SILIGURI INSTITUTE OF TECHNOLOGY

Minor Project: MCAN-381

"SMART ATTENDANCE SYSTEM"

Submitted by

Raghav Kumar Jha (Roll No: 33671024009)

Yojana Adhikari (Roll No: 33671024027)

Sonam Sharma (Roll No: 33671024029)

Shruti Kumari Singh (Roll No: 33671024030)

Under the guidence of

Dr. Tumpa Banerjee

Submitted to the Department of Master of Computer Application in partial fulfillment of the requirements for the award of the degree MCA .

Year of Submission: 2025

Siliguri Institute of Technology
P.O. SUKNA, SILIGURI, DIST. DARJEELING, PIN: 734009
Tel: (0353)2778002/04, Fax: (0353) 2778003

DECLARATION

We, the undersigned group members, hereby solemnly declare that the project report entitled “Smart Attendance System” is submitted in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications (MCA) at Siliguri Institute of Technology under Maulana Abul Kalam Azad University of Technology, West Bengal.

We affirm that the work contained within this report is original and constitutes our own joint effort. We have taken assistance from other published and unpublished sources only when necessary, and all such sources have been properly acknowledged within the text. Furthermore, we certify that this report has not been submitted to any other Institute or University for the award of any other degree.

Project Group Signatures

We, the members of the project group, affirm the originality and integrity of the work presented herein:

1. Raghav Kumar Jha (Roll No: 33671024009)

Signature: ____________________________

2. Yojana Adhikari (Roll No: 33671024027)

Signature: ____________________________

3. Sonam Sharma (Roll No: 33671024029)

Signature: ____________________________

4. Shruti Kumari Singh (Roll No: 33671024030)

Signature: ____________________________

Date: ____________________________

Place: Siliguri, West Bengal

CERTIFICATE

This is to certify that the project report entitled “Smart Attendance System” submitted to the Department of Master of Computer Applications (MCA) of Siliguri Institute of Technology in partial fulfillment of the requirement for the award of the degree of MCA during the academic year 2024-2025, is a bonafide record of the project work carried out by the following students under my guidance and supervision.

Student's Name	Roll No	Registration No
Raghav Kumar Jha	33671024009	243360510029
Yojana Adhikari	33671024027	243360510049
Sonam Sharma	33671024029	243360510039
Shruti Kumari Singh	33671024030	243360510030

____

Signature of Project Guide

Dr. Tumpa Banerjee

____

Signature of the HOD

Department of MCA

Submitted to:

Maulana Abul Kalam Azad University of Technology, West Bengal

Acknowledgement

We, the project group, wish to express our profound gratitude to our esteemed guide, **Dr. Tumpa Banerjee**, for her unwavering mentorship, insightful criticism, and expert guidance throughout the entire process of developing the Smart Attendance System. Her dedication, technical expertise, and support were instrumental in the successful completion of this academic endeavor.

Our sincere thanks are also extended to the entire faculty and the **Department of MCA, Siliguri Institute of Technology**, for providing the excellent academic environment, essential laboratory resources, and the platform necessary to undertake this major project.

Finally, we are deeply grateful to our friends and families for their constant motivation, patience, and unwavering encouragement, which provided the necessary support system during the challenging phases of this work.

Project Group Signatures

Name	Student ID	Signature	Date
Raghav Kumar Jha	33671024009	____________________	______________
Yojana Adhikari	33671024027	____________________	______________
Sonam Sharma	33671024029	____________________	______________
Shruti Kumari Singh	33671024030	____________________	______________

Abstract

Traditional attendance systems are critically flawed by inefficiency, susceptibility to proxy attendance, and data forgery. This project focuses on the development of the Smart Attendance System, leveraging advanced facial recognition technology to introduce a secure and fully automated solution for modern institutional and organizational needs. The project is strategically driven by the ultimate vision of a comprehensive platform that includes integrated liveness detection (using CNNs) to counter active spoofing, alongside robust Human Resource Management (HRM) capabilities designed for streamlining institutional administration through automated working hour calculations and detailed leave tracking.

The resulting solution, developed using a Minimum Viable Product (MVP) strategy and a two-tiered architecture based on Python, is a very robust Desktop Application incorporating Computer Vision (OpenCV) and Deep Metric Learning using `face_recognition` libraries for reliable real-time identification of a user based on a 128-D face embedding. The Desktop Application is secured using a Tkinter GUI interface for Administration purposes and MySQL storage with a persistent connection to not only ensure transactional integrity but also record minute presence detail with an accurate In-Time (`first_time`) and Out-Time (`last_time`) stamping. The Desktop Application successfully enables authenticatable biometric security, optimized admin efficiency with specialized CRUD handling, and a historical auditing capability via automated and exportable reports in the standard format of .xlsx, making it a foolproof platform for complete Attendance Management.

Table of Index

Smart Attendance System

1. Introduction

A desktop application called the Smart Attendance System was developed due to the increased demand for a secure, automated, and efficient method of tracking students' attendance in educational institutions. The Smart Attendance System provides a verifiable biometric method for tracking student attendance through the use of computer vision; the Smart Attendance System was designed to be superior to the current methods of tracking student attendance through manual means.

1.1 Objective

The primary objectives of the Smart Attendance System are directly tied to overcoming the limitations of conventional methods and establishing a robust administrative tool:

- **To Ensure Biometric Security:** To implement a facial recognition system capable of accurately identifying registered students in real-time, thereby guaranteeing non-repudiation and eliminating proxy attendance.

- **To Achieve Real-Time Data Persistence:** To develop a robust backend system that continuously logs student presence by recording highly accurate first time and last time time stamps directly into a secure MySQL database.

- **To Maximize Operational Efficiency:** To automate the entire attendance logging process, reducing the time spent by educators on administrative tasks and increasing overall classroom productivity.

- **To Provide Comprehensive Administrative Utility:** To build an intuitive Tkinter-based GUI that provides integrated tools for student data management (CRUD operations) and flexible reporting (daily and custom date range export in .xlsx format).

1.2 Problem Statement

Traditional attendance procedures—relying on paper sign-in sheets, manual roll calls, or RFID badges—are fundamentally flawed. These methods are susceptible to proxy enrollment, consume significant class time, and introduce substantial delays and errors in data compilation and analysis. The absence of real-time monitoring leads to reactive, rather than proactive, academic administration. Furthermore, the lack of immediate digital data aggregation makes historical auditing cumbersome and hinders the timely identification of critical attendance patterns requiring intervention.

1.3 Solution and Scope

The system presents a robust, multi-layered solution:

Feature	Value Proposition	Technical Implementation
Biometric Verification	Ensures non-repudiation and eliminates proxy attendance fraud.	Utilizes Deep Metric Learning (via face_recognition) to identify students based on unique 128-D face embeddings.
Real-time Data Logging	Provides granular, auditable proof of presence and tracks student duration in class.	High-frequency webcam polling (cv2) with dynamic first_time and last_time updates to MySQL.
Secure Centralization	Guarantees data integrity, scalability, and secure long-term archival.	Data persistence achieved using a MySQL relational database backend.
Administrative Utility	Streamlines administrative workload and supports data-driven decision-making.	Tkinter GUI provides integrated tools for student database management (CRUD) and instantaneous report generation (openpyxl).

The scope is a single, self-contained Python application designed for deployment on a workstation equipped with a webcam and network access to the MySQL server instance.

2. Components of Attendance System

The application is built using three major functional layers, each responsible for a specific role in the system.

2.1 Image Capture Workflow

- **Camera Initialization:** This step establishes the live connection to the webcam hardware and manages the system resource.
  - **Technical Detail:** The `cv2.VideoCapture(0)` command initializes the input stream from the default webcam device. This resource is managed through the `self.cap` object and is explicitly released upon system shutdown via `self.cap.release()`.

- **Frame Processing and Optimization:** This crucial step ensures the system maintains high performance and a fast frame rate for real-time tracking.

  - **Technical Detail:** To sustain a high frame rate for real-time tracking, the `video_stream` method implements frame downsampling: `cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)`. This reduces the input resolution for the face detection algorithm, which speeds up processing time.

- **GUI Integration Bridge:** This final step converts the video data into a format that the graphical user interface can display continuously.

  - **Technical Detail:** Frames are converted from the standard OpenCV NumPy array (BGR format) to the RGB format required for recognition, and then specifically converted to the Tkinter-compatible `tk.PhotoImage` format for display on the `self.video_label`.

2.2 Image Recognition Workflow

- **Loading Known Faces (`load_known_faces`):** This is the system preparation step, ensuring all necessary reference data is cached for rapid verification.
  - **Technical Detail:** This routine retrieves student records from MySQL (`SELECT name, image_path FROM students`), loads the image file, and generates the unique 128-D embedding using `face_recognition.face_encodings(image)`. These embeddings are cached in Python memory (`self.known_faces`) for rapid lookup.

- **Runtime Verification and Annotation:** This is the core logic loop, responsible for processing the live video frame, generating face features, and matching them against the cached profiles.

  - **Technical Detail:** In the `video_stream` loop, Face detection is performed using `face_recognition.face_locations()`; encodings for the live faces are computed using `face_recognition.face_encodings()`; and finally, `face_recognition.compare_faces` is called against the cached `self.known_faces` to identify the student.

- **Visual Feedback Protocol:** This step provides instant, unambiguous visual confirmation to the administrator regarding the identity and logging status of the detected subject.

  - **Technical Detail:** The system uses OpenCV drawing functions: **Green bounding boxes** (`(0, 255, 0)` in BGR) confirm successful identification and logging, while **Red bounding boxes** (`(0, 0, 255)`) indicate an unidentified face.

2.3 Database for Candidate (Data Persistence)

The system relies on a central **MySQL** database named `smart_attendance` to maintain persistent, structured data. The data persistence layer is critical for ensuring data integrity, transactional reliability, and long-term archival.

The database serves as the single source of truth for all application data, facilitating two primary functions:

1. **Reference Data Storage:** It stores static administrative data, including user authentication details (`admin` table) and persistent student profiles (`students` table).

2. **Transactional Logging:** It is the destination for all real-time attendance events, specifically recording the auditable `first_time` and `last_time` stamps in the `attendance` table.

This centralized structure ensures data consistency and allows the application to query, manage (CRUD), and export historical records reliably.

2.4 Report

The reporting module provides administrators with powerful data extraction capabilities, ensuring seamless output in the professional .xlsx format via the `openpyxl` library. This system transforms raw attendance events, anchored by the critical In-Time and Out-Time stamps, into comprehensive intelligence. We categorize our reports into two distinct levels: Core Log Reports (generated instantly via SQL functions for daily and period logs) and Derived Analysis Reports (advanced metrics like Late Entry, Absence, and Percentage, which are fully calculable from the exported raw data). This structure ensures both auditable integrity and maximum flexibility for external data analysis.

3. Workflow of Project

The project workflow is strictly governed by administrative authorization and is designed for operational reliability, directly mirroring the sequence of function calls in the main program.

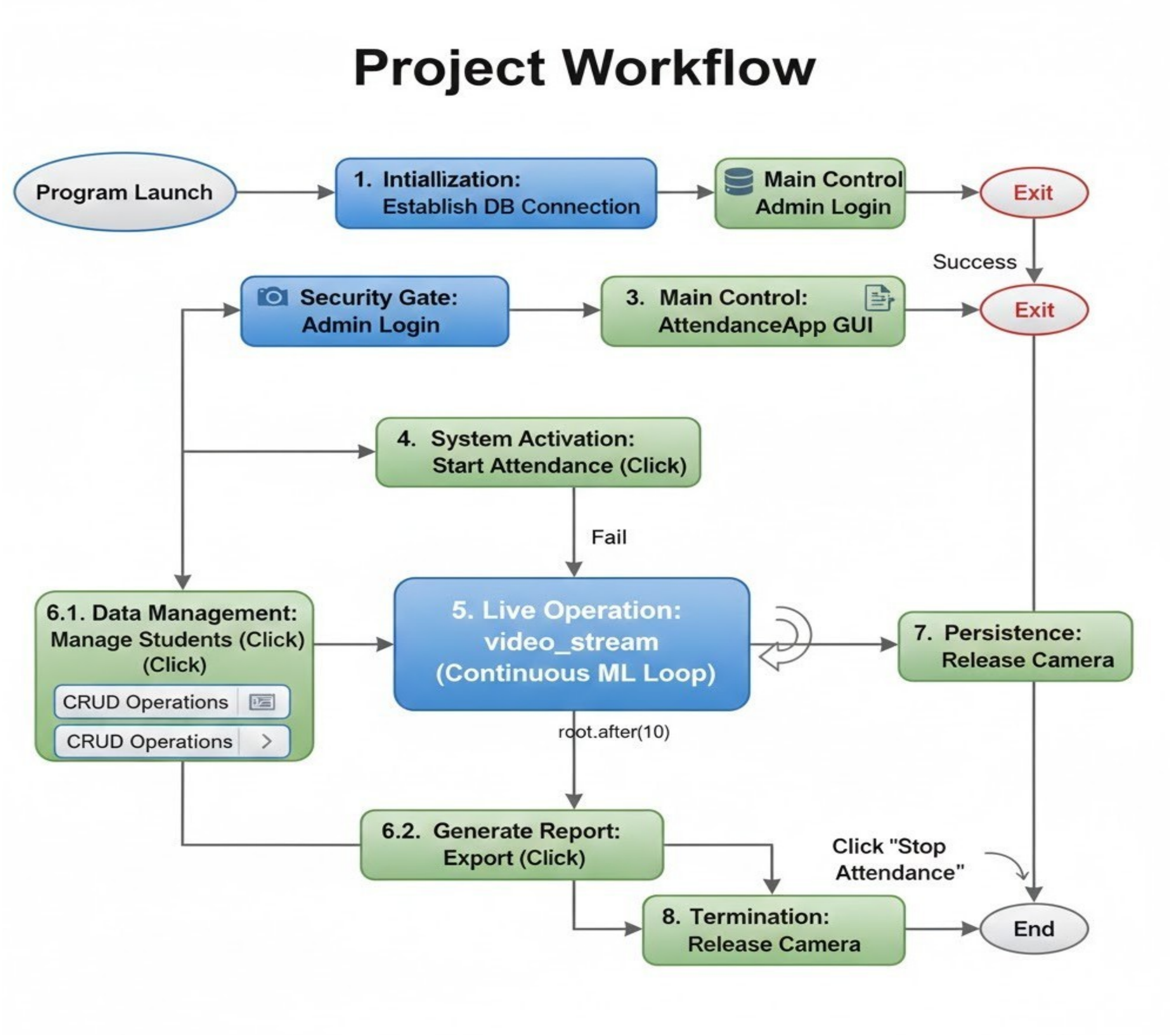

Project Workflow
Program Launch
1. Intiallization: Establish DB Connection
Main Control Admin Login
Exit
Success
Security Gate: Admin Login
3. Main Control: AttendanceApp GUI
Exit
4. System Activation: Start Attendance (Click)
Fail
6.1. Data Management: Manage Students (Click) (Click)
CRUD Operations
CRUD Operations
5. Live Operation: video_stream (Continuous ML Loop)
7. Persistence: Release Camera
root.after(10)
6.2. Generate Report: Export (Click)
Click "Stop Attendance"
8. Termination: Release Camera
End

Figure 1: *Workflow of the Project*

Step	**Functional Module**	**Trigger**	**Core Action and Security Gate**
Initialization	Database Connection	Program Launch	Establishes mysql.connector global variables (db, cursor).
Security Gate	AdminLogin Class	User Input/Click "Login"	Verifies credentials against the admin table via cursor.execute("SELECT * FROM admin...").
Main Control	AttendanceApp.__init	Successful Login	Instantiates the AttendanceApp Tkinter GUI.
System Activation	start_attendanc e	Click "Start Attendance"	Calls load_known_faces and initializes cv2.VideoCapture(0).
Live Operation	video_stream	Continuous root.after(1 0) loop	Executes the ML pipeline (Section 4) for tracking and logging via mark_attendance.
Data Management	manage_students	Click"Manage Students"	Opens a modal window allowing authorized CRUD operations (INSERT,U P D A T E ,D E L E T E) using raw SQL queries.
Persistence	db.commit()	Called after every database write operation	Ensures data modifications are permanently stored in MySQL.
Termination	stop_attendance	Click"Stop Attendance"	Clears the camera resource (self.cap.release()) and sets self.running = False.

4. Workflow of Model

The face recognition and logging model implements a continuous loop managed by the Tkinter event system, focusing on the processing of each video frame.

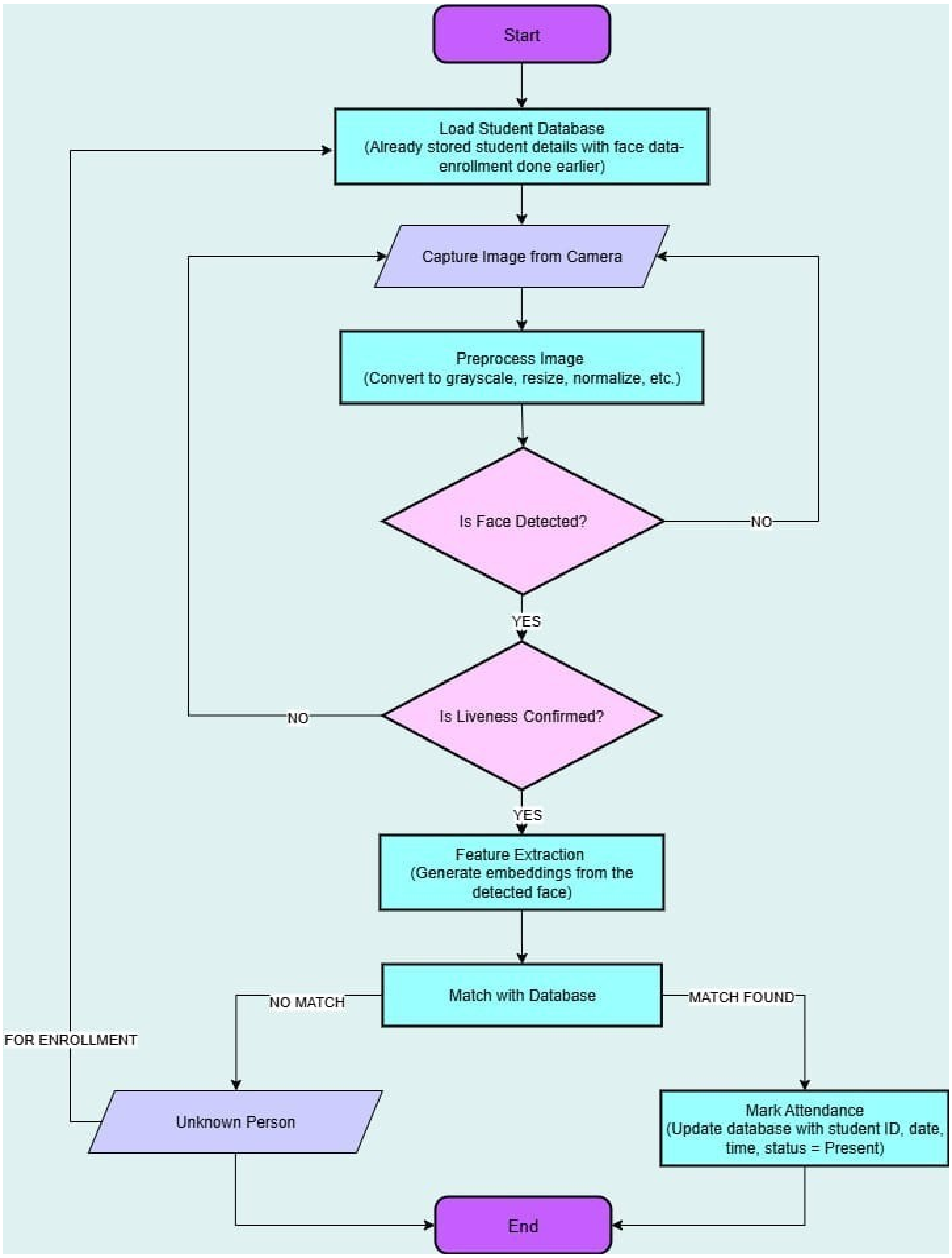

Start
Load Student Database
(Already stored student details with face data-enrollment done earlier)
Capture Image from Camera
Preprocess Image
(Convert to grayscale, resize, normalize, etc.)
Is Face Detected?
NO
YES
Is Liveness Confirmed?
NO
YES
Feature Extraction
(Generate embeddings from the detected face)
Match with Database
NO MATCH
MATCH FOUND
FOR ENROLLMENT
Unknown Person
Mark Attendance
(Update database with student ID, date, time, status = Present)
End

Figure 2: *Workflow of Model*

The face recognition and logging model implements a continuous loop managed by the Tkinter event system, focusing on the processing of each video frame.

- **Loading Reference Data:** This initial step prepares the system by fetching and processing all known biometric identifiers from the database for fast comparison.

  - **Technical Detail:** `load_known_faces` queries the `students` table, loads images based on `image_path`, and generates the **128-D embedding vectors**, caching them in Python lists.

- **Frame Acquisition and Preprocessing:** This step captures the live feed and optimizes the image data to maximize processing speed for the ML model.

  - **Technical Detail:** `video_stream` captures the frame (`self.cap.read()`), resizes it (`fx=0.25`), and converts it to RGB.

- **Face Detection and Localization:** This step accurately pinpoints the location of faces within the current video frame.

  - **Technical Detail:** `face_recognition.face_locations()` identifies bounding box coordinates.

- **Encoding Generation (Feature Extraction):** This step converts the detected face region into the unique numerical vector used for identification.

  - **Technical Detail:** `face_recognition.face_encodings()` computes the 128-D vector for each detected face.

- **Biometric Matching:** This step compares the unique feature vector of the live face against the database of known embeddings to establish identity.

  - **Technical Detail:** The live vector is compared against all cached vectors using `face_recognition.compare_faces()`.

- **Mark Attendance (`mark_attendance`):** This function executes the business logic for transactional logging in the database based on the verified identity.

  - **Technical Detail:**

    - **Fetch ID:** Retrieves `student_id` using `SELECT id FROM students WHERE name=%s`.

    - **Check Record:** Queries `attendance` table for an existing record for today.

    - **Update Logic:** If a record exists, executes `UPDATE attendance SET last_time=%s...`.

    - **Insert Logic:** If no record exists, executes `INSERT INTO attendance (student_id, date, first_time, last_time) VALUES...`.

- **GUI Update and Loop Continuation:** This final step refreshes the visual output and schedules the next cycle of the verification loop.
    - **Technical Detail:** The annotated frame is displayed, and the process is immediately rescheduled via `root.after(10)`.

5. Technology Used

This section details the selection and utilization of key third-party libraries based strictly on their use in the code.

Library	Rationale for Selection	Specific Implementation Details
Tkinter	Used for creating the cross-platform, multi-window desktop interface.	Used for the AdminLogin modal, AttendanceApp window, and managing the display of the video feed (tk.PhotoImage).
mysql.connector	Provides stable, direct database connectivity to MySQL.	Initializes the global db connection and executes all transactional SQL queries (CRUD, SELECT) via the cursor object.
OpenCV (cv2**)**	Industry-standard for real-time camera interfacing and image manipulation.	Used for hardware access (VideoCapture(0)), performance optimization (cv2.resize), and drawing visual feedback (cv2.putText, cv2.rectangle).
face_recognition	High-level API used to simplify the complex deep learning pipeline of face detection, feature extraction, and comparison.	Provides the core biometric functions: face_locations (detection), face_encodings (feature generation), and compare_faces (verification).
shutil	Standard Python module used for high-level file operations.	Used in manage_students via shutil.copy to move the student's reference image into the system's known_faces/ directory.
openpyxl	Used to handle native Excel XLSX file format, ensuring high compatibility for administrative reporting.	Used in export functions to create the Workbook (), append data rows, and save the final report file.

6. Face Recognition Model

6.1 About Face recognition Model

The system utilizes a **Deep Metric Learning** approach implemented through the **Dlib library's ResNet architecture**, accessed via the face_recognition API. This architecture is pre-trained to generate unique face embeddings.

- **Underlying Architecture:** A **29-layer residual neural network (ResNet)** is executed to extract invariant facial features.
- **Training Objective:** The model was trained using the **Triplet Loss** function, which minimizes the distance between embeddings of the same person and maximizes the distance between embeddings of different people.

6.2 Input/output - Details about model

Stage	**Input/Output**	**Detail and Technical Specification**
Input	Image Data (Frame)	A pre-processed RGB image array captured by `self.cap.read()`.
Intermediate Output	Face Locations	A tuple of four integers defining the bounding box: `(top, right, bottom, left)`.
Core Output	Face Embedding	A **128-dimensional floating-point vector (NumPy array)**.
Verification Logic	Comparison Method	**Euclidean Distance** is calculated between the live embedding and the stored reference embeddings.
Decision Parameter	Threshold/Tolerance	A match is declared if the distance falls below a pre-defined tolerance, handled by `face_recognition.compare_faces()`.

6.3 Available Model

The system utilizes the **pre-trained Dlib ResNet model**, which is optimized for performance on desktop CPUs.

6.4 Our Library:- face_recognition: This Python wrapper provides a clean API for:

1. face_recognition.face_locations(): Face detection.
2. face_recognition.face_encodings(): Encoding generation.
3. face_recognition.compare_faces(): Distance calculation and threshold check.

6.5 Model Accuracy and Operational Performance

The model's operational performance is subject to specific environmental and spatial constraints:

- **Calculated Identification Accuracy:** In controlled multi-subject scenarios (up to five people), the system demonstrated an identification accuracy of approximately **80.0%** (a 20% error rate). This marginal error rate is primarily observed when multiple subjects are moving or positioned close together.

- **Spatial Limitation:** Reliable detection and recognition of student faces is constrained to an operational distance of **100 cm** (approximately 3.3 feet) from the camera, beyond which accuracy degrades significantly.

- **Field of View Degradation:** In crowded scenarios, the system experiences performance reduction. Specifically, high density can lead to occasional **False Positives** (misidentifying a person) or **False Negatives** (failing to identify a known person).

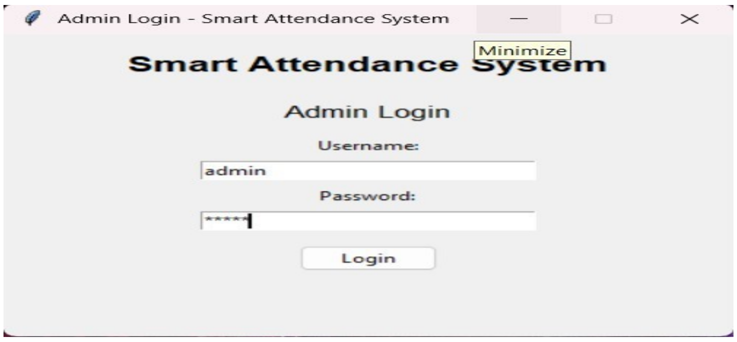
- **Motion Sensitivity:** Excessive or rapid movement by a subject can disrupt the face detection and encoding pipeline, resulting in temporary loss of tracking or inaccurate identification.

7. User Interface

The User Interface (UI) is the presentation layer, built entirely with Python's standard library.

7.1 Key things in UI

- **Security Gateway:** The AdminLogin class provides mandatory password-protected access to the application functionalities.
- **Real-time Monitoring:** A primary tk.Label (self.video_label) is used to continuously display the annotated video feed, providing immediate visual confirmation of student recognition and logging status.
- **Hierarchical Control:** The main AttendanceApp window uses a simple array of ttk.Button widgets for primary navigation (Start/Stop, Manage, View, Export).
- **Data Visualization:** The **View Attendance** screen utilizes the ttk.Treeview widget to present structured, sortable attendance records fetched from the MySQL database.
  - **Modular Forms:** tk.Toplevel windows are used for administrative forms (Student Management, Custom Range Export) to keep the main application uncluttered


Admin Login - Smart Attendance System

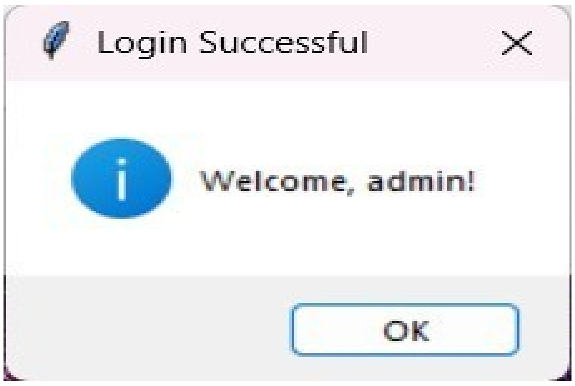
Smart Attendance System

Admin Login

Username: admin

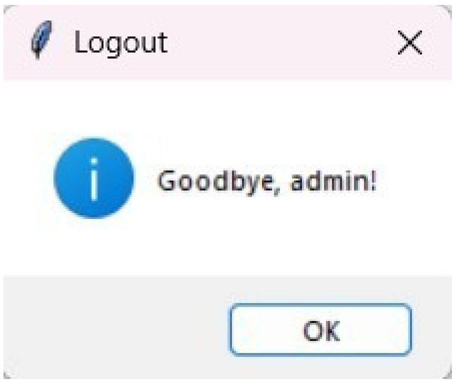
Password: *****

Figure 3: *System Entry: Administrative Login Interface*


Login Successful

Welcome, admin!

Figure 4: *Authentication Success Confirmation*


Logout

Goodbye, admin!

Figure 5: *System Termination Confirmation (Simulated Logout)*

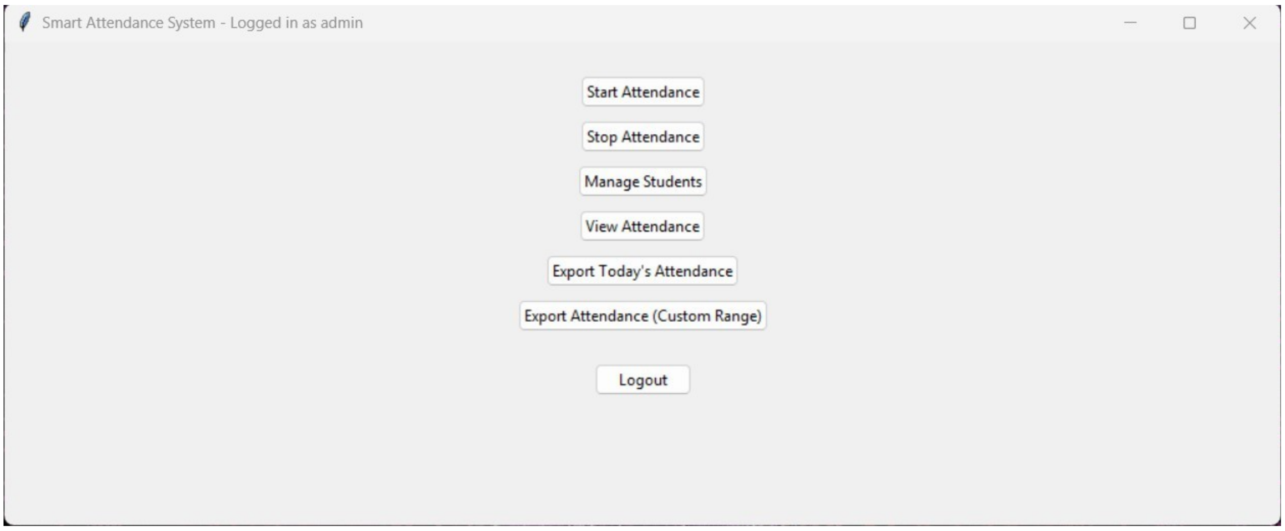

Smart Attendance System - Logged in as admin

Figure 6: *Main Control Dashboard (AttendanceApp)*

"The operational lifecycle of the Smart Attendance System is initiated and secured through a mandatory administrative session, as depicted across these figures. **Figure 3** illustrates the **System Entry: Administrative Login Interface**, which is managed by the AdminLogin class. This interface acts as the necessary security gate, requiring credentials to be input and subsequently validated against the secure admin table in the MySQL database via the verify_login method. Upon successful authentication, the administrator receives immediate feedback through the standard **Authentication Success Confirmation** message box (**Figure 4**), and the application transitions its state. This successful transition leads to the **Main Control Dashboard (AttendanceApp)** (**Figure 6**), which is the primary operational interface instantiated by the AttendanceApp class. This dashboard centralizes all administrative and operational controls, including the critical activation buttons ('Start Attendance', 'Stop Attendance') and access points for management and reporting. Finally, the session is terminated via the 'Stop Attendance' control, triggering the necessary resource cleanup (self.cap.release()) and returning the system state to the login requirement. The **System Termination Confirmation (Simulated Logout)** (**Figure 5**) represents the final user feedback confirming the end of the secure administrative session before the application closes or resets."

"The application’s core function revolves around two key interfaces: real-time monitoring and data maintenance. The **Real-Time Video Stream Monitoring** interface shows the live operational view during attendance tracking, directly driven by the AttendanceApp.video_stream method. The webcam feed is displayed on the self.video_label and is overlaid with bounding boxes and identification labels (Green for recognized, Red for unknown), as determined by the face_recognition.compare_faces pipeline **(Figure 7)** and **(Figure 8**, which likely refers to the unknown face monitoring figure**)**.

Complementary to the live stream is the administrative function managed via the **Student Management Module (CRUD Interface)**. This modal window, opened by the manage_students method, displays a **ttk.Treeview** list of current students, populated by the load_students function. The process of enrollment begins with the **Form Input for Adding a New Student** (**Figure 9,10,11**), where the administrator collects 'Name' and 'Roll No' and selects the student's reference image for processing. After execution of the database operation, the system provides immediate feedback through the **Success Notification: Student Creation/Deletion** (**Figure 12,13)**, confirming the successful persistence of data following the MySQL INSERT or DELETE commands."

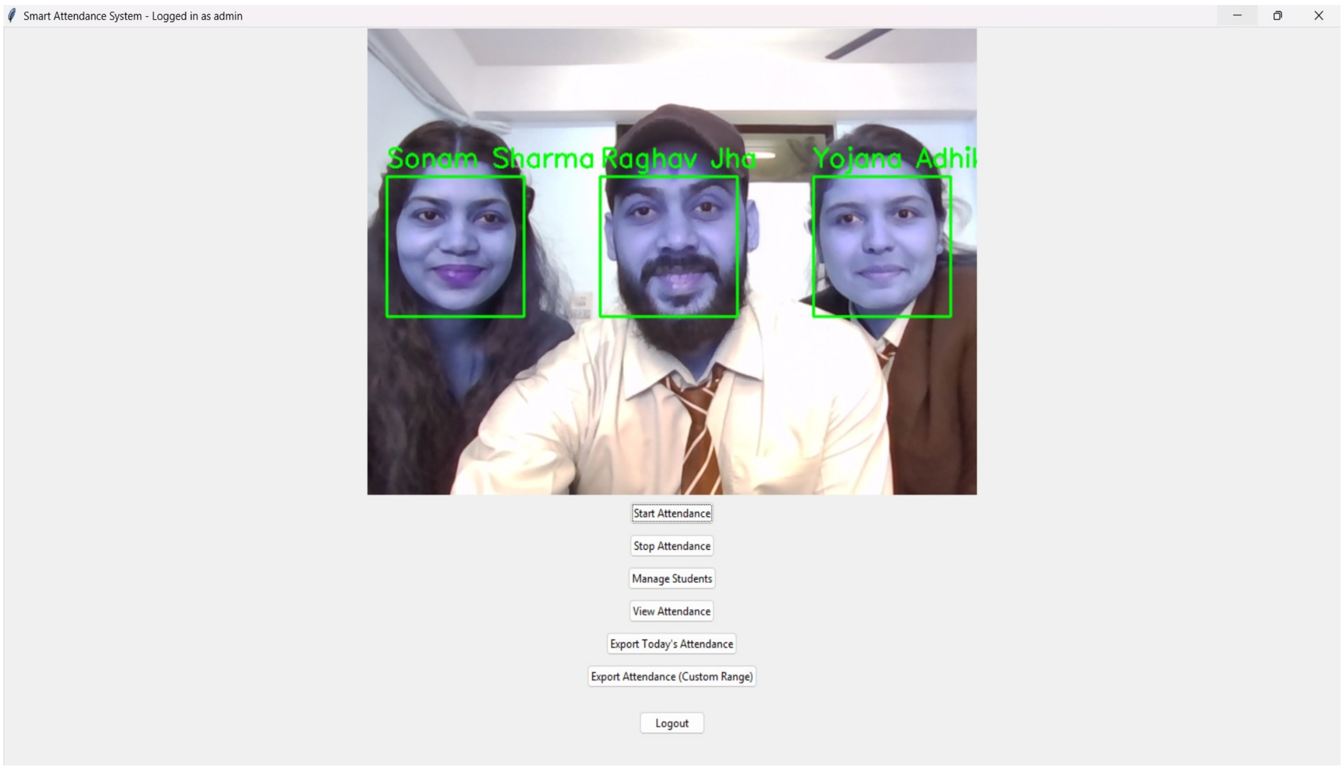

Smart Attendance System - Logged in as admin
Sonam Sharma
Raghav Jha
Yojana Adhil
Start Attendance
Stop Attendance
Manage Students
View Attendance
Export Today's Attendance
Export Attendance (Custom Range)
Logout

Figure 7: *Real-Time Video Stream Monitoring*

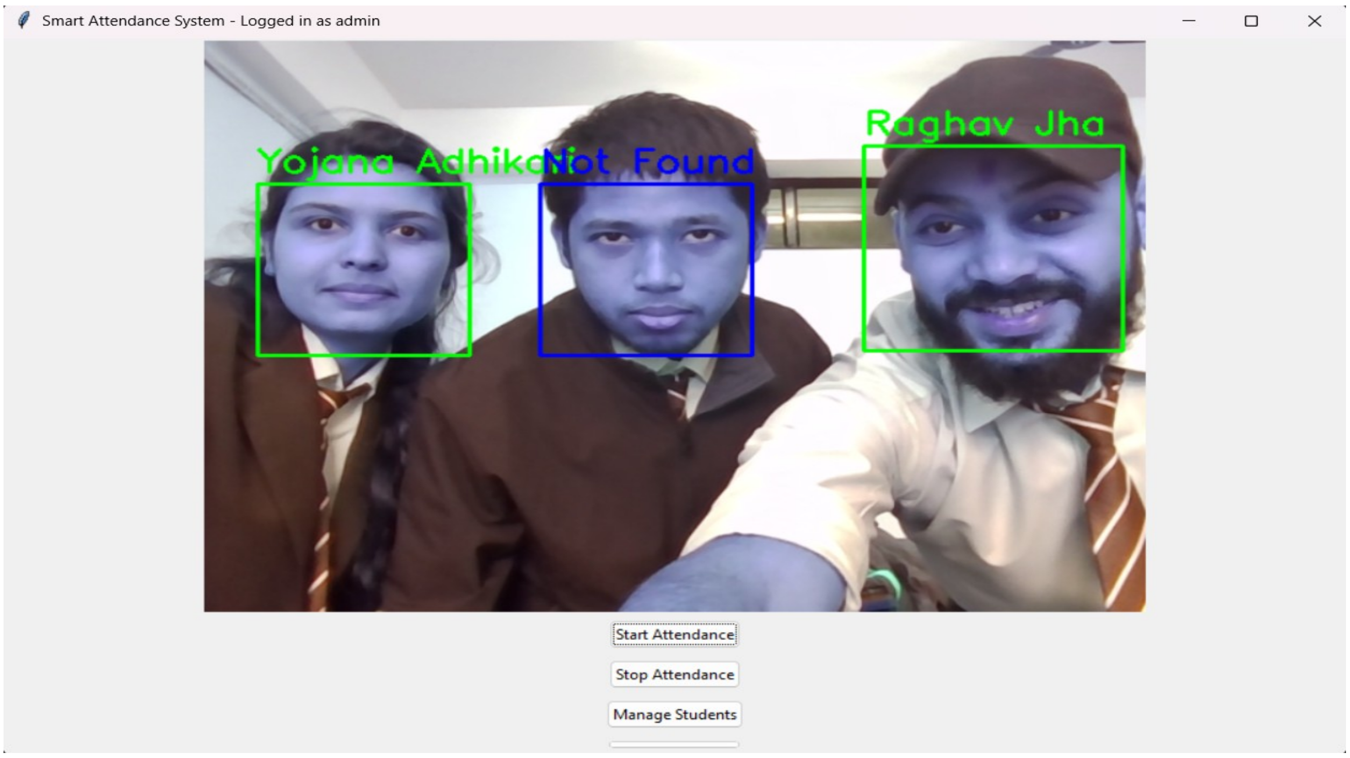
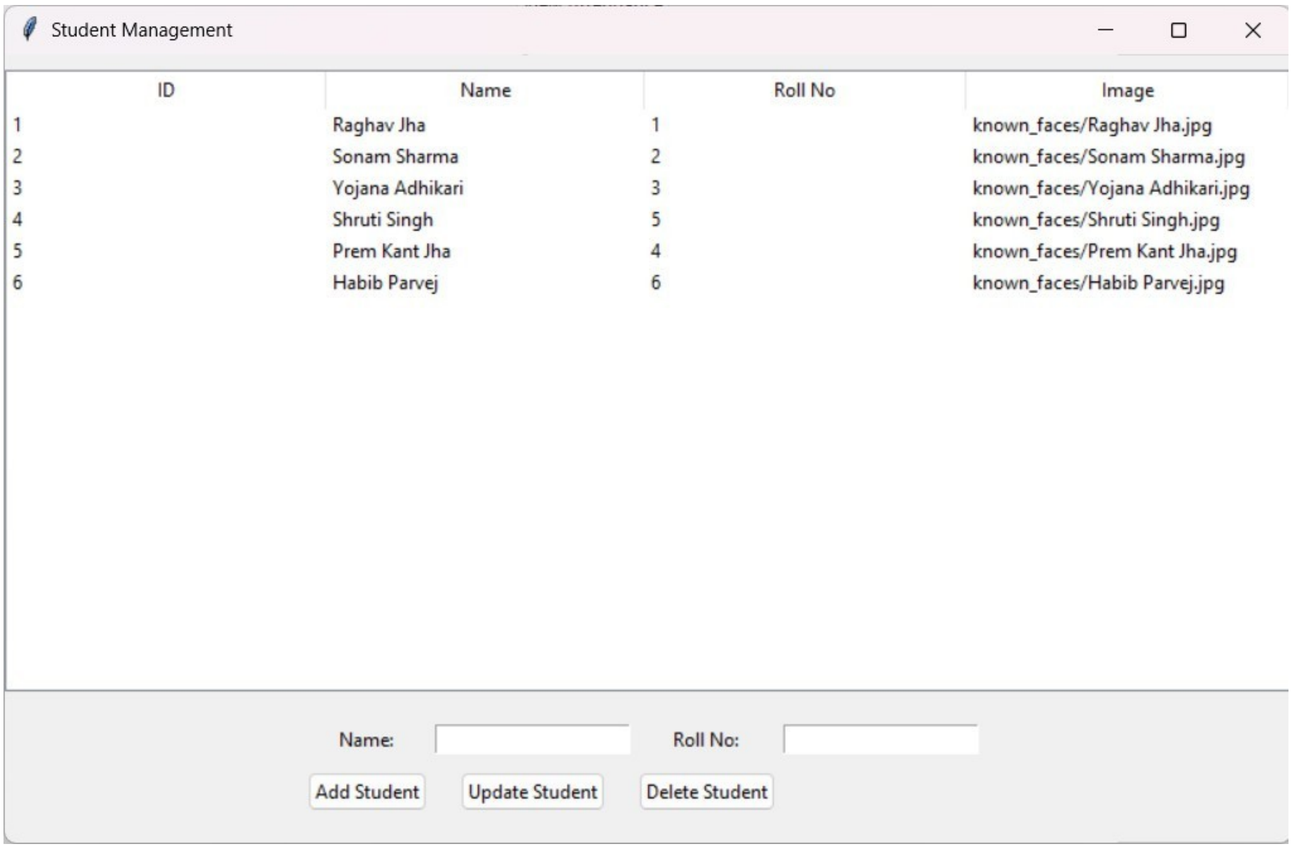
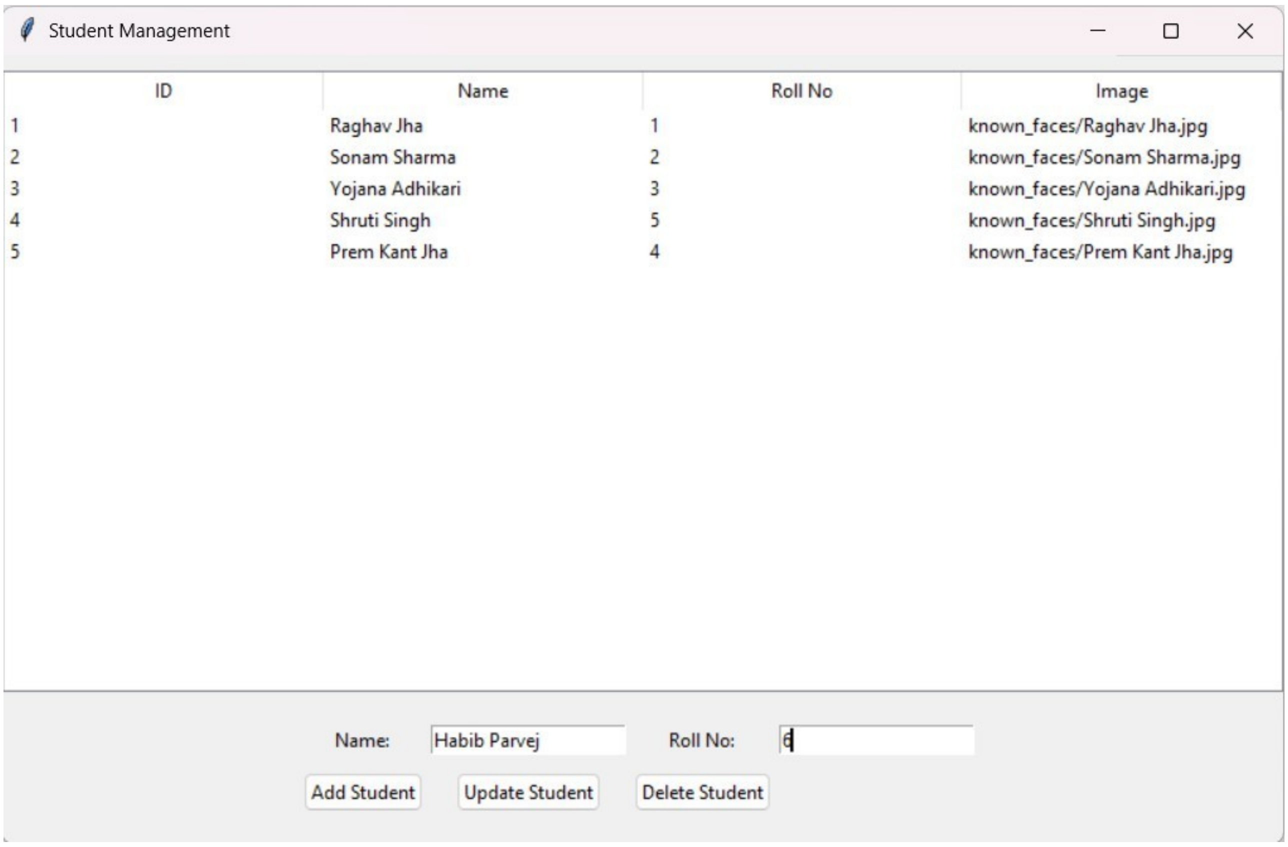


Figure 8: *Real-Time Monitoring: Detection of an Unidentified Subject*



ID	Name	Roll No	Image
1	Raghav Jha	1	known_faces/Raghav Jha.jpg
2	Sonam Sharma	2	known_faces/Sonam Sharma.jpg
3	Yojana Adhikari	3	known_faces/Yojana Adhikari.jpg
4	Shruti Singh	5	known_faces/Shruti Singh.jpg
5	Prem Kant Jha	4	known_faces/Prem Kant Jha.jpg
6	Habib Parvej	6	known_faces/Habib Parvej.jpg

Figure 9: *Student Management Module (CRUD Interface)*

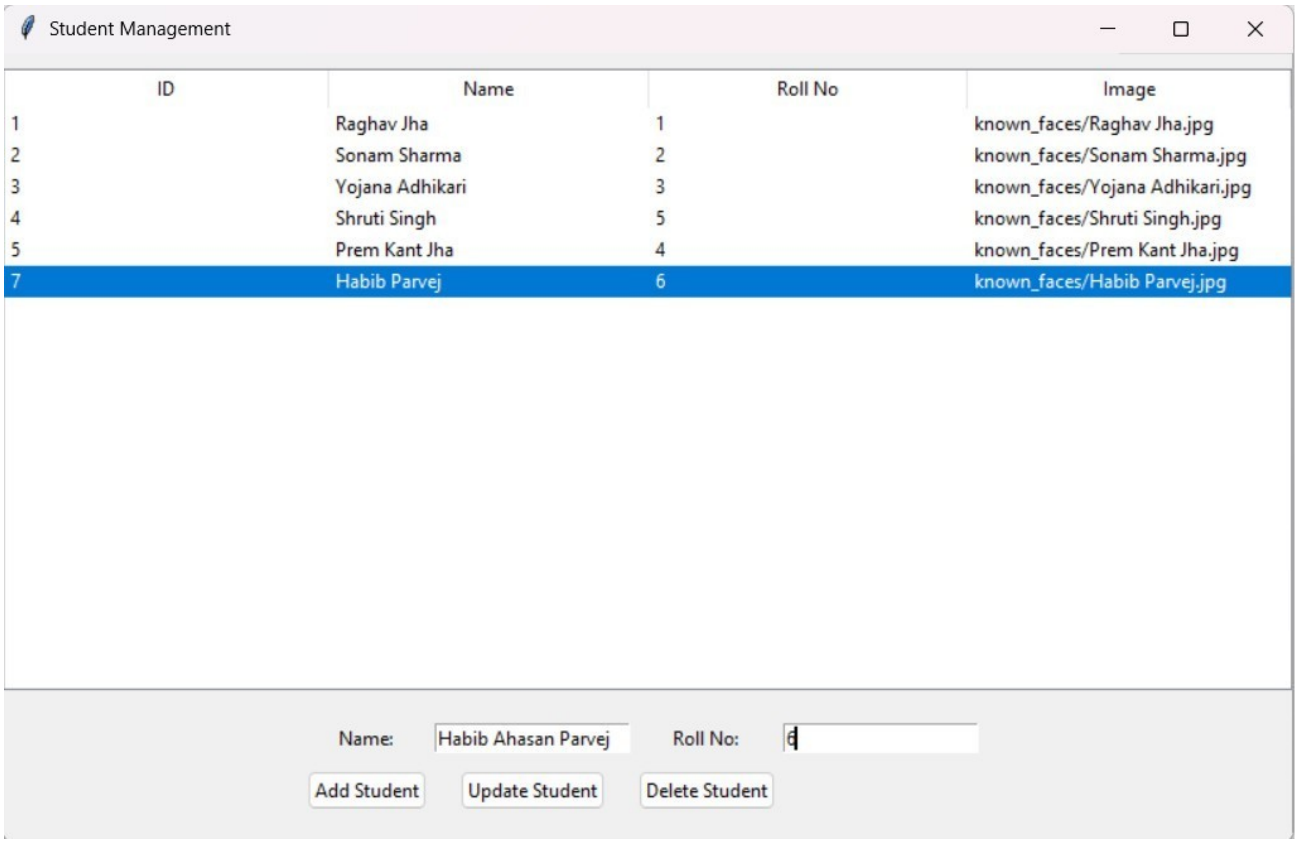


Student Management

ID	Name	Roll No	Image
1	Raghav Jha	1	known_faces/Raghav Jha.jpg
2	Sonam Sharma	2	known_faces/Sonam Sharma.jpg
3	Yojana Adhikari	3	known_faces/Yojana Adhikari.jpg
4	Shruti Singh	5	known_faces/Shruti Singh.jpg
5	Prem Kant Jha	4	known_faces/Prem Kant Jha.jpg

Name: Habib Parvej Roll No: 6

Figure 10: *Form Input for Adding a New Student*

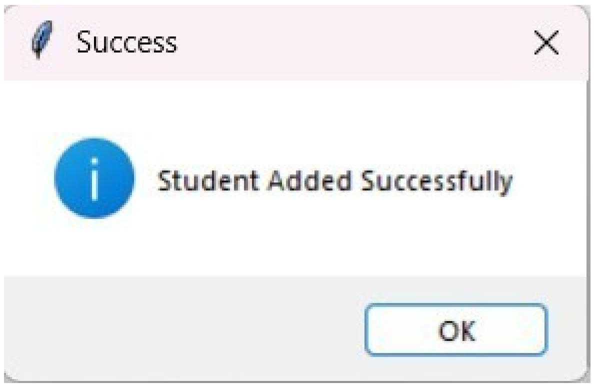


Student Management

ID	Name	Roll No	Image
1	Raghav Jha	1	known_faces/Raghav Jha.jpg
2	Sonam Sharma	2	known_faces/Sonam Sharma.jpg
3	Yojana Adhikari	3	known_faces/Yojana Adhikari.jpg
4	Shruti Singh	5	known_faces/Shruti Singh.jpg
5	Prem Kant Jha	4	known_faces/Prem Kant Jha.jpg
7	Habib Parvej	6	known_faces/Habib Parvej.jpg

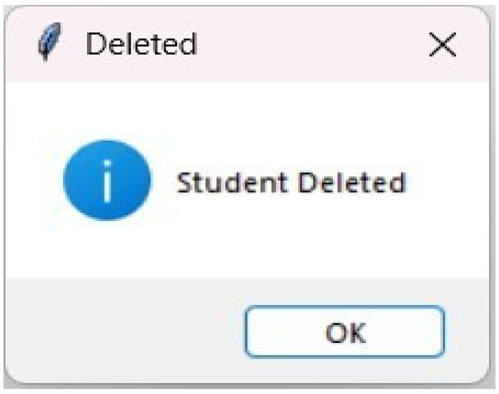
Name: Habib Ahasan Parvej Roll No: 6

Figure 11: *Real-Time Database Update in Treeview*


Success

Student Added Successfully

Figure 12: *Success Notification: Student Creation*


Deleted

Student Deleted

Figure 13: *Success Notification: Student Deletion*

7.2 Available library for UI design

While several powerful cross-platform GUI libraries are available (e.g., PyQt, Kivy, WxPython), the system adopted a simpler approach.

7.3 Library what we have used

- **Tkinter:** The standard, built-in Python GUI toolkit. It eliminates external dependency installation, ensuring maximum portability and easy deployment.
- **tkinter.ttk:** Provides themed widgets, improving the aesthetic appeal of the interface elements (buttons, treeview, etc.) compared to the default Tkinter widgets.

8. Database

8.1 Need of Database

Data persistence is mandatory for an attendance system. The database serves to:

- **Maintain Integrity:** Ensure records are accurate, consistent, and follow relational rules.
- **Security:** Store critical information (admin credentials, attendance logs) securely.
- **Scalability:** Provide a structured environment capable of handling increasing volumes of student data and historical logs over multiple academic years.

8.2 What database we have used

- **MySQL:** A widely-used, mature, and reliable open-source Relational Database Management System (RDBMS). It was selected for its performance, transactional integrity, and stability in handling large datasets.

8.3 Structure of database

The logical and physical structure of the MySQL database (smart_attendance) is defined as follows:

Table	Purpose	Fields and Structure
students	Stores persistent student profile data.	id (INT, PK, AUTO_INC), name (VARCHAR), roll_no (VARCHAR), image_path (VARCHAR).
attendance	Stores time-based attendance events.	id (INT, PK, AUTO_INC), student_id (INT, FK to students.id), date (DATE), first_time (TIME), last_time (TIME).
admin	Stores system administrator credentials.	username (VARCHAR), password (VARCHAR).

Table: students

Column	Type	Description
id	INT (Primary Key)	Auto incremented ID
name	VARCHAR (100)	Student’s name
roll_no	VARCHAR (50)	Unique roll number
image_path	TEXT	Path of the face image

Table: attendance

Column	Type	Description
id	INT (Primary Key)	Auto incremented
student_id	INT	Foreign key (from students.id)
date	DATE	Attendance date
time_in	TIME	First time detected
time_out	TIME	Last time detected

Table: admin

Column	Type	Description
id	INT (Primary Key)	Auto-incremented ID
username	VARCHAR (50)	Admin username (login)
password	VARCHAR (255)	Admin password (hashed)

9. Report

9.1 Need of Report

Reports transform raw logged data into actionable information, essential for:

- **Auditing:** Verifying compliance and reviewing historical attendance accuracy.
- **Analysis:** Identifying patterns of presence/absence for academic intervention.
- **Integration:** Providing data in a standard format (.xlsx) for transfer to external HR or school management systems.

9.2 Types of report

The reporting module is designed to provide administrative users with flexible data extraction capa bilities, leveraging SQL filtering and the openpyxl library for professional output in the standard `.xlsx` format. The system supports a comprehensive range of reports, derived from the core `attendance` and students tables, all relying on the logged First Time and `Last Time` data points.

9.2.1. Core Log Reports (Directly Exported via Functions)

These reports are generated directly by functions in the Python code (`export_today_attendance` and `export_range_attendance`).

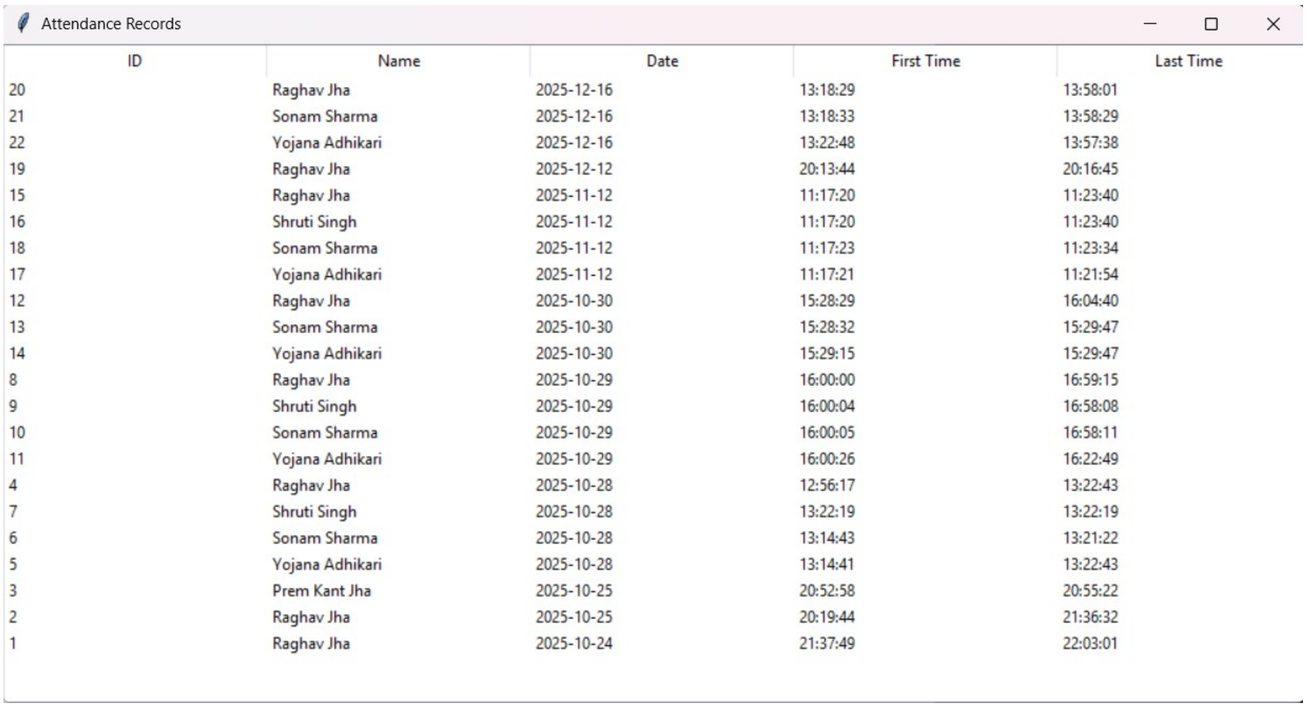
- **Daily Attendance Report (`export_today_attendance`):**
  - **Purpose:** Provides a real-time, granular log of all attendance activity recorded during the current operational day.
  - **Mechanism:** Data is filtered using the comparison logic: `WHERE a.date = CURRENT_DATE()`.
  - **Content:** Student Name, Roll Number, Date, `First Time`**, and** `Last Time.`
- **Attendance Report over a Period (`export_range_attendance`):**
  - **Purpose:** Consolidates all logged attendance data over a specific, administrator-defined period for historical auditing.
  - **Mechanism:** Data is filtered using the inclusive range operator based on user input: `WHERE a.date BETWEEN %s AND %s`.
  - **Content:** Aggregates records across the specified time window, including Student Name, Roll Number, Date, `First Time`, and `Last Time`
- **Exportable Reports:**
  - **Purpose:** All reports generated by the system are instantly formatted for export (as `.xlsx` files using `openpyxl`) for official use, auditing, and integration with external-

systems.

9.2.2. Derived Analysis Reports (Calculated from Exported Data)

These specific reports are achievable using the exported raw data, even though the calculation logic itself is not present in the Python code and must be performed externally (e.g., within Excel).

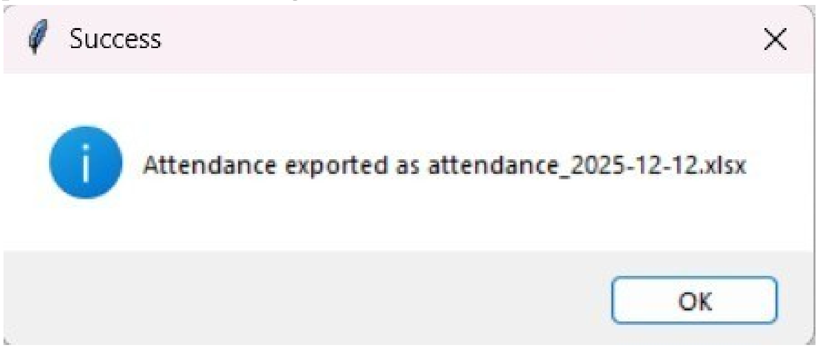
- **Individual Attendance Report:**
    - **Purpose:** Shows attendance details for a specific individual over a selected period.
    - **Mechanism:** Achieved by filtering the **Custom Range Report** output based on the specific `Roll No` or `Student Name` column in the spreadsheet.
- **Biometric / Face Recognition Log Report:**
    - **Purpose:** Maintains a log of verified biometric recognition events.
    - **Mechanism:** The **Daily Attendance Report** implicitly serves this function, as every entry represents a successful, timestamped biometric verification event. The `First Time` and `Last Time` are the core recognition log stamps.
- **Late Entry Report:**
    - **Purpose:** Identifies individuals who arrived after a scheduled official start time.
    - **Mechanism:** Derivable from the raw data by comparing the logged `First Time` column in the exported report against a predefined organizational threshold (e.g., 9:00 AM).
- **Absent Report:**
    - **Purpose:** Lists individuals absent on a particular date.
    - **Mechanism:** Derivable by cross-referencing the total student roster (available in the `students` table) against the `Name` entries found in the **Daily Attendance Report**.
- **Attendance Percentage Report:**
    - **Purpose:** Calculates overall attendance percentages for eligibility or compliance purposes.
    - **Mechanism:** Derivable from the **Custom Range Report** by calculating the ratio of unique attendance days logged per student against the total number of working days in the specified period.



ID	Name	Date	First Time	Last Time
20	Raghav Jha	2025-12-16	13:18:29	13:58:01
21	Sonam Sharma	2025-12-16	13:18:33	13:58:29
22	Yojana Adhikari	2025-12-16	13:22:48	13:57:38
19	Raghav Jha	2025-12-12	20:13:44	20:16:45
15	Raghav Jha	2025-11-12	11:17:20	11:23:40
16	Shruti Singh	2025-11-12	11:17:20	11:23:40
18	Sonam Sharma	2025-11-12	11:17:23	11:23:34
17	Yojana Adhikari	2025-11-12	11:17:21	11:21:54
12	Raghav Jha	2025-10-30	15:28:29	16:04:40
13	Sonam Sharma	2025-10-30	15:28:32	15:29:47
14	Yojana Adhikari	2025-10-30	15:29:15	15:29:47
8	Raghav Jha	2025-10-29	16:00:00	16:59:15
9	Shruti Singh	2025-10-29	16:00:04	16:58:08
10	Sonam Sharma	2025-10-29	16:00:05	16:58:11
11	Yojana Adhikari	2025-10-29	16:00:26	16:22:49
4	Raghav Jha	2025-10-28	12:56:17	13:22:43
7	Shruti Singh	2025-10-28	13:22:19	13:22:19
6	Sonam Sharma	2025-10-28	13:14:43	13:21:22
5	Yojana Adhikari	2025-10-28	13:14:41	13:22:43
3	Prem Kant Jha	2025-10-25	20:52:58	20:55:22
2	Raghav Jha	2025-10-25	20:19:44	21:36:32
1	Raghav Jha	2025-10-24	21:37:49	22:03:01

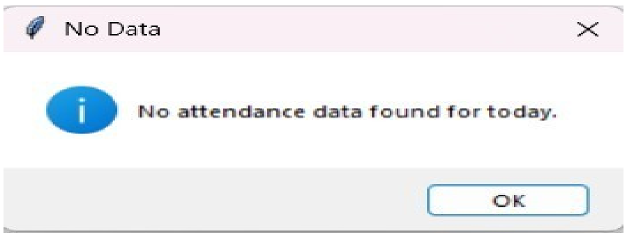
Figure 14: *Historical Attendance Records Interface*

"The system allows administrators to view all historical attendance logs via the view_attendance method, displaying records (ID, Name, Date, Times) in a **ttk.Treeview** table using a SQL JOIN query (**Figure 14**). For permanent records, the system exports data using openpyxl. Successful reports trigger a notification (**Figure 15,17**) confirming file creation (attendance_YYYY-MM-DD.xlsx), while a failed query results in a 'No Data Found' notification (**Figure 16**).(**Figure 18)** illustrates the final output of the `export_today_attendance` function**:**the Daily Attendance Spreadsheet showcasing the verified student data.



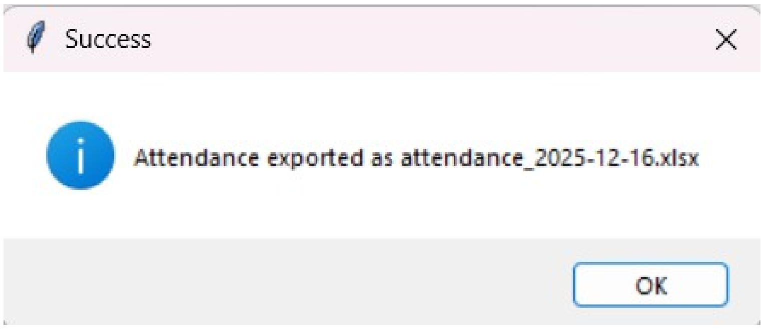
Attendance exported as attendance_2025-12-12.xlsx

Figure 15: *Notification: Successful Daily Report Export*



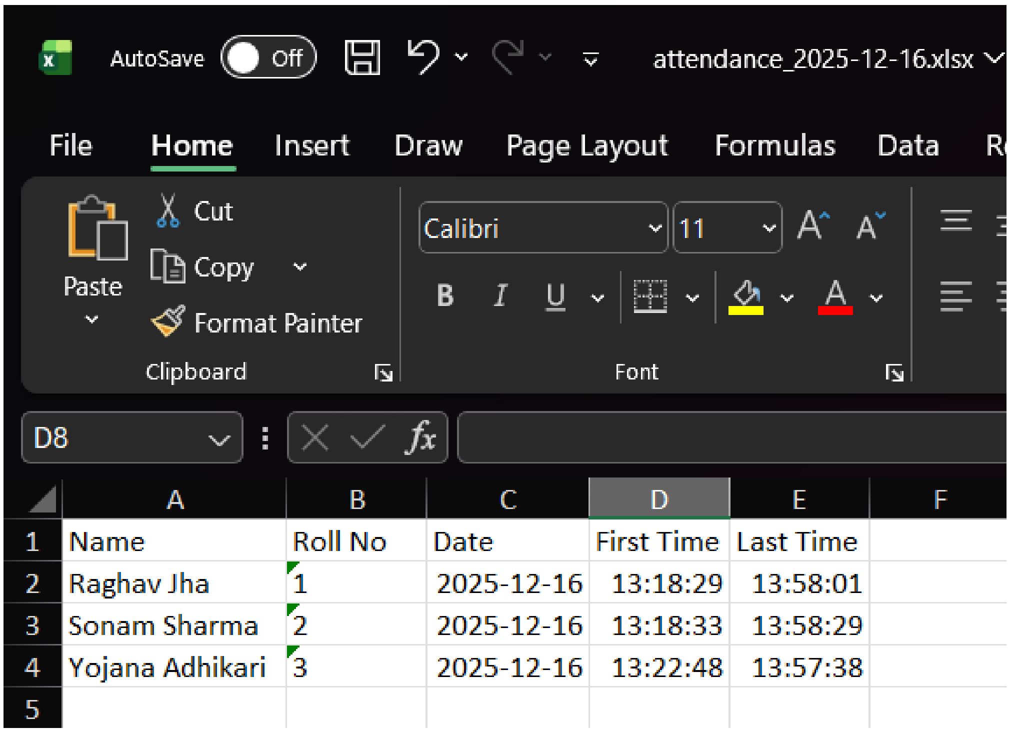
No attendance data found for today.

Figure 16: Notification: No Attendance Data Found



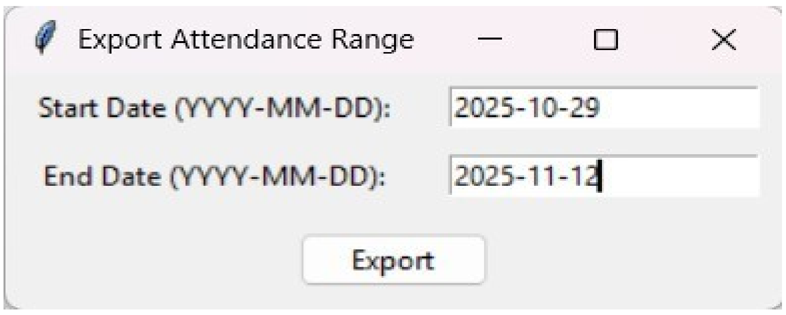
Attendance exported as attendance_2025-12-16.xlsx

Figure 17: Notification: Successful Daily Report Ex-



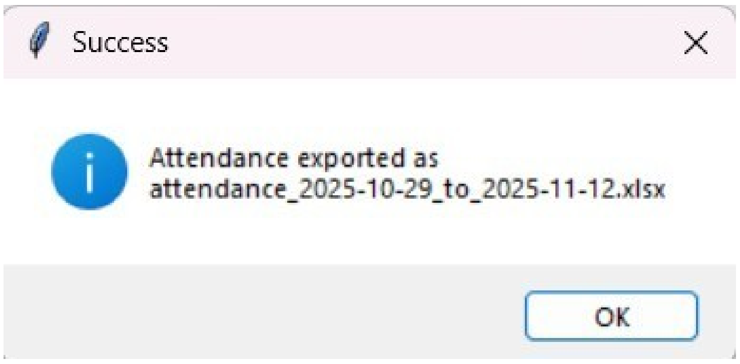
	A	B	C	D	E	F
1	Name	Roll No	Date	First Time	Last Time	
2	Raghav Jha	1	2025-12-16	13:18:29	13:58:01	
3	Sonam Sharma	2	2025-12-16	13:18:33	13:58:29	
4	Yojana Adhikari	3	2025-12-16	13:22:48	13:57:38	
5						

Figure 18: Final Report Output: Daily Attendance Spreadsheet



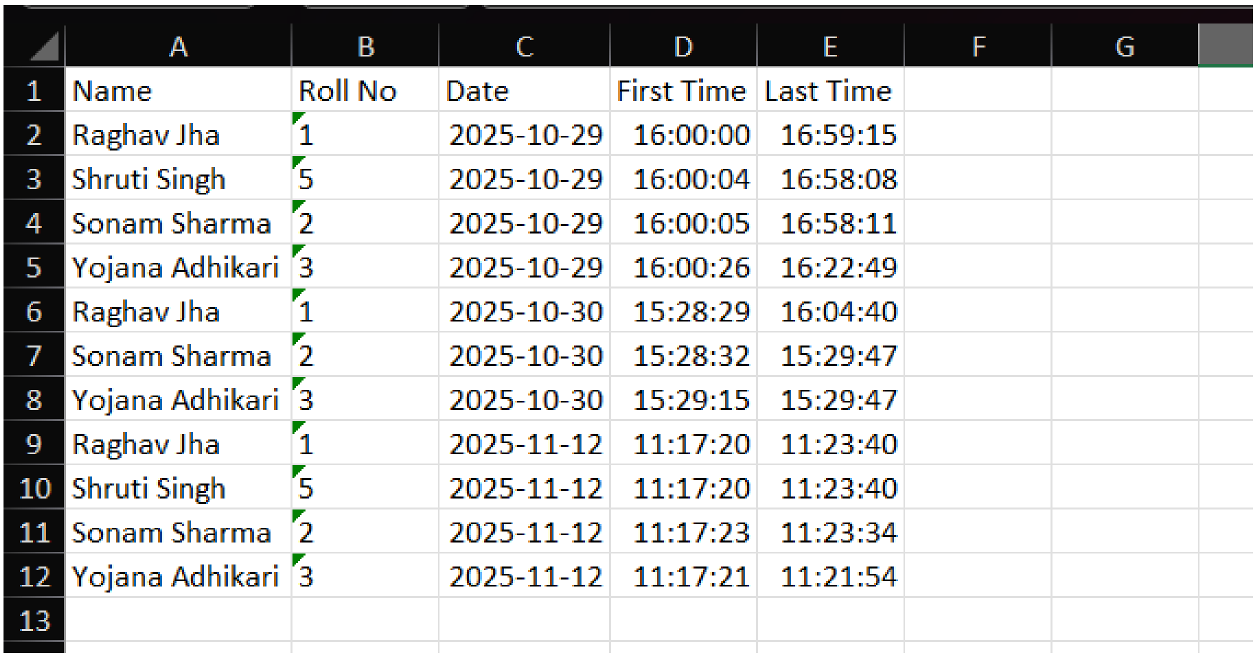
Start Date (YYYY-MM-DD):	2025-10-29
End Date (YYYY-MM-DD):	2025-11-12

Figure 19: Report Export Control: Custom Date RangeForm



Attendance exported as
attendance_2025-10-29_to_2025-11-12.xlsx

Figure 20: Notification: Successful Daily Report Export



	A	B	C	D	E	F	G
1	Name	Roll No	Date	First Time	Last Time		
2	Raghav Jha	1	2025-10-29	16:00:00	16:59:15		
3	Shruti Singh	5	2025-10-29	16:00:04	16:58:08		
4	Sonam Sharma	2	2025-10-29	16:00:05	16:58:11		
5	Yojana Adhikari	3	2025-10-29	16:00:26	16:22:49		
6	Raghav Jha	1	2025-10-30	15:28:29	16:04:40		
7	Sonam Sharma	2	2025-10-30	15:28:32	15:29:47		
8	Yojana Adhikari	3	2025-10-30	15:29:15	15:29:47		
9	Raghav Jha	1	2025-11-12	11:17:20	11:23:40		
10	Shruti Singh	5	2025-11-12	11:17:20	11:23:40		
11	Sonam Sharma	2	2025-11-12	11:17:23	11:23:34		
12	Yojana Adhikari	3	2025-11-12	11:17:21	11:21:54		
13							

Figure 21: Final Report Output (Excel Spreadsheet)

"Custom report generation begins with the **Custom Date Range Form (Figure 19)**, displayed by the export_range_attendance method, which captures the Start and End Dates for the query boundary. Upon successful execution of the WHERE a.date BETWEEN... SQL query, a confirmation box (**Figure 20**) notifies the user of the **Successful Custom Range Report Export**. The final output is the **Excel Spreadsheet (Figure 21)**, generated using the openpyxl library, which contains the structured attendance data retrieved from MySQL, including the required header row and time logs."

10. Conclusion

The Smart Attendance System project successfully achieved its core objectives by delivering a robust Minimum Viable Product (MVP) that automates and secures the student attendance process. By replacing error-prone manual methods, the system leverages Deep Metric Learning and Computer Vision to establish non-repudiable biometric verification, now enhanced by a foundational layer of Liveness Detection to mitigate basic spoofing attempts. The two-tiered architecture—combining a user-friendly Tkinter GUI for administrative control and a persistent MySQL backend for transactional logging—ensures high data integrity and consistency. The application accurately captures auditable time stamps (In-Time and Out-Time) and provides flexible reporting through exportable `.xlsx` files, thereby demonstrating a successful proof-of-concept for integrating advanced security and streamlined administrative utility. The system provides a solid foundation for institutional attendance management.

Source Code Repository

The complete source code and associated files for this project are publicly available for review and future development at: https://github.com/Raghavjha6/SmartAttendanceSystem

11. Challenges

As an MVP (Minimum Viable Product) and due to architectural choices, the following challenges are present in the current deployed system:

1. **Lack of Liveness Detection:** The system relies purely on 2D image matching. It does not contain any logic (e.g., blink detection, head movement analysis) to prevent spoofing via printed photos or video replays, representing a security vulnerability.
2. **No Multi-threading for GUI:** The video_stream and ML processing runs on the primary Tkinter thread (root.after). Under heavy load or on slower machines, the GUI may experience freezing or reduced responsiveness.
3. **No Duration Enforcement:** The original requirement of marking attendance only after a 40-minute presence is **not implemented**. The current system only logs first_time and last_time; the final attendance status (Present/Absent) is left to manual calculation based on the exported times.
4. **Hardcoded Credentials:** Database access credentials (host="localhost", user="root", password="root") are hardcoded, which is a significant security risk for production deployment.
5. **Single Image Registration:** The manage_students function only allows the upload and storage of a single image per student, which may reduce the accuracy and robustness of the recognition model across varied lighting and poses.

12. Future Scope

While the current implementation fulfills the primary functional requirements (identification, logging, and reporting), the system can be significantly enhanced to address identified limitations and fully realize the comprehensive vision of a complete organizational management platform. The future scope focuses on three key areas: security, computation, and architecture.

12.1 Advanced Security and Robustness

- **Advanced Passive Liveness Techniques:** To improve the system's defense against sophisticated presentation attacks, future work should focus on integrating passive, multimodal Liveness Detection (e.g., analyzing subtle texture variations, depth cues, or advanced spatio-temporal features) to eliminate the need for active user compliance (like blinking or head movement).

- **Secure Credential Management:** Implement industry-standard security practices by removing hardcoded database credentials and employing secure configuration files or environment variables.

12.2 Advanced Computational Logic

- **Duration Enforcement and Status Calculation:** Implement the required business logic to automatically calculate the duration between first time and last time. The system should then update the final attendance status (e.g., "Present," "Absent," or "Short Duration") based on a minimum presence requirement (e.g., the planned 40-minute rule).

- **Working Hour Calculation:** Extend the reporting module to automatically compute derived metrics, such as total hours attended per day or aggregate hours per student over a defined month or semester, supporting detailed time auditing.

12.3 Architectural Scalability

- **Multi-threading Implementation:** To prevent GUI freezing under heavy ML load, the `video_stream` and face processing should be migrated to a dedicated worker thread, allowing the main Tkinter thread to remain responsive.

- **Web Application Interface:** Migrate the Tkinter GUI to a **Web Application** platform (e.g., using Flask/Django or a JavaScript framework). This would allow for centralized deployment, remote monitoring, and greater accessibility for administrative users across an organization's network.

REFERENCES

This project utilizes established open-source libraries and industry-standard technologies. The core references that enable the system's functionality are listed below:

1. **Dlib and Face Recognition Model:** The underlying architecture for the 128-D face embeddings is based on the Dlib library's implementation of a deep residual network (ResNet).

   - *Reference:* King, D. E. (2009). Dlib-ML: A C++ Library for Machine Learning. *Journal of Machine Learning Research*, 10, 1755-1758. https://www.jmlr.org/papers/volume10/king09a/king09a.pdf

2. **`face_recognition` Library:** The high-level Python API used for simplified face detection, encoding, and comparison logic.

   - *Source:* Adam Geitgey. Available at: https://github.com/ageitgey/face_recognition

3. **OpenCV (Open Source Computer Vision Library):** Used for video stream management (`cv2.VideoCapture`), frame processing, optimization, and visualization.

   - *Official Documentation:* OpenCV Documentation. Available at: https://docs.opencv.org/

4. **MySQL Database:** The relational database management system used for secure data persistence of student profiles, admin credentials, and all attendance logs.

   - *Official Documentation:* MySQL Documentation. Available at: https://dev.mysql.com/doc/

5. **`mysql.connector`:** The standard Python driver used to establish a persistent connection and execute all SQL transactions with the MySQL database.

   - *Official Documentation:* Python Developer Guide for MySQL Connector. Available at: https://dev.mysql.com/doc/connector-python/en/

6. **Tkinter and `tkinter.ttk`:** The standard Python GUI toolkit used to develop the cross-platform desktop application, including the administration interface and real-time video display.

   - *Source:* Python Standard Library Documentation. Available at: https://docs.python.org/3/library/tkinter.html

7. **`openpyxl`:** The Python library used to handle and generate the final administrative reports in the standardized Microsoft Excel (`.xlsx`) format.

   - *Official Documentation:* openpyxl Documentation. Available at: https://openpyxl.readthedocs.io/en/stable/

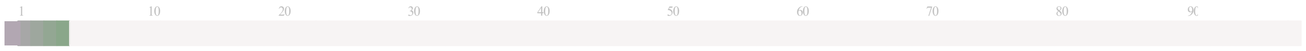
The Report is Generated by DrillBit Plagiarism Detection Software

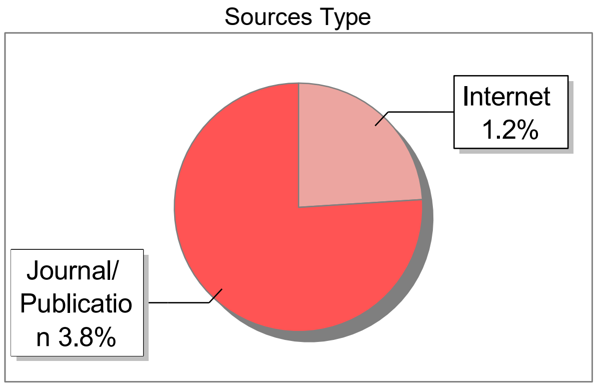
Submission Information

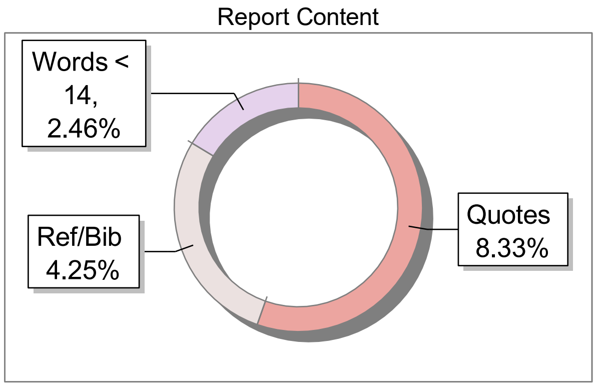
Author Name	Yojana Adhikari
Title	Smart Attendance System
Paper/Submission ID	4974415
Submitted by	hod_mca@sittechno.org
Submission Date	2025-12-17 17:18:07
Total Pages, Total Words	31, 5603
Document type	Project Work

Result Information

Similarity **5 %**







Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

5	20	A	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	alameenedu.co.in	1	Publication
2	dspace.srmist.edu.in	1	Publication
3	wjaets.com	1	Publication
4	Thesis Submitted to Shodhganga Repository	<1	Publication
5	Applied GPU Computing in Computational Biology to Study Disparate Multicellular By Avimanyu Bandyopadhyay, Yr-2020,2,9	<1	Publication
6	journal.untar.ac.id	<1	Internet Data
7	ir.juit.ac.in8080	<1	Publication
8	wjaets.com	<1	Publication
9	artsdocbox.com	<1	Internet Data
10	worldwidescience.org	<1	Internet Data
11	arxiv.org	<1	Publication
12	www.freepatentsonline.com	<1	Internet Data
13	www.ncbi.nlm.nih.gov	<1	Internet Data

14	IEEE Standard for Design and Verification of Low Power Integrated Circuits by -	<1	Publication
15	Thesis submitted to shodhganga - shodhganga.inflibnet.ac.in	<1	Publication
16	www.ecpe.nu.ac.th	<1	Publication
17	www.ijfans.org	<1	Publication
18	www.irjmets.com	<1	Publication
19	www.jetir.org	<1	Publication
20	www.mdpi.com	<1	Internet Data