NEW SUMMIT COLLEGE

Affiliated To

Tribhuvan University

Institute of Science and Technology

A Final Year Project Report

On

“Smart Attendance System”

Submitted to

Department of Computer Science and Information Technology

New Summit College

In partial fulfillment of the requirement of the Bachelor Degree in Computer Science and Information Technology

Under The Supervision of

Abhishek Koirala

Submitted By:

Bibek Parajuli [23424/076]

Saman Raut [23450/076]

Sanju Thanait [23454/076]

March, 2024

Acknowledgement

It is a great pleasure to have the opportunity to extend our heartfelt gratitude to everyone who helped us throughout this project. We are profoundly grateful to our supervisor, **Mr. Abhishek Koirala**, for his expert guidance, continuous encouragement and ever willingness to spare time from his otherwise busy schedule for the project's progress reviews. His continuous inspiration has made us complete this project and achieve its target.

We would also like to express our deepest appreciation to **Mr. Chok Raj Dawadi**, Principal, New Summit College, for his constant motivation, support and for providing us with a suitable working environment.

We sincerely acknowledge direct and indirect help, suggestions and feedback offered by our colleagues before, during and after the development and implementation of the project. At last, our special thanks go to all staff members of the CSIT department at New Summit College who kindly extended their hands in making this project work a success.

Abstract

The Smart Attendance System introduces a cutting-edge solution for attendance management in Nepal, challenging traditional methods. This system leverages the power of Python, Flask, HTML, CSS, and JavaScript, providing a seamless and efficient experience. Fueled by Python's Flask framework, the system ensures robustness and scalability, while HTML, CSS, and JavaScript contribute to a user-friendly interface. This platform addresses the shortcomings of traditional attendance systems by incorporating smart features that automate and streamline the attendance tracking process. Through the integration of Python and Flask, the system offers a dynamic and responsive interface, allowing users to conveniently manage attendance records.

Key features of the Smart Attendance System include real-time tracking, automated data analysis, and personalized dashboards for administrators and users. The use of HTML, CSS, and JavaScript enhances the user experience, making it intuitive and engaging. The system emphasizes inclusivity and collaboration by providing a comprehensive solution for attendance management. The Smart Attendance System aspires to revolutionize traditional attendance tracking in Nepal, offering a smart, efficient, and user-friendly platform that ensures accurate attendance data while promoting real-time interaction and collaboration.

Keywords: *Attendance, Tracking, Records, Automation, Robustness*

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI: Artificial Intelligence

CSS: Cascading Style Sheet

HOG: Histograms of Oriented Gradients

IDE: Integrated Development Environment

JS: Java Script

LBPH: Local Binary Pattern Histogram

Chapter 1
Introduction

1.1. Introduction

An effective attendance management system must be implemented in any organization to ensure that the attendance of individuals is tracked accurately and efficiently. In recent years, technological advances have changed the methods of attendance recording, and one such innovation is the attendance camera. With the advancement of technology, we can now control and manage attendance in a variety of settings, including educational institutions, offices, and public events. To record attendance, the attendance system employs advanced computer vision technology and facial recognition systems. It captures and verifies identity by analyzing facial features, providing a secure and convenient method of tracking attendees. Cameras will be used to capture individual by using face recognition algorithm. Face recognition algorithms are applied to match the detected faces against a database of known individuals, enabling the system to identify people and update the attendance list if known face is detected. Cameras in collaboration with computer vision and machine learning algorithms will work together to provide a simple and accurate method of attendance management, ensuring accuracy and reliability.

1.2. Problem Statement

The main issue is the modernization of attendance systems that have been in use for many years in educational institutions and other institutions. Existing methods, which are primarily manual and time-consuming, hamper efficiency and frequently produce incorrect results. The goal is to modify this trend through the use of Camera technology and computer vision. The system aims to automate face detection using advanced facial recognition algorithms, providing simple and accurate solutions for recording attendance. The difficulties include performing facial recognition and recognizing accuracy, which can work well in a variety of environmental conditions. In educational institutions, organizations, and workplaces, successful implementation promises to improve attendance management, optimize resource utilization, and boost productivity.

1.3. Objectives

The objective of smart attendance system using facial recognition are:

i. To automate attendance recording through Camera-based computer vision.

ii. To enhance accuracy by utilizing advanced face detection and recognition algorithms, ensuring real-time updates and precise attendance tracking.

1.4. Scope and Limitation

1.4.1. Scope

The scope of the smart facial attendance encompasses a comprehensive attendance using Cameras for facial recognition in Nepal. Key areas within the scope of this system include:

i. **Efficiency and Accuracy:** Implementation of a dynamic pricing model allowing buyers to actively participate in adjusting product prices through bidding. Integration of a bidding system that enhances user engagement and offers a competitive and exciting shopping experience.

ii. **Convenience:** The system can offer a convenient and non-intrusive way for individuals to mark attendance without the need for physical contact, such as fingerprint scanners or RFID cards.

iii. **Real-time Monitoring:** The system can enable real-time monitoring of attendance data, allowing for quick identification of attendance patterns and potential issues.

iv. **Security:** Facial recognition adds an additional layer of security, as it is harder to forge or manipulate compared to traditional methods like paper attendance sheets or cards.

v. **Integration with Other Systems:** The system can be integrated with other databases and systems, facilitating seamless attendance management and record-keeping.

vi. **User Identification:** Facial recognition allows for individualized identification, reducing the chances of attendance fraud or proxy attendance.

1.4.2. Limitation

i. **Accuracy Challenges:** Environmental factors like lighting conditions, camera angles, and changes in appearance (e.g., facial hair, glasses) can affect the accuracy of facial recognition.

ii. **Limited Coverage**: The effectiveness of the system may be limited if individuals are not facing the cameras, leading to potential gaps in attendance tracking.

1.5. Development Methodology

Iterative Waterfall Model:

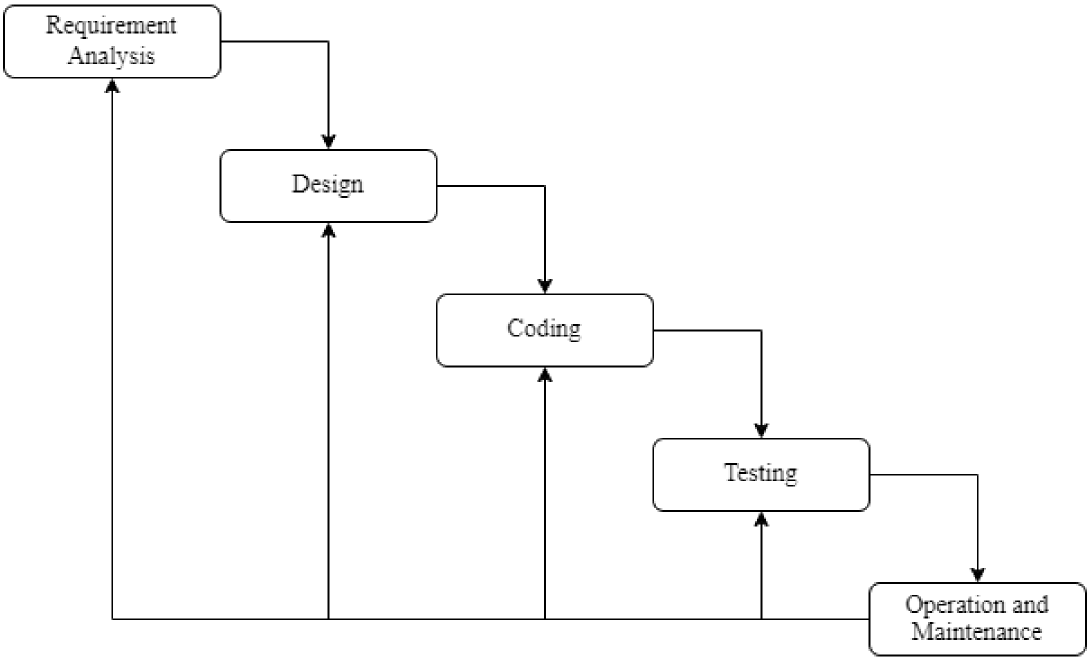


Figure 1.1: Iterative Waterfall Model

1. **Requirements analysis:** It is the first step of iterative waterfall model. During this phase, the system's desires are precisely defined. In this phase we gathered and analyzed all the requirements that was required to build the Smart Attendance System.
2. **Design:** The design phase involves creating the architectural and system design. In this stage we designed the system such that it can be easily used and adapted by different organizations.
3. **Coding:** The coding phase is carried out after the design is complete. Here, we used Python programming language combining different libraries and packaged like Dlib,

to built the entire system. We implemented Cosine similarity algorithm to mark attendance of individuals.

4. **Testing:** Testing is performed after the completion of each iteration. The Unit testing of the system is done in this stage.
5. **Operation and maintenance:** The system is operated in the real world; maintaining it regularly and making it better based on user feedback, new needs, and changes in technology.

1.6. Report Organization

This report is organized into five chapters:

Chapter 1: "Introduction"- This chapter introduces the problem statement, objectives and limitations of the project.

Chapter 2: "Requirement and Feasibility Analysis"- This chapter describes about the functional and non-functional requirements, economic feasibility, technical feasibility, operational feasibility and scheduling feasibility.

Chapter 3: "System Design"- This chapter introduces about the system and interface design of the project app.

Chapter 4: "Implementation and Testing- This chapter clearly illustrates the methods and tools used to implement the project.

Chapter 5: "Conclusion and Future Works"- This is the final chapter that concludes the project and talks about our future plans with the project.

Chapter 2

Background Study and Literature Review

2.1. Background Study

Attendance management is key to organizational efficiency, but traditional manual methods have proven time-consuming and inaccurate Responding to these challenges, recent technological advances have brought innovative solutions. An easily accessible method is face recognition algorithm. Biometric technology, particularly facial recognition, is gaining ground as a preferred method of attendance control in educational settings, due to its convenience and reliability. Unlike traditional methods based on manual input or card scanning, facial recognition provides a sophisticated yet easy-to-use solution. Researchers have delved into several different algorithms aimed at refining the accuracy and efficiency of facial recognition and detection and have applied these advances when exploring whether to use facial recognition and associated technologies such as RFID web-based platforms aimed at improving productivity and user experience Educational institutions can fine-tune attendance tracking procedures, freeing up valuable time and resources for students and faculty are independent and ensure accurate records of attendance [1].

The anticipated benefits of implementing a camera-based attendance system is expected to bring about more advantages than just improving efficiency. The helps to improve management, infrastructure, and increase overall productivity. However, there are some challenges to ensure accurate face recognition, especially in environmental conditions. Despite these challenges, this retrospective study highlights the importance of adopting a modern approach to attendance management, paving the way for and building a successful integration of camera-based systems across organizational contexts emphasizing the potential for transformational efficiency and accuracy in attendance management

2.2. Literature Review

Face recognition is a technology created by Woodrow Wilson Bleadsoe in 1966 that works to match human faces through digital images or video footage through a facial database. Face recognition became an idea to allow computers to find and recognize human faces quickly and precisely. Many algorithms have been developed to improve the performance of face recognition [2].

The Local Binary Patterns Histogram (LBPH) algorithm is also utilized for face recognition due to its effectiveness in handling low-light conditions and its ability to recognize both front and side faces. The LBPH algorithm works by dividing the input image into cells, typically 4x4 pixels in size, for feature encoding. Within each cell, the intensity values of the central pixel are compared to those of its surrounding neighbors in a clockwise or counter-clockwise direction. Based on these comparisons, binary values 1 or 0 are assigned to each neighbor pixel. This process results in an 8-bit binary number representing each cell. Histograms are then used to summarize the frequency of these binary patterns within larger cells, making the system robust to variations in illumination and aiding in edge detection. Finally, feature vectors are obtained by combining the histograms of all cells, and face images are classified based on similarity with the trained dataset using a classifier, such as Haar cascade classifier [3].

Haar Cascade classifier, a foundational component in facial recognition systems, enables robust and efficient face detection by leveraging a cascade of simple classifiers trained on Haar-like features. Its adaptability to variations in facial appearances, such as changes in expressions, occlusions, and accessories like glasses and beards, ensures reliable performance across diverse operational scenarios. By integrating Haar Cascade with recognition algorithms like the Local Binary Pattern Histogram (LBPH), facial recognition systems achieve comprehensive capabilities encompassing both detection and identification tasks [4].

With the advancement in the technology and use of proper software and hardware made the smart attendance system more accurate. Dlib is a widely-utilized C++ library in computer vision and machine learning. It is instrumental in facilitating various tasks related to face recognition. It uses Histogram of Oriented Gradients (HOG) algorithm to efficiently detects faces within images by analyzing gradients which enables accurate identification of facial regions. Furthermore, Dlib's robust strong and steady landmark estimation capabilities ensure precise positioning and alignment of detected faces, enhancing the system's accuracy. By employing deep learning techniques, Dlib computes 128-D facial feature vectors for each detected face, facilitating the matching of faces against a database for recognition purposes [5].

HOG is utilized as a feature extraction technique, where it analyzes gradients in localized image regions to construct histograms capturing the distribution of edge directions which provides a simplified as well as informative representation of facial features. This approach enables robust face detection across various poses and lighting conditions which is very crucial for attendance tracking. Furthermore, SVM is employed for classification tasks, trained on a dataset of facial feature vectors to distinguish between different individuals. By optimizing the decision boundary, SVM can efficiently classify unseen faces based on their extracted features, enabling accurate recognition. Hence, the development process is simplified by the seamless integration of dlib, which makes it easier to implement the functions of SVM-based face recognition and HOG-based face detection [6].

ResNet is an effective architecture for computer vision applications like facial identification. Studies on face recognition for the blind have made use of ResNet50, which is renowned for its capacity to learn complex facial features and generalize effectively. Furthermore, research conducted by Islam et al. assessed ResNet50's efficacy in emotion recognition from facial expressions, demonstrating its capacity to extract important characteristics for these kinds of tasks. ResNet50 has demonstrated encouraging performance when compared to other models, such as VGG16, highlighting the significance of appropriate weight initialization for best outcomes. CNNs have completely changed the field of facial identification. By utilizing convolutional layers, pooling processes, and nonlinear activations, CNNs are able to extract and interpret relevant features from face data that include both high-level and low-level properties. What is also interesting is that CNNs are very good at identifying abstract and complicated characteristics, which makes them very good at tasks where sophisticated feature representations are required. The most notable application of CNNs has been in facial expression identification, where they can accurately distinguish between a wide range of emotional indicators, including happy, surprise, and neutrality [7].

Chapter 3
System Analysis

3.1. System Analysis

The Smart Attendance System is designed to revolutionize traditional attendance tracking methods by introducing a user-friendly platform that utilizes camera technology for seamless and efficient attendance management. This innovative solution eliminates manual record-keeping, allowing people to conveniently mark their attendance. The system aims to enhance overall convenience and accuracy, providing administrators with real-time insights into attendance patterns and reducing the administrative burden associated with traditional methods. By streamlining the attendance tracking process, the Smart Attendance System seeks to deliver a more efficient and hassle-free experience for both institutions and attendees.

3.1.1. Requirement Analysis

The requirements for the system can be termed as functional and no- functional requirements.

Functional

i. The system should have accurate face detection and recognition from cameras.

ii. The system should perform automated attendance recording for recognized individuals.

iii. The system should be capable of sending real time notification to the guardian about the attendance.

Use Case Diagram

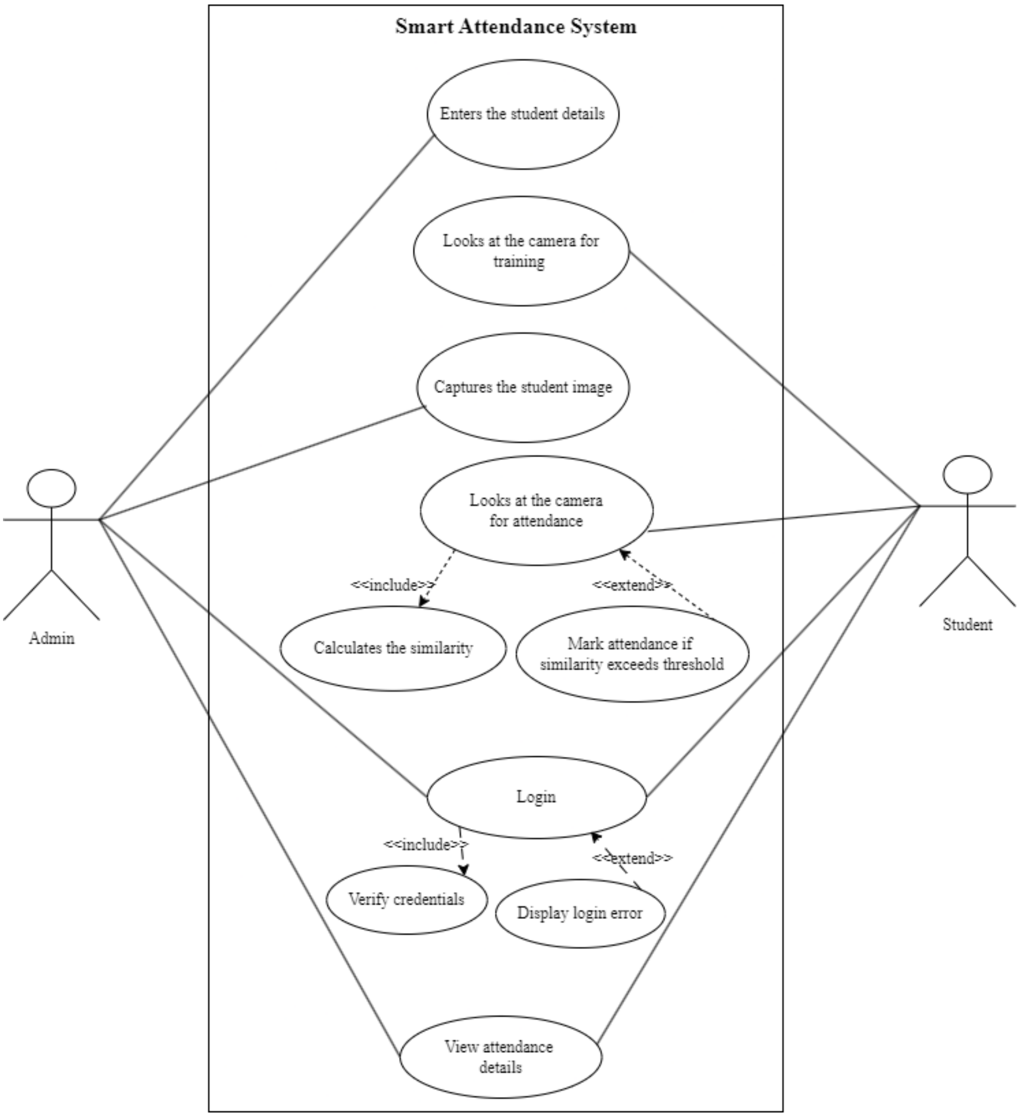

Smart Attendance System
Enters the student details
Looks at the camera for training
Captures the student image
Looks at the camera for attendance
<<include>>
<<extend>>
Calculates the similarity
Mark attendance if similarity exceeds threshold
Login
<<include>>
<<extend>>
Verify credentials
Display login error
View attendance details
Admin
Student

Figure 3.1: Use Case Diagram

Non-Functional

i. The system should be easy to maintain and update, with a clear and well-documented codebase.

ii. The system should be easy to use and understand for every user.

3.1.2. Feasibility Analysis

A feasibility study is an extensive review and evaluation of a proposed project's probability of success and profitability. It investigates a number of factors, including technical, operational, financial, legal, regulatory, and environmental possibilities. The assessment aims to determine the project's feasibility under specific constraints, taking into account factors such as technical readiness, resource availability, and cost compliance.

i. **Technical Feasibility:** Technical feasibility determines whether a project is feasible in terms of software, hardware, and expertise. Since, the system runs on any computer that has Python installed. Python, a popular and innovative technology, ensures that the system is technically feasible and meets modern standards. It confirms that we have the necessary tools and expertise to build and implement the system effectively.

ii. **Operational Feasibility:** The operational feasibility determines whether the new system truly benefits the users. It depends on having the right people for the project and predicts whether or not people will use the system once it is completed. This project makes sense for the users as almost everyone nowadays, including teachers and staff, is familiar with digital technology. As a result, using our project should be simple for them. Hence, the system is operationally feasible.

iii. **Economic Feasibility:** Economic feasibility, also known as a cost/benefit analysis, assists us in determining whether the value of the system's anticipated benefits exceeds the cost of construction and operation. This figure also includes the cost of acquiring and installing cameras. This study shows that the benefits we expect from this system should outweigh the costs of development and operation, making it economically feasible.

iv. **Legal Feasibility:** It is legal to bid on products and buy them in Nepal. There are no specific laws prohibiting smart attendance using Cameras. Thus, the proposed system is legally feasible.

v. **Schedule Feasibility:** The time required to complete the project, and the time spent on each activity in the following weeks are represented in the Gantt chart below.

Table 3.1: Gantt Chart

Working Time In	**October 2023**				**November 2023**				**January 2023**				**February 2023**
Week	**1st**	**2nd**	**3rd**	**4th**	**1st**	**2nd**	**3rd**	**4th**	**1st**	**2nd**	**3rd**	**4th**	**1st**
Requirement Analysis	■	■											
Design			■	■									
Coding					■	■	■	■	■	■			
Testing										■	■	■	
Operation													■
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■

Chapter 4

System Design

4.1. Design

4.1.1. Flow Chart

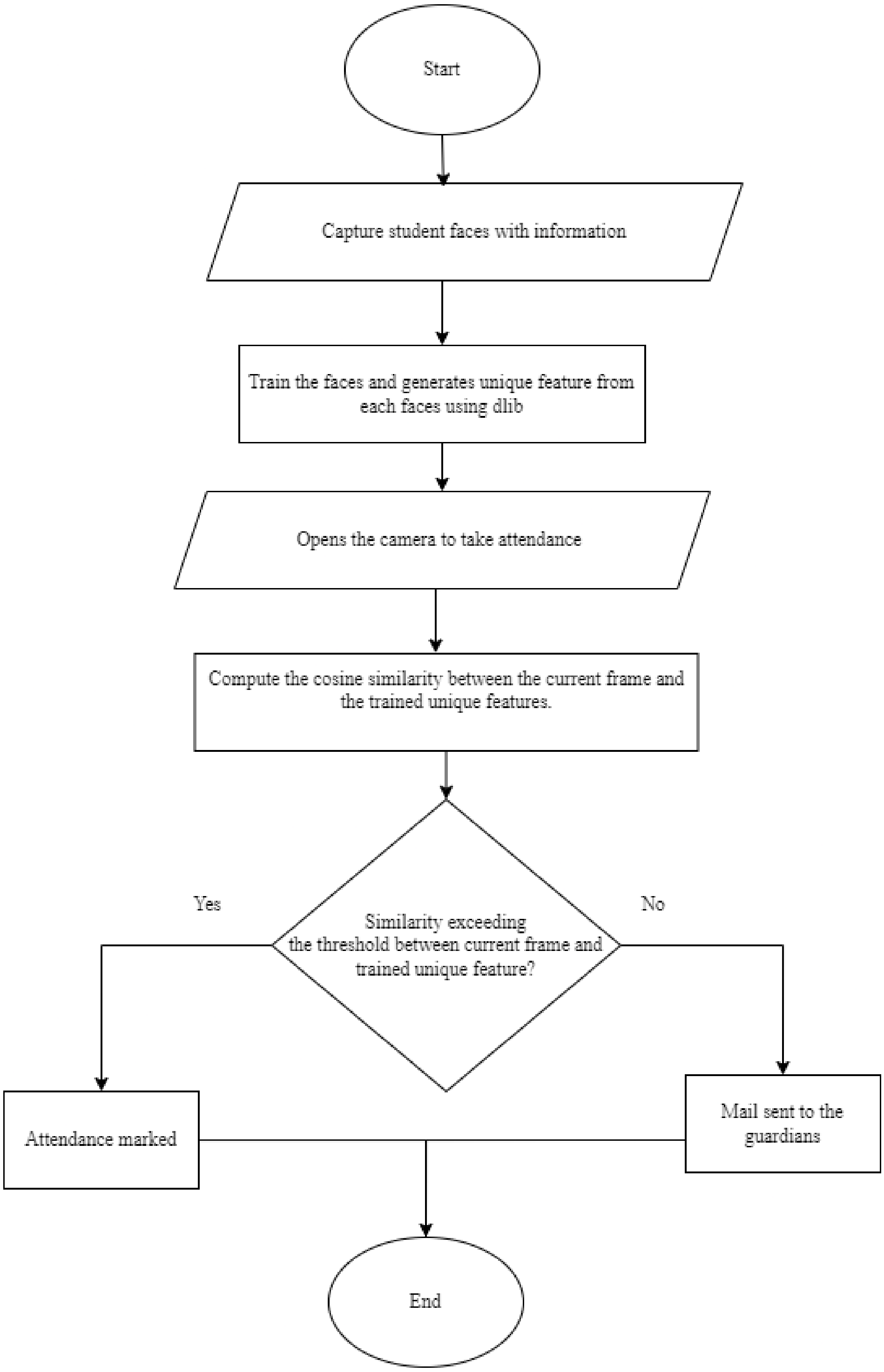

Start
Capture student faces with information
Train the faces and generates unique feature from each faces using dlib
Opens the camera to take attendance
Compute the cosine similarity between the current frame and the trained unique features.
Similarity exceeding the threshold between current frame and trained unique feature?
Yes
No
Attendance marked
Mail sent to the guardians
End

Figure 4.1: Flow Chart of Smart Attendance System

4.1.2. Sequence Diagram

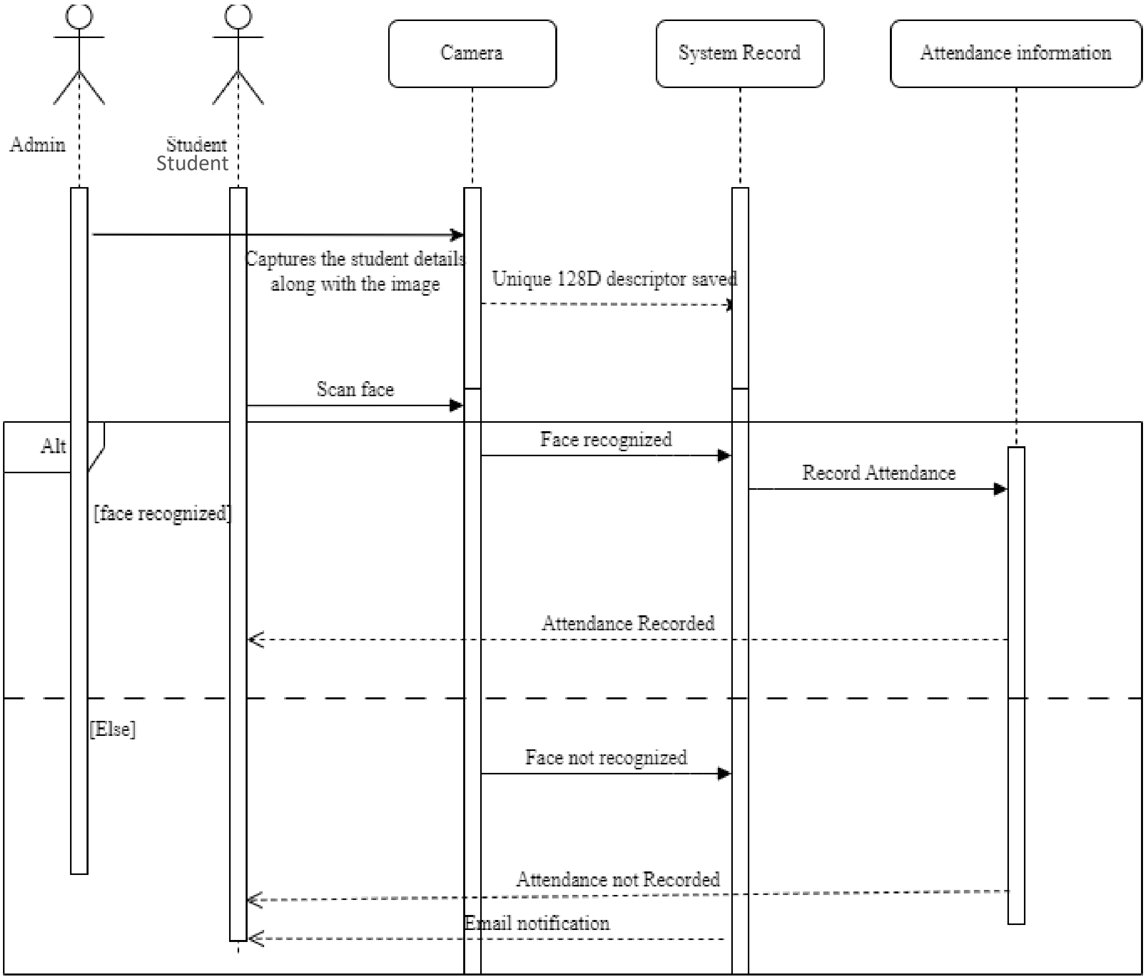
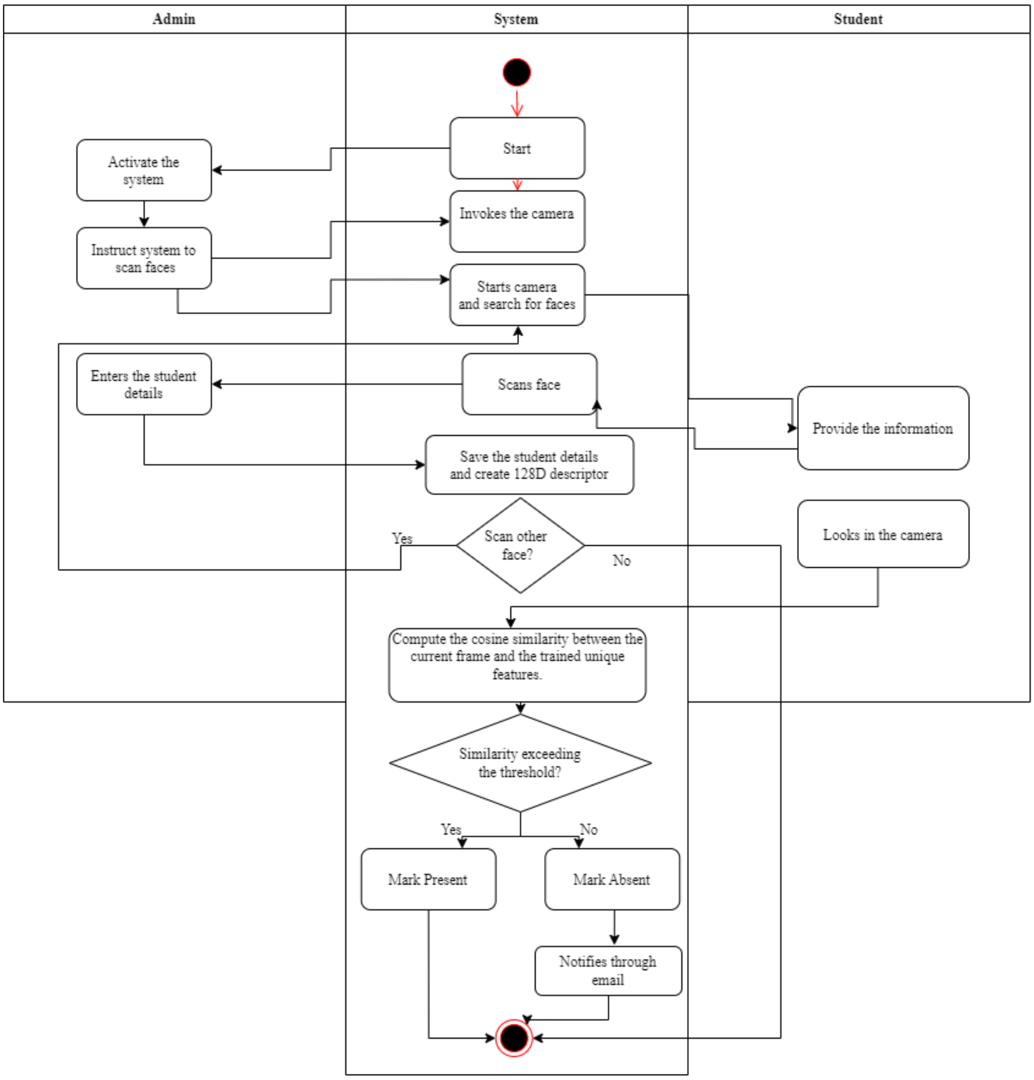


Figure 4.2: Sequence Diagram of Smart Attendance System

4.1.3. Activity Diagram



Admin
System
Student
Start
Activate the system
Invokes the camera
Instruct system to scan faces
Starts camera and search for faces
Enters the student details
Scans face
Provide the information
Save the student details and create 128D descriptor
Yes
Scan other face?
No
Looks in the camera
Compute the cosine similarity between the current frame and the trained unique features.
Similarity exceeding the threshold?
Yes
No
Mark Present
Mark Absent
Notifies through email

Figure 4.3: Activity Diagram of Smart Attendance System

4.1.4. Component Diagram

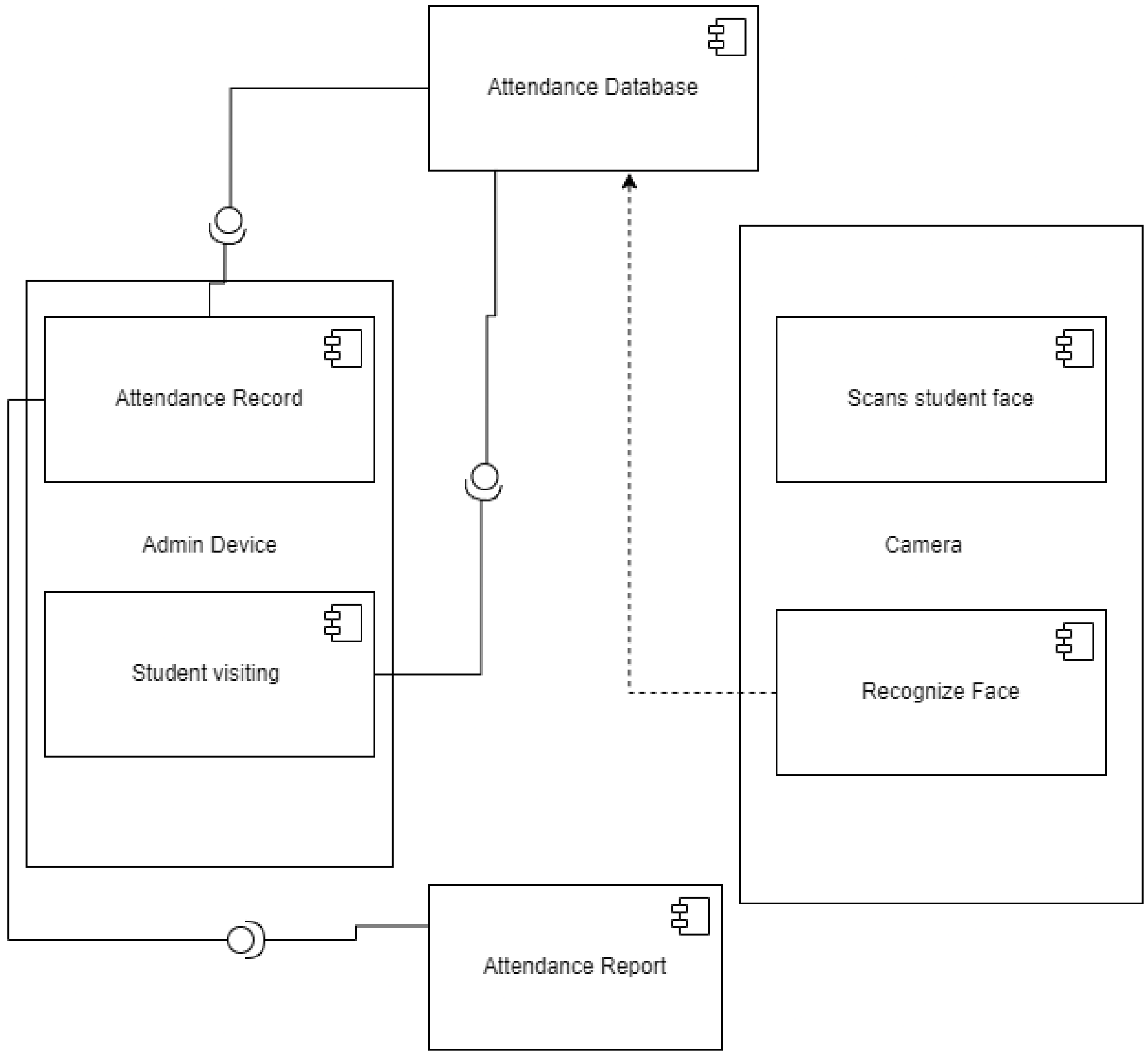


Figure 4.4: Component Diagram of Smart Attendance System

4.1.5. Class Diagram

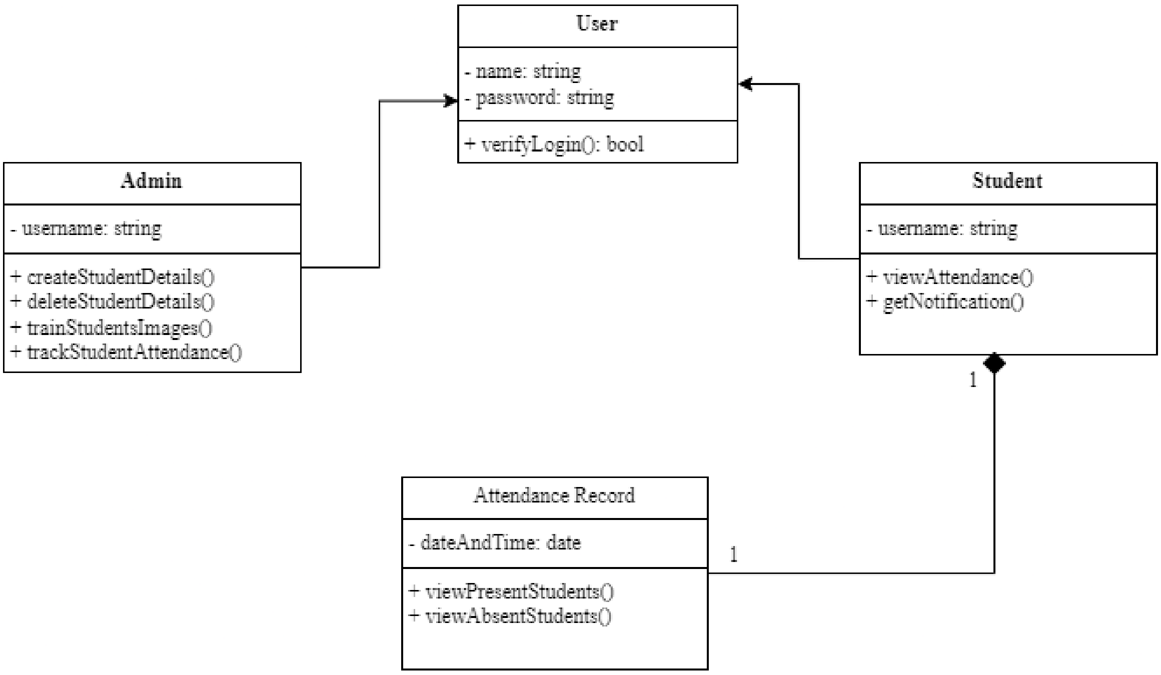


Figure 4.5: Class Diagram of Smart Attendance System

4.2. Algorithm Used

First, we collect facial images and attendance data for training. Then, we develop a facial recognition model using the cosine similarity algorithm to match the trained facial data with live camera data. This model incorporates factors such as attendance history to enhance accuracy. During designated times, the system automatically marks attendance based on facial recognition. Additionally, we implement an email notification system to alert guardians if a student is absent, providing relevant information.

Dlib: Dlib is a powerful and user-friendly C++ library that provides tools for developing applications related to computer vision, machine learning, and image processing. It comes with a wide range of pre-trained models and tools, making it easier for developers to integrate features like facial recognition, object detection, and shape prediction into their projects. Dlib is known for its efficiency and flexibility, making it suitable for various tasks in the field of artificial intelligence. Additionally, it is open-source, allowing developers to access and modify the source code, fostering a collaborative community for continuous improvement and innovation in computer vision applications.

In Python, Dlib is a popular library that extends the capabilities of the C++ version to a Python environment. It offers a set of tools for tasks such as facial recognition, object detection, image processing, and machine learning. One of the key features of the Dlib Python library is its simplicity and ease of use, allowing developers to leverage its functionality without extensive knowledge of C++.

Dlib working mechanism:

- Face Detection: Dlib first detects faces using a pre-trained deep learning model. This model identifies the locations of faces within the image.
- Facial Landmark Detection: After detecting the faces in the image, Dlib's facial landmark detection algorithm is applied to locate key points on the face, such as the corners of the eyes, the tip of the nose, and the corners of the mouth.
- Face Alignment: Using the detected facial landmarks, it aligns the faces in a standardized position. This step is essential because it minimizes variations caused by differences in head pose, scale, and orientation. This increases the accuracy of subsequent steps.

- Vectorization: After face alignment, Dlib utilizes a pre-trained deep neural network to extract features from the aligned face regions. This network is trained specifically for the task of face recognition and produces a fixed-length numerical vector representing the unique characteristics of each face. This vector representation captures various facial attributes like shape, texture, and appearance.

Cosine Similarity Algorithm:

Cosine similarity is a fundamental metric utilized in various fields of data science to assess the similarity between two vectors. It quantifies this similarity by computing the cosine of the angle formed between the two vectors.

Widely recognized for its versatility and effectiveness, cosine similarity finds extensive application across diverse domains within data science. Its utility spans tasks such as identifying comparable documents in natural language processing (NLP), retrieving information, locating analogous DNA sequences in bioinformatics, identifying instances of plagiarism, and numerous other scenarios. Cosine Similarity can be calculated as:

$$cos\,\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

Where,

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

$$\|A\| = \sqrt{\sum_{i=1}^{n} A_i^2}$$

$$\|B\| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

Here, A represents the feature extracted from current frame and B represents the features of the trained images.

In this smart attendance system, cosine similarity serves as a crucial metric for identifying individuals based on their facial features. Initially, during the setup phase, the system extracts and stores facial feature vectors of known individuals in a database using dlib.

These feature vectors encapsulate unique characteristics of each individual's face i.e. generate the 128 descriptors.

During runtime, when processing each frame from a video stream or camera feed, the system extracts the facial feature vector of the face detected in the frame. This vector represents the essential facial characteristics of the current face being analyzed.

To determine the identity of the current face, the system calculates the cosine similarity between the current face's feature vector and each known face's feature vector stored in the database. The cosine similarity quantifies the resemblance between the facial features of the current face and those of the known faces. A higher cosine similarity indicates a closer match between the features, suggesting a higher likelihood of the current face belonging to the known individual.

By comparing the cosine similarity scores against a predefined threshold, the system decides whether the current face corresponds to any known individual or if it remains unidentified. If the similarity score surpasses the threshold for a particular known individual, the system recognizes and marks the attendance of that individual for the current date and time.

This approach enables efficient and accurate recognition of individuals in real-time, facilitating automatic attendance tracking based on facial recognition technology.

In summary:

1. **Cosine Similarity in Attendance System:**

   In the smart attendance system code, cosine similarity is used to compare the facial feature vectors of known faces with the feature vector of the current face being processed. Here's how it works within the system:

2. **Feature Extraction:**

   The facial feature vectors of known faces are precomputed and stored in the database during the initialization phase. During runtime, the facial feature vector of the current face is extracted using a deep learning model such as Dlib's ResNet.

3. **Similarity Calculation:**

   Cosine similarity is calculated between the feature vector of the current face and each known face in the database. The similarity score indicates how closely the current face resembles each known face.

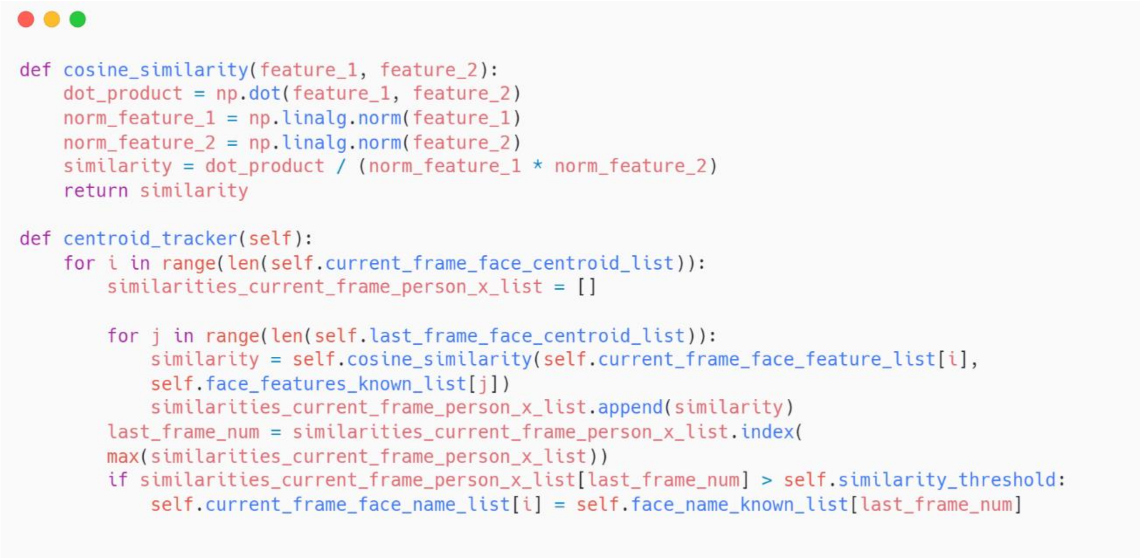
4. **Thresholding:**

   If the similarity score exceeds a predefined threshold, the current face is recognized as the corresponding known face. Otherwise, the current face is labeled as "unknown."

5. **Attendance Marking:**

   If the recognized face corresponds to a known individual, their attendance is marked for the current date and time.

Similarity function:



```
def cosine_similarity(feature_1, feature_2):
    dot_product = np.dot(feature_1, feature_2)
    norm_feature_1 = np.linalg.norm(feature_1)
    norm_feature_2 = np.linalg.norm(feature_2)
    similarity = dot_product / (norm_feature_1 * norm_feature_2)
    return similarity

def centroid_tracker(self):
    for i in range(len(self.current_frame_face_centroid_list)):
        similarities_current_frame_person_x_list = []

        for j in range(len(self.last_frame_face_centroid_list)):
            similarity = self.cosine_similarity(self.current_frame_face_feature_list[i],
            self.face_features_known_list[j])
            similarities_current_frame_person_x_list.append(similarity)
        last_frame_num = similarities_current_frame_person_x_list.index(
        max(similarities_current_frame_person_x_list))
        if similarities_current_frame_person_x_list[last_frame_num] > self.similarity_threshold:
            self.current_frame_face_name_list[i] = self.face_name_known_list[last_frame_num]
```

Chapter 5
Implementing and Testing

5.1. Implementation

Software implementation is the process of converting the designed system into the programs. This process includes not only the actual writing of the code but also the preparation of the requirements and objectives, the design of what is to be coded, and confirmations that what is developed has met the predefined objectives.

5.1.1. Tools Used

i. **Python:** As the core programming language, Python provides a versatile and user-friendly environment for developing the smart attendance system. Its extensive libraries, including dlib for facial recognition, enable efficient coding and seamless integration with various components of the project.

ii. **Flask:** Flask, a lightweight web framework for Python, facilitates the development of the project's web-based interface. Its simplicity and modularity make it ideal for creating interactive user interfaces to manage and visualize attendance data in real-time.

iii. **PyCharm:** PyCharm serves as the integrated development environment (IDE) for this project. Its intelligent code assistance, debugging tools, and seamless integration with version control systems enhance the development workflow, ensuring efficient coding, testing, and project management.

iv. **Draw.io:** Draw.io is free online diagram software for making Class diagram, Sequence diagram, Component and Deployment diagram. It is an open-source technology stack & most widely used browser-based end-user diagramming application.

5.2. Testing

Table 5.1: Unit Testing

S.N.	Test Case ID	Test Description	Steps Executed	Expected Result	Actual Result	Pass/ Fail
1	UT-001	Opening the Tkinter GUI for entering the students details.	Run the file capture_faces _from_ camera.py file.	GUI should be visible.	GUI successfully displayed.	Pass
2	UT-002	Opening camera of laptop.	Run the file capture_faces _from_ camera.py file.	Camera should get open to capture the faces.	Camera was successfully opened.	Pass
3	UT-003	Entering details of student along with their images.	Enter name, email and capture the faces.	Detail of the student along with images must be saved.	Student detail was successfully saved.	Pass
4	UT-004	Entering the same name which has been already used.	Enter the same name for two or more than two students.	Error message must be shown, and numbers can be appended for same name of the student.	Error message shown.	Pass
5	UT-005	Training the images.	Run the feature_extraction.py file.	Unique 128 D vectors should be saved in feature_all.csv file.	128 D unique vectors saved in features all.csv, i.e. different for different student.	

6	UT-006	Opening the camera for attendance.	Run the attendance_taker.py file.	Camera should get opened.	Camera opened for taking the attendance.	Pass
7	UT-007	Similarity calculation using cosine similarity.	Run the attendance_taker.py file.	Cosine Similarity must be calculated for the image in the current frame and the trained image previously.	Similarity is calculated for the image in the current frame and the trained image previously.	Pass
8	UT-008	Marking the attendance similarity threshold meets.	Run the attendance_taker.py file.	If similarity threshold meets the calculated similarity then mark student as present.	Attendance Marked successfully .	Pass
9	UT-009	Sending notification to absent students.	Run the attendance_taker.py file.	Sending notification to the guardian for absent students.	Email and SMS notification sent to guardian.	Pass
10	UT-010	Marking attendance for the single time in a day.	Run attendance_taker.py file.	Student whose attendance is already marked for the day should not be marked again.	Attendance was taking taken for only single time in a day.	Pass

11	UT-011	Viewing the attendance in the web app.	Run app.py file and go to the browser and put the url.	Attendance should be viewed in the table format.	Attendance viewed in table format.	Pass
12	UT-012	Downloading the attendance in the excel file.	Click on the download button in the given web app.	Attendance should be downloaded in excel format.	Attendance downloaded successfully.	Pass
13	UT-013	Login into student to view their attendance record.	Enter the username and password.	Attendance record should be shown to only the specific user.	Attendance record shown successfully.	Pass
14	UT-014	Logout from the student profile.	Press the logout button.	Page must return to the login page.	Login page successfully opened.	Pass

Chapter 6

Conclusion and Future Recommendations

6.1. Conclusion

In conclusion, the implementation of a smart attendance system using facial recognition technology is expected to revolutionize traditional attendance tracking processes. The system, equipped with facial recognition and a decision tree algorithm for optimal timing allocation, aims to automate attendance marking and enhance overall efficiency. The inclusion of a guardian notification system for student absences further strengthens communication channels. The anticipated outcomes include reduced manual workload, improved accuracy, and a technologically advanced educational environment. Careful consideration of privacy concerns, regular updates, and proactive issue resolution are essential for successful deployment and sustained effectiveness.

6.2. Future Recommendations

The expected outcomes of implementing a smart attendance system using facial recognition technology include:

i. **Efficiency Improvement:** Automation of attendance tracking through facial recognition is anticipated to significantly reduce manual effort, leading to a more efficient and streamlined process.

ii. **Accuracy Enhancement:** The use of facial recognition technology is expected to improve the accuracy of attendance records, reducing the likelihood of errors associated with manual entry or traditional methods.

iii. **Adaptive Timing Allocation:** The incorporation of a decision tree algorithm is designed to optimize the timing for attendance marking, considering factors like historical attendance patterns and class schedules, resulting in a more adaptive and effective system.

References

[1] S. H. P. S. Hegde and A. , "Face Recognition based Attendance Management System," *International Journal of Engineering Research and Technology,* vol. 9, 2020.

[2] R. Jayaswal and M. Dixit, "Comparative analysis of human face recognition by traditional methods and deep learning in real-time environment," in *Communication Systems and Network Technologies (CSNT)*, 2020.

[3] S. K. S Manvi, A. P. Singh, P. Nimbal and G. K. Shyam, "Face Recognition System Based on LBPH Algorithm," *International Journal of Engineering and Advanced Technology (IJEAT),* vol. 8, no. 5s, pp. 26-29, 2019.

[4] T. Chinimilli, V. R. Kaipu, A. T, A. Kotturi and J. V. Mandapati, " Face Recognition based Attendance System using Haar Cascade and Local Binary Pattern Histogram Algorithm," Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India, 2023.

[5] S. R. Wagh and K. Chaudhari, "Face recognition-based student's attendance system using DLIB," *International Journal of Advance Research, Ideas and Innovations in Technology,* vol. 8, no. 1, pp. 265-268, 2022.

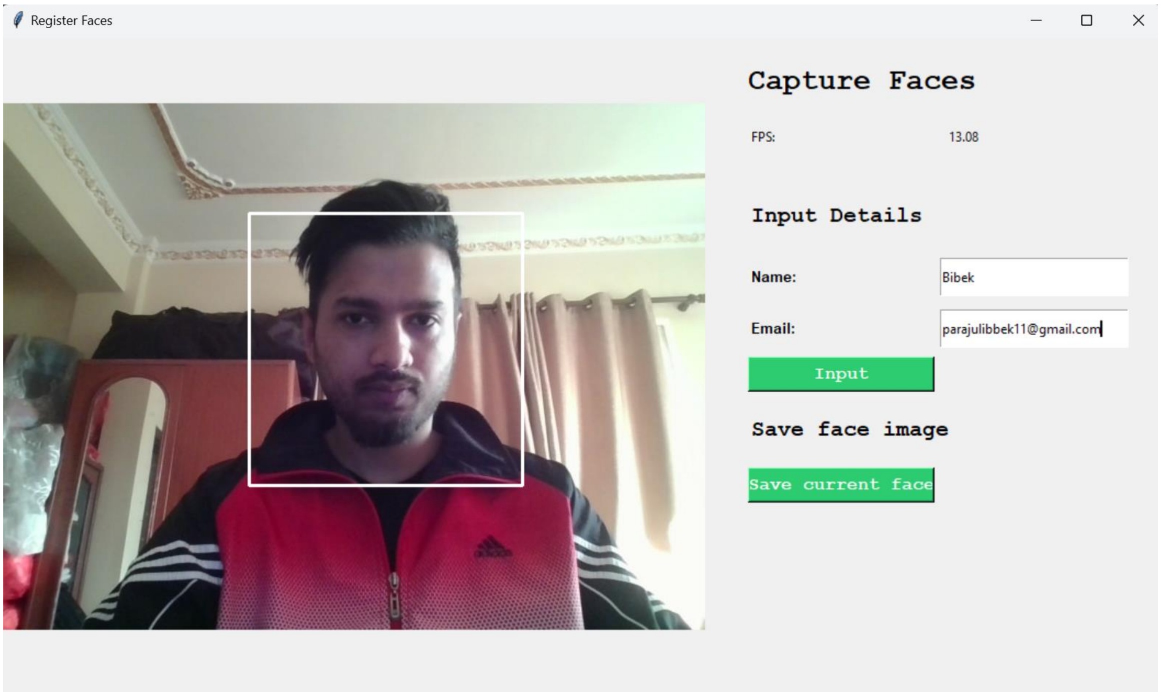
[6] S. R. Boyapally, "Facial Recognition and Attendance System using dlib and face_recognition libraries," Hyderabad, India, 2023.

[7] J.-R. Lee, K.-W. Ng and Y.-J. Yoong, "Face and facial expressions recognition system for blind people using ResNet50 architecture and CNN," *Journal of Informatics and Web Engineering,* vol. 2, no. 2, 2023.

APPENDIX

Screenshots

Snapshot 1: Inserting student details and faces

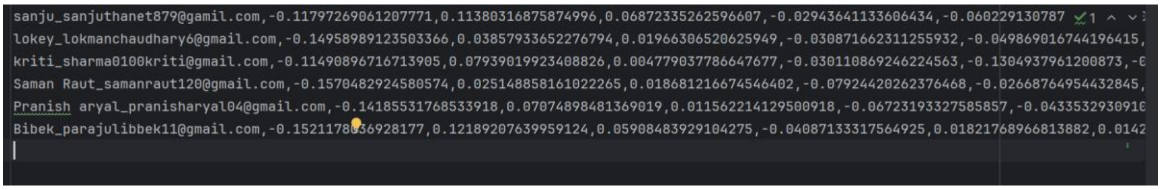


FPS: 13.08

Name: Bibek

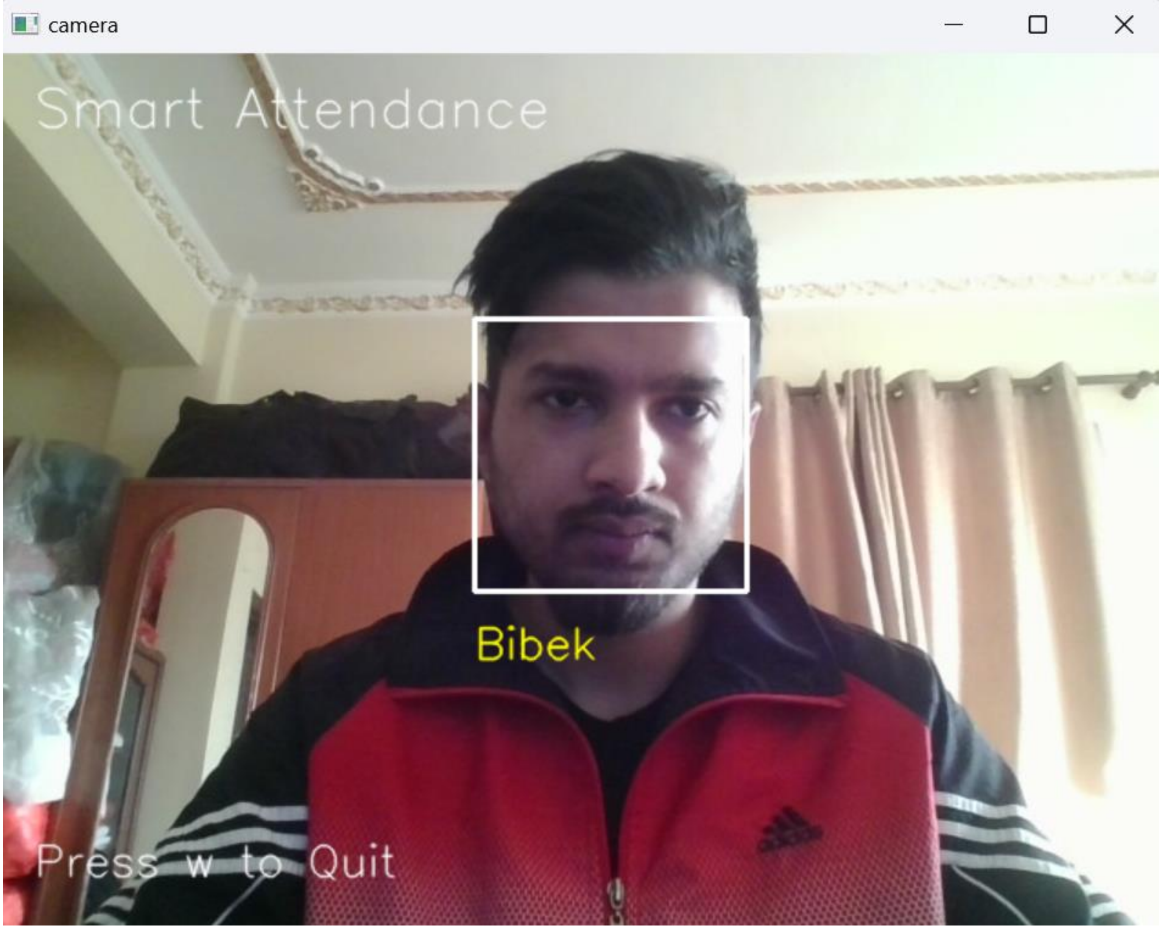
Email: parajulibbek11@gmail.com

Snapshot 2: Generating face vectors

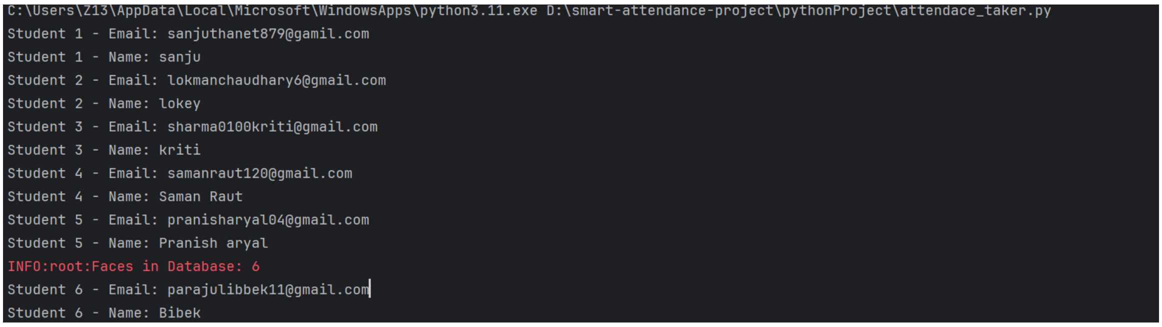


```
sanju_sanjuthanet879@gamil.com,-0.11797269061207771,0.11380316875874996,0.06872335262596607,-0.02943641133606434,-0.060229130787
lokey_lokmanchaudhary6@gmail.com,-0.14958989123503366,0.03857933652276794,0.01966306520625949,-0.030871662311255932,-0.049869016744196415,
kriti_sharma0100kriti@gmail.com,-0.11490896716713905,0.07939019923408826,0.004779037786647677,-0.030110869246224563,-0.1304937961200873,-0
Saman Raut_samanraut120@gmail.com,-0.1570482924580574,0.025148858161022265,0.018681216674546402,-0.07924420262376468,-0.02668764954432845,
Pranish aryal_pranisharyal04@gmail.com,-0.14185531768533918,0.07074898481369019,0.011562214129500918,-0.06723193327585857,-0.0433532930910
Bibek_parajulibbek11@gmail.com,-0.1521178036928177,0.12189207639959124,0.05908483929104275,-0.04087133317564925,0.01821768966813882,0.0142
```

Snapshot 3: Facial recognition

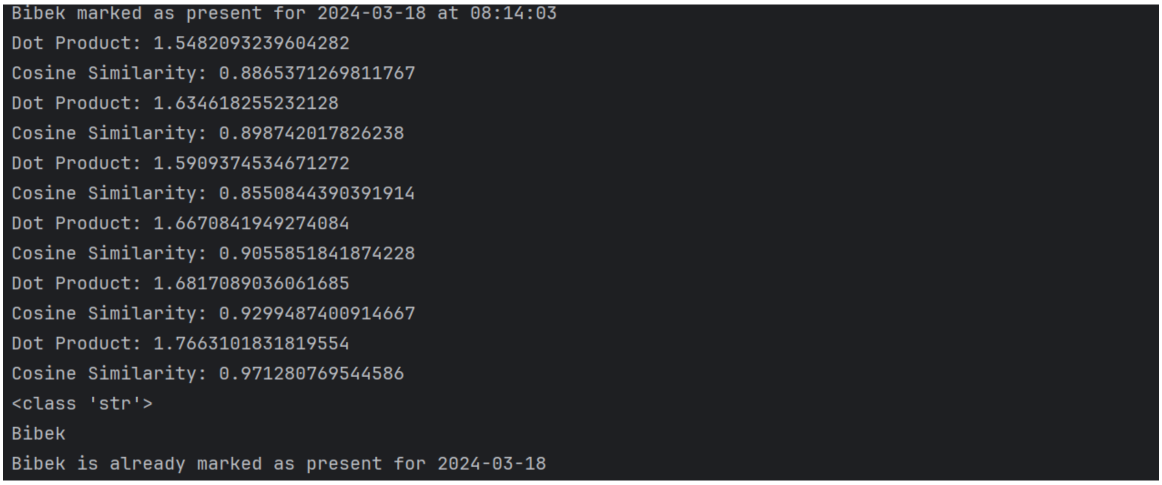


Snapshot 4: Student details



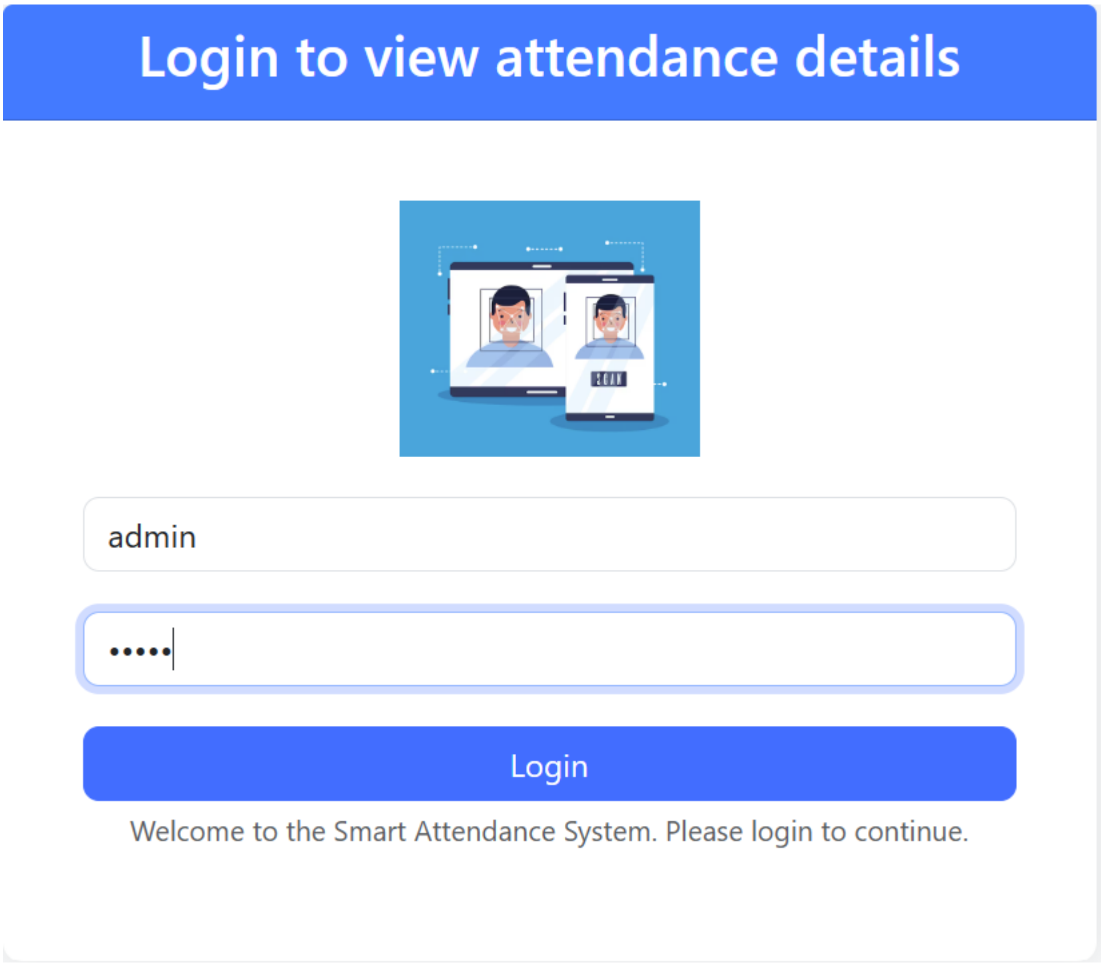
```
C:\Users\Z13\AppData\Local\Microsoft\WindowsApps\python3.11.exe D:\smart-attendance-project\pythonProject\attendace_taker.py
Student 1 - Email: sanjuthanet879@gamil.com
Student 1 - Name: sanju
Student 2 - Email: lokmanchaudhary6@gmail.com
Student 2 - Name: lokey
Student 3 - Email: sharma0100kriti@gmail.com
Student 3 - Name: kriti
Student 4 - Email: samanraut120@gmail.com
Student 4 - Name: Saman Raut
Student 5 - Email: pranisharyal04@gmail.com
Student 5 - Name: Pranish aryal
INFO:root:Faces in Database: 6
Student 6 - Email: parajulibbek11@gmail.com
Student 6 - Name: Bibek
```

Snapshot 5: Marking attendance using cosine similarity



```
Bibek marked as present for 2024-03-18 at 08:14:03
Dot Product: 1.5482093239604282
Cosine Similarity: 0.8865371269811767
Dot Product: 1.634618255232128
Cosine Similarity: 0.898742017826238
Dot Product: 1.5909374534671272
Cosine Similarity: 0.8550844390391914
Dot Product: 1.6670841949274084
Cosine Similarity: 0.9055851841874228
Dot Product: 1.6817089036061685
Cosine Similarity: 0.9299487400914667
Dot Product: 1.7663101831819554
Cosine Similarity: 0.971280769544586
<class 'str'>
Bibek
Bibek is already marked as present for 2024-03-18
```

Snapshot 6: Admin login



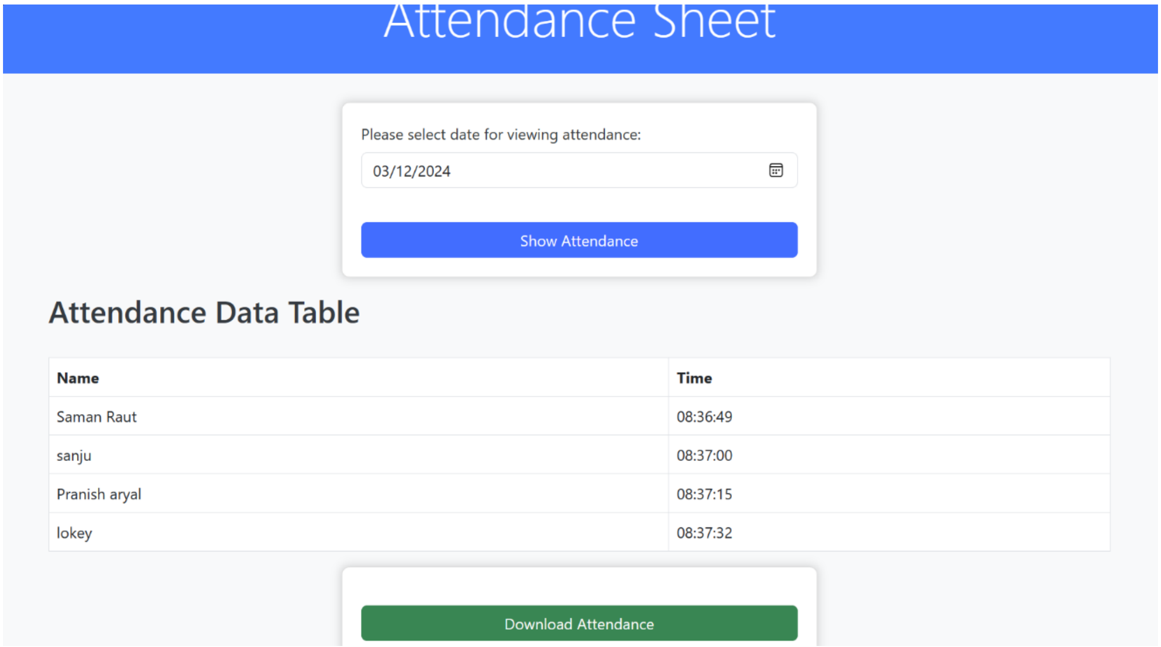
Login to view attendance details

admin

•••••

Welcome to the Smart Attendance System. Please login to continue.

Snapshot 7: Admin view



Attendance Sheet

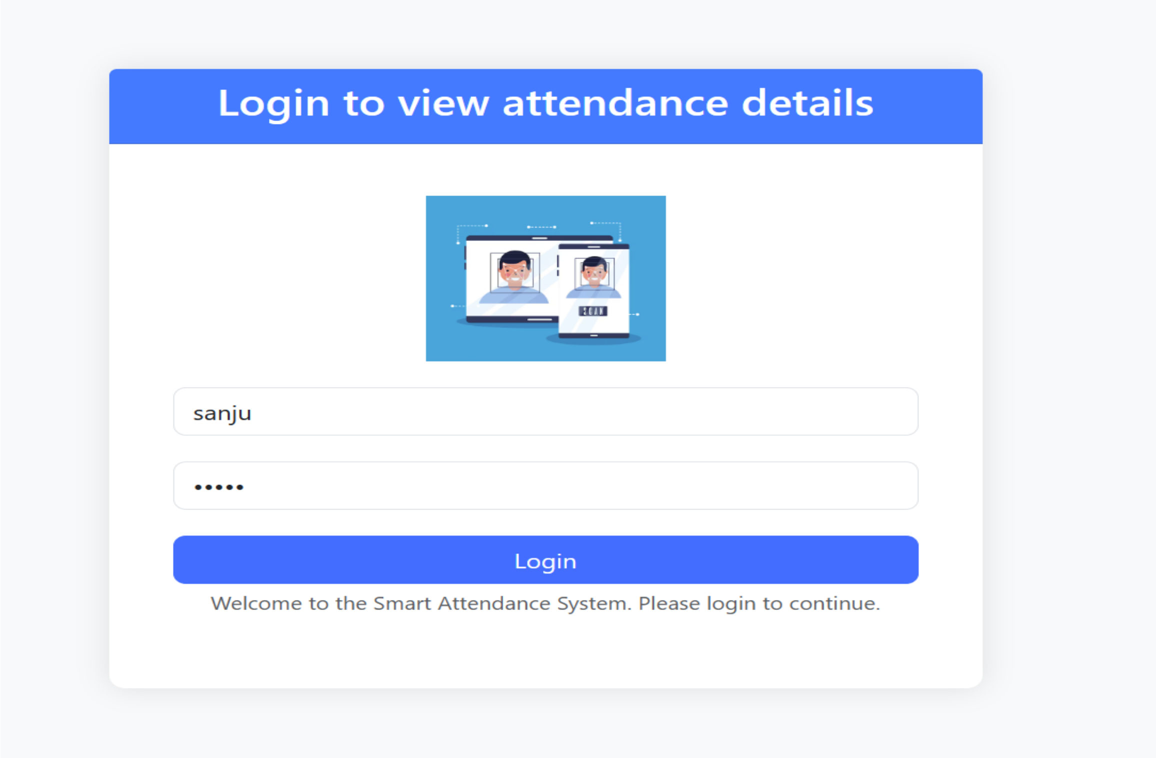
Please select date for viewing attendance:

03/12/2024

Attendance Data Table

Name	Time
Saman Raut	08:36:49
sanju	08:37:00
Pranish aryal	08:37:15
lokey	08:37:32

Snapshot 8:Student login



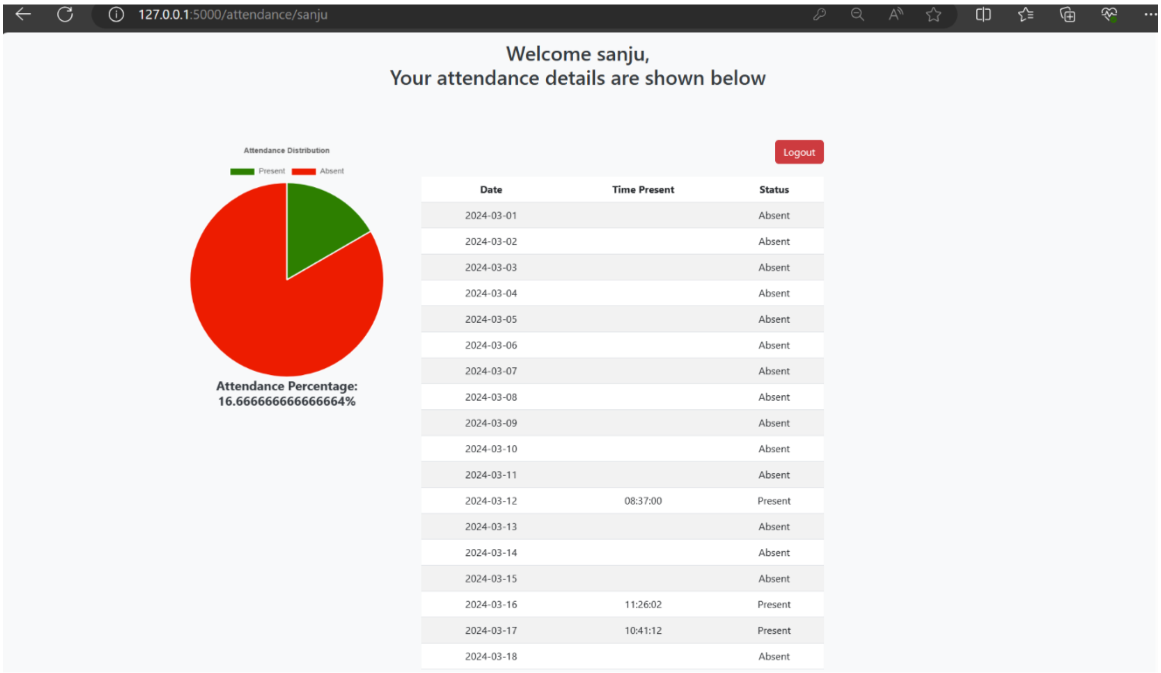
Login to view attendance details

sanju

•••••

Welcome to the Smart Attendance System. Please login to continue.

Snapshot 9: Student view



Welcome sanju,
Your attendance details are shown below

Attendance Percentage:
16.666666666666664%

Date	Time Present	Status
2024-03-01		Absent
2024-03-02		Absent
2024-03-03		Absent
2024-03-04		Absent
2024-03-05		Absent
2024-03-06		Absent
2024-03-07		Absent
2024-03-08		Absent
2024-03-09		Absent
2024-03-10		Absent
2024-03-11		Absent
2024-03-12	08:37:00	Present
2024-03-13		Absent
2024-03-14		Absent
2024-03-15		Absent
2024-03-16	11:26:02	Present
2024-03-17	10:41:12	Present
2024-03-18		Absent

Source Code

```
# For GUI creation and saving student details
import dlib
import numpy as np
import cv2
import os
import shutil
import time
import logging
import tkinter as tk
from tkinter import font as tkFont
from PIL import Image, ImageTk

detector = dlib.get_frontal_face_detector()

    def GUI_get_input_name(self):
        self.input_name_char = self.input_name.get().strip()
        self.input_email_char = self.input_email.get().strip()

        # Check if both name and email are provided
        if not self.input_name_char or not self.input_email_char:
            self.label_warning['text'] = "Please provide both name and email"
            self.label_warning['fg'] = 'red'
            return

        # Check if the name already exists
        existing_names = [name.split('_')[2] for name in
os.listdir("data/data_faces_from_camera/") if
                          name.startswith("person")]
        if self.input_name_char in existing_names:
            self.label_warning['text'] = "Name already exists"
            self.label_warning['fg'] = 'red'
            return

        self.create_face_folder(self.input_email_char)
        self.label_warning['text'] = ""
        self.label_cnt_face_in_database['text'] = str(self.existing_faces_cnt)

    def GUI_info(self):
        tk.Label(self.frame_right_info,
                 text="Capture Faces",
                 font=self.font_title).grid(row=0, column=0, columnspan=3, sticky=tk.W,
padx=2, pady=20)


        tk.Label(self.frame_right_info, text="FPS: ").grid(row=1, column=0, sticky=tk.W,
padx=5, pady=2)
        self.label_fps_info.grid(row=1, column=1, sticky=tk.W, padx=5, pady=2)
        self.label_warning.grid(row=4, column=0, columnspan=3, sticky=tk.W, padx=5,
pady=2)

        tk.Label(self.frame_right_info,
                 font=self.font_step_title,
                 text="Input Details").grid(row=5, column=0, columnspan=2, sticky=tk.W,
padx=5, pady=20)

        tk.Label(self.frame_right_info, text="Name: ", font=('Arial', 10, 'bold')).grid(row=6,
column=0, sticky=tk.W, padx=5, pady=0)
        self.input_name.grid(row=6, column=1, sticky=tk.W, padx=0, pady=4, ipadx=24,
ipady=8)

        tk.Label(self.frame_right_info, text="Email: ", font=('Arial', 10, 'bold')).grid(row=7,
column=0, sticky=tk.W, padx=5, pady=0)
        self.input_email.grid(row=7, column=1, sticky=tk.W, padx=0, pady=8, ipadx=24,
ipady=8)

        tk.Button(self.frame_right_info,
                  text='Input',
                  command=self.GUI_get_input_name, bg="#2ecc71", fg="white", width=16,
height=1, font=('Courier New', 12, 'bold')).grid(row=8, column=0, padx=5)

        tk.Label(self.frame_right_info,
                 font=self.font_step_title,
                 text="Save face image").grid(row=9, column=0, columnspan=2, sticky=tk.W,
padx=5, pady=20)

        tk.Button(self.frame_right_info,
                  text='Save current face',
                  command=self.save_current_face, bg="#2ecc71", fg="white", width=16,
height=1, font=('Courier New', 12, 'bold')).grid(row=10, column=0, padx=5,
columnspan=3, sticky=tk.W)


        self.log_all.grid(row=11, column=0, columnspan=20, sticky=tk.W, padx=5,
pady=20)

        self.frame_right_info.pack()

    def create_face_folder(self, email):
        self.existing_faces_cnt += 1
        if self.input_name_char:
            person_name = "person_" + str(self.existing_faces_cnt) + "_" +
self.input_name_char
        else:
            person_name = "person_" + str(self.existing_faces_cnt)
        person_name_with_email = f"{person_name}_{email}"


        self.current_face_dir = os.path.join(self.path_photos_from_camera,
person_name_with_email)
        os.makedirs(self.current_face_dir)


        self.log_all["text"] = f"\"{self.current_face_dir}/\" created!"
        logging.info("\n%-40s %s", "Create folders:", self.current_face_dir)
        self.ss_cnt = 0
        self.face_folder_created_flag = True
        self.email = email


    #  Main process of face detection and saving
    def process(self):
        ret, self.current_frame = self.get_frame()
        faces = detector(self.current_frame, 0)
        # Get frame
        if ret:
            self.update_fps()
            self.label_face_cnt["text"] = str(len(faces))
            #  Face detected
            if len(faces) != 0:
                #   Show the ROI of faces
                for k, d in enumerate(faces):
                    self.face_ROI_width_start = d.left()
                    self.face_ROI_height_start = d.top()
                    #  Compute the size of rectangle box
                    self.face_ROI_height = (d.bottom() - d.top())
                    self.face_ROI_width = (d.right() - d.left())
                    self.hh = int(self.face_ROI_height / 2)
                    self.ww = int(self.face_ROI_width / 2)


                    # If the size of ROI > 480x640
                    if (d.right() + self.ww) > 640 or (d.bottom() + self.hh > 480) or (d.left() -
self.ww < 0) or (
                            d.top() - self.hh < 0):
                        self.label_warning["text"] = "OUT OF RANGE"
                        self.label_warning['fg'] = 'red'
                        self.out_of_range_flag = True
                        color_rectangle = (255, 0, 0)
                    else:
                        self.out_of_range_flag = False
                        self.label_warning["text"] = ""
                        color_rectangle = (255, 255, 255)
                    self.current_frame = cv2.rectangle(self.current_frame,
                                                       tuple([d.left() - self.ww, d.top() - self.hh]),
                                      tuple([d.right() + self.ww, d.bottom() + self.hh]),
                                      color_rectangle, 2)
            self.current_frame_faces_cnt = len(faces)


            # Convert PIL.Image.Image to PIL.Image.PhotoImage
            img_Image = Image.fromarray(self.current_frame)
            img_PhotoImage = ImageTk.PhotoImage(image=img_Image)
            self.label.img_tk = img_PhotoImage
            self.label.configure(image=img_PhotoImage)


# For extracting features
#  Get face landmarks
predictor = dlib.shape_predictor('data/data_dlib/shape_predictor_68_face_landmarks.dat')


#  Use Dlib resnet50 model to get 128D face descriptor
face_reco_model =
dlib.face_recognition_model_v1("data/data_dlib/dlib_face_recognition_resnet_model_v1
.dat")

#  Return 128D features for single image
def return_128d_features(path_img):
    img_rd = cv2.imread(path_img)
    faces = detector(img_rd, 1)
    if len(faces) != 0:
        shape = predictor(img_rd, faces[0])
        face_descriptor = face_reco_model.compute_face_descriptor(img_rd, shape)
    else:
        face_descriptor = 0
        logging.warning("no face")
    return face_descriptor

def return_features_mean_personX(path_face_personX):
    features_list_personX = []
    photos_list = os.listdir(path_face_personX)
    if photos_list:
        for i in range(len(photos_list)):
            features_128d = return_128d_features(path_face_personX + "/" + photos_list[i])
            if features_128d == 0:
                i += 1
            else:
                features_list_personX.append(features_128d)
    else:
        logging.warning(" Warning: No images in%s/", path_face_personX)


    if features_list_personX:
        features_mean_personX = np.array(features_list_personX,
dtype=object).mean(axis=0)
    else:
        features_mean_personX = np.zeros(128, dtype=object, order='C')
    return features_mean_personX


# For Taking Attendance
import dlib
import numpy as np
import cv2
import os
import pandas as pd
import time
import logging
import sqlite3
import datetime
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from twilio.rest import Client

    def send_email_notification(self, absent_students=None):
        if absent_students is None:
            # If absent_students is not provided, use the entire list of known students
            absent_students = self.face_name_known_list


        if absent_students:
            subject = "Attendance Notification"
            body = "Dear Parent,\n\nYour child was absent today."


            # Map names to email addresses
            absent_students_emails =
[self.face_email_known_list[self.face_name_known_list.index(name)] for name in
                                absent_students]


            msg = MIMEMultipart()
            msg['From'] = self.email_sender
            msg['To'] = ', '.join(absent_students_emails)
            msg['Subject'] = subject
            msg.attach(MIMEText(body, 'plain'))


            try:
                server = smtplib.SMTP(self.smtp_server, self.smtp_port)
                server.starttls()
                server.login(self.email_sender, self.email_password)
                text = msg.as_string()
                server.sendmail(self.email_sender, absent_students_emails, text)
                server.quit()
                logging.info("Email notification sent to absent students.")
            except Exception as e:
                logging.error(f"Failed to send email notification. Error: {str(e)}")

    def get_face_database(self):
        if os.path.exists("data/features_all.csv"):
            path_features_known_csv = "data/features_all.csv"
            csv_rd = pd.read_csv(path_features_known_csv, header=None)


            for i in range(csv_rd.shape[0]):
                features_someone_arr = []


                # Extract email (assumed to be at the beginning of each row)
                email = csv_rd.iloc[i][0].split('_')[1]  # Assuming email is in the format
"name_email"
                name = csv_rd.iloc[i][0].split('_')[0]
                # Extract features
                for j in range(1, 129):
                    if csv_rd.iloc[i][j] == '':
                        features_someone_arr.append('0')
                    else:
                        features_someone_arr.append(csv_rd.iloc[i][j])


                self.face_name_known_list.append(name)
                self.face_email_known_list.append(email)
                self.face_features_known_list.append(features_someone_arr)


                # Print email for debugging purposes
                print(f"Student {i + 1} - Email: {email}")
                print(f"Student {i + 1} - Name: {name}")

            logging.info("Faces in Database: %d", len(self.face_features_known_list))
            return 1
        else:
            logging.warning("'features_all.csv' not found!")
            return 0

    def update_fps(self):
        now = time.time()
        # Refresh fps per second
        if str(self.start_time).split(".")[0] != str(now).split(".")[0]:
            self.fps_show = self.fps
        self.start_time = now
        self.frame_time = now - self.frame_start_time
        self.fps = 1.0 / self.frame_time
        self.frame_start_time = now

    @staticmethod
    def cosine_similarity(feature_1, feature_2):  # feature_1 come from
compute_face_descriptor method of face_reco_model
        dot_product = np.dot(feature_1, feature_2)
        norm_feature_1 = np.linalg.norm(feature_1)
        norm_feature_2 = np.linalg.norm(feature_2)
        # print("Feature 1:", feature_1)
        # print("Feature 2:", feature_2)
        print("Dot Product:", dot_product)
        # print("Norm of Feature 1:", norm_feature_1)
        # print("Norm of Feature 2:", norm_feature_2)
        similarity = dot_product / (norm_feature_1 * norm_feature_2)
        print("Cosine Similarity:", similarity)  # Print the similarity value
        return similarity

    def centroid_tracker(self):
        for i in range(len(self.current_frame_face_centroid_list)):
            similarities_current_frame_person_x_list = []

            for j in range(len(self.last_frame_face_centroid_list)):
                similarity = self.cosine_similarity(
                    self.current_frame_face_feature_list[i],
                    self.face_features_known_list[j]
                )
                similarities_current_frame_person_x_list.append(similarity)

            last_frame_num = similarities_current_frame_person_x_list.index(
                max(similarities_current_frame_person_x_list))
            if similarities_current_frame_person_x_list[last_frame_num] >
self.similarity_threshold:
                self.current_frame_face_name_list[i] =
self.face_name_known_list[last_frame_num]


    def draw_note(self, img_rd):
        cv2.putText(img_rd, "Smart Attendance", (20, 40), self.font, 1, (255, 255, 255), 1,
                    cv2.LINE_AA)
        cv2.putText(img_rd, "Press w to Quit", (20, 450), self.font, 0.8, (255, 255, 255), 1,
cv2.LINE_AA)

        for i in range(len(self.current_frame_face_name_list)):
            img_rd = cv2.putText(img_rd, self.current_frame_face_name_list[i],
                                 self.current_frame_face_position_list[i], self.font, 0.8, (0, 255, 255),
1,
                                 cv2.LINE_AA)
        return img_rd
    def attendance(self, name):
        current_date = datetime.datetime.now().strftime('%Y-%m-%d')
        conn = sqlite3.connect("attendance.db")
        cursor = conn.cursor()


        cursor.execute("SELECT * FROM attendance WHERE name = ? AND date = ?",
(name, current_date))
        existing_entry = cursor.fetchone()


        if existing_entry:
            print(f"{name} is already marked as present for {current_date}")
        else:
            current_time = datetime.datetime.now().strftime('%H:%M:%S')
            cursor.execute("INSERT INTO attendance (name, time, date) VALUES (?, ?, ?)",
                           (name, current_time, current_date))
            conn.commit()
            print(f"{name} marked as present for {current_date} at {current_time}")

            if name in self.absent_students:
                self.absent_students.remove(name)
                print(f"{name} removed from absent_students list")

            if name != 'unknown':
                self.absent_students.append(name)
                # self.send_sms_notification()

        conn.close()

    def process(self, stream):
        if self.get_face_database():
            while stream.isOpened():
                self.frame_cnt += 1
                logging.debug("Frame " + str(self.frame_cnt) + " starts")
                flag, img_rd = stream.read()
                kk = cv2.waitKey(1)

                faces = detector(img_rd, 0)
                self.last_frame_face_cnt = self.current_frame_face_cnt
                self.current_frame_face_cnt = len(faces)

                self.last_frame_face_name_list = self.current_frame_face_name_list[:]
                self.last_frame_face_centroid_list = self.current_frame_face_centroid_list
                self.current_frame_face_centroid_list = []

                if (self.current_frame_face_cnt == self.last_frame_face_cnt):
                    logging.debug("No face count changes in this frame")
                    self.current_frame_face_position_list = []

                    if "unknown" in self.current_frame_face_name_list:
                        self.reclassify_interval_cnt += 1


                    if self.current_frame_face_cnt != 0:
                        for k, d in enumerate(faces):
                            self.current_frame_face_position_list.append(tuple(
                                [faces[k].left(), int(faces[k].bottom() + (faces[k].bottom() -
faces[k].top()) / 4)]))
                            self.current_frame_face_centroid_list.append(
                                [int(faces[k].left() + faces[k].right()) / 2,
                                 int(faces[k].top() + faces[k].bottom()) / 2])


                            img_rd = cv2.rectangle(img_rd,
                                                   tuple([d.left(), d.top()]),
                                                   tuple([d.right(), d.bottom()]),
                                                   (255, 255, 255), 2)


                    if self.current_frame_face_cnt != 1:
                        self.centroid_tracker()


                    for i in range(self.current_frame_face_cnt):
                        img_rd = cv2.putText(img_rd, self.current_frame_face_name_list[i],
                                             self.current_frame_face_position_list[i], self.font, 0.8, (0,
255, 255), 1,
                                             cv2.LINE_AA)
                    self.draw_note(img_rd)


                else:
                    logging.debug("Faces count changes in this frame")
                    self.current_frame_face_position_list = []
                    self.current_frame_face_X_similarity_list = []
                    self.current_frame_face_feature_list = []
                    self.reclassify_interval_cnt = 0


                    if self.current_frame_face_cnt == 0:
                        logging.debug("No faces in this frame")
                        self.current_frame_face_name_list = []


                    else:
                        logging.debug("Get faces in this frame and do face recognition")
                        self.current_frame_face_name_list = []
                        for i in range(len(faces)):
                            shape = predictor(img_rd, faces[i])    #feature_1 comes from here
                            self.current_frame_face_feature_list.append(
                                face_reco_model.compute_face_descriptor(img_rd, shape))
                            self.current_frame_face_name_list.append("unknown")


                        for k in range(len(faces)):
                            logging.debug("For face %d in current frame:", k + 1)
                            self.current_frame_face_centroid_list.append(
                                [int(faces[k].left() + faces[k].right()) / 2,
                                 int(faces[k].top() + faces[k].bottom()) / 2])

                            self.current_frame_face_X_similarity_list = []


                            self.current_frame_face_position_list.append(tuple(
                                [faces[k].left(), int(faces[k].bottom() + (faces[k].bottom() -
faces[k].top()) / 4)]))


                            for i in range(len(self.face_features_known_list)):
                                if str(self.face_features_known_list[i][0]) != '0.0':
                                    similarity_tmp = self.cosine_similarity(
                                        self.current_frame_face_feature_list[k],
                                        self.face_features_known_list[i]
                                    )
                                    logging.debug("With person %d, the similarity: %f", i + 1,
similarity_tmp)
                                    self.current_frame_face_X_similarity_list.append(similarity_tmp)
                                else:
                                    self.current_frame_face_X_similarity_list.append(0)


                            similar_person_num = self.current_frame_face_X_similarity_list.index(
                                max(self.current_frame_face_X_similarity_list))


                            if self.current_frame_face_X_similarity_list[similar_person_num] >
self.similarity_threshold:
                                self.current_frame_face_name_list[k] =
self.face_name_known_list[similar_person_num]
                                logging.debug("Face recognition result: %s",
                                              self.face_name_known_list[similar_person_num])


                                nam = self.face_name_known_list[similar_person_num]
                                print(type(self.face_name_known_list[similar_person_num]))
                                print(nam)
```

```
                                self.attendance(nam)
                            else:
                                logging.debug("Face recognition result: Unknown person")


                        self.draw_note(img_rd)

                if kk == ord('w'):
                    print("Available name: ", self.face_name_known_list)
                    present_students = [name for name in self.face_name_known_list if name
not in self.absent_students]
                    if present_students:
                        # Send email notification to absent students
                        self.send_email_notification(present_students)
                        print("Absent Students:", present_students)
                    break


                self.update_fps()
                cv2.namedWindow("camera", 1)
                cv2.imshow("camera", img_rd)


                logging.debug("Frame ends\n\n")

# For student to view attendance

from flask import Flask, render_template, request, redirect, url_for, session
import sqlite3
import os
from datetime import datetime, timedelta


app = Flask(__name__)


app.secret_key = 'abcd'

# Path of cropped faces
path_images_from_camera = "data/data_faces_from_camera/"


# Load attendance data from SQLite database
def fetch_attendance(username):
    conn = sqlite3.connect('attendance.db')
    cursor = conn.cursor()


    # Get start and end date for current month
    today = datetime.today()
    start_of_month = today.replace(day=1)
    end_of_month = start_of_month.replace(month=today.month+1) - timedelta(days=1)

    # Fetch attendance data up to current date of the month
    cursor.execute("SELECT date, time FROM attendance WHERE name = ? AND date
BETWEEN ? AND ?",
               (username, start_of_month.strftime('%Y-%m-%d'), today.strftime('%Y-%m-
%d')))
    attendance_data = cursor.fetchall()

    conn.close()

    # Prepare attendance details for each date up to current date in the month
    attendance_details = []
    present_count = 0
    total_days = (today - start_of_month).days + 1
    current_date = start_of_month
    while current_date <= today:
        formatted_date = current_date.strftime('%Y-%m-%d')
        present = any(entry[0] == formatted_date for entry in attendance_data)
        # Fetch time if present, else set to empty string
        time_present = next((entry[1] for entry in attendance_data if entry[0] ==
formatted_date), '')
        status = 'Present' if present else 'Absent'
        attendance_details.append((formatted_date, status, time_present))
        if present:
            present_count += 1
        current_date += timedelta(days=1)

    # Calculate attendance percentage
    attendance_percentage = (present_count / total_days) * 100 if total_days > 0 else 0

    return attendance_details, attendance_percentage, total_days, present_count

# Load student directories and extract names for authentication
student_directories = os.listdir(path_images_from_camera)
student_usernames = [directory.split('_')[2] for directory in student_directories]

# Login route
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        name = request.form.get('username')
        password = request.form.get('password')


        if name in student_usernames and name == password:
            return redirect(url_for('attendance', username=name))
        else:
            return render_template('login.html', error='Invalid credentials')
    return render_template('login.html')

# Authentication route
@app.route('/authenticate', methods=['POST'])
def authenticate():
    name = request.form.get('username')
    password = request.form.get('password')

    if name in student_usernames and name == password:
        return redirect(url_for('attendance', username=name))
    else:
        return render_template('login.html', error='Invalid credentials')

# Logout route
@app.route('/logout')
def logout():
    # Clear the session
    session.clear()
    return redirect(url_for('login'))

# Attendance route
@app.route('/attendance/<username>')
def attendance(username):
    if username not in student_usernames:
        return redirect(url_for('login'))

    attendance_data, attendance_percentage, total_days, present_count =
fetch_attendance(username)

    if not attendance_data:
        return render_template('attendance.html', username=username, no_data=True,
attendance_percentage=attendance_percentage,
total_days=total_days,present_count=present_count)
    else:
        return render_template('attendance.html', username=username,
attendance_data=attendance_data, attendance_percentage=attendance_percentage,
total_days=total_days, present_count=present_count)


if __name__ == '__main__':
    try:
        app.run(debug=True)
    except RuntimeError as e:
        if 'The session is unavailable because no secret key was set' in str(e):
            print("Error: Please set the 'secret_key' for the application.")
```