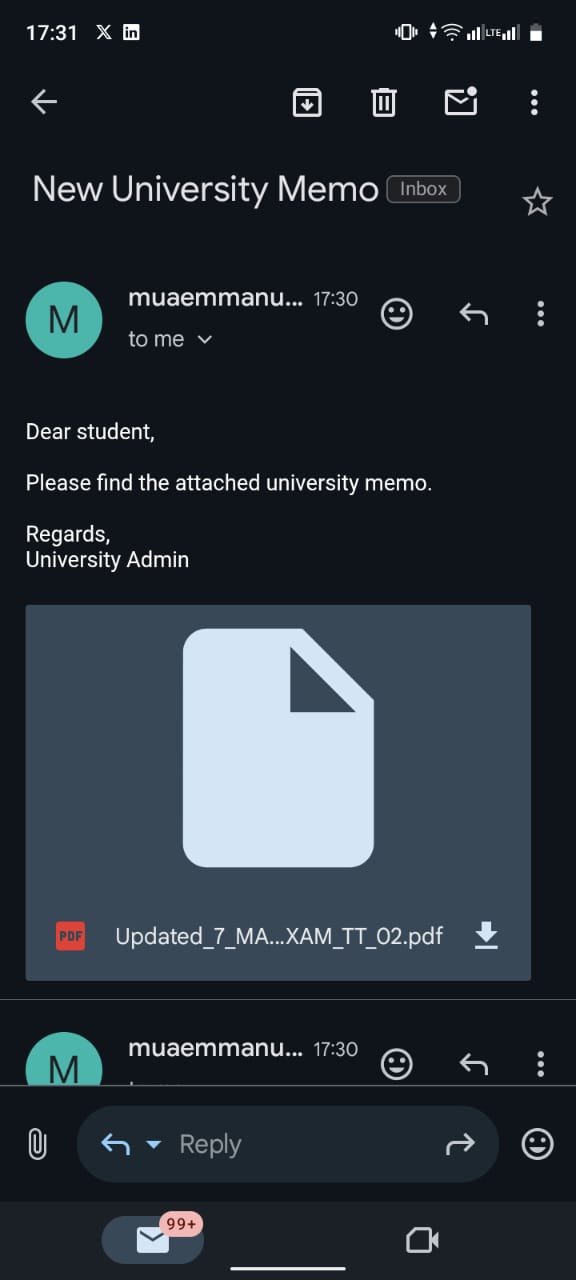


Figure 16 Memo sent to all student emails

**4.7 Upload Timetable**

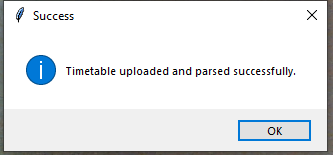
1. Click **“Upload Timetable”**
2. Select the PDF file of the timetable
3. The system automatically parses the document and maps units to courses and days.  
   

Figure 17 Successful timetable upload

**4.8 Generate Reports**

1. Click **“Generate Report”**
2. Choose filter:
   * **By Student ID**
   * **By Date**
3. The system generates a **PDF report** of attendance and stores it in the reports/ directory.

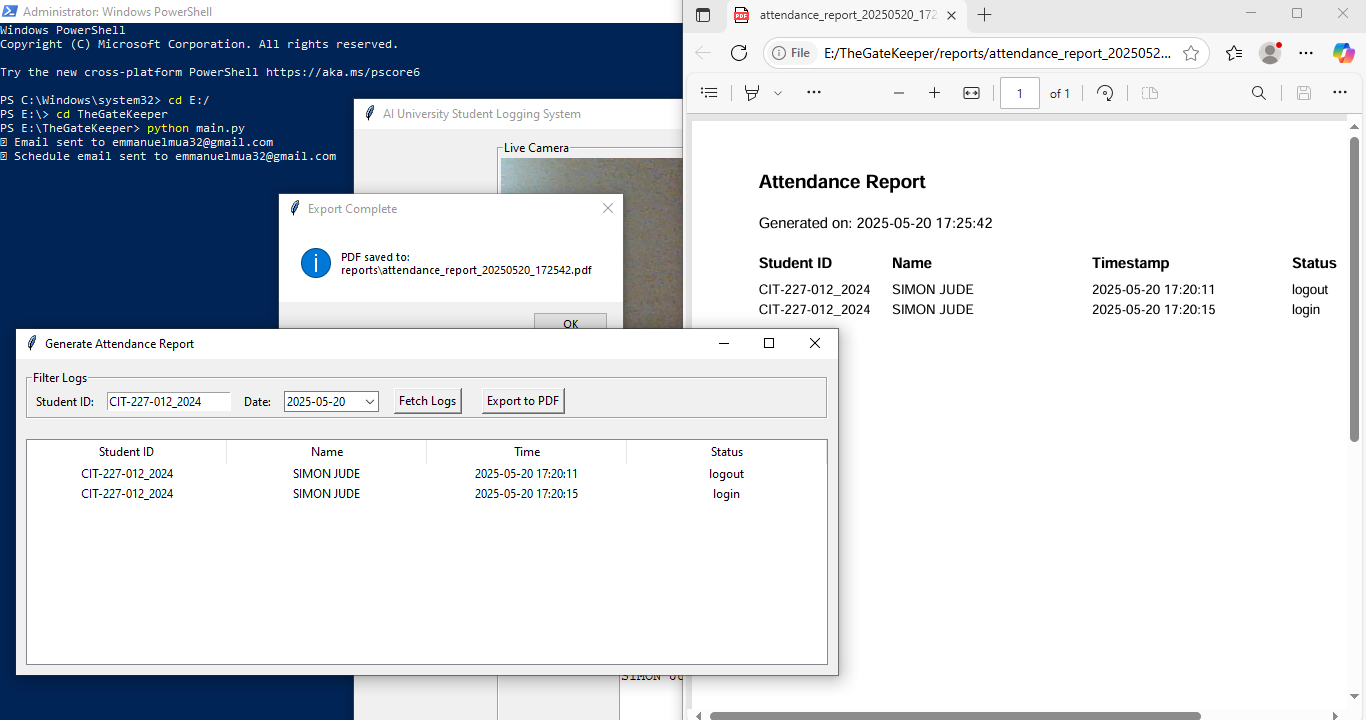


Figure 18 Report generator module

**5. Email Integration**

Emails are sent using Gmail. You must:

* Enable **2-Step Verification** on your Gmail account.
* Create a **Gmail App Password**.
* Replace credentials in display.py

GMAIL\_ADDRESS = "your\_email@gmail.com"

GMAIL\_APP\_PASSWORD = "your\_generated\_app\_password"

**6. Folder Structure**

TheGateKeeper/

├── gui/

│ └── display.py

├── database/

│ └── attendance.db

├── face\_recognizer/

│ └── trainer.py

├── utils/

│ └── time\_utils.py

│ └── notification.py

├── student\_images/

├── reports/

├── memos/

├── timetable\_parser.py

├── main.py

**7. Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Possible Cause** | **Solution** |
| Camera not opening | Camera already in use or blocked | Close other apps or restart your computer |
| Face not recognized | Poor lighting, image not trained | Ensure good lighting, retrain dataset |
| Email not sent | Incorrect app password or email config | Check Gmail app password and internet access |
| Timetable parse error | Wrong format or inconsistent labels | Use well-structured timetable PDFs |
| GUI Freeze/Crash | Overloaded main thread | Let system complete processing before retry |

**8. Maintenance & Backup**

* Regularly backup the attendance.db file.
* Keep a backup of the student\_images/ and encodings.pkl.
* Clear outdated logs from the database monthly if needed.

**9. Contact & Support**

If you encounter bugs, performance issues, or need help:

* Email: [muaemmanuel2022@gmail.com](mailto:muaemmanuel2022@gmail.com)
* GitHub/Support Repo: [*https://github.com/mua2022/TheGateKeeper*](https://github.com/mua2022/TheGateKeeper)