

Automated Attendance System Using Face Recognition

A

Mini Project Report

Submitted to



Jawaharlal Nehru Technological University, Hyderabad

*In partial fulfillment of the requirements for the
award of the degree of*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

ROHITH REDDY MADDI

(21VE1A0556)

POOJITHA NALAVATH

(21VE1A0545)

UDAY BABU PARISE

(21VE1A0548)

CH VINAY KUMAR

(22VE5A0501)

Under the Guidance

of

Mrs. A. ANITHA

ASSISTANT PROFESSOR



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)

Beside Indu Aranya, Nagole, Hyderabad-500068, Ranga Reddy Dist.

(2021-2025)



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project Report on **“AUTOMATED ATTENDANCE SYSTEM USING FACE RECOGNITION”** submitted by **Poojitha Nalavath, Uday Babu Parise, Rohith Reddy Maddi, CH Vinay Kumar** bearing Hall ticket numbers: **21VE1A0545, 21VE1A0548, 21VE1A0556, 22VE5A0501** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2024-2025 is a record of bonafide work carried out by them under our guidance and Supervision.

Project Coordinator
Dr. U.M FERNANDES DIMLO
Professor & Head

Head of the Department
Dr.U .M FERNANDES DIMLO
Professor & Head

Internal Guide
Mrs. A. ANITHA
Assistant Professor

External Examiner



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We **Poojitha Nalavath , Uday Babu Parise , Rohith Reddy Maddi , CH Vinay Kumar** bearing Hall ticket numbers: **21VE1A0545, 21VE1A0548, 21VE1A0556, 22VE5A0501** hereby declare that the Mini Project titled **AUTOMATED ATTENDANCE SYSTEM USING FACE RECOGNITION** done by us under the guidance of **Mrs.A.ANITHA, Assistant Professor** which is submitted in the partial fulfillment of the requirement for the award of the B.Tech degree in **Computer Science and Engineering** at **Sreyas Institute of Engineering and Technology** for **Jawaharlal Nehru Technological University, Hyderabad** is our original work.

POOJITHA NALAVATH

(21VE1A0545)

UDAY BABU PARISE

(21VE1A0548)

ROHITH REDDY MADDI

(21VE1A0556)

CH VINAY KUMAR

(22VE5A0501)

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs. A. ANITHA, Assistant Professor, Department of Computer Science and Engineering** for her constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **DR. U. M. FERNANDES DIMLO, Head of the Department and Project Coordinator** who has been a source of Continuous motivation and support. He had taken time and effort to guide and correct us all through the span of this work.

We owe very much to the **Department Faculty, Principal** and the **Management** who made our team at Sreyas Institute of Engineering and Technology a stepping stone for our career. We treasure every moment we had spent in college.

Last but not the least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

POOJITHA NALAVATH	(21VE1A0545)
UDAY BABU PARISE	(21VE1A0548)
ROHITH REDDY MADDI	(21VE1A0556)
CH VINAY KUMAR	(22VE5A0501)

ABSTRACT

Attendance management is a vital daily activity in educational institutions, workplaces, and organizations, traditionally carried out using manual methods such as roll calls or name-based identification. These conventional approaches are time-intensive, prone to human error, and inefficient, especially in large groups. This project proposes an automated attendance system that leverages face recognition technology to address these challenges and modernize the process. The system is designed to be installed in classrooms or similar environments, where it captures and processes the facial data of individuals for attendance tracking. Students' information, including their name, roll number, class, section, and facial images, is pre-registered and stored in a structured dataset. Using the OpenCV library, the system extracts, processes, and trains the facial images to create a robust recognition model. Before the start of a class, students interact with the system, which scans their faces, matches the captured data with the stored dataset, and automatically records attendance upon successful identification. By eliminating the need for manual intervention, the system enhances accuracy, saves time, and ensures a seamless and efficient attendance process. Furthermore, it aligns with the growing need for digital transformation by introducing a secure, user-friendly, and technologically advanced solution that improves operational efficiency and minimizes errors.

KEYWORDS: Attendance System, Face Recognition, Automated Process, Manual Process, OpenCV, Student Information, Roll Number, Photographs, Time Management, Classroom Device, Dataset Training, Modernization, Automation.

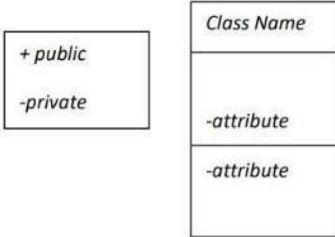

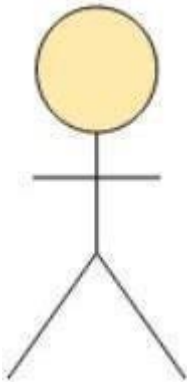
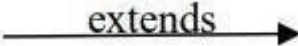
S.NO		TABLE OF CONTENTS	PAGE NO.
1		INTRODUCTION	1-5
	1.1	GENERAL	1
	1.2	PROBLEM STATEMENT	2
	1.3	EXISTING SYSTEM	2
	1.3.1	DRAWBACKS	3
	1.4	PROPOSED SYSTEM	4
	1.4.1	ADVANTAGES	5
2		LITERATURE SURVEY	6-7
	2.1	TECHNICAL PAPERS	6
3		REQUIREMENTS	8
	3.1	GENERAL	8
	3.2	HARDWARE REQUIREMENTS	8
	3.3	SOFTWARE REQUIREMENTS	8
4		SYSTEM DESIGN	9-24
	4.1	GENERAL	9
	4.2	SYSTEM ARCHITECTURE	11
	4.3	FLOW CHART	12
	4.4	UML DISGN	13
	4.4.1	USE-CASE DIAGRAM	14
	4.4.2	CLASS DIAGRAM	16
	4.4.3	ACTIVITY DIAGRAM	18
	4.4.4	SEQUENCE DIAGRAM	21
	4.5	E-R DIAGRAM	22
	4.6	DATA FLOW DIAGRAM	23







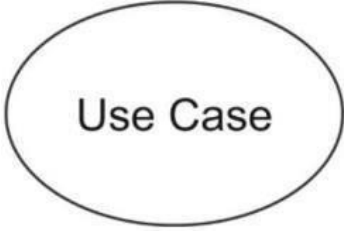
5		TECHNOLOGY DESCRIPTION	25-40
	5.1	WHAT IS PYTHON?	25
	5.2	ADVANTAGES OF PYTHON	26
	5.3	LIBRARIES	29
	5.4	DISADVANTAGES OF PYTHON	39
6		IMPLEMENTATION	41-52
	6.1	METHODOLOGY	41
	6.2	SAMPLE CODE	44
7		TESTING	53-56
	7.1	GENERAL	53
	7.2	TYPES OF TESTING	53
	7.3	TEST CASES	56
8		RESULTS	57-60
	8.1	RESULTS SCREENSHOTS	57
9		FUTURE SCOPE	61
10		CONCLUSION	62
11		REFERENCES	63

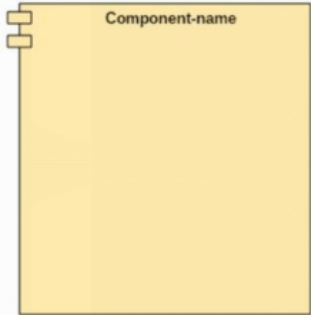
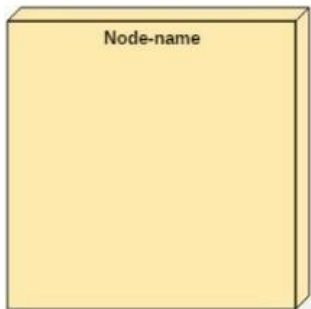
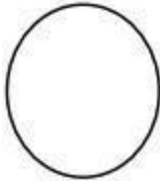


FIG. NO/TAB.NO	LIST OF FIGURES AND TABLES	PAGE NO.
4.1	Architecture Diagram	11
4.2	Flow Chart	12
4.3	Use-Case Diagram	16
4.4	Class Diagram	18
4.5	Activity Diagram	21
4.6	Sequence Diagram	22
4.7	E-R Diagram	23
4.8	Data Flow diagram	24
7.2	Test Cases	56



SCREENSHOT. NO	LIST OF SCREENSHOTS	PAGE. NO
8.1	GUI Page	57
8.2	Registration Page	58
8.3	Mark Attendance	58
8.4	Attendance Saved	59
8.5	View Attendance	59
8.6	Send Mail	60
8.7	Output	60

LIST OF SYMBOLS

SNO.	Name of Symbol	Notation	Description
1	CLASS		Represents a collection of similar entities grouped together.
2	ASSOCIATION		Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3	ACTOR		It aggregates several classes into a single class.
4	RELATION (uses)	<i>Uses</i>	Used for additional process communication.
5	RELATION (extends)		Extends relationship is used when one use case is similar to another use case.

6	COMMUNICATION		Communication between various use cases.
7	STATE		State of the process
8	INITIAL STATE		Initial state of the object
9	FINAL STATE		Final state of the object
10	CONTROL FLOW		Represents various control flow between the states.
11	DECISION BOX		Represents decision making process from a constraint
12	USE CASE		Interaction between the system and external environment.

13	COMPONENT		Represents physical modules which is a collection of components.
14	NODE		Represents physical modules which are a collection of components.
15	DATA PROCESS/ STATE		A circle in DFD represents a state or process which has been triggered due to some event or action.
16	EXTERNAL ENTITY		Represents external entities such as keyboard, sensors, etc
17	TRANSITION		Represents communication that occurs between processes.

18	OBJECT LIFELINE		Represents the vertical dimensions that the object communications.
19	MESSAGE		Represents the message exchanged.

CHAPTER 1

INTRODUCTION

1.1 GENERAL

To verify the student attendance record, the personnel staff ought to have an appropriate system for approving and maintaining the attendance record consistently. By and large, there are two kinds of student attendance framework, i.e. Manual Attendance System (MAS) and Automated Attendance System (AAS). Practically in MAS, the staff may experience difficulty in both approving and keeping up every student's record in a classroom all the time. In a classroom with a high teacher-to-student ratio, it turns into an extremely dreary and tedious process to mark the attendance physically and cumulative attendance of each student. Consequently, we can execute a viable framework which will mark the attendance of students automatically via face recognition. AAS may decrease the managerial work of its staff. Especially, for an attendance system which embraces Human Face Recognition (HFR), it normally includes the students' facial images captured at the time he/she is entering the classroom, or when everyone is seated

in the classroom to mark the attendance. Generally, there are two known methodologies to deal with HFR, one is the feature-based methodology and the other is the brightness-based methodology. The feature-based methodology utilizes key point features present on the face, called landmarks, of the face, for example, eyes, nose, mouth, edges or some other unique attributes. In this way, out of the picture that has been extricated beforehand, just some part is covered during the calculation process. Then again, the brightness-based methodology consolidates and computes all parts of the given picture. It is also called holistic-based or image-based methodology. Since the overall picture must be considered, the brightness-based methodology takes longer handling time and is likewise more complicated. There are different advances that are done during the process of this face recognition framework, yet the essential steps of these are face detection and face recognition. Firstly, to mark the attendance, the images of students' faces will be required. This image can be captured from the camera, which will be installed in the classroom at a position from where the entire classroom is visible. This image will be considered as an input to the system. For efficient face identification, the picture should be upgraded by utilizing some image processing methods like grayscale conversion and histogram equalization. After image quality upgrade, the image will be passed to perform face detection. The face identification

process is trailed by face recognition process. There are different strategies accessible for face recognition like Eigen face, PCA and LDA hybrid algorithm. In the Eigen face, when faces are identified, they are trimmed from the picture. With the assistance of the element extractor, different face highlights are extracted. Utilizing these faces as Eigen features, the student is recognized and by coordinating with the face database, their attendance is marked. Developing the face database is required with the end goal of comparison.

1.2 PROBLEM STATEMENT

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as 4 calling student names or checking respective identification cards. There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names, attendance sheet is passed around the classroom during the lecture sessions. The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class. Thus, face recognition attendance system is proposed in order to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance. Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

1.3 EXISTING SYSTEM

To verify the student attendance record, the personnel staff ought to have an appropriate system for approving and maintaining the attendance record consistently. By and large, there are two kinds of student attendance framework, i.e. Manual Attendance System (MAS) and Automated Attendance System (AAS). Practically in MAS, the staff may

experience difficulty in both approving and keeping up every student's record in a classroom all the time. In a classroom with a high teacher-to-student ratio, it turns into an extremely dreary and tedious process to mark the attendance physically and cumulative attendance of each student. Consequently, we can execute a viable framework which will mark the attendance of students automatically via face recognition. AAS may decrease the managerial work of its staff.

1.3.1 DISADVANTAGES OF EXISTING SYSTEM

1. Privacy Concerns

- Storing facial images can raise concerns about data privacy and unauthorized usage.

2. High Initial Setup Cost

- Installing cameras and setting up the system may require significant initial investment.

3. Accuracy Issues

- The system may struggle in cases of poor lighting, image quality, or changes in facial appearance (e.g., facial hair, glasses).

4. Dependence on Technology

- System malfunctions, camera failures, or software bugs can disrupt attendance tracking.

1.4 PROPOSED SYSTEM

The proposed face recognition-based attendance system leverages K-Nearest Neighbors (KNN) for real-time, automated attendance management, specifically designed for educational institutions. This system begins with a data collection phase, where facial images of students are captured and stored in a structured database. During this stage, images undergo preprocessing techniques such as grayscale conversion and histogram equalization to enhance facial features and optimize recognition accuracy.

For real-time attendance, a live camera feed captures images as students enter the classroom. Using OpenCV's Haar Cascade classifier, the system identifies faces within each frame, cropping and resizing each detected face to a standard format. These facial images are then converted into feature vectors, which serve as input for the KNN classifier. By comparing these vectors to existing records in the database, the KNN algorithm determines the closest match based on Euclidean distance, effectively identifying each student in real-time. When a match is confirmed, the system records the student's attendance automatically in an Excel file, updating the attendance database without requiring manual input.

1.4.1 ADVANTAGES OF PROPOSED SYSTEM

1. Automation and Efficiency

- The system automatically records attendance, reducing manual effort and minimizing errors.

2. Fraud Prevention

- It eliminates issues like proxy attendance and fake sign-ins, ensuring accurate attendance records.

3. Real-time Processing

- Using machine learning and image processing, attendance can be marked instantly as students enter the classroom.

4. Time-saving

- The system reduces the time required for traditional roll calls or signing attendance sheets.

CHAPTER 2

LITERATURE SURVEY

2.1 Student Attendance System in Classroom Using Face Recognition Technique

Authors: S. Lukas, A. R. Mitra, R. I. Desanti and D. Krisnadi

Abstract:

Nowadays, many applications such as video monitoring/surveillance system, human-computer interaction, door access control system and network security use biometric authentication. One of the biometric identification is using fingerprint. It is considered to be the best and fastest method because every person has unique fingerprint and does not change in one's lifetime. Fingerprint recognition is a mature field today, but using face recognition technique is still better to be applied in capturing the presence of the student in the class. Other advantages using face recognition are knowing the attitude of students in class such as students readiness or interestedness in lecture. This paper discusses a method for managing student attendance system in classroom using multiple facial images for classifying the facial objects. From the experiments conducted by involving 19 students situated in classroom setting, it results in 174 out of 205 successful faces recognition. Recognition rate is about 85%.

2.2 Attendance System based on Face Recognition using Eigen face and PCA Algorithms

Authors: P. Wagh, S. Patil, J. Chaudhari and R. Thakare

Abstract:

A Face recognition system is an application of computer vision and image processing which is capable of performing two major tasks of identifying and verifying a person from an image or a video database. The objective of this paper is to automate the attendance system by integrating the face recognition technology using Eigen Face database and PCA algorithm with Matlab GUI. In Conventional attendance system there are several issues like fake attendance, lot of time consumption, manipulation of attendance, information cannot be secure. There are many limitations in implementing face recognition technologies like Image Quality, Image Size, Face angle, varying intensity of light. In order to overcome these issues various techniques like Illumination Invariant, Histogram equalization, PCA are used. By using this system attendance is updated automatically after comparing the detected face with original Eigen database in Excel sheet integrated with Matlab GUI.

2.3 Robust real-time face detection

Authors: Viola, M. J. Jones and Pau

Abstract:

This paper describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions. The first is the introduction of a new image representation called the “Integral Image” which allows the features used by our detector to be computed very quickly. The second is a simple and efficient classifier which is built using the AdaBoost learning algorithm (Freund and Schapire, 1995) to select a small number of critical visual features from a very large set of potential features. The third contribution is a method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions. A set of experiments in the domain of face detection is presented. The system yields face detection performance comparable to the best previous systems (Sung and Poggio, 1998; Rowley et al., 1998; Schneiderman and Kanade, 2000; Roth et al., 2000). Implemented on a conventional desktop, face detection proceeds at 15 frames per second.

CHAPTER 3

TECHNICAL REQUIREMENTS

3.1 GENERAL

These are the requirements for doing the project.

They are:

1. Hardware Requirements
2. Software Requirements

3.2 HARDWARE REQUIREMENTS

- **Processor** : minimum intel i3
- **Ram** : 4GB and Higher
- **Hard disk** : 500GB:Minimum

3.3 SOFTWARE REQUIREMENTS

- **Windows OS (windows 7,10,11) Or Linux**
- **Language : Python**
- **GUI : Graphical User Interface**

CHAPTER-4

SYSTEM DESIGN

4.1 GENERAL

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
 - **TECHNICAL FEASIBILITY**
 - **SOCIAL FEASIBILITY**
-
- **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited.

The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

- **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.2 SYSTEM ARCHITECTURE

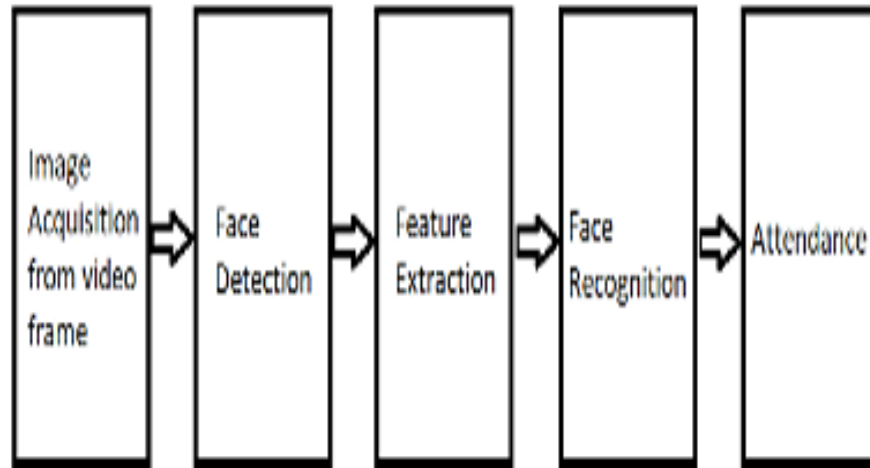


Figure 4.1: Architecture Diagram

4.3 FLOW CHART

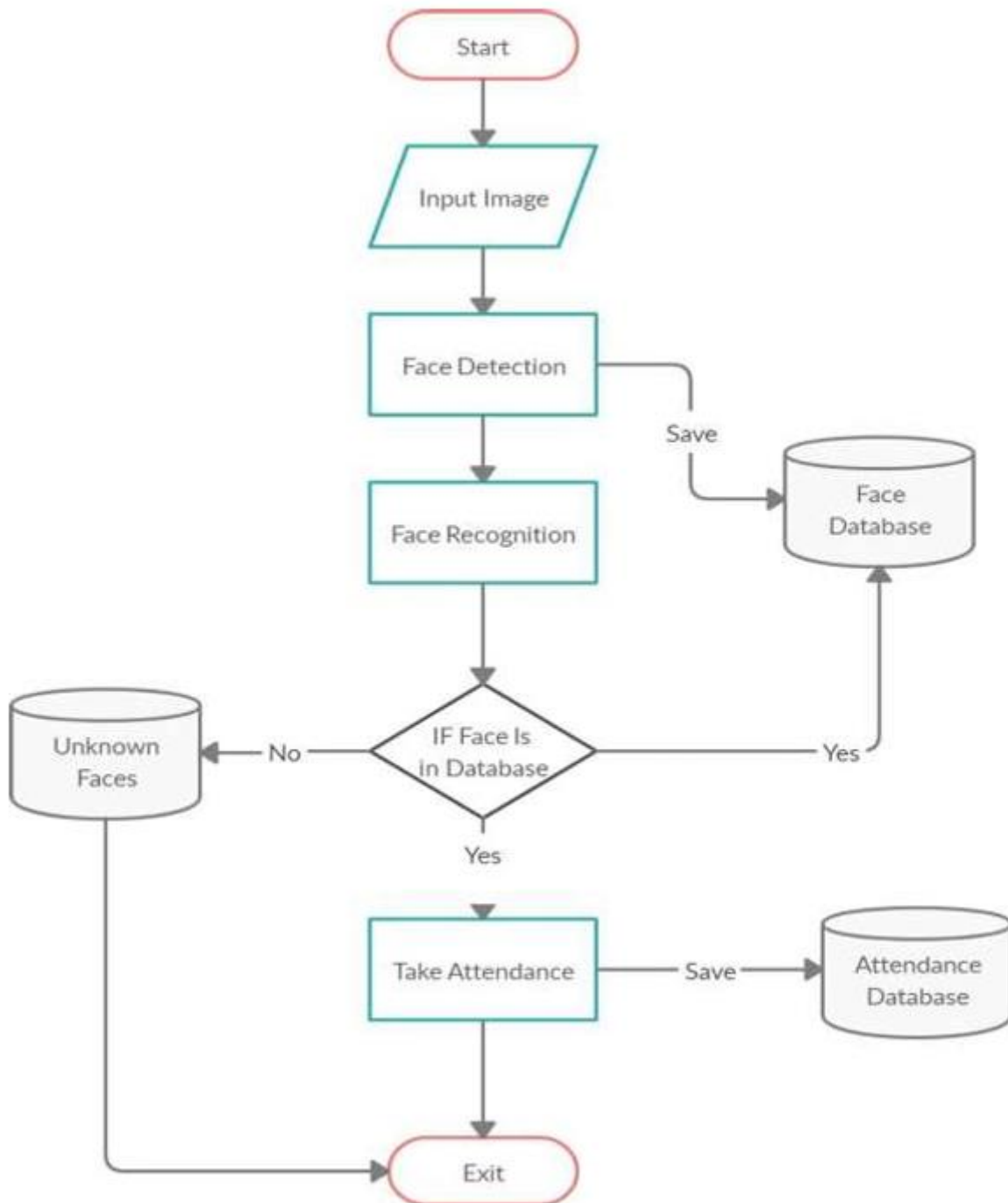


Figure 4.2: Flow Chart

4.4 UML DESIGN

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed.

It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language. Use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis.

It's been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as: ▪ Actors ▪ Business processes ▪ (logical) Components ▪ Activities ▪ Programming Language Statements ▪ Database Schemes ▪ Reusable software components.

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non-programmer's essential requirements, functionalities and processes of the system.
- A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
- UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

The Primary goals in the design of the UML are as follows

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

4.4.1 USE-CASE DIAGRAM

A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior.

Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure.

These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional .

The primary components of a use case diagram include:

- **Actor**

An actor is an external entity that interacts with the system. Actors can be people, other systems, or even hardware devices. Actors are represented as stick figures or simple icons. They are placed outside the system boundary, typically on the left or top of the diagram.

- **Use Case**

A use case represents a specific functionality or action that the system can perform in response to an actor's request. Use cases are represented as ovals within the system boundary.

The name of the use case is written inside the oval.

- **Association Relationship**

An association relationship is a line connecting an actor to a use case. It represents the interaction or communication between an actor and a use case.

The arrowhead indicates the direction of the interaction, typically pointing from the actor to the use case.

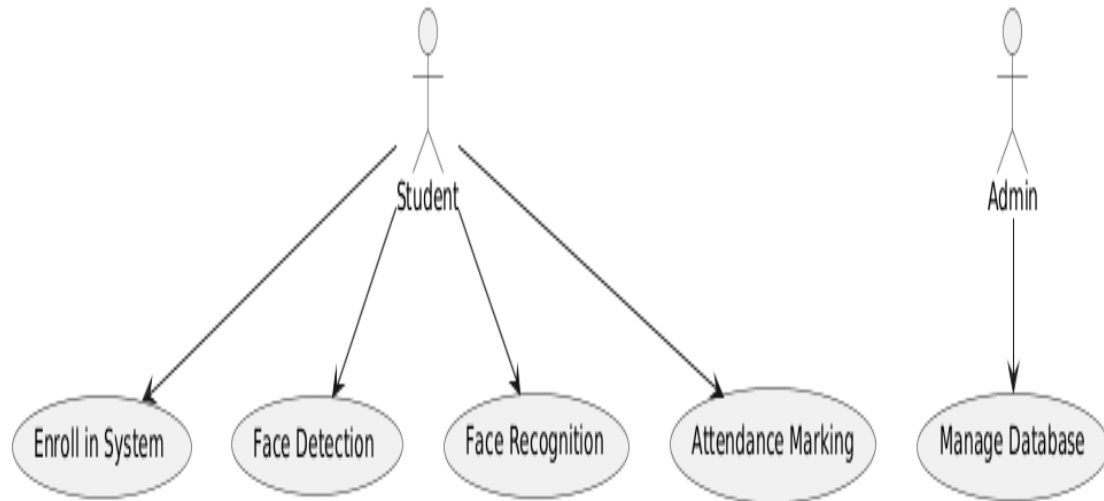


Figure 4.3: Use-Case-Diagram

4.4.2 CLASS DIAGRAM

A class diagram in Unified Modeling Language (UML) is a type of structural diagram that represents the static structure of a system by depicting the classes, their attributes, methods, and the relationships between them. Class diagrams are fundamental in object-oriented design and provide a blueprint for the software's architecture.

Here are the key components and notations used in a class diagram:

- **Class**

A class represents a blueprint for creating objects. It defines the properties (attributes) and behaviors (methods) of objects belonging to that class. Classes are depicted as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes, and the bottom compartment lists the class methods.

- **Attributes**

Attributes are the data members or properties of a class, representing the state of objects. Attributes are shown in the middle compartment of the class rectangle and are typically listed as a name followed by a colon and the data type (e.g., name: String).

- **Methods**

Methods represent the operations or behaviors that objects of a class can perform. Methods are listed in the bottom compartment of the class rectangle and include the methodname, parameters, and the return type (e.g., calculateCost(parameters): ReturnType).

- **Visibility Notations**

Visibility notations indicate the access level of attributes and methods. The common notations are:

+ (public): Accessible from anywhere.

- (private): Accessible only within the class.

(protected): Accessible within the class and its subclasses.

~ (package or default): Accessible within the package.

- **Associations**

Associations represent relationships between classes, showing how they are connected. Associations are typically represented as a solid line connecting two classes. They may have multiplicity notations at both ends to indicate how many objects of each class can participate in the relationship (e.g., 1..*).

Aggregations and Compositions: Aggregation and composition are special types of associations that represent whole-part relationships. Aggregation is denoted by a hollow diamond at the diamond end, while composition is represented by a filled diamond.

Aggregation implies a weaker relationship, where parts can exist independently, while composition implies a stronger relationship, where parts are dependent on the whole.

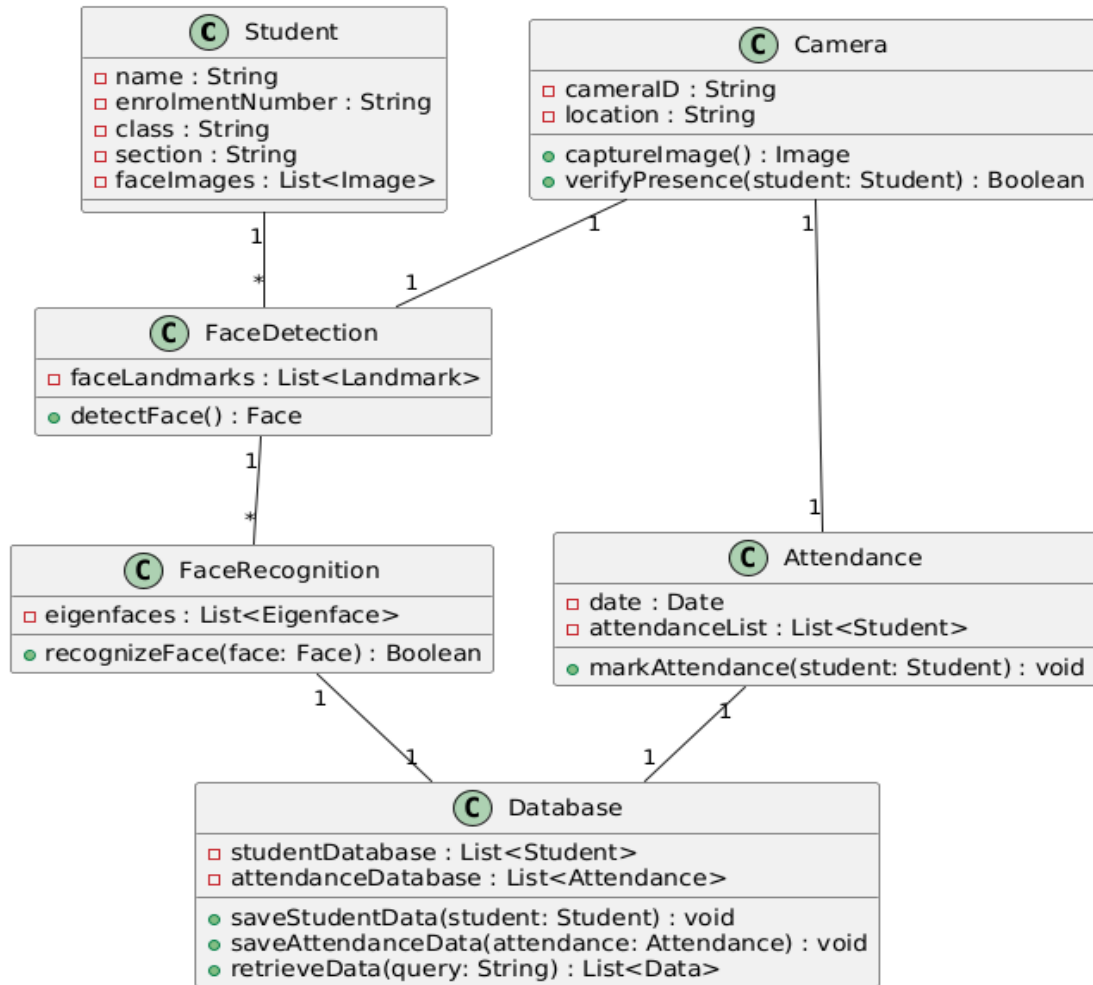


Figure 4.4: Class diagram

4.4.3 ACTIVITY DIAGRAM

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

The diagram might start with an initial activity such as "User approaches the door." This activity triggers the system to detect the presence of the user's Bluetooth-enabled device, initiating the authentication process.

Next, the diagram could depict a decision point where the system determines whether the detected device is authorized. If the device is recognized as authorized, the diagram would proceed to the activity "Unlock the door." Conversely, if the device is not authorized, the diagram might show alternative paths such as prompting the user for additional authentication credentials or denying access.

The key components and notations used in an activity diagram:

- **Initial Node**

An initial node, represented as a solid black circle, indicates the starting point of the activity diagram. It marks where the process or activity begins.

- **Activity/Action**

An activity or action represents a specific task or operation that takes place within the system or a process. Activities are shown as rectangles with rounded corners. The name of the activity is placed inside the rectangle.

- **Control Flow Arrow**

Control flow arrows, represented as solid arrows, show the flow of control from one activity to another. They indicate the order in which activities are executed.

- **Decision Node**

A decision node is represented as a diamond shape and is used to model a decision point or branching in the process. It has multiple outgoing control flow arrows, each labeled with a condition or guard, representing the possible paths the process can take based on condition.

- **Merge Node**

A merge node, also represented as a diamond shape, is used to show the merging of multiple control flows back into a single flow.

- **Fork Node**

A fork node, represented as a black bar, is used to model the parallel execution of multiple activities or branches. It represents a point where control flow splits into multiple concurrent paths.

- **Join Node**

A join node, represented as a black bar, is used to show the convergence of multiple control flows, indicating that multiple paths are coming together into a single flow.

- **Final Node**

A final node, represented as a solid circle with a border, indicates the end point of the activity diagram. It marks where the process or activity concludes.

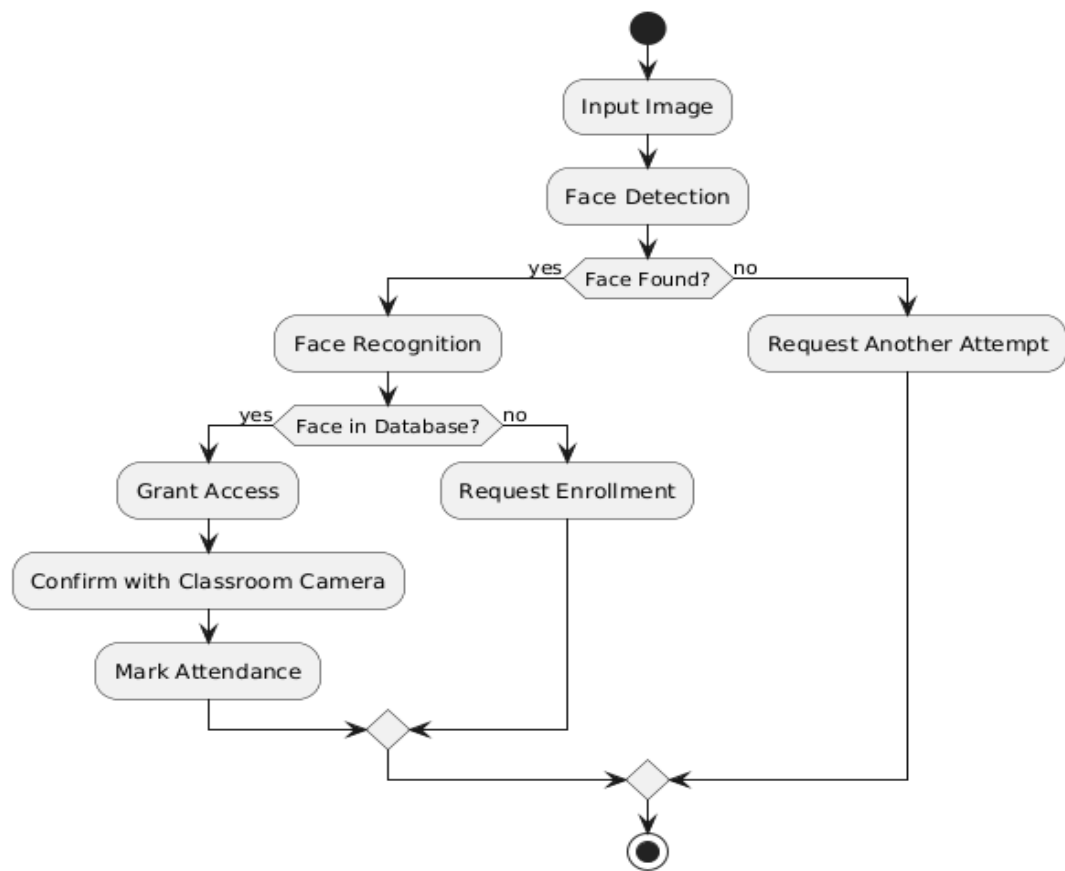


Figure 4.5: Activity diagram

4.4.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

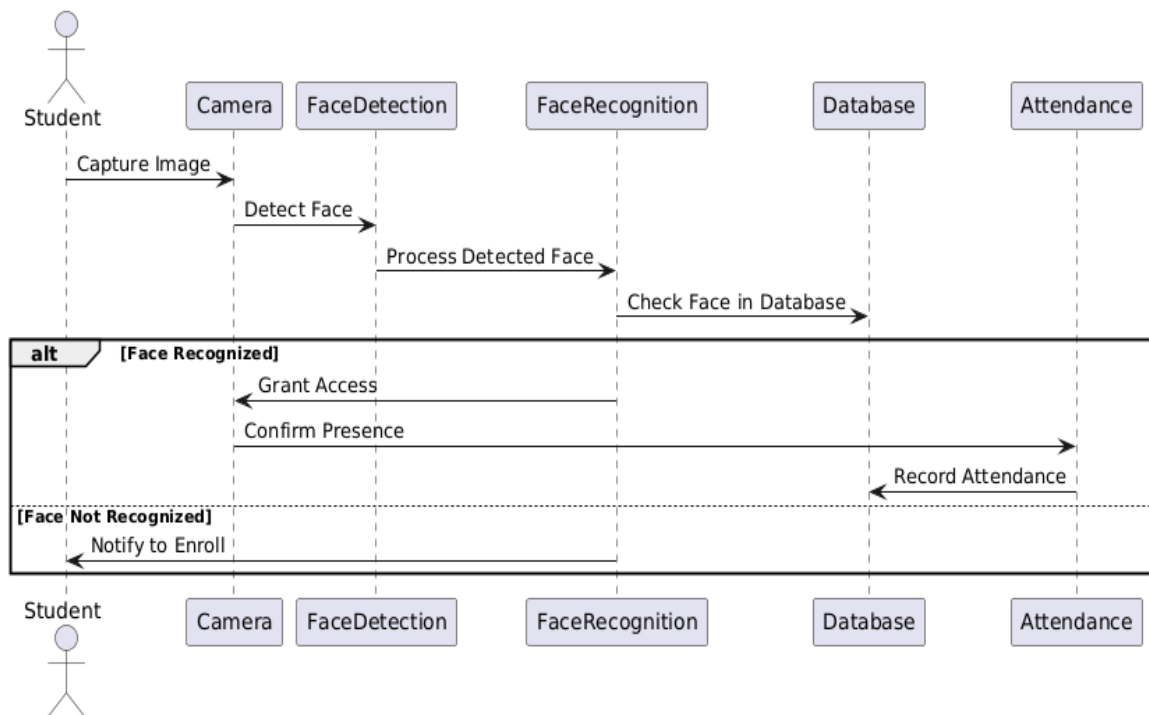


Figure 4.6: Sequence Diagram

4.5 E-R Diagram

An entity-relationship (ER) diagram is a particular kind of graphical model that assists in defining the structure of a database. [ER diagrams](#) give a more enhanced understanding of the actual physical objects and characteristics of a database and the relation that exists between those characteristics. They provide a logical way to map the structures of a system that would enable the database designers to establish relations. Likewise, ER diagrams are mostly used during the design of databases for the purpose of showing how entities and how they are related.

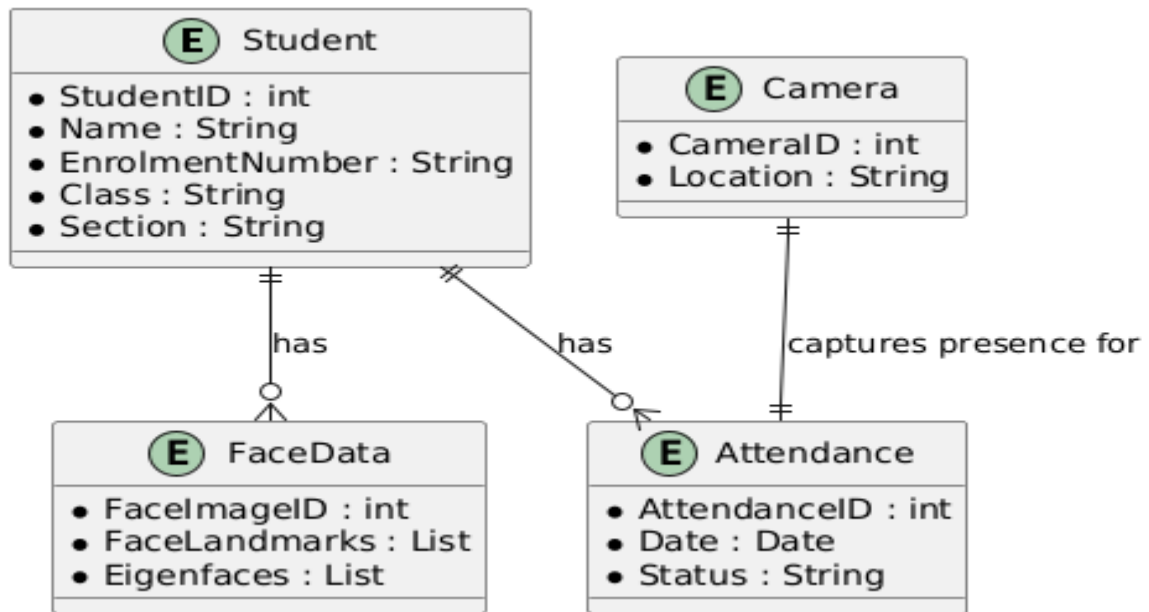


Figure 4.7: E-R Diagram

4.6 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

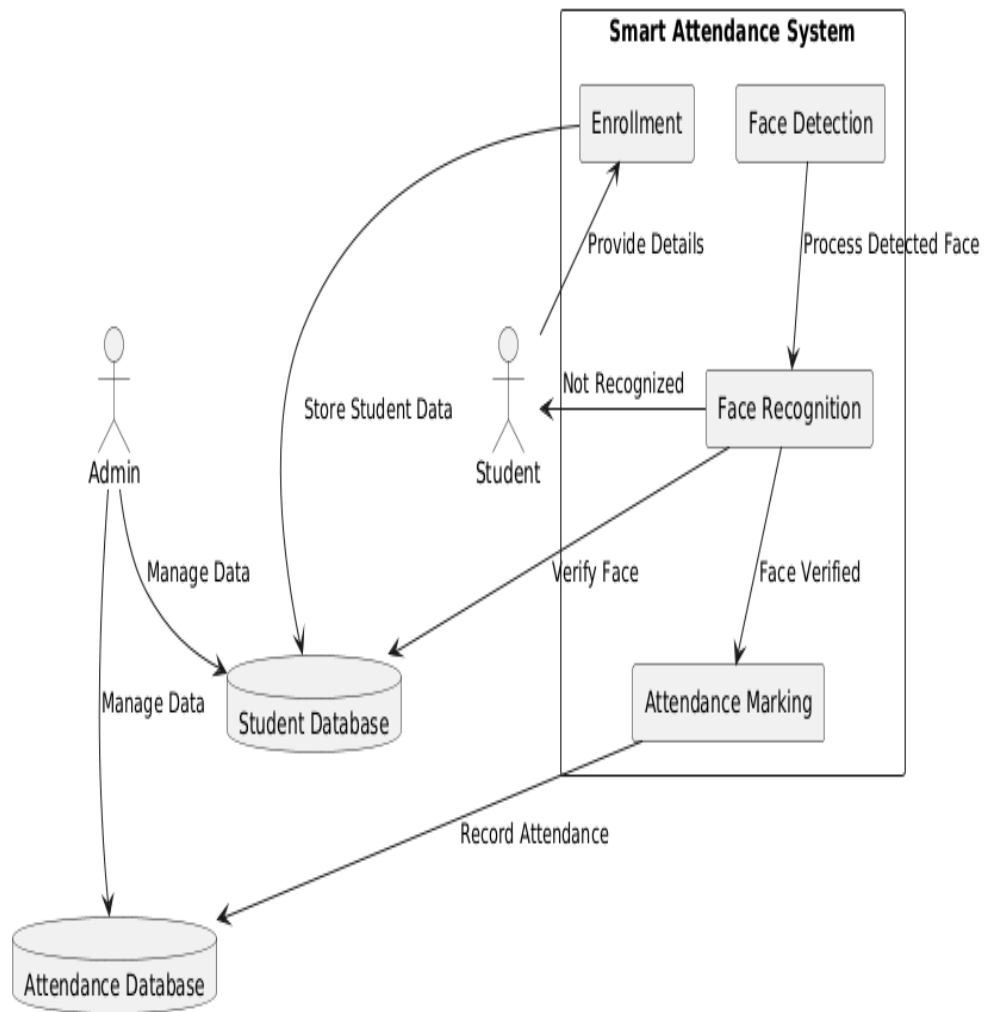


Figure 4.8: Data Flow Diagram

CHAPTER-5

TECHNOLOGY DESCRIPTION

5.1 WHAT IS PYTHON

Python is a high-level, interpreted programming language created by Guido van Rossum in 1989 and first released in 1991. Named after the British comedy group *Monty Python's Flying Circus*, Python was designed to prioritize simplicity and readability, making it accessible for both beginners and seasoned developers. Over the years, Python has evolved significantly, with major milestones including the release of Python 2.0 in 2000, which introduced features like list comprehensions, and Python 3.0 in 2008, which revamped the language to address inconsistencies, albeit at the cost of backward compatibility. Known for its versatility, Python has become one of the most popular programming languages in the world, powering applications in fields like web development, data science, artificial intelligence, and automation. Its extensive libraries, active community, and cross-platform compatibility continue to drive its widespread adoption and success.

Key Features of Python:

1. **Easy to Learn and Use:** Python has a simple syntax that is easy to read and write, which helps reduce development time.
2. **Interpreted Language:** Python code is executed line by line, which makes debugging easier.
3. **Cross-Platform:** Python works on many platforms, including Windows, macOS, Linux, and more.
4. **Extensive Libraries:** Python has a vast standard library and many third-party libraries for various applications (e.g., NumPy for numerical computation, Pandas for data manipulation, TensorFlow for AI/ML).
5. **Object-Oriented and Procedural:** Python supports multiple programming paradigms, including object-oriented, procedural, and functional programming.
6. **Community Support:** Python has a large and active community, providing extensive documentation, forums, and tutorials.

Common Applications:

- **Web Development:** Frameworks like Django and Flask make building websites easier.
- **Data Science and Machine Learning:** Libraries like Pandas, NumPy, TensorFlow, and Scikit-learn are commonly used.
- **Automation:** Python scripts can automate repetitive tasks.
- **Game Development:** Frameworks like Pygame help in developing games.
- **Embedded Systems:** Python is used in microcontroller programming with frameworks like MicroPython.

5.2 ADVANTAGES OF PYTHON

1. **Simple and Readable Syntax:**
 - a. Python's syntax is clean and easy to understand, making it beginner-friendly.
2. **Versatile and Multi-Purpose:**
 - a. Python can be used for web development, data science, machine learning, AI, automation, and more.
3. **Extensive Standard Library:**
 - a. Includes modules for file I/O, databases, web services, and more, reducing the need for external libraries.
4. **Large Community Support:**
 - a. Python has a vast and active community that provides tutorials, libraries, and quick solutions to problems.
5. **Cross-Platform Compatibility:**
 - a. Python works on all major platforms, such as Windows, macOS, and Linux, ensuring code portability.
6. **Support for Multiple Paradigms:**
 - a. It supports object-oriented, procedural, and functional programming, offering flexibility in coding style.
7. **Rich Ecosystem of Libraries and Frameworks:**
 - a. Libraries like Pandas, NumPy, TensorFlow, and frameworks like Django and Flask accelerate development.

8. Ease of Integration:

- a. Python can easily integrate with other languages like C, C++, and Java, and tools like databases or web services.

9. Open Source and Free:

- a. Python is free to use and distribute, even for commercial purposes.

10. Ideal for Rapid Prototyping:

- a. With its simplicity and powerful libraries, Python allows quick development and testing of ideas.

11. High Demand in Industry:

- a. Python is widely used in modern fields like data science, AI, and automation, making it a sought-after skill.

12. Built-in Garbage Collection:

- a. Python automates memory management, reducing the risk of memory leaks.
- b. These features make Python one of the most preferred programming languages in the world.

13. Dynamic Typing:

- a. Python does not require explicit declaration of variable types, making it faster to write and easier to adapt.

14. Interpreted Language:

- a. Python code is executed line-by-line, which simplifies debugging and error detection.

15. Strong Support for Data Handling:

- a. Libraries like NumPy, Pandas, and Matplotlib make Python a powerhouse for data manipulation, analysis, and visualization.

16. Popularity in Academia and Research:

- a. Due to its simplicity and powerful tools, Python is extensively used in education and scientific research.

17. Readable Code:

- a. Python's focus on indentation and minimal syntax improves code readability and maintainability.

18. Automation and Scripting:

- a. Python excels in automating repetitive tasks, making it a favorite for DevOps and system administrators.

19. Scalability:

- a. Python can handle small projects to enterprise-level solutions, scaling with ease.

20. Testing Frameworks:

- a. Python provides tools like unittest and pytest, simplifying testing and debugging processes.

21. Embedded Development:

- a. Python can be used with platforms like Raspberry Pi and MicroPython for IoT and embedded systems development.

22. Extensive Documentation:

- a. Python has comprehensive and well-structured documentation for developers at all levels.

23. Support for Concurrent Development:

- a. Libraries like asyncio allow developers to write concurrent and asynchronous programs efficiently.

24. Compatibility with Big Data and Cloud Computing:

- a. Python is widely used in big data processing (e.g., Apache Spark) and cloud platforms for scalability and analytics.

25. Built-in REPL (Read-Eval-Print Loop):

- a. Python provides an interactive shell, making it easier to test code snippets in real-time.

26. Powerful IDEs and Editors:

- a. Python is supported by various IDEs and editors, such as PyCharm, VS Code, Jupyter Notebook, and Spyder, offering enhanced development tools.

27. Global Recognition:

- a. Python is consistently ranked as one of the top programming languages in the world, ensuring its relevance for future projects.

28. Continuous Updates and Evolution:

- a. Python evolves regularly, with new features and optimizations added to address modern needs.

29. Comprehensive Error Handling:

- a. Python's try-except blocks and error handling mechanisms make applications robust and fault-tolerant.

30. Rich Web Frameworks:

- a. Frameworks like Flask, Django, and FastAPI simplify web development, enabling developers to build scalable web applications.

31. Integration with AI/ML Tools:

- a. Python works seamlessly with TensorFlow, PyTorch, and Scikit-learn, making it the go-to language for artificial intelligence and machine learning.

32. Backward Compatibility:

- a. While Python 3 was a significant shift, newer versions maintain consistency, making upgrades smoother.

33. Game Development Support:

- a. Python libraries like Pygame are used for creating 2D and 3D games.

34. Customizability:

- a. Python's open-source nature allows developers to modify its behavior or write extensions in other languages like C/C++.

35. Community-Supported Events and Tutorials:

- a. Global events like PyCon and online platforms provide opportunities to learn and grow in Python.

5.3 LIBRARIES

NumPy:

Description:

NumPy is a core library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices. It also offers a wide range of mathematical functions to operate on these arrays. NumPy serves as the foundation for most scientific computing tasks in Python, enabling efficient operations with large datasets and complex mathematical models.

Features:

- **N-dimensional Arrays:**

- Provides the ndarray object for working with large, multi-dimensional arrays and matrices.
- **Mathematical Functions:**
 - Includes functions for linear algebra, statistical operations, and Fourier transforms.
- **Array Broadcasting:**
 - Allows for operations on arrays of different shapes and sizes with efficient broadcasting.
- **Integration with C/C++/Fortran:**
 - Can be interfaced with low-level languages like C, C++, and Fortran for performance optimizations.
- **Random Module:**
 - Includes functionality for generating random numbers and performing random sampling.
- **Array Manipulation:**
 - Tools for reshaping, slicing, and manipulating arrays with high performance.

Pandas:

Description:

Pandas is a powerful library for data manipulation and analysis, providing data structures like Series and DataFrame to handle structured data. It simplifies tasks like reading, cleaning, transforming, and analyzing data, and is a key tool in the data science ecosystem.

Features:

- **DataFrame:**
 - A two-dimensional, labeled data structure ideal for representing tabular data.
- **Data Handling:**
 - Supports easy data manipulation like filtering, sorting, grouping, and merging datasets.
- **File I/O:**
 - Built-in functions to read/write data from various formats, including CSV, Excel, SQL databases, and JSON.

- **Handling Missing Data:**
 - Tools to handle and fill missing values efficiently.
- **Time Series:**
 - Supports working with time series data, including resampling, time zone conversions, and date arithmetic.
- **Pivot Tables and Aggregations:**
 - Tools for reshaping data and performing complex aggregation operations.

Matplotlib:

Description:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It allows users to create a wide variety of plots, including line plots, scatter plots, bar charts, and more, with customization options to suit diverse data visualization needs.

Features:

- **Basic Plotting:**
 - Functions for creating line plots, scatter plots, bar charts, histograms, and pie charts.
- **Customization:**
 - Offers control over plot elements like labels, titles, colors, and legends.
- **Interactive Plots:**
 - Supports interactive plots that allow zooming and panning for better data exploration.
- **Animation:**
 - Allows the creation of animated plots for dynamic data visualization.
- **Export Options:**
 - Supports saving plots in various formats, including PNG, PDF, SVG, and EPS.
- **Integration with Pandas:**
 - Works seamlessly with Pandas DataFrames for quick visualization of data.

TensorFlow:

Description:

TensorFlow is an open-source framework developed by Google for building machine learning and deep learning models. It provides a comprehensive ecosystem of tools and libraries for constructing, training, and deploying ML models across various platforms.

Features:

- **Flexible Model Building:**
 - Supports building and training neural networks using both high-level APIs (like Keras) and low-level control for fine-grained customization.
- **Keras Integration:**
 - Keras, a user-friendly neural networks API, is included in TensorFlow for rapid prototyping and model building.
- **TensorFlow Lite:**
 - A lightweight solution for deploying models on mobile devices and embedded systems.
- **TensorFlow Serving:**
 - A system for serving machine learning models in production environments.
- **Distributed Training:**
 - Provides tools for scaling training across multiple GPUs and machines, improving model performance on large datasets.
- **Visualization with TensorBoard:**
 - Integrated visualization tools for monitoring the training process, visualizing model graphs, and debugging.

Flask:**Description:**

Flask is a lightweight web framework for building web applications in Python. It is designed to be simple, flexible, and easy to use, making it ideal for small to medium-sized applications or microservices. Flask provides the essentials for building web apps while allowing developers to choose other components as needed.

Features:

- **Minimalist Core:**
 - Provides basic functionality like URL routing, request handling, and templating, while leaving the rest to the developer.
- **Jinja2 Templating:**
 - Uses the Jinja2 template engine to render HTML and manage dynamic content in web pages.
- **Built-in Development Server:**
 - Comes with a simple web server for development purposes, supporting debugging and hot reloading.
- **Extensible:**
 - Can be extended with plugins and libraries like Flask-SQLAlchemy, Flask-WTF, and Flask-Login to handle additional tasks such as databases and form validation.
- **RESTful Support:**
 - Easily supports RESTful API design and integration for web services and applications.
- **Secure:**
 - Offers tools for protecting against common security vulnerabilities, such as CSRF and XSS.

Scikit-learn:

Description:

Scikit-learn is a powerful library for machine learning in Python, providing simple and efficient tools for data mining and data analysis. It includes various algorithms for classification, regression, clustering, and dimensionality reduction, making it one of the most widely used libraries for machine learning in Python.

Features:

- **Supervised Learning:**
 - Includes algorithms for classification (e.g., SVM, Decision Trees, Naive Bayes) and regression (e.g., Linear Regression, Ridge, Lasso).
- **Unsupervised Learning:**

- Offers clustering algorithms (e.g., K-Means, DBSCAN) and dimensionality reduction methods (e.g., PCA, t-SNE).
- **Model Selection:**
 - Tools for splitting datasets, cross-validation, and hyperparameter tuning (e.g., GridSearchCV, RandomizedSearchCV).
- **Preprocessing:**
 - Includes methods for scaling, encoding, and transforming data (e.g., StandardScaler, OneHotEncoder).
- **Pipeline:**
 - Support for creating end-to-end pipelines that integrate preprocessing and model training.

Requests:

Description:

Requests is a simple and elegant HTTP library for Python, designed to make sending HTTP requests easier and more human-friendly. It abstracts away the complexity of handling HTTP requests, allowing developers to send HTTP requests with a few lines of code.

Features:

- **HTTP Methods:**
 - Supports all HTTP methods like GET, POST, PUT, DELETE, PATCH, etc.
- **URL Handling:**
 - Easily handles URL parameters, headers, and cookies.
- **Authentication:**
 - Built-in support for HTTP authentication methods like Basic, Digest, and OAuth.
- **Session Objects:**
 - Allows persistent sessions across requests to handle cookies and keep connections alive.
- **Response Handling:**
 - Provides easy access to response content, status codes, and metadata.

PyTorch:

Description:

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab. It is widely used for creating and training deep neural networks and supports both research and production environments. PyTorch provides a flexible and dynamic approach to building neural networks with high-performance computing capabilities.

Features:

- **Dynamic Computational Graphs:**
 - Uses dynamic computation graphs, allowing for flexible model building and easier debugging.
- **Tensors:**
 - Supports multi-dimensional arrays (tensors), similar to NumPy, but with GPU acceleration.
- **Neural Networks:**
 - Provides modules for building neural networks, including predefined layers, loss functions, and optimizers.
- **Autograd:**
 - Automatic differentiation for computing gradients, which simplifies backpropagation in training deep learning models.
- **TorchScript:**
 - A way to serialize models for running them in production environments outside of Python.
- **GPU Acceleration:**
 - Seamlessly integrates with CUDA for GPU acceleration, making training large models efficient.

BeautifulSoup:

Description:

BeautifulSoup is a Python library used for web scraping purposes to extract data from HTML and XML documents. It provides methods for navigating and modifying the parse tree and is especially helpful for working with poorly structured or malformed web pages.

Features:

- **HTML/XML Parsing:**
 - Parses HTML and XML documents and converts them into a tree structure, making it easy to navigate and search.
- **Search and Navigation:**
 - Allows searching for tags, attributes, and text using methods like `find()`, `find_all()`, and CSS selectors.
- **Modifying the Parse Tree:**
 - Supports modifying the document's tree, such as inserting, deleting, or replacing tags and content.
- **Unicode Support:**
 - Handles Unicode content and encodings gracefully.
- **Automatic Error Recovery:**
 - Can handle and recover from malformed HTML, ensuring a smooth parsing experience.

SQLAlchemy:

Description:

SQLAlchemy is a powerful SQL toolkit and Object Relational Mapper (ORM) for Python. It provides an easy-to-use interface to interact with relational databases, and its ORM allows you to map Python objects to database tables, simplifying database management.

Features:

- **ORM (Object Relational Mapper):**

- Allows developers to define Python classes that map to database tables and interact with the database using Python objects.
- **SQL Expression Language:**
 - Provides an expressive way to construct SQL queries programmatically, using Pythonic syntax.
- **Database Support:**
 - Supports a wide range of database backends, including MySQL, PostgreSQL, SQLite, and Oracle.
- **Connection Pooling:**
 - Manages database connections efficiently with built-in connection pooling.
- **Migration Support:**
 - Works with libraries like Alembic for database schema migrations.

Keras:

Description:

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It simplifies the process of building deep learning models by providing an intuitive and modular interface for defining layers, loss functions, optimizers, and more.

Features:

- **User-Friendly API:**
 - Provides a simple and modular interface for quickly building and prototyping deep learning models.
- **Model Building:**
 - Supports building complex models, including sequential models and functional API models.
- **Extensibility:**
 - Easily extendable with custom layers, metrics, and loss functions.
- **GPU Acceleration:**
 - Supports running on multiple GPUs with minimal code changes for improved performance.

- **TensorFlow Backend:**

- Works seamlessly with TensorFlow, leveraging its capabilities for model training, optimization, and deployment.

Flask-SQLAlchemy:

Description:

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy, allowing developers to easily integrate relational databases into Flask web applications. It combines the simplicity of Flask with the power of SQLAlchemy, making it easier to work with databases.

Features:

- **SQLAlchemy Integration:**

- Simplifies the integration of SQLAlchemy with Flask by handling database connections and sessions.

- **Declarative Base:**

- Uses SQLAlchemy's declarative base to define models with a simple, intuitive syntax.

- **Session Management:**

- Manages database sessions automatically within Flask routes, ensuring clean and efficient transactions.

- **Database URI:**

- Supports configuration for database URIs, enabling connection to various database types (e.g., MySQL, SQLite).

5.4 DISADVANTAGES OF PYTHON

1. **Slower Execution Speed:**

- Python is an interpreted language, making it slower compared to compiled languages like C++ or Java.

2. **Global Interpreter Lock (GIL):**

- Python's GIL restricts the execution of multiple threads simultaneously, limiting its performance in multi-threaded applications.

3. Weak in Mobile Development:

- Python is not widely used for mobile app development due to slower performance and lack of native support.

4. High Memory Consumption:

- Python's dynamic nature and data types consume more memory, making it less suitable for memory-constrained environments.

5. Database Access Limitations:

- Python's database access layers are less robust compared to languages like Java, which has JDBC for seamless database connectivity.

6. Runtime Errors:

- Python's dynamically-typed nature can lead to runtime errors that are harder to catch during development.

7. Dependency Management:

- Managing dependencies across different environments can be challenging, although tools like virtualenv and pip help mitigate this.

8. Not Ideal for Low-Level Programming:

- Python is not suitable for system-level programming like drivers or operating systems, as it lacks low-level access to hardware.

9. Limited Mobile Frameworks:

- Python has fewer frameworks for mobile development compared to languages like Swift or Kotlin.

10. Poor Performance in Multi-Core Applications:

- Due to GIL, Python is not efficient for CPU-bound tasks requiring multi-core processing.

11. Lack of Commercial Support:

- Unlike some proprietary languages, Python does not have dedicated commercial support; it relies on community contributions.

12. Compatibility Issues Between Versions:

- The transition from Python 2 to Python 3 caused compatibility issues, and some projects still face legacy code challenges.

13. Not Always Suitable for Large Applications:

- Python's slower execution and higher memory use can limit its scalability for very large applications.

14. Steep Learning Curve for Libraries:

- While Python itself is easy to learn, mastering its extensive libraries and frameworks can be time-consuming.

15. Less Suitable for Performance-Critical Applications:

- Applications requiring high-speed processing, like games or 3D rendering, are not Python's strong suit.

16. Limited Mobile and Browser Support:

- Python is not commonly used for mobile apps or in-browser applications due to a lack of native support.

17. Dependency on External Libraries:

- Many advanced features in Python rely heavily on third-party libraries, which can lead to dependency management challenges.

18. No True Private Access Modifiers:

- Python does not enforce strict private variables, making encapsulation less secure than in languages like Java or C++.

19. Less Suitable for Real-Time Applications:

- Python's slower speed and higher latency make it less ideal for real-time systems like embedded or time-sensitive applications.

20. Verbose Code for Simple Tasks:

- Sometimes, Python code can be more verbose compared to specialized languages for specific tasks (e.g., R for statistical modeling).

CHAPTER 6

IMPLEMENTATION

6.1 METHODOLOGY

To develop the smart attendance management system, some steps are required to be followed for accomplishing this task successfully. The steps can be defined in the following ways:

1. Enrolment
2. Face Detection
3. Face Recognition
4. Confirmation by the class camera
5. Attendance Marking

1. Enrollment

In this step, the student is enrolled in the student database. General information like Name, Enrolment Number is stored in the database. Along with all this information, pictures of the student's face appearing in the camera window are also stored in the student database. With the help of all the images stored in the student database, facial recognition can be performed for all the students are coming to attend a lecture.

2. Face Detection

In this face recognition-based attendance system, the **face detection** process is a crucial first step, responsible for accurately identifying faces in a live video feed. Implemented using **OpenCV's Haar Cascade Classifier**, this detection system is designed for real-time performance, using pre-trained data that can recognize essential facial features like eyes, nose, and mouth. When the camera captures images. Once a face is detected, the system highlights it by drawing a bounding box around the face area, making it easy to monitor the success of each detection. This identified face region is then cropped and resized,

preparing it for feature extraction and subsequent recognition by the KNN classifier. By employing Haar Cascade for detection, the system achieves a balance of speed and accuracy, making it suitable for real-time, efficient attendance tracking in educational settings.

3. Face Recognition

To implement the facial recognition in this model, we will make use of the Principle Component Analysis (PCA). PCA is a methodology utilized for lessening the quantity of variables which are used in face recognition. In PCA, each picture in the training dataset is represented as a linearly weighted eigenvector called eigenfaces. This methodology change faces into a small arrangement of basic qualities, eigenfaces, which are the principal parts of the underlying arrangement of learning pictures. Recognition is implemented by anticipating another picture in the eigenface subspace, after which the individual is arranged by contrasting its current position in eigenface space and the position of known people. The main benefits of using PCA for facial recognition is ease of use, speed and not changing its judgment based on changes on the human face. The students, appearing on the camera present outside the class, will have their face recognized in order to get access to enter the classroom. If the student's face is present in the respective database, then he is allowed the access to enter the class, else if his face image is not present in the database then the system will ask the student to enrol himself in the student database before gaining access in the classroom.

4. Confirmation by the classroom camera

After the face of a student is recognized successfully and the student is allowed access to the classroom, in order to confirm that the student is present in the class for the lecture, a second camera installed inside the classroom will be set up in such a way that all the students are visible. This will help in cancelling out the proxies.

5. Attendance Marking

At the end of the lectures, the camera inside the classroom will be used to provide the list of students present in the classroom. With the help of this, attendance for that lecture will be marked in the attendance database.

Opencv

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Computer Vision

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields –

- **Image Processing** – It focuses on image manipulation.
- **Pattern Recognition** – It explains various techniques to classify patterns.
- **Photogrammetry** – It is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

Image processing deals with image-to-image transformation. The input and output of image processing are both images.

Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

6.2 SAMPLE CODE

```
pimport tkinter as tk
from tkinter import messagebox, scrolledtext
import cv2
import os
import numpy as np
import pandas as pd
import pickle
from datetime import datetime, timedelta
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Initialize Tkinter
root = tk.Tk()
root.title("Automated Attendance System")
root.geometry("600x400")
root.configure(bg="#E8F0F2")

# Global variables
student_name_var = tk.StringVar()
student_id_var = tk.StringVar()
attendance_file = 'Attendance.csv'
trained_model_file = 'face_trained.yml'
labels_file = 'labels.pkl'
faces_directory = 'faces'
attendance_df = None
email_file = 'students_email.csv'
```

```

# Ensure directories and files exist
if not os.path.exists(faces_directory):
    os.makedirs(faces_directory)
if not os.path.exists(attendance_file):
    attendance_df = pd.DataFrame(columns=["ID", "Name", "Timestamp", "Status"])
    attendance_df.to_csv(attendance_file, index=False)
else:
    attendance_df = pd.read_csv(attendance_file)

# Load email data if exists
if not os.path.exists(email_file):
    pd.DataFrame(columns=["ID", "Email"]).to_csv(email_file, index=False)

def show_main_menu():
    for widget in root.winfo_children():
        widget.destroy()

    tk.Label(root, text="Automated Attendance System", font=("Helvetica", 18, "bold"),
bg="#E8F0F2").pack(pady=20)

    tk.Button(root, text="Register Student", font=("Helvetica", 14),
command=register_student).pack(pady=10)

    tk.Button(root, text="Train Model", font=("Helvetica", 14),
command=train_model).pack(pady=10)

    tk.Button(root, text="Mark Attendance", font=("Helvetica", 14),
command=mark_attendance).pack(pady=10)

    tk.Button(root, text="View Attendance", font=("Helvetica", 14),
command=view_attendance).pack(pady=10)

    tk.Button(root, text="Send Email", font=("Helvetica", 14),
command=send_email).pack(pady=10)

def register_student():
    for widget in root.winfo_children():
        widget.destroy()

```

```

tk.Label(root, text="Register Student", font=("Helvetica", 18, "bold"),
bg="#E8F0F2").pack(pady=20)

tk.Label(root, text="Enter Student Name:", font=("Helvetica", 14),
bg="#E8F0F2").pack(pady=10)

tk.Entry(root, textvariable=student_name_var, font=("Helvetica", 14)).pack(pady=5)

tk.Label(root, text="Enter Student ID:", font=("Helvetica", 14),
bg="#E8F0F2").pack(pady=10)

tk.Entry(root, textvariable=student_id_var, font=("Helvetica", 14)).pack(pady=5)

tk.Label(root, text="Enter Student Email:", font=("Helvetica", 14),
bg="#E8F0F2").pack(pady=10)

student_email_var = tk.StringVar()

tk.Entry(root, textvariable=student_email_var, font=("Helvetica", 14)).pack(pady=5)

tk.Button(root, text="Capture Image", font=("Helvetica", 14), command=lambda:
capture_student_image(student_email_var)).pack(pady=10)

tk.Button(root, text="Back", font=("Helvetica", 14),
command=show_main_menu).pack(pady=10)

def capture_student_image(student_email_var):
    student_name = student_name_var.get().strip()
    student_id = student_id_var.get().strip()
    student_email = student_email_var.get().strip()
    if not student_name or not student_id or not student_email:
        messagebox.showerror("Error", "Please enter all fields")
    return

face_cascade = cv2.CascadeClassifier(cv2.data.harcascades +
'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

cv2.namedWindow("Capture Student Image")

while True:
    ret, frame = cap.read()
    if not ret:
        break

```

```

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow("Capture Student Image", frame)

k = cv2.waitKey(1)
if k % 256 == 32: # Press SPACE to capture image
    img_name = os.path.join(faces_directory, f'{student_id}.jpg')
    cv2.imwrite(img_name, gray[y:y+h, x:x+w]) # Save the detected face region
    messagebox.showinfo("Success", f'{img_name} written!')

    # Save student email information
    students_df = pd.read_csv(email_file)
    students_df = pd.concat([students_df, pd.DataFrame([{"ID": student_id, "Email":
student_email}])], ignore_index=True)
    students_df.to_csv(email_file, index=False)

    break

cap.release()
cv2.destroyAllWindows()
tk.Button(root, text="Back", font=("Helvetica", 14),
command=show_main_menu).pack(pady=10)

def train_model():
    try:
        face_recognizer = cv2.face.LBPHFaceRecognizer_create()
        faces, labels = [], []

        label_map = {}

```

```

image_size = (200, 200) # Set a consistent size for all images

for filename in os.listdir(faces_directory):
    if filename.endswith(".jpg"):
        image_path = os.path.join(faces_directory, filename)
        gray_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        resized_image = cv2.resize(gray_image, image_size) # Resize the image
        faces.append(resized_image)
        labels.append(int(filename.split('.')[0])) # Use student ID as label
        label_map[int(filename.split('.')[0])] = filename.split('.')[0]

faces = np.array(faces)
labels = np.array(labels)
face_recognizer.train(faces, labels)
face_recognizer.write(trained_model_file)
with open(labels_file, 'wb') as f:
    pickle.dump(label_map, f)

messagebox.showinfo("Success", "Model trained successfully!")
except Exception as e:
    messagebox.showerror("Error", str(e))
show_main_menu()

def mark_attendance():
    global attendance_df # Ensure attendance_df is accessible in this function
    try:
        face_recognizer = cv2.face.LBPHFaceRecognizer_create()
        face_recognizer.read(trained_model_file)

        with open(labels_file, 'rb') as f:
            label_map = pickle.load(f)

        face_cascade = cv2.CascadeClassifier(cv2.data.harcascades +
'haarcascade_frontalface_default.xml')

```

```

cap = cv2.VideoCapture(0)
cv2.namedWindow("Attendance System")

marked_names = set()

while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

    for (x, y, w, h) in faces:
        face = gray[y:y+h, x:x+w]
        label, confidence = face_recognizer.predict(face)
        name = label_map[label]

        if name not in marked_names:
            now = datetime.now()
            date_str = now.strftime("%Y-%m-%d")
            existing_entry = attendance_df[(attendance_df["Name"] == name) &
(attendance_df["Timestamp"].str.startswith(date_str))]

            if existing_entry.empty:
                # Mark entry time
                time_stamp = now.strftime("%Y-%m-%d %H:%M:%S")
                new_entry = pd.DataFrame({"ID": [label], "Name": [name], "Timestamp":
[time_stamp], "Status": ["Entry"]})
                attendance_df = pd.concat([attendance_df, new_entry],
ignore_index=True)
                print(f"{name} entry marked at {time_stamp}")
            else:
                last_entry = pd.to_datetime(existing_entry["Timestamp"]).max()
                if now - last_entry >= timedelta(minutes=2):

```

```

        if "Exit" not in existing_entry["Status"].values:
            # Mark exit time
            time_stamp = now.strftime("%Y-%m-%d %H:%M:%S")
            new_entry = pd.DataFrame({"ID": [label], "Name": [name],
"Timestamp": [time_stamp], "Status": ["Exit"]})
            attendance_df = pd.concat([attendance_df, new_entry],
ignore_index=True)
            print(f'{name} exit marked at {time_stamp}')
        else:
            print(f'{name} exit already marked for today.')
        else:
            print(f'{name} entry already marked for today.')

    marked_names.add(name)

    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
    cv2.putText(frame, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(255, 0, 0), 2)

    cv2.imshow("Attendance System", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    cap.release()
    cv2.destroyAllWindows()

    attendance_df.to_csv(attendance_file, index=False)
    messagebox.showinfo("Success", "Attendance saved successfully!")
except Exception as e:
    messagebox.showerror("Error", str(e))
show_main_menu()

def send_email():

```

```

try:
    students_df = pd.read_csv(email_file)
    attendance_df = pd.read_csv(attendance_file)

    for _, student in students_df.iterrows():
        student_id = student['ID']
        email = student['Email']

        # Check if both Entry and Exit are recorded for the student
        student_attendance = attendance_df[attendance_df["ID"] == student_id]

        entry_recorded = not student_attendance[student_attendance["Status"] ==
"Entry"].empty
        exit_recorded = not student_attendance[student_attendance["Status"] ==
"Exit"].empty

        if entry_recorded and exit_recorded:
            status = "Your attendance is successfully recorded for the day."
        else:
            status = "your entry has been marked."

        subject = "Your Attendance Status"
        body = f"Dear Student,\n\nHere is your attendance
status:\n\n{status}\n\nRegards,\nAutomated Attendance System"

        # Setup the email server
        sender_email = "saivamshi939@gmail.com" # Replace with your email
        sender_password = "fkaclisiwjawdsnf" # Replace with your email password

        msg = MIMEMultipart()
        msg['From'] = sender_email
        msg['To'] = email
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))

```



```

# Send email
with smtplib.SMTP('smtp.gmail.com', 587) as server:
    server.starttls()
    server.login(sender_email, sender_password)
    server.sendmail(sender_email, email, msg.as_string())

print(f'Email sent to {email}')

messagebox.showinfo("Success", "Emails sent successfully!")
except Exception as e:
    messagebox.showerror("Error", str(e))

def view_attendance():
    global attendance_df
    for widget in root.winfo_children():
        widget.destroy()
    tk.Label(root, text="View Attendance", font=("Helvetica", 18, "bold"),
bg="#E8F0F2").pack(pady=20)

    text_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=60, height=10,
font=("Helvetica", 12))
    text_area.pack(pady=10)
    attendance_data = pd.read_csv(attendance_file).to_string(index=False)
    text_area.insert(tk.END, attendance_data)

    tk.Button(root, text="Back", font=("Helvetica", 14),
command=show_main_menu).pack(pady=10)

# Show main menu
show_main_menu()

# Run Tkinter event loop
root.mainloop()

```

CHAPTER 7

TESTING

7.1 GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing for a Multilevel Data Concealing Technique that integrates Steganography and Visual Cryptography is crucial to ensure its functionality, security, and reliability. The testing process involves several stages, including unit testing, integration testing, and security testing.

7.2 TYPES OF TESTING

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic

outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or

requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.3 TEST CASES

Test Case	Scenario	Input	Expected Output	Result
TC001	Student Registration	Enter details and click capture	User created Image captured	Pass
TC002	Model Training	Click train model button	Model trained successfully	Pass
TC003	Attendance Marking	Click mark attendance	Entry,Exit time stamps marked	Pass
TC004	View Attendance	Click the button	Attendance log displayed	Pass
TC005	Send Email	Click send button	Email is sent	Pass

Table 7.2 Test Cases

CHAPTER 8

RESULTS

8.1 SCREEN SHOTS

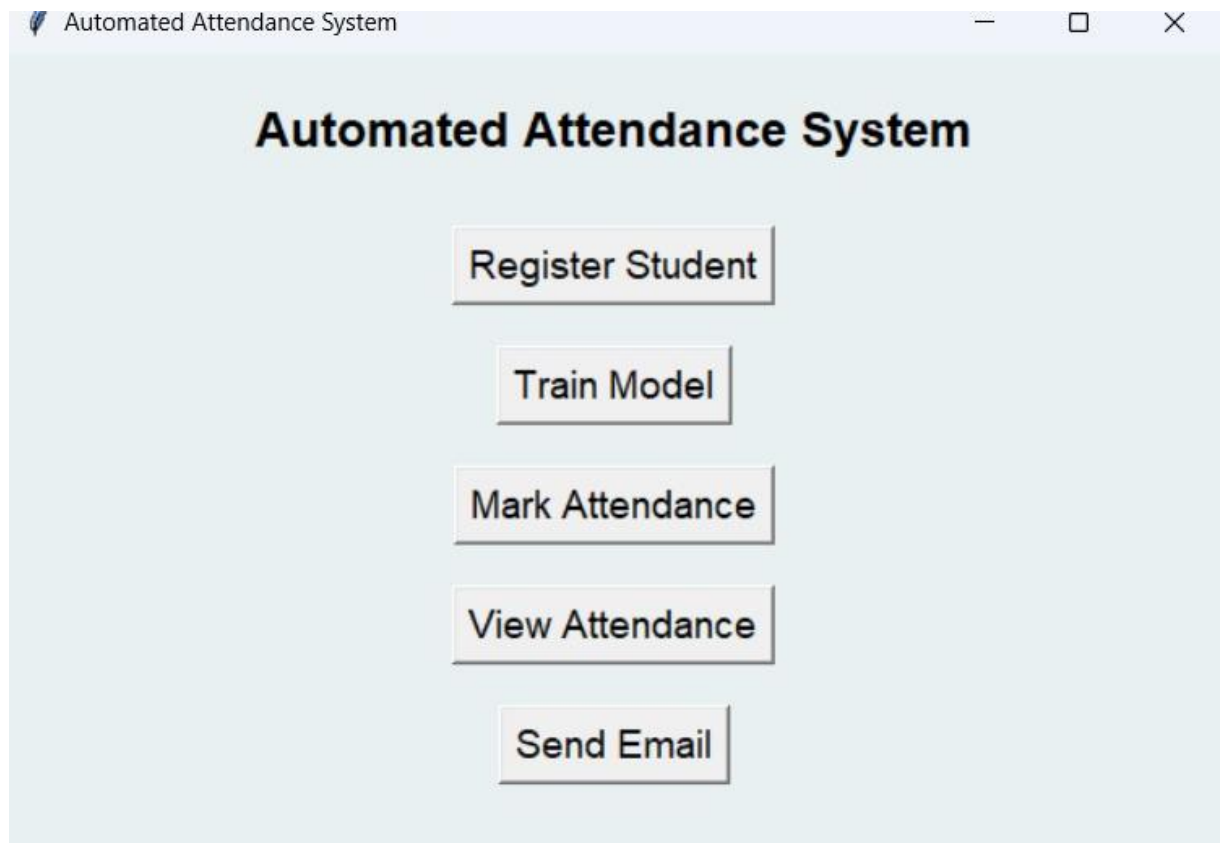


Figure 8.1 GUI Page

Register Student

Enter Student Name:

Rohith Reddy

Enter Student ID:

556

Enter Student Email:

reddymaddi28@gmail.com

Capture Image

Figure 8.2 Registration Page

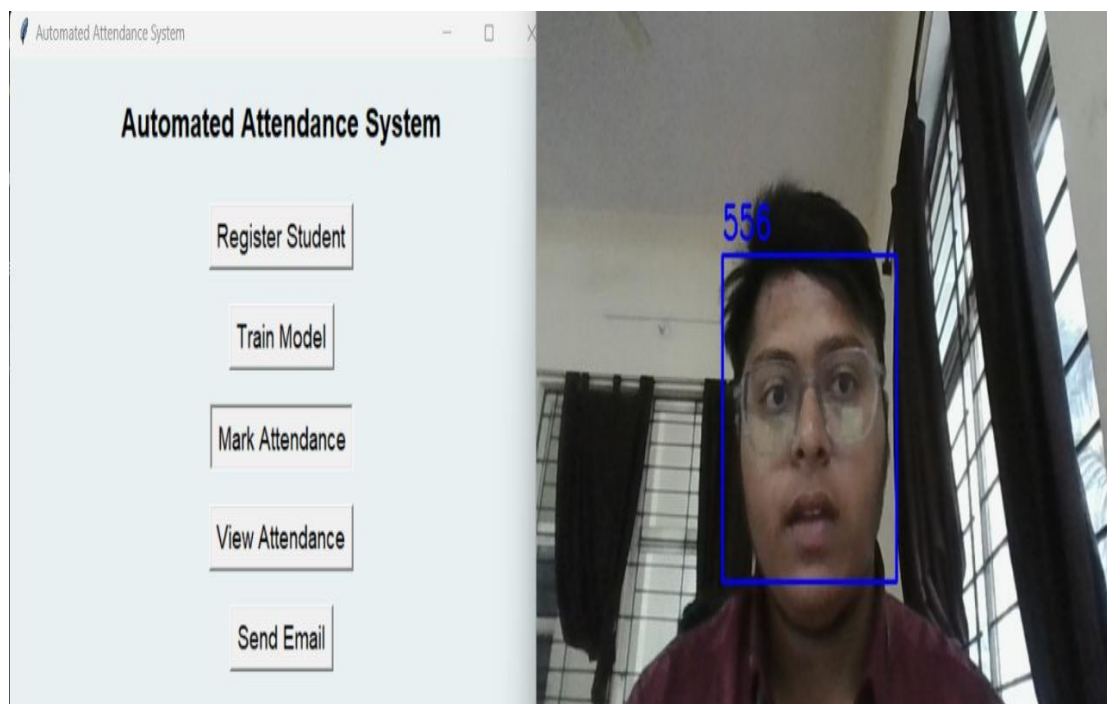


Figure 8.3 Mark Attendance

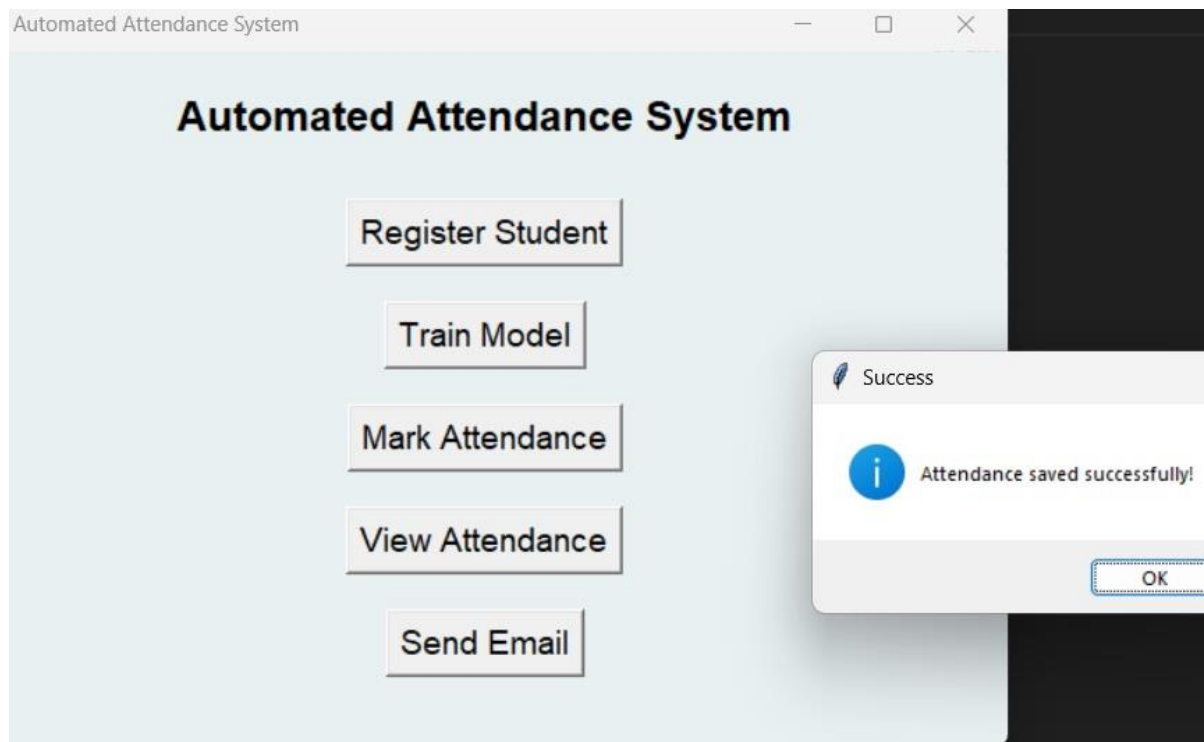


Figure 8.4 Attendance Saved

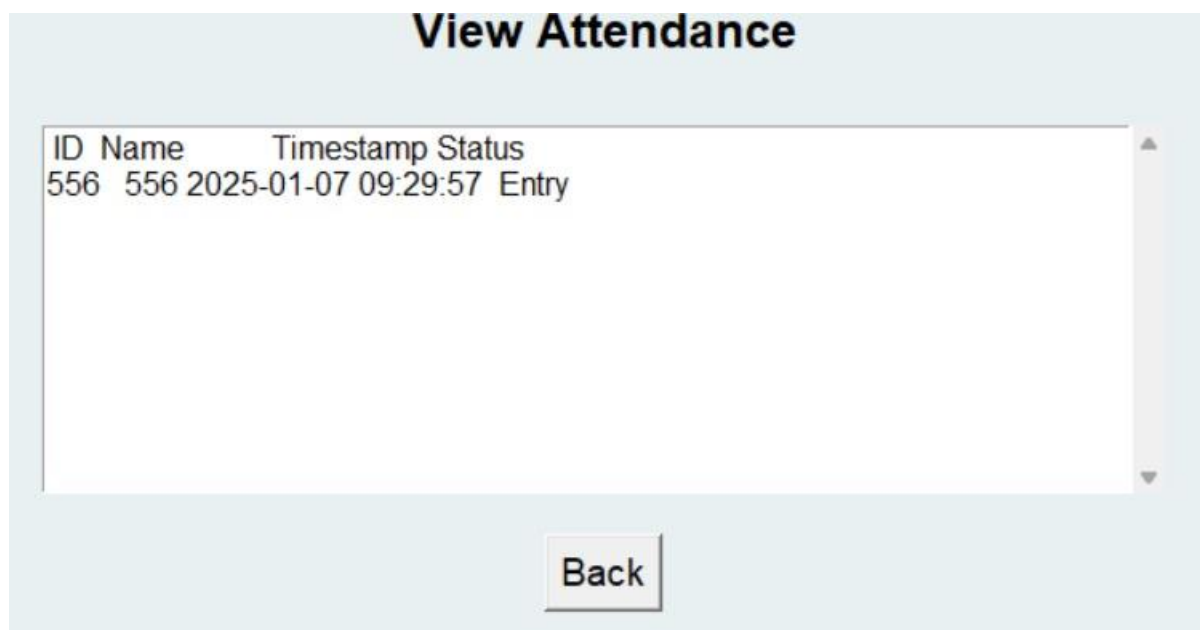


Figure 8.5 View Attendance

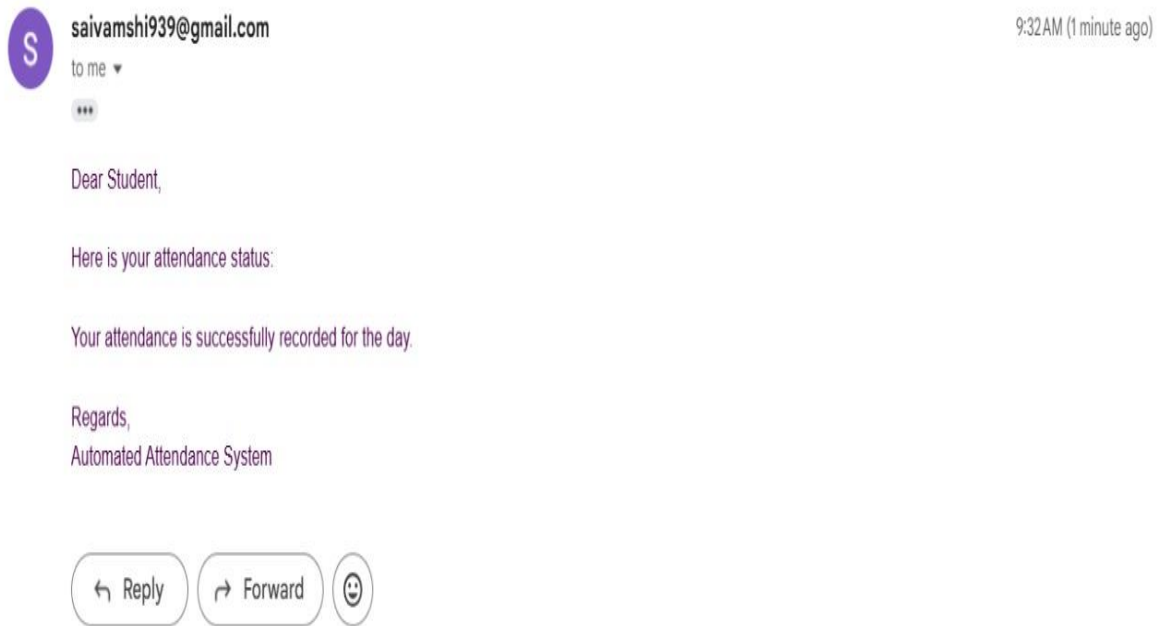


Figure 8.6 Send Mail

```
556 entry marked at 2025-01-07 09:29:57
556 entry already marked for today.
556 exit marked at 2025-01-07 09:32:09
556 entry already marked for today.
Email sent to rohithreddymaddi28@gmail.com
```

Figure 8.7 Output

CHAPTER - 9

FUTURE SCOPE

9.1 FUTURE SCOPE

The proposed system here is only used for classroom attendance for students. However, this system can be improved and enhanced in a way that it can also be used in multi-national companies for maintaining the surveillance of a much larger database, filled with huge amount of entries of the employees working in a particular organization. This will be able to help in maintaining security and also the company will be able to keep a track on its workers whether they are completing the desired working hours in a day or not. This can also be implemented in banks. The ATM machines can be equipped with a facial recognition algorithm. The customers will only be able to access their bank accounts, once their faces have been recognized by the ATM machine on comparison with images which are already saved in the database. This can help in preventing money thefts hence increasing the security while operating ATMs.

CHAPTER-10

CONCLUSION

10.1 CONCLUSION

The proposed automated attendance system using face recognition is a great model for marking the attendance of students in a classroom. This system also assists in overcoming the chances of proxies and fake attendance. In the modern world, a large number of systems using biometrics are available. However, the facial recognition turns out to be a viable option because of its high accuracy along with minimum human intervention. This system is aimed at providing a significant level of security. Hence, a highly pro-efficient attendance system for classroom attendance needs to be developed which can perform recognition on multiple faces at one instance. Also, there is no requirement of any special hardware for its implementation. A camera, a PC and database servers are sufficient for constructing the smart attendance system.

CHAPTER-11

REFERENCES

11.1 REFERENCES

1. S. Lukas, A. R. Mitra, R. I. Desanti and D. Krisnadi, "Student Attendance System in Classroom Using Face Recognition Technique," in ICTC 2016, Karawaci, 2016.
2. P. Wagh, S. Patil, J. Chaudhari and R. Thakare, "Attendance System based on Face Recognition using Eigen face and PCA Algorithms," in 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015.
3. N. M. Ara, N. S. Simul and M. S. Islam, "Convolutional Neural Network Approach for Vision Based Student Recognition System," in 2017 20th International Conference of Computer and Information Technology (ICCIT), 22-24 December, 2017, Sylhet, 2017.
4. N. Khan and Balcoh, "Algorithm for efficient attendance management: Face recognition based approach," in JCSI International Journal of Computer Science Issues 9.4, 2012.
5. KAWAGUCHI and Yohei, "Face Recognition-based Lecture Attendance System.," in The 3rd AEARU Workshop on Network Education. 2005., 2005.
6. MuthuKalyani.K, "Smart Application For AMS using Face Recognition," in CSEIJ 2013, 2013.
7. M. Arsenovic, S. Skadojevic and A. Anderla, "FaceTime- Deep Learning Based Face Recognition Attendance system.," in IEEE 15th International Symposium on Intelligent Systems and Informatics, Serbia, 2017.
8. K. Goyal, K. Agarwal and R. Kumar, "Face Detection and tracking using OpenCV," in International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017.
9. Viola, M. J. Jones and Paul, "Robust real-time face detection.," in International journal of computer vision 57.2 (2004), 2004.
10. A. Jha, ""Class Room Attendance System Using Facial Recognition System.," in International journal of Mathematical science technology and management 2(3). 2007, 2007.