

A Real time Project Report on
Real-Time Face Recognition Attendance System with Python and Excel Integration

Submitted to
Jawaharlal Nehru Technological University, Hyderabad in partial
fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Pandem Varun Kumar Reddy	22831A05E4
Ranga Aswitha	22831A05G6
Kondapuram Sharadha	22831A05H8
Dipak kumar Yadav	22831A05K6
Aklesh Mishra	22831A05K7

Under the Esteemed Guidance of
Mrs.G.Rashmi
Professor & Head of Department



**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING GURU NANAK INSTITUTE OF
TECHNOLOGY**

(Affiliated to JNTU - Hyderabad)

Ranga Reddy District –

5015062023-2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GURU NANAK INSTITUTE OF TECHNOLOGY

(Affiliated to JNTU - Hyderabad)

Ranga Reddy District - 501506



CERTIFICATE

This is to certify that the project entitled “Real-Time Face Recognition Attendance System with Python and Excel Integration” is being presented with a report by Pandem Varun Kumar Reddy (22831A05E4), Ranga Aswitha (22831A05G6), Kondapuram Sharadha (22831A05H8), Dipak kumar Yadav (22831A05K6), and Aklesh Mishra (22831A05K7) in partial fulfilment for the award of Degree of Bachelor of Technology in Computer Science of Engineering, to Jawaharlal Nehru Technological University, Hyderabad.

Mrs.G.Rashmi
Co-ordinator

Dr. B. SANTHOSH KUMAR
Professor & Head of Department



GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpattanam, R.R. Dist. – 501506.

VISION OF GNIT

To be a world-class educational and research institution in the service of humanity by promoting high quality engineering and management education.

MISSION OF GNIT

- Imbibe soft skills and technical skills.
- Develop the faculty to reach international standards.
- Maintain high academic standards and teaching quality that promotes the typical thinking and independent judgment.
- Promote research, innovation and product development by collaboration with reputed
- Foreign universities.



GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpattanam, R.R. Dist. – 501506.

VISION OF DEPARTMENT

To be recognized as a leading department of Computer science and Engineering in the region by students, employers and be known for leadership, Ethics, and commitment to fostering quality teaching-learning, research, and innovation.

MISSION OF DEPARTMENT

- Nurture young individuals into knowledgeable, skill-full and ethical professionals in their Pursuit of computer science and Engineering.
- Nurture the faculty to expose them to world-class infrastructure.
- Sustain high performance by excellence in teaching, research and innovations.
- Extensive partnerships and collaborations with foreign universities for technology upgradation.
- Develop industry-interaction for innovation and product development.



GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpattanam, R.R. Dist. – 501506.

Program Educational Objectives (PSO's)

PSO-1: Graduates shall have the ability to apply knowledge and technical skills in emerging areas of Computer Science and Engineering for higher studies, research, employability, product development and handle realistic problems.

PSO-2: Graduates shall maintain ethical conduct, a sense of responsibility to serve society and protect the environment.

PSO-3: Graduates shall possess academic excellence with innovative insight, soft skills, managerial skills, leadership qualities, knowledge of contemporary issues for successful professional career.

Program Outcomes (PO's)

PO-1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3: Design/development of solutions: Design solutions for complex engineering problems and design system components

or processes that meet specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

PO-4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information

PO-5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations

PO-6: The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7: Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO-9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for and have the preparations and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES(PSO'S)

PSO-1: Professional Skills: The ability to understand the principles and working of computer systems. Students can assess the hardware and software aspects of computer systems.

PSO-2: Problem Solving Skills: The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

PSO-3: Successful Career and Entrepreneurship: The ability to employ modern computer languages, environments, and platforms in creating

DECLARATION

We hereby declare that the major project report entitled “Real-Time Face Recognition Attendance System with Python and Excel Integration” is the work done by Pandem Varun kumar Reddy, Ranga. Aswitha, Kondapuram Sharadha, Dipak Kumar Yadav, and Aklesh Mishra bearing the roll no’s 22831A05E4, 22831A05G6, 22831A05H8, 22831A05K6, and 22831A05K7 towards the fulfillment of the requirement for the award of the Degree of Bachelor of Technology in Computer Science & Engineering, to Jawaharlal Nehru Technological University, Hyderabad, is the result of the work carried out under the Mrs G. Rashmi, Assistant Professor of Computer Science and Engineering, Guru Nanak Institute of Technology, Hyderabad.

We further declare that this project report has not been previously submitted either in part or full for the award of any degree or diploma by any organization or university.

Pandem Varun Kumar Reddy	22831A05E4
Ranga Aswitha	22831A05G6
Kondapuram Sharadha	22831A05H8
Dipak kumar Yadav	22831A05K6
Aklesh Mishra	22831A05K7

ACKNOWLEDGEMENT

“Task successful” makes everyone happy. But happiness would be gold without glitter if we didn’t state the persons who have supported us to make it a success for us. We would like to express our sincere thanks and gratitude to our Principal, **Dr. S. SREENATHA REDDY** and **Dr. B. SANTHOSH KUMAR** ,Professor and head of Department of **Computer Science and Engineering, Guru Nanak Institute of Technology** for having guided me in developing the requisite capabilities for taking up this project. We thank Coordinator **Mrs G..Rashmi**, Assistant Professor **CSE, Guru Nanak Institute of Technology** for providing seamless support and right suggestions that are given in the development of the project. We would also like to thank all our lecturers for helping me in every possible way whenever the need arose. On a more personal note, we thank our beloved parents and friends for their moral support during the course of our project.

ABSTRACT

The Python GUI Integrated Attendance System Using Face Recognition is a comprehensive solution designed to simplify and automate the process of taking attendance. This project leverages the power of face recognition technology through OpenCV to accurately identify individuals, coupled with a user-friendly Graphical User Interface (GUI) created using tkinter.

Key features of this system include its ease of use, with an intuitive interface that allows for quick and efficient attendance tracking. The system also includes password protection for registration, ensuring data security. Additionally, the system maintains detailed student records, creating and updating CSV files for student details and daily attendance.

The system enhances user experience by providing live attendance updates on the main screen, presenting information such as ID, name, date, and time in a clear and organized manner. Overall, this project offers a practical and effective solution for managing attendance in various educational or organizational settings.

LIST OF CONTENTS

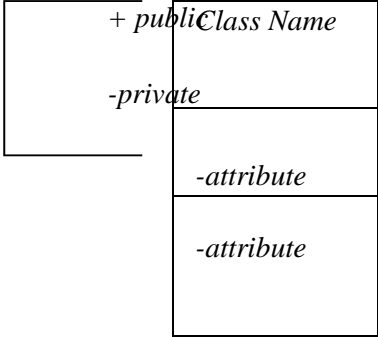
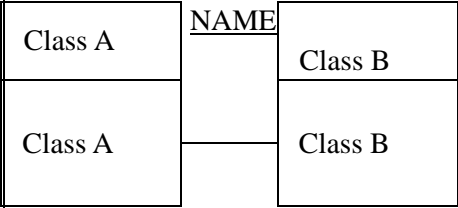
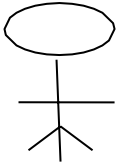
CHAPTER NO.	TITLE	PAGE NO
1.	CHAPTER 1: INTRODUCTION 1.1 General 1.2 Literature Survey	 1 2-4
2.	CHAPTER 2: SCOPE OF THE PROJECT 2.1 General 2.2 Problem Statement 2.3 Existing System and Disadvantages 2.4 Proposed System and Advantages 2.5 System Architecture	 5 5 6 7 8-9
3.	CHAPTER 3: PROJECT DESCRIPTION 3.1 General 3.2 Module Name 3.3 Module Explanation and Diagram 3.4 Techniques or Algorithm	 10 10 11-12 13-14
4.	CHAPTER 4: REQUIREMENT 4.1 General 4.2 Hardware Requirement 4.3 Software Requirement 4.4 Functional Requirement 4.5 Non-Functional Requirement	 15 15 16 17 18-19

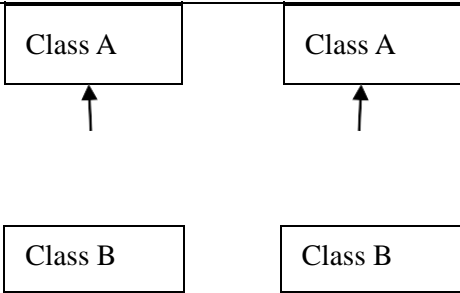
5.	CHAPTER 5: SOFTWARE SPECIFICATION 5.1 Use Case Diagram 5.2 Class Diagram 5.3 Sequence Diagram 5.4 Activity Diagram 5.5 Deployment Diagram	 20 21 22-23 24 25-26
6.	CHAPTER 6: IMPLEMENTATION 6.1 General 6.2 Implementation 6.3 Courses Code	 27 27-37 38
7.	CHAPTER 7: RESULTS & DISCUSSION 7.1 General 7.2 Results & Discussion	 39 39-42
8.	CHAPTER 8 :CONCLUSION 8.1 Conclusion 8.2 Future Enhancement	 43 44
9.	References	45

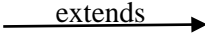


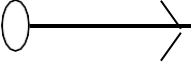
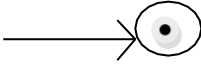
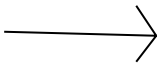
LIST OF FIGURES

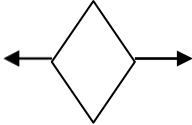
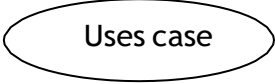
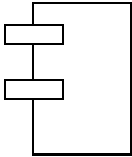
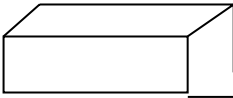
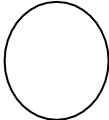


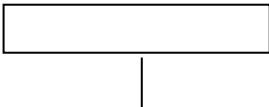
FIGURE NO	NAME OF THE FIGURE	Page no
2.5	System Architecture	8-9
3.3	Modules-Connectivity Diagram	11
5.1	Use Case Diagram	20
5.2	Class Diagram	21
5.3	Sequence Diagram	22
5.4	Activity Diagram	24
5.5	Deployment diagram	25
6.2.1	Coding	27-38
7.2.1	Results	45

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.

4.	Aggregation		Interaction between the system and external environment
----	-------------	-----------------------------------------------------------------------------------	---------------------------------------------------------

5.	Relation (uses)	uses	Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the processes.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.

12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system and external environment.
14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.

20.	Message	<div> <div>Message</div> <div>  </div> </div>	Represents the message exchanged.
-----	---------	--------------------------------------------------------------------------------------------------------------------------------	-----------------------------------

LIST OF ABBREVIATION

S.no	Abbreviation	Expansion
1.	GUI	Graphical User Interface
2.	NFC	Near Field Communication
3.	CSV	Comma-Separated Values
4.	TK	tkinter (Python library for GUI)
5.	CV2	OpenCV (Open Source Computer Vision Library)
6.	SDK	Software Development Kit
7.	HTTP	Hypertext Transfer Protocol
8.	IoT	Internet of Things

Chapter -1

Introduction

1.1 GENERAL

The GNIT Smart Attendance System represents a pivotal evolution in attendance management, addressing critical issues that plague traditional systems. One of the foremost challenges is the accuracy of attendance data, often compromised by manual recording or proxy attendance. In the past, students could easily circumvent the system by having others sign on their behalf, leading to inaccuracies and data manipulation. This lack of enforcement undermines the reliability of attendance records, rendering them unsuitable for meaningful analysis.

Moreover, the previous system's inefficiency is evident in its time-consuming nature. With manual sign-ins, only a limited number of students can be recorded in an hour, creating bottlenecks and administrative burdens. The cumbersome process not only hampers productivity but also raises concerns about data integrity and reliability.

Furthermore, accessibility to attendance information is a pressing issue for stakeholders, particularly parents.

In the absence of a streamlined system, parents lack the means to track their child's attendance and verify their whereabouts, raising concerns about accountability and transparency.

To address these challenges, the GNIT Smart Attendance System leverages cutting-edge facial recognition technology, ensuring the accuracy and integrity of attendance data. By automating the process, the system eliminates the possibility of proxy attendance and ensures that only the rightful individuals can record their attendance, enhancing data reliability.

Additionally, the system's efficiency is unparalleled, significantly reducing the time required to record attendance. With facial recognition, the process becomes seamless and efficient, enabling swift and accurate data collection for analysis and reporting.

Moreover, the system enhances accessibility by providing stakeholders, including parents, with secure access to attendance information. Through a user-friendly interface, parents can track their child's attendance, fostering transparency and accountability.

In conclusion, the GNIT Smart Attendance System represents a paradigm shift in attendance management, offering unparalleled accuracy, efficiency, and accessibility. By addressing the shortcomings of traditional systems, the system ensures that attendance data is reliable, efficient, and accessible to all legitimate parties, ushering in a new era of attendance management.

1.2 Literature Survey

The benefits and disadvantages identified in the research publications are addressed below.

Title: Attendance System Using NFC Technology with Embedded Camera on Mobile Device

Author: Bhise, Khichi, Korde, Lokare (2015)

Description: This research explores an attendance system that integrates Near Field Communication (NFC) technology and a mobile application. Each student is provided with a unique NFC tag during enrollment, which they use to mark their attendance by touching or moving the tag near the lecturer's mobile phone. The embedded camera on the phone captures the student's face for validation and verification.

Advantages:

- Simple to use.

Disadvantages:

- Vulnerable to proxy attendance if NFC tags are not personally used.
 - Inconvenient for lecturers who forget their mobile phones or are concerned about privacy.
 - Suggests replacing NFC tags with biometrics or face recognition for more reliable attendance tracking.
-

Title: Face Recognition Based Attendance Marking System.

Author: SenthamilSelvi, Chitrakala, Antony Jenitha (2014)

Description: This system utilizes face recognition technology for attendance marking. It captures images of employees using a camera, performs face detection and recognition, and compares the images with a face database to mark attendance securely on the server.

Advantages:

- Highly secure attendance marking.
- Improved face detection accuracy using skin classification techniques.

Disadvantages:

- Not portable, as it requires a standalone computer and constant power supply.
 - Suitable only for staff attendance due to lack of portability for daily student attendance marking.
-

Title: Fingerprint Based Attendance System Using Microcontroller and LabView.

Author: Kumar Yadav, Singh, Pujari, Mishra (2015)

Description: This system employs fingerprint recognition for attendance marking. It uses two microcontrollers, with one handling

the fingerprint sensor and the other managing the database and communication with the PC.

Advantages:

- Accelerates development and simplifies testing.
- Maintains design flexibility.

Disadvantages:

- Not portable, as it is attached to a PC.
- Database information not easily accessible; requires uploading to a web server for convenient access by parents.

Title: RFID based Student Attendance System.

Author: Hussain, Dugar, Deka, Hannan (2014)

Description: This system utilizes RFID technology for student attendance tracking. A tag and a reader are used, and attendance information can be accessed through a web portal.

Advantages:

- Convenient information retrieval through a web portal.

Disadvantages:

- Not portable; RFID reader requires connection to a PC.
-

chapter-2

Scope of the Project

2.1 Scope of the Project

The primary goal of this project is to address the shortcomings of the traditional attendance system by developing a new, innovative smart system that enhances convenience for educational institutions. The project involves the development of an application capable of identifying individuals and recording their attendance in Excel sheets, eliminating the need for manual attendance marking. This automated system aims to improve accuracy and efficiency in attendance management, providing real-time data for analysis and monitoring.

2.2 Problem Statement and Motivation

The current attendance management system faces several challenges that hinder its efficiency and reliability. The foremost issue is the accuracy of the data collected, as it can be compromised by proxy attendance. This occurs when a student's attendance is marked by someone else, leading to inaccurate records. This practice undermines the credibility of the attendance data for analysis and monitoring purposes. Additionally, the manual process of recording attendance is time-consuming and inefficient. With each student taking approximately a minute to sign their attendance, the process becomes laborious, allowing only a limited number of students to be recorded in an hour. This inefficiency not only wastes time but also hampers productivity.

These issues highlight the need for an evolution in the attendance management system to improve efficiency, data accuracy, and accessibility for all stakeholders involved.

2.3 Existing system

In the context of existing biometric attendance systems in colleges, several critical issues necessitate the development of a more reliable

and efficient solution like the GNIT Smart Attendance system. The key disadvantages of the current biometric systems are outlined as follows:

1. **Accuracy and Reliability Issues:**

- Attendance might not be recorded personally by the original person, allowing for proxy attendance and compromising data integrity.

2. **Time-Consuming Process:**

- Biometric systems can be slow, with each student's attendance taking significant time, leading to delays and inefficiencies.

3. **Dependency on Hardware:**

- Biometric systems rely on specific hardware, which can malfunction or require maintenance, disrupting the attendance process.

4. **Lack of Accessibility:**

- Current systems do not provide easy access to attendance information for parents, leading to a lack of transparency and accountability.

5. **Manual Handling and Data Entry Errors:**

- Manual intervention is often needed, increasing the workload and potential for data entry errors.

2.4 Objectives

To address the challenges of the manual attendance management system and improve efficiency and accuracy, the GNIT Smart Attendance project aims to achieve the following objectives:

1. Automated Attendance Recording: Develop a system that automates the process of recording attendance, eliminating the need for manual entry and reducing the possibility of errors.

2.Excel Sheet Integration: Implement functionality to store attendance data directly into Excel sheets ,ensuring compatibility with the existing system infrastructure.

3. Accuracy Enhancement: Utilize facial recognition technology to accurately identify students and record their attendance, reducing the risk of proxy attendance and ensuring data integrity.

4. Time Efficiency :Create a system that significantly reduces the time required to record attendance, improving efficiency and productivity for administrators and faculty.

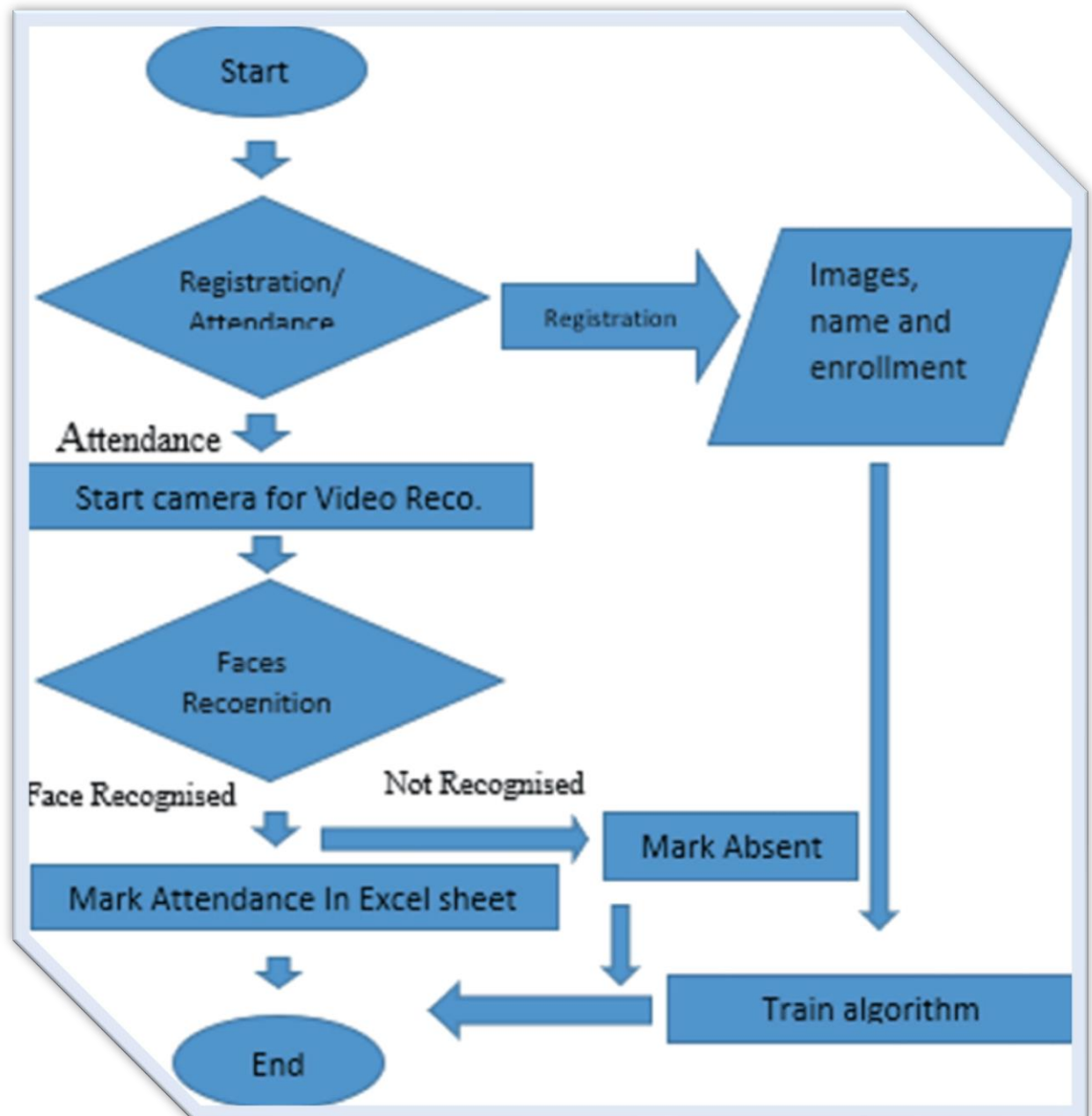
5.Parental Access : Develop features that allow parents to access their child's attendance records directly from the Excel sheets, fostering transparency and accountability.

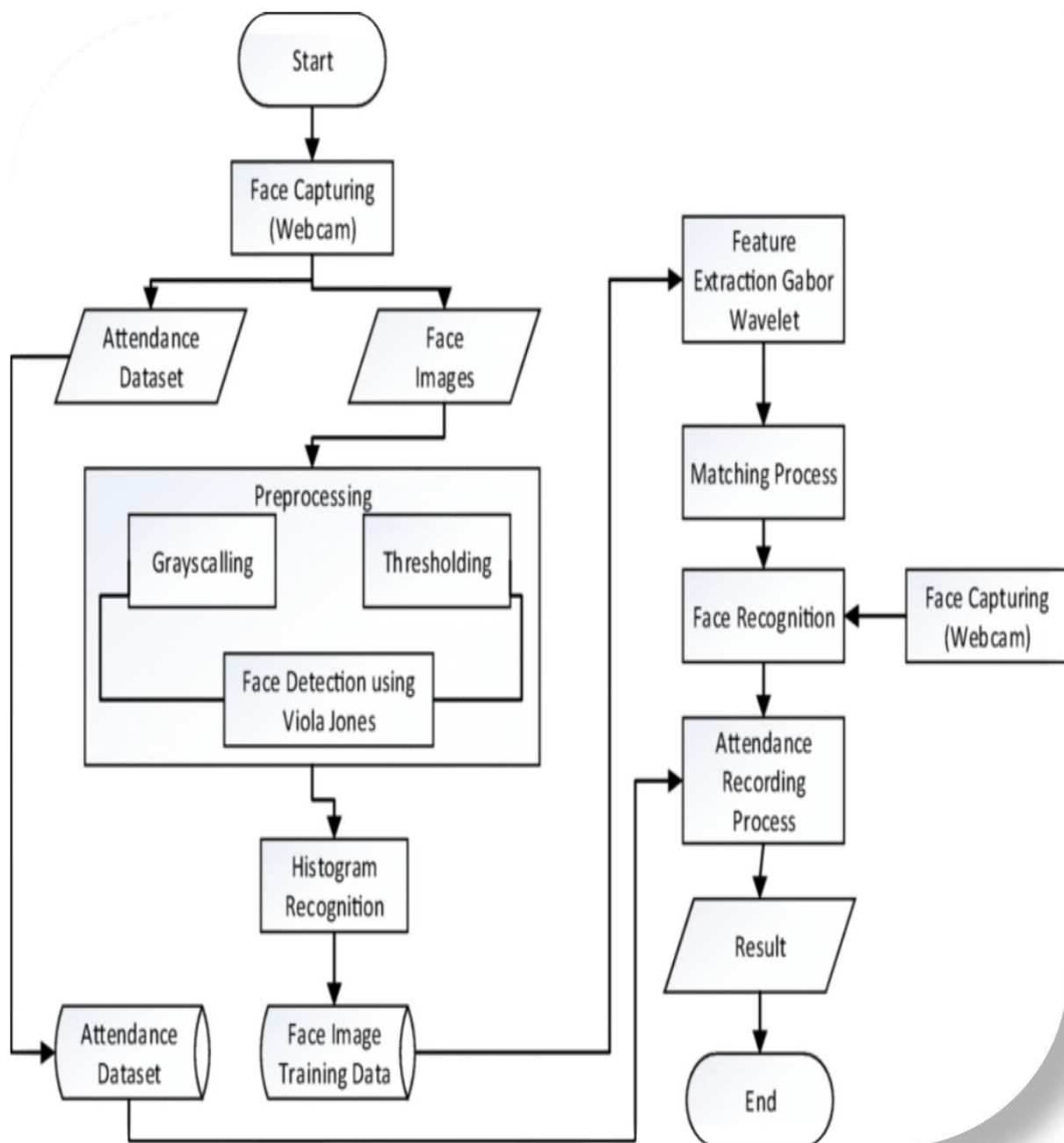
6. User-Friendly Interface: Design an intuitive and easy-to-use interface for administrators to manage attendance records and for parents to view attendance data, enhancing user experience and adoption.

7. Feedback Mechanism: Implement visual indicators to provide immediate feedback to users on the success of the facial recognition process, ensuring user awareness and confidence in the system.

8. Scalability and Flexibility: Build a system that can accommodate future growth and changes in attendance management requirements, ensuring long-term sustainability and adaptability.

2.5 System Architecture





The flow chart presents a process for taking attendance using face recognition technology. The process begins with “Start” and proceeds with an “Input Image” step. Following this, “Face Detection” is performed to locate faces within the image. Once faces are detected, “Face Recognition” is conducted, and if a face is recognized as known, it is saved to the “Face Database.” The system then checks “IF Face Is In Database,” and if the face is known (“Yes”), attendance is taken and saved in the “Attendance Database.”

Chapter -3

Project Description

3.1 Project description

The GNIT Smart Attendance System is an innovative solution designed to streamline attendance tracking using face recognition technology. The system captures images and accurately recognizes faces with OpenCV's `cv2.face.LBPHFaceRecognizer_create()` method. An intuitive GUI built with Tkinter makes the system user-friendly and accessible to everyone, regardless of technical skills. The technology stack includes Tkinter for the GUI, OpenCV for image capture and face recognition, and CSV, Numpy, Pandas, and datetime for data manipulation and management.

Key features of the system include an interactive GUI that supports easy navigation and clear instructions. Security is ensured through password-protected new person registration, allowing only authorized users to add entries. The system creates or updates a CSV file for student details upon registration and generates a new CSV file each day for attendance, accurately marking entries with date and time. The main screen provides live attendance updates in a tabular format, displaying ID, name, date, and time for real-time monitoring and verification.

3.2 MODULES NAME

This project having the following 6 modules:

- **Modules-Connectivity Diagram**
- **Interface**
- **Data Collection**
- **Face Recognition and Attendance Marking**
- **Attendance Data Management**
- **Live Attendance Updates**

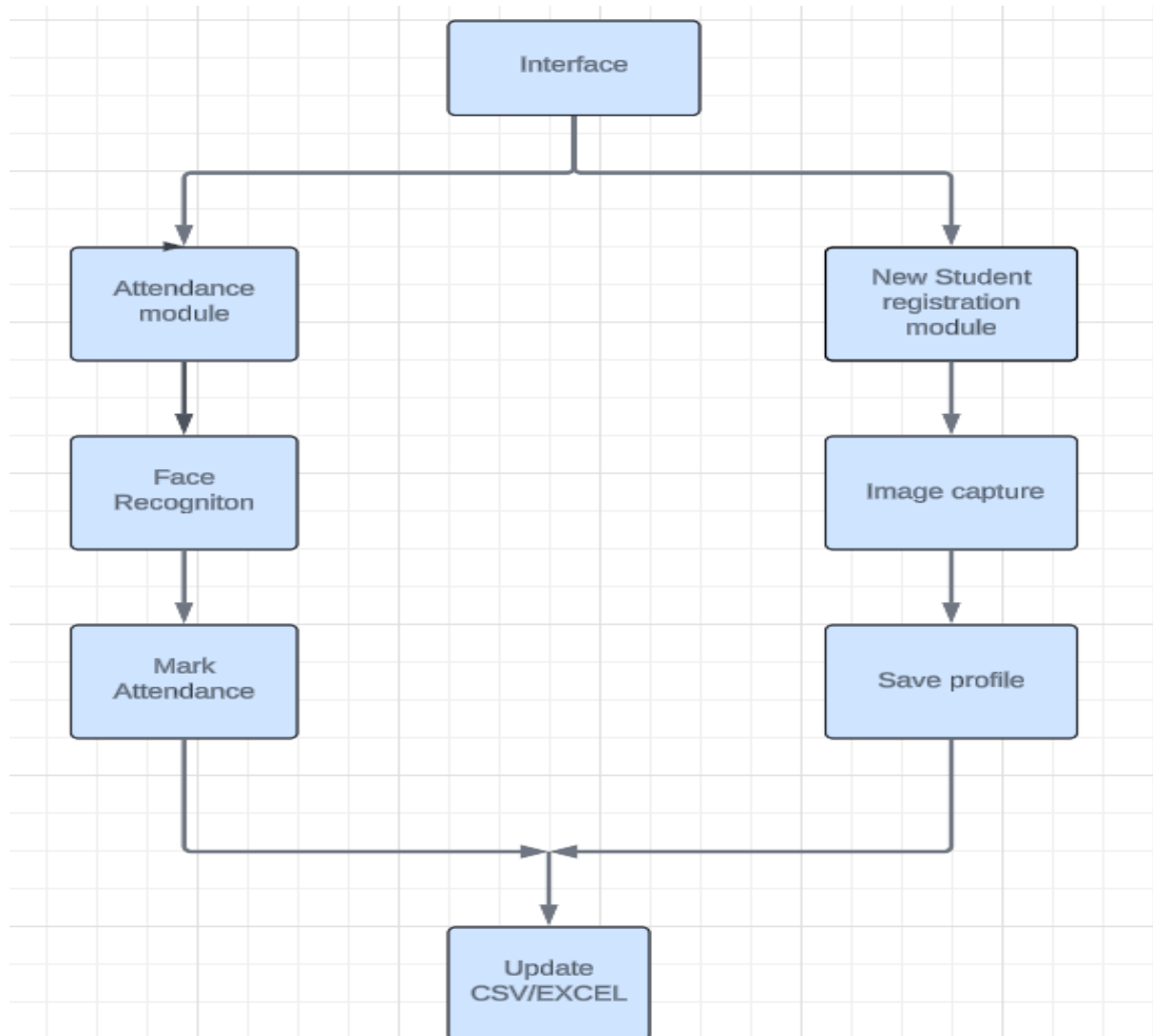
3.3 Modules Explanation

● Modules-Connectivity Diagram

The module diagram is an expanded data flow diagram used to define a system's business logic. It is composed of the following key elements:

Components: Components, such as computation or message displays, are functional components in the system. These elements have input and output ports, which allow data to be transmitted between them and are represented as rectangles.

Data Flow: Flow lines indicate the movement of data within a component. These lines define the order in which elements are implemented, as well as the amount of data that flows through the system. Depending on the nature of data, they can be represented as lines or color codes.



Control Flow Constructs: Control flow constructs, such as those used with Schneiderman boxes, allow you to design module logic in

accordance with Structured Programming concepts, making them a more pleasant addition to your module diagram. These constructs include timed execution, looping, dependent branching, and recursive module calls.

Module Events: The module diagram will cover one or more modules, and each of these events is displayed separately within it. By clicking on an event or selecting a group of events, the user can view the specifics of each module event. It is a module event, as you can tell by looking underneath the module name. Module Events can be triggered by a variety of events, such as the beginning of a related module, interacting with your user interface, selecting an override button, or using internal triggers within this module.

Interface: The graphical user interface (GUI) built using Tkinter. It features easy-to-use interfaces for user registration, attendance marking, and displaying live attendance updates. The interface includes password protection for new user registration and various buttons for interacting with the system.

Data Collection: This module handles the processes for capturing images and gathering student details during registration. It stores the captured images for training the face recognition model. The data collection process is designed to be seamless and efficient, ensuring accurate and reliable data for the face recognition system.

Face Recognition and Attendance Marking: Utilizes OpenCV to recognize faces from the live camera feed and mark attendance by matching the faces against the registered profiles. This module incorporates dependent branching to handle different scenarios, such as recognizing a registered student or identifying an unregistered individual. The attendance marking process is automatic and real-time, providing immediate feedback to the user.

Attendance Data Management: Manages creation and updating of CSV files for student details and daily attendance, with timed execution for daily file generation.

Live Attendance Updates: Displays real-time attendance data on the main screen, triggered by internal module events and user interactions.

3.4 Technique Used: LBPH Algorithm for Face Recognition

The Local Binary Patterns Histogram (LBPH) algorithm is a robust method for face recognition, known for its simplicity and effectiveness under various lighting conditions and facial expressions. LBPH captures local texture information, making it resilient to changes in illumination and facial features.

3.4.1 Local Binary Patterns (LBP)

LBPH uses the Local Binary Pattern (LBP) operator, which labels image pixels by thresholding the neighborhood of each pixel and creating binary numbers:

1. Grayscale Conversion: Convert the image to grayscale.
2. Neighborhood Thresholding: Compare each pixel with its neighbors; assign 1 if the neighbor's value is greater than or equal to the center pixel, otherwise assign 0.
3. Binary Pattern Formation: Convert the binary numbers to decimal values representing the image texture.

3.4.2 Histogram Calculation

After computing LBP values, the next step is creating a histogram:

1. Dividing the Image into Grids: Split the image into grids (e.g., 8x8).
2. Histogram of Each Grid: Calculate LBP histograms for each grid.
3. Concatenating Histograms: Combine histograms to form a single feature vector representing the image.

3.4.3 Face Recognition Using LBPH

The concatenated histograms form the feature vector used for face recognition:

1. Training Phase: Compute LBPH feature vectors for training images, storing them with corresponding labels.
2. Recognition Phase: Compute the LBPH feature vector for the input image and compare it with stored vectors using a distance metric (e.g., Euclidean distance).
3. Prediction: Assign the label of the closest matching feature vector to the input image, identifying the person.

Chapter 4

Requirements

These are the requirements for doing the project. Without using these tools & software's we cannot do the project. Therefore, we have two requirements to do the project. They are

- Hardware Requirements.
- Software Requirements.

Hardware Requirements:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. Software engineers use them as the starting point for the system design. It should what the system and not how it should be implemented.

1. Camera: A high-resolution camera for capturing clear images for face recognition.

2. Computer System:

- Processor: Intel i5 or higher
- RAM: 8GB or more
- Storage: 256GB SSD or higher

3. Power Supply: Reliable power source to ensure continuous operation.

4. Internet Connection Stable internet connection for potential remote access and updates.

Software Requirements:

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks, tracking the teams, and tracking the team's progress throughout the development activity.

1. Operating System: Windows 10 or higher, or any compatible Linux distribution.

2. Python: Version 3.6 or higher.

3. Libraries and Frameworks:

- Tkinter
- OpenCV
- Numpy
- Pandas
- Datetime

4. Database: CSV files for data storage (or any other suitable database system if needed).

5. Development Environment: Visual Studio Code or any other suitable IDE.

Functional Requirements:

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, and outputs. The outsourced computation is data is more secure.

1. Programming Languages and Libraries:

- **Programming Language:** Python 3.6 or higher.
- **Libraries:**
 - Tkinter: For creating the graphical user interface.
 - OpenCV: For capturing images and face recognition(using`cv2.face.LBPHFaceRecognizer_create()`).
 - Numpy: For numerical operations.
 - Pandas: For data manipulation and storage.
 - Datetime: For handling date and time operations.

2.Data Preprocessing Tools:

- OpenCV: For image processing.
- Numpy: For handling numerical data.
- Pandas: For data cleaning and manipulation.

3.Integrated Development Environment (IDE):

- Visual Studio Code: For code development and debugging.

4. User Registration:

- The system should allow new users to register by capturing their face and personal details.
- Registration should be password protected to ensure security.

5. Attendance Marking:

- The system should recognize faces and mark attendance accurately.
- Attendance should be logged with the correct date and time.

6. CSV File Management:

- The system should create and update CSV files for student details during registration.
- A new CSV file should be generated daily for recording attendance.

7. Live Attendance Display:

- The main screen should display live attendance updates in a tabular format, including ID, name, date, and time.

8.GUI Interaction:

- The GUI should be intuitive and easy to navigate for users.

Non-Functional Requirements:

1. Performance:

- The system should perform face recognition and attendance marking in real-time.
- The GUI should respond promptly to user actions.

2. Reliability:

- The system should accurately recognize faces and avoid false positives/negatives.
- Attendance data should be correctly logged without errors.

3. Usability:

- The GUI should be user-friendly and require minimal training for users.

4. Security:

- User registration should be protected by a password.

- Attendance data should be stored securely to prevent unauthorized access.

5. Scalability:

- The system should handle an increasing number of users and attendance records without performance degradation.

6. Maintainability:

- The code should be well-documented and modular to facilitate easy maintenance and updates.

7. Portability:

- The system should be compatible with different operating systems, including Windows and Linux.

CHAPTER 5

SOFTWARE SPECIFICATION

5.1 Use Case Diagram

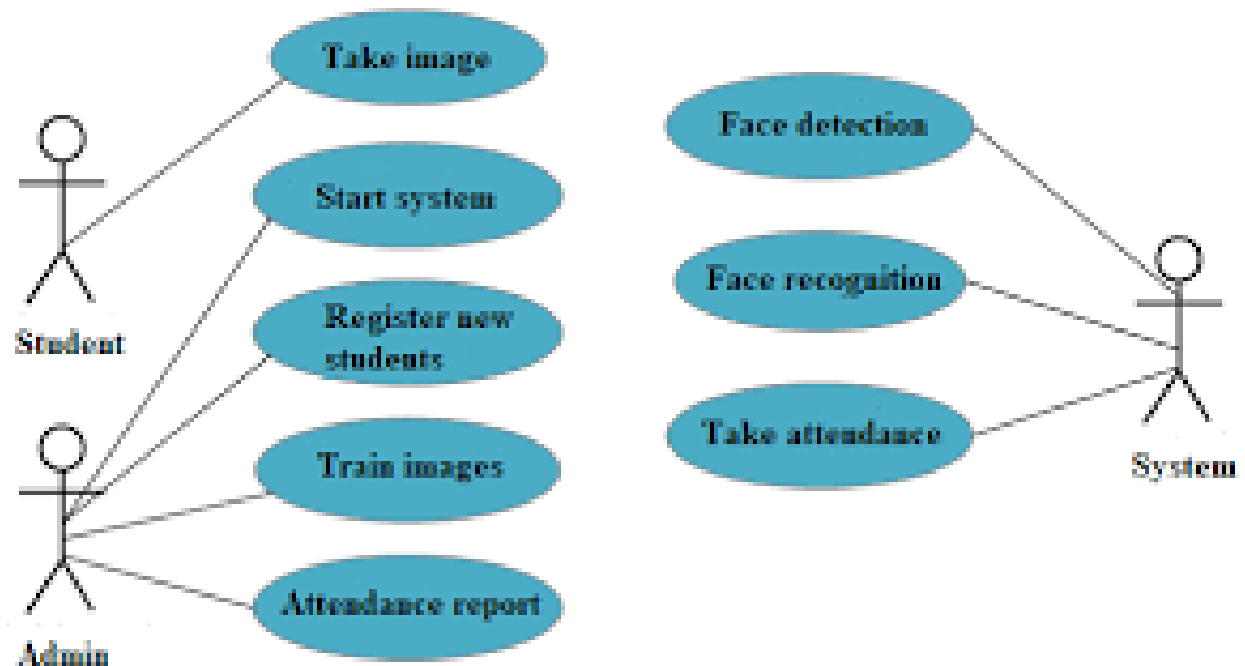
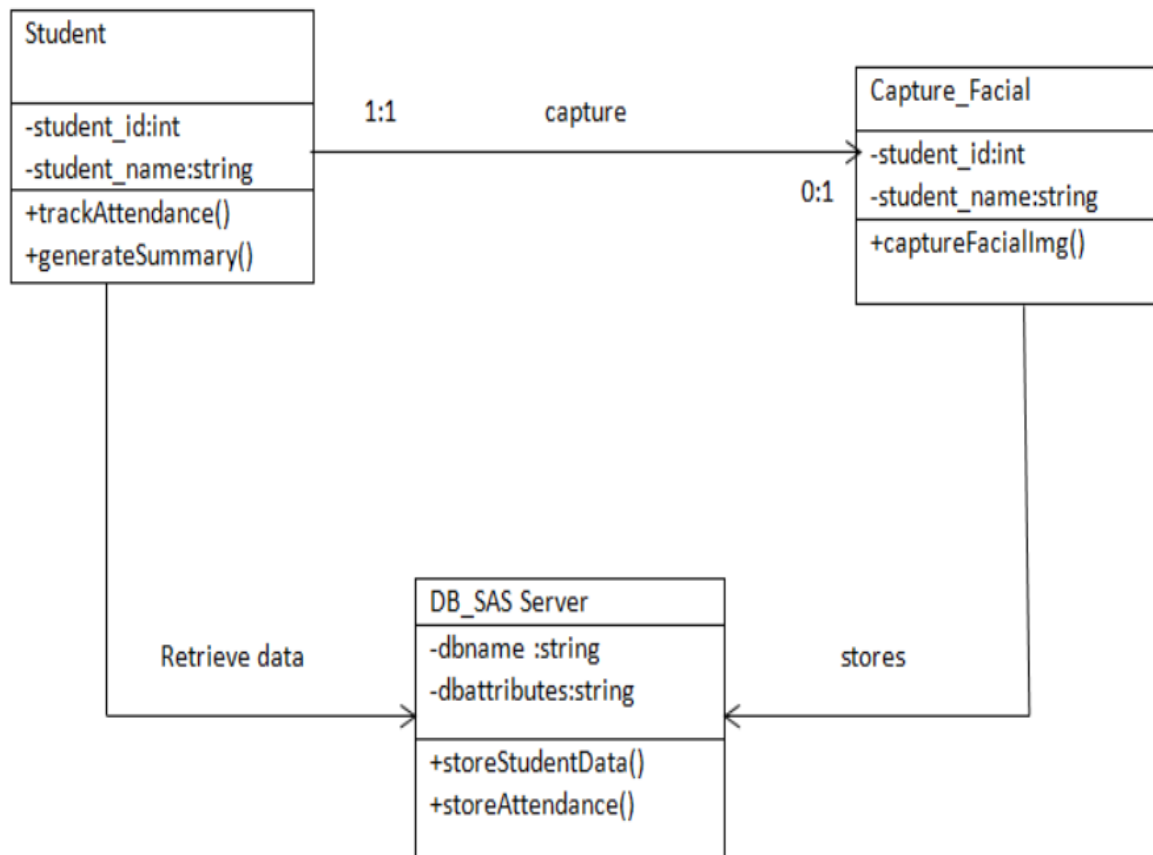


Figure 5.1 Use Case diagram

Explains the connection between the use cases and the relationships between use cases and the actors. A user case diagram provides an easy representation of the system and its users, which visually displays interactions between different elements. It provides an overview of the events that occur in the system and its flow but does not give detailed information on how they are implemented

5.2 Class Diagram



1. Student Class

- Represents individual students.
- Manages student identification and name.
- Tracks and generates summaries of student attendance.

2. Capture Facial Class

- Handles facial recognition for students.
- Captures and processes facial images to mark attendance.

3. DB SAS Server Class

- Acts as the central database server.
- Stores student data and attendance records.

5.3 Sequence Diagram

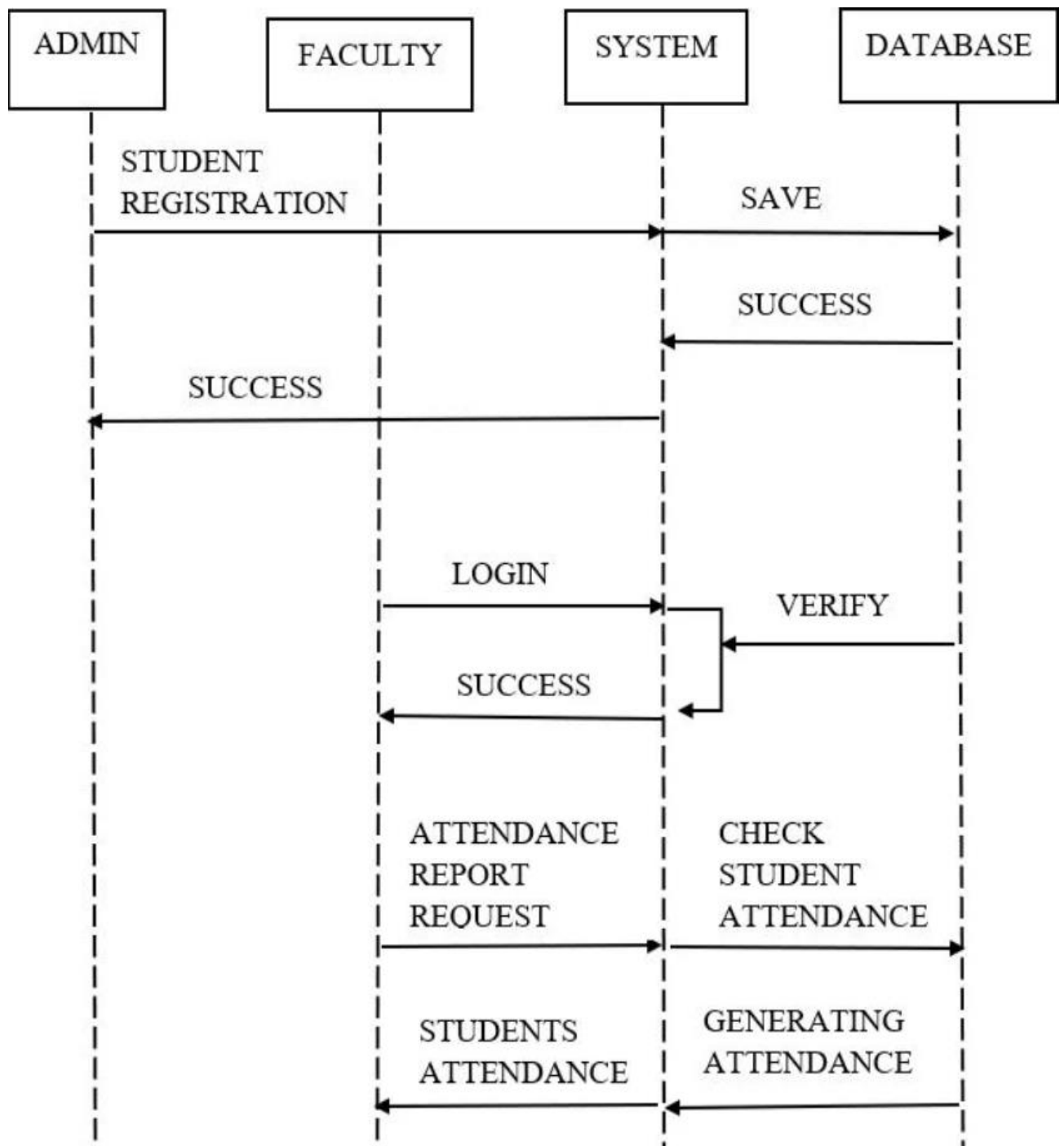


Figure 5.3 Sequence diagram

Sequence Diagrams are a type of interaction diagram that provides a detailed representation of how operations are executed within a system. They capture the dynamic interaction between objects within a collaboration, showcasing the chronological order of messages exchanged between them

. Sequence Diagrams serve to capture:

1. Interaction within a collaboration: They illustrate the interaction that occurs within a collaboration, which can be associated with the realization of a use case or an operation. This includes both instance diagrams, which represent specific object instances, and generic diagrams, which showcase high-level interactions between active objects.

2. High-level interactions: Sequence Diagrams can depict interactions between users and the system, interactions between the system and other external systems, or interactions between subsystems. These interactions provide a broad overview of the system's functionality and communication patterns.

5.4 Activity Diagram

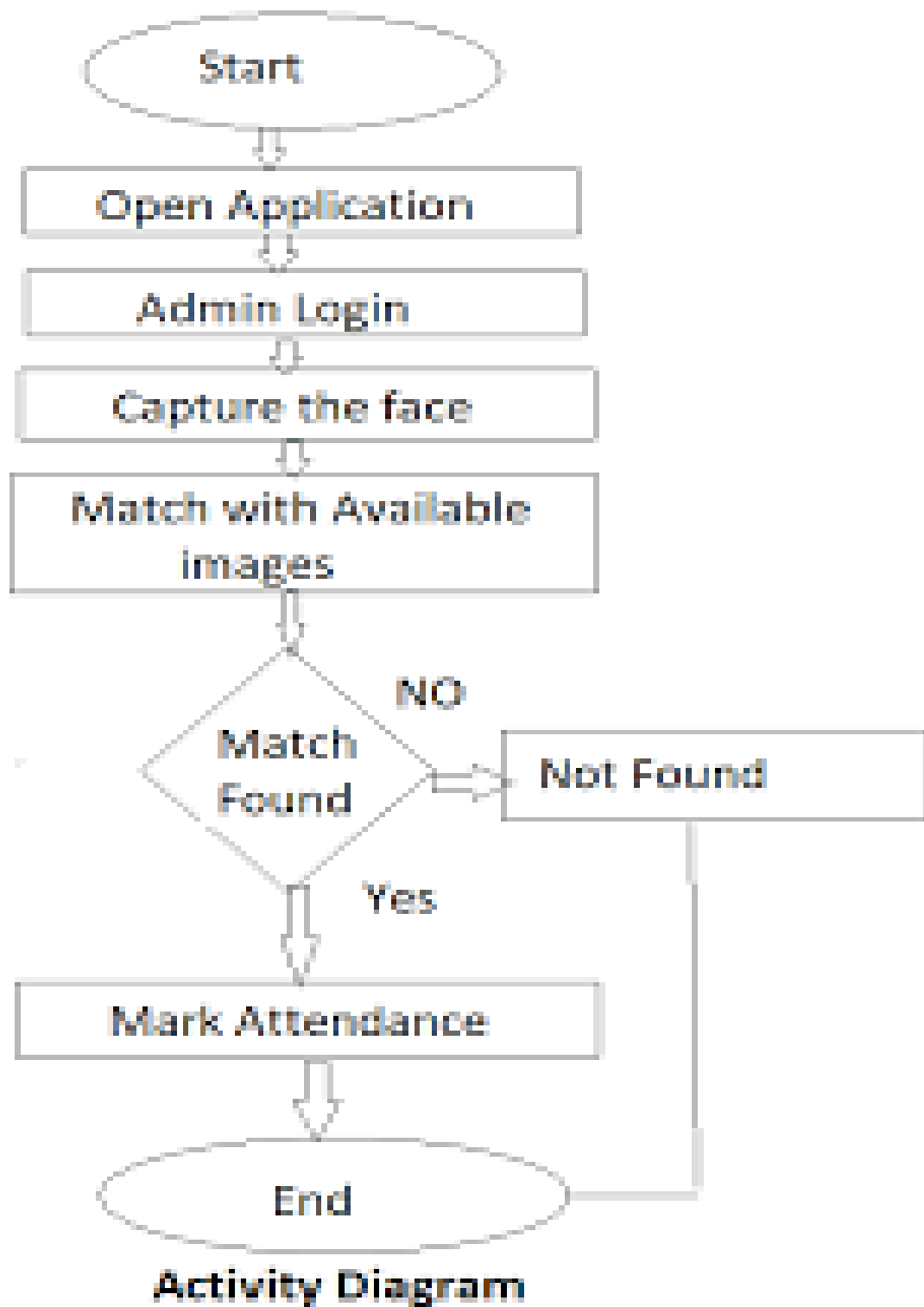


Figure 5.4 Activity diagram

Explains the execution flow of the program. An activity diagram not only facilitating the visualization of a system's dynamic nature, but also makes it possible to build functional systems by means of future and reverse engineering techniques. Nevertheless, there is a lack of illustration of the flow of messages across activities in activity diagrams. Although they are similar to flowcharts, the activity diagram has a number of additional features and provides various flow types like parallel, branch, continuous or individual flows. It is possible to define the objectives of the activity diagram as follows: Depict the flow of activities within the system.

5.5 Deployment Diagram

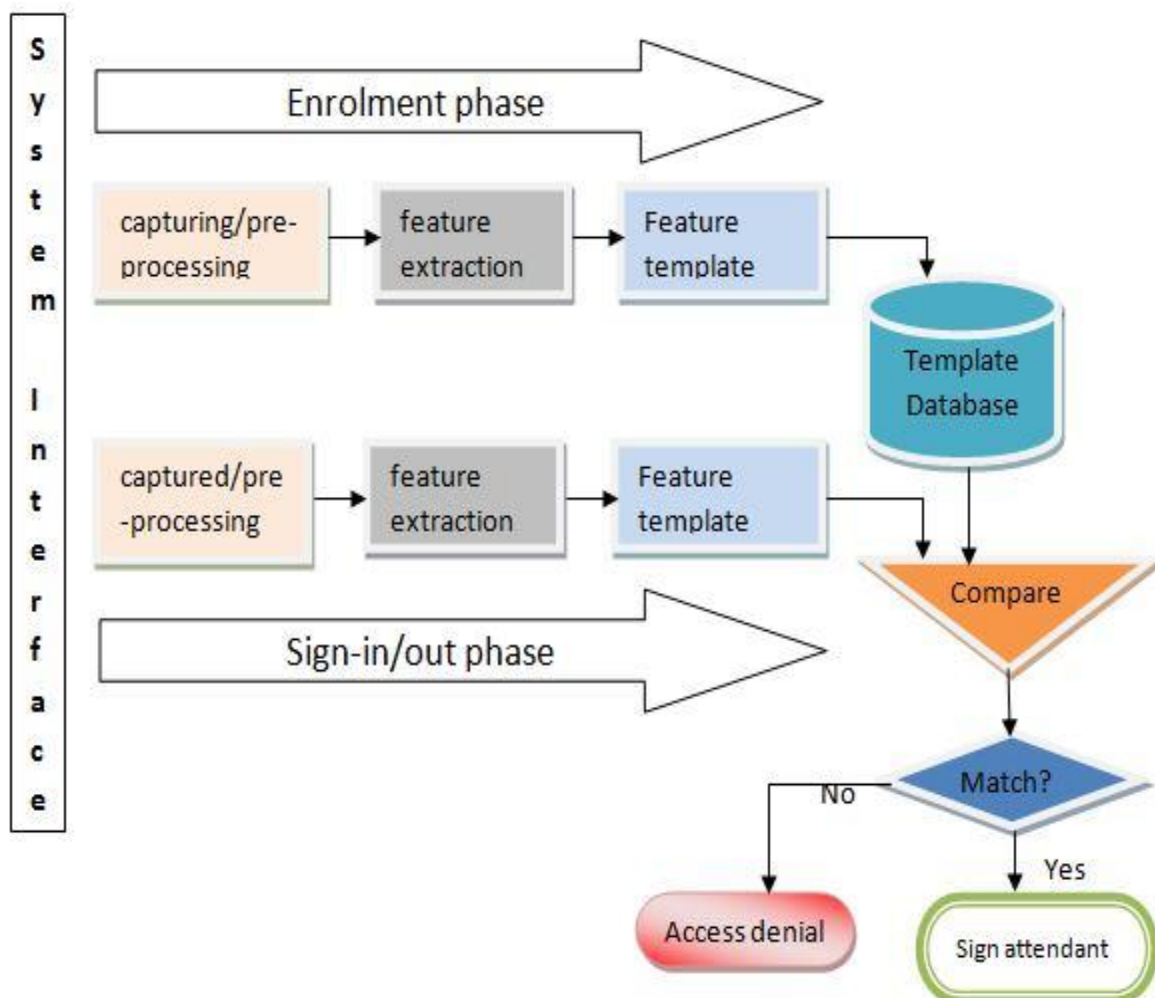


Fig.5.4 Deployment Diagram

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it. A Deployment Diagram is a visual representation that defines the physical hardware infrastructure on which a software system is intended to run. It goes beyond software components, outlining the configuration and arrangement of hardware elements. This diagram illustrates the allocation of software modules to the specific devices or nodes within the system, providing a clear mapping of how the software is deployed on the underlying hardware. In essence, it serves as a crucial tool for understanding the distribution and execution environment of a software system.

CHAPTER 6

IMPLEMENTATION

6.1 GENERAL

The Implementation is nothing but source code of project

6.2 IMPLEMENTATION

CODING:

```
##### IMPORTING
#####
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox as mess
import tkinter.simpledialog as tsd
import cv2,os
import csv
import numpy as np
from PIL import Image
import pandas as pd
import datetime
import time

##### FUNCTIONS
#####

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

#####
####

def tick():
    time_string = time.strftime('%H:%M:%S')
    clock.config(text=time_string)
    clock.after(200,tick)

#####
#####

def contact():
    mess._show(title='Contact us', message="Please contact us on :
'vr657658@gmail.com' ")
```

```
#####
#####

def check_haarcascade_file():
    exists = os.path.isfile("haarcascade_frontalface_default.xml")
    if exists:
        pass
    else:
        mess._show(title='Some file missing', message='Please contact us for
help')
        window.destroy()

#####
#####

def save_pass():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile(r"TrainingImageLabel\psd.txt")
    if exists1:
        tf = open(r"TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not
set!! Please try again')
        else:
            tf = open(r"TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
    op = (old.get())
    newp = (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open(r"TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old
password.')
```

```

        return
    mess._show(title='Password Changed', message='Password changed
successfully!!')
    master.destroy()

#####
#####

def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
    master.configure(background="white")
    lbl4 = tk.Label(master,text='    Enter Old
Password',bg='white',font=('comic', 12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('comic', 12,
' bold '),show='*')
    old.place(x=180,y=10)
    lbl5 = tk.Label(master, text='    Enter New Password', bg='white',
font=('comic', 12, ' bold '))
    lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('comic',
12, ' bold '),show='*')
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white',
font=('comic', 12, ' bold '))
    lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black",
relief='solid',font=('comic', 12, ' bold '),show='*')
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy
,fg="black" ,bg="red" ,height=1,width=25 , activebackground = "white"
,font=('comic', 10, ' bold '))
    cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black",
bg="#00fcca", height = 1,width=25, activebackground="white", font=('comic',
10, ' bold '))
    save1.place(x=10, y=120)
    master.mainloop()

#####
#####

```

```

def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile(r"TrainingImageLabel\psd.txt")
    if exists1:
        tf = open(r"TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not
set!! Please try again')
        else:
            tf = open(r"TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
        password = tsd.askstring('Password', 'Enter Password', show='*')
        if (password == key):
            TrainImages()
        elif (password == None):
            pass
        else:
            mess._show(title='Wrong Password', message='You have entered wrong
password')

#####

def clear():
    txt.delete(0, 'end')
    res = "1)Take Images >>> 2)Save Profile"
    message1.configure(text=res)

def clear2():
    txt2.delete(0, 'end')
    res = "1)Take Images >>> 2)Save Profile"
    message1.configure(text=res)

#####

def TakeImages():
    check_haarcascade()
    columns = ['SERIAL NO.', '', 'ID', '', 'NAME']
    assure_path_exists(r"StudentDetails/")

```

```

assure_path_exists(r"TrainingImage/")
serial = 0
exists = os.path.isfile(r"StudentDetails\StudentDetails.csv")
if exists:
    with open(r"StudentDetails\StudentDetails.csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            serial = serial + 1
    serial = (serial // 2)
    csvFile1.close()
else:
    with open(r"StudentDetails\StudentDetails.csv", 'a+') as csvFile1:
        writer = csv.writer(csvFile1)
        writer.writerow(columns)
        serial = 1
    csvFile1.close()
Id = (txt.get())
name = (txt2.get())
if ((name.isalpha()) or (' ' in name)):
    cam = cv2.VideoCapture(0)
    harcascadePath = r"haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    sampleNum = 0
    while (True):
        ret, img = cam.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = detector.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            # incrementing sample number
            sampleNum = sampleNum + 1
            # saving the captured face in the dataset folder TrainingImage
            cv2.imwrite(r"TrainingImage\ " + name + "." + str(serial) +
                "." + Id + "." + str(sampleNum) + ".jpg",
                gray[y:y + h, x:x + w])
            # display the frame
            cv2.imshow('Taking Images', img)
            # wait for 100 milliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum > 100:
                break
    cam.release()
    cv2.destroyAllWindows()
    res = "Images Taken for ID : " + Id
    row = [serial, '', Id, '', name]
    with open(r'StudentDetails\StudentDetails.csv', 'a+') as csvFile:

```



```

        writer = csv.writer(csvFile)
        writer.writerow(row)
    csvFile.close()
    message1.configure(text=res)
else:
    if (name.isalpha() == False):
        res = "Enter Correct name"
        message.configure(text=res)

#####
#####

def TrainImages():
    check_haarcascadeFile()
    assure_path_exists("TrainingImageLabel/")
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, ID = getImagesAndLabels("TrainingImage")
    try:
        recognizer.train(faces, np.array(ID))
    except:
        mess._show(title='No Registrations', message='Please Register someone first!!!')
        return
    recognizer.save(r"TrainingImageLabel\Trainer.yml")
    res = "Profile Saved Successfully"
    message1.configure(text=res)
    message.configure(text='Total Registrations till now : ' + str(ID[0]))

#####
#####3

def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePath = [os.path.join(path, f) for f in os.listdir(path)]
    # create empty face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the
    images
    for imagePath in imagePath:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image

```

```

        ID = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(ID)
    return faces, Ids

#####
#####

def TrackImages():
    check_haarcascadefile()
    assure_path_exists("Attendance/")
    assure_path_exists("StudentDetails/")
    for k in tv.get_children():
        tv.delete(k)
    msg = ''
    i = 0
    j = 0
    recognizer = cv2.face.LBPHFaceRecognizer_create() #
cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile(r"TrainingImageLabel\Trainer.yml")
    if exists3:
        recognizer.read(r"TrainingImageLabel\Trainer.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile
to reset data!!')
        return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);

    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', '', 'Name', '', 'Date', '', 'Time']
    exists1 = os.path.isfile(r"StudentDetails\StudentDetails.csv")
    if exists1:
        df = pd.read_csv(r"StudentDetails\StudentDetails.csv")
    else:
        mess._show(title='Details Missing', message='Students details are
missing, please check!')
        cam.release()
        cv2.destroyAllWindows()
        window.destroy()
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)

```

```

        serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
        if (conf < 50):
            ts = time.time()
            date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-
%Y')

            timeStamp =
datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
            aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values
            ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
            ID = str(ID)
            ID = ID[1:-1]
            bb = str(aa)
            bb = bb[2:-2]
            attendance = [str(ID), ',', bb, ',', str(date), ',',
str(timeStamp)]

        else:
            Id = 'Unknown'
            bb = str(Id)
            cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
            cv2.imshow('Taking Attendance', im)
            if (cv2.waitKey(1) == ord('q')):
                break
            ts = time.time()
            date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
            exists = os.path.isfile(r"Attendance\Attendance_" + date + ".csv")
            if exists:
                with open(r"Attendance\Attendance_" + date + ".csv", 'a+') as
csvFile1:
                    writer = csv.writer(csvFile1)
                    writer.writerow(attendance)
                    csvFile1.close()
            else:
                with open(r"Attendance\Attendance_" + date + ".csv", 'a+') as
csvFile1:
                    writer = csv.writer(csvFile1)
                    writer.writerow(col_names)
                    writer.writerow(attendance)
                    csvFile1.close()
            with open(r"Attendance\Attendance_" + date + ".csv", 'r') as csvFile1:
                reader1 = csv.reader(csvFile1)
                for lines in reader1:
                    i = i + 1
                    if (i > 1):
                        if (i % 2 != 0):
                            iidd = str(lines[0]) + ' '
                            tv.insert(' ', 0, text=iidd, values=(str(lines[2]),
str(lines[4]), str(lines[6])))

```

```

csvFile1.close()
cam.release()
cv2.destroyAllWindows()

##### USED STUFFS
#####

global key
key = ''

ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
day,month,year=date.split("-")

mont={'01':'January',
      '02':'February',
      '03':'March',
      '04':'April',
      '05':'May',
      '06':'June',
      '07':'July',
      '08':'August',
      '09':'September',
      '10':'October',
      '11':'November',
      '12':'December'
      }

##### GUI FRONT-END
#####

window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
window.title("Attendance System")
window.configure(background='black')

frame1 = tk.Frame(window, bg="#D3D3D3")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#D3D3D3")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="GNIT Smart Attendance System"
,fg="#FFD700",bg="black" ,width=55 ,height=1,font=('comic', 29, ' bold '))
message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="red")

```

```

frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="yellow")
frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+" | ",
fg="white",bg="black" ,width=55 ,height=1,font=('comic', 22, ' bold '))
datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="white",bg="black" ,width=55
,height=1,font=('comic', 22, ' bold '))
clock.pack(fill='both',expand=1)
tick()

head2 = tk.Label(frame2, text="                For New Student
Registrations                ", fg="black",bg="yellow" ,font=('comic',
17, ' bold '))
head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text="                For Already Registered
Students                ", fg="black",bg="yellow" ,font=('comic', 17, '
bold '))
head1.place(x=0,y=0)

lbl = tk.Label(frame2, text="Enter
ID",width=20 ,height=1 ,fg="#000080" ,bg="#c79cff" ,font=('comic', 17, '
bold '))
lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))
txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter name",width=20 ,fg="Red" ,bg="#c79cff"
,font=('comic', 17, ' bold '))
lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))
txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile"
,bg="#c79cff" ,fg="black" ,width=39 ,height=1, activebackground = "#3ffc00"
,font=('comic', 15, ' bold '))
message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#c79cff"
,fg="black" ,width=39,height=1, activebackground = "#3ffc00" ,font=('comic',
16, ' bold '))
message.place(x=7, y=450)

```

```

lbl3 = tk.Label(frame1,
text="Attendance",width=20 ,fg="black" ,bg="#c79cff" ,height=1
,font=('comic', 17, ' bold '))
lbl3.place(x=100, y=115)

res=0
exists = os.path.isfile(r"StudentDetails\StudentDetails.csv")
if exists:
    with open(r"StudentDetails\StudentDetails.csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            res = res + 1
    res = (res // 2) - 1
    csvFile1.close()
else:
    res = 0
message.configure(text='Total Registrations till now : '+str(res))

##### MENUBAR #####

menubar = tk.Menu(window,relief='ridge')
filemenu = tk.Menu(menubar,tearoff=0)
filemenu.add_command(label='Change Password', command = change_pass)
filemenu.add_command(label='Contact Us', command = contact)
filemenu.add_command(label='Exit',command = window.destroy)
menubar.add_cascade(label='Help',font=('comic', 29, ' bold '),menu=filemenu)

##### TREEVIEW ATTENDANCE TABLE #####

tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))
tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)
tv.heading('#0',text = 'ID')
tv.heading('name',text = 'NAME')
tv.heading('date',text = 'DATE')
tv.heading('time',text = 'TIME')

##### SCROLLBAR #####

scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)
scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')
tv.configure(yscrollcommand=scroll.set)

##### BUTTONS #####

```

```

clearButton = tk.Button(frame2, text="Clear",
command=clear ,fg="black" ,bg="#ff7221" ,width=11 ,activebackground =
"white" ,font=('comic', 11, ' bold '))
clearButton.place(x=335, y=86)
clearButton2 = tk.Button(frame2, text="Clear",
command=clear2 ,fg="black" ,bg="#ff7221" ,width=11 , activebackground =
"white" ,font=('comic', 11, ' bold '))
clearButton2.place(x=335, y=172)
takeImg = tk.Button(frame2, text="Take Images",
command=TakeImages ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))
takeImg.place(x=30, y=300)
trainImg = tk.Button(frame2, text="Save Profile", command=psw
,fg="white" ,bg="#6d00fc" ,width=34 ,height=1, activebackground = "white"
,font=('comic', 15, ' bold '))
trainImg.place(x=30, y=380)
trackImg = tk.Button(frame1, text="Take Attendance",
command=TrackImages ,fg="black" ,bg="#3ffc00" ,width=35 ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))
trackImg.place(x=30,y=50)
quitWindow = tk.Button(frame1, text="Quit",
command=window.destroy ,fg="black" ,bg="#eb4600" ,width=35 ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))
quitWindow.place(x=30, y=450)

```

```

##### END #####

```

```

window.configure(menu=menubar)
window.mainloop()

```

```

#####
#####

```

CHAPTER 7

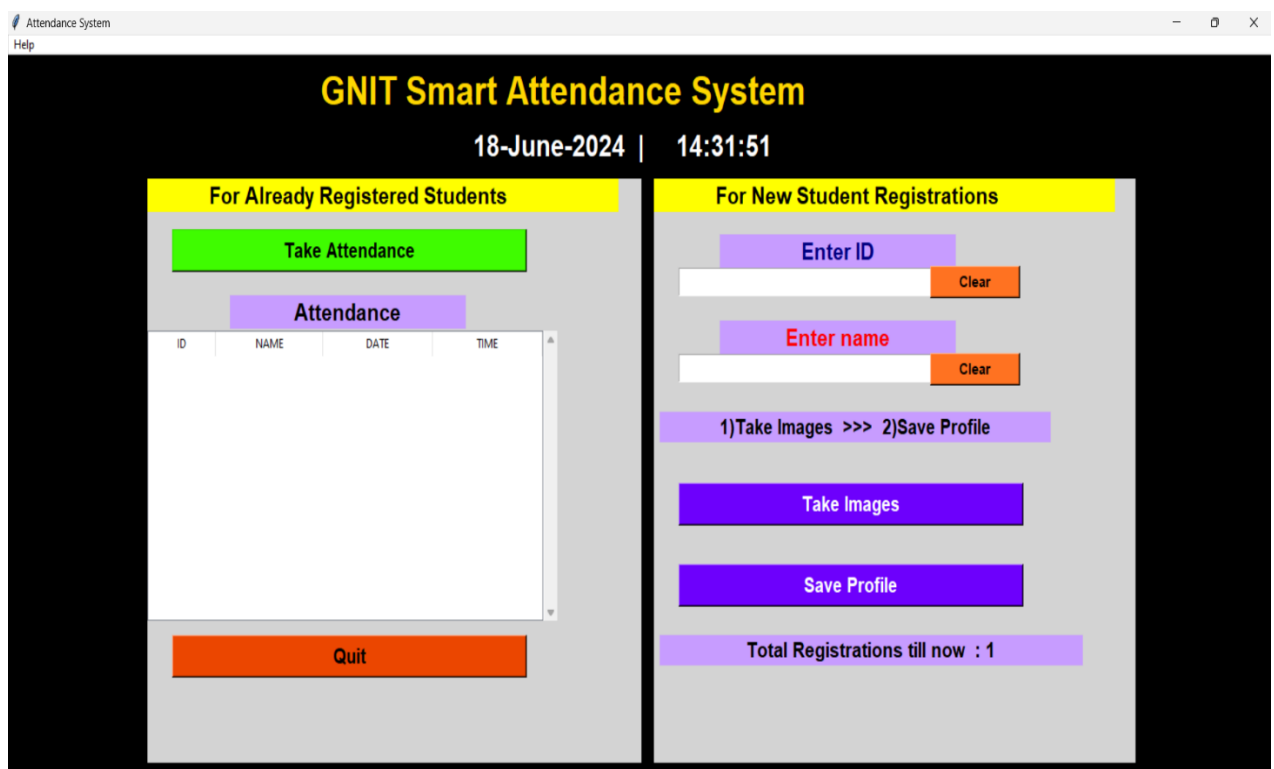
RESULTS & DISCUSSION

7.1 GENERAL

The entire project is done on vscode using python in which opencv and other powerful libraries is used to run the project

7.2 RESULTS & DISCUSSION

The interface of the project



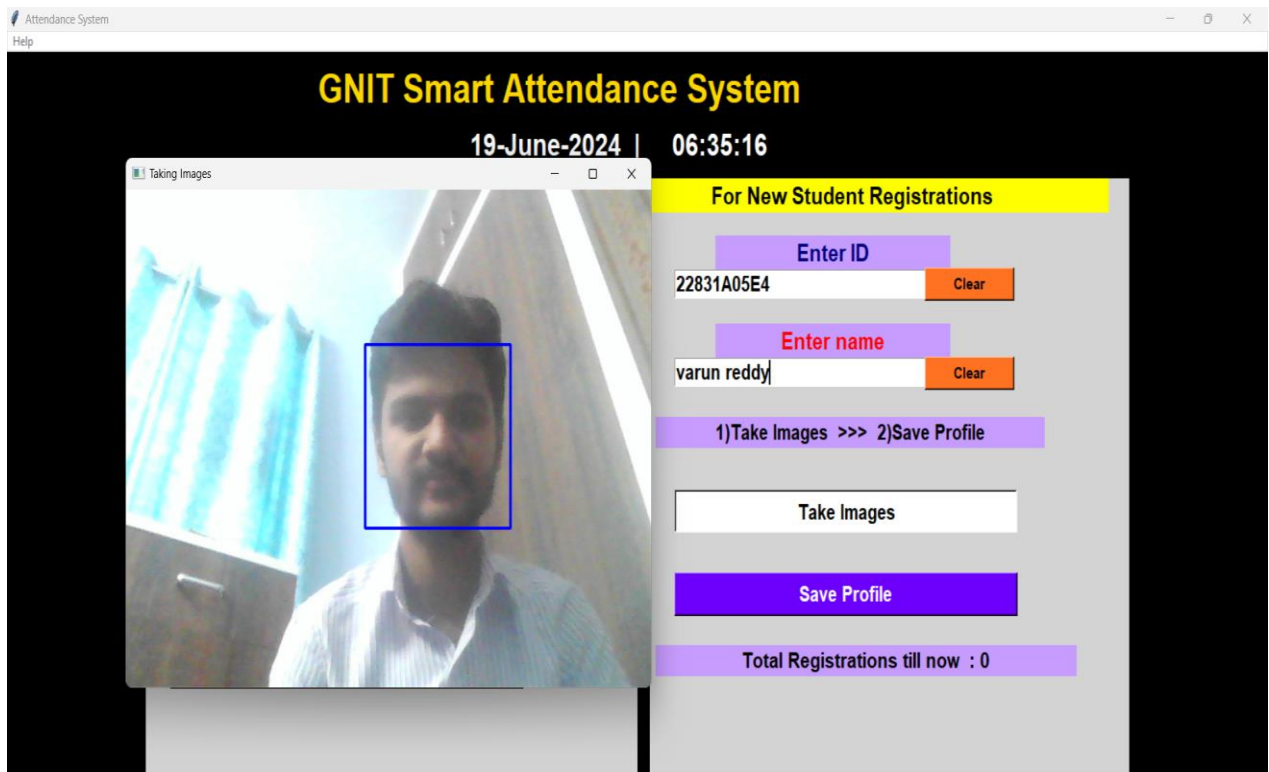
For new student registrations

1. Enter Your Details:

Input your Student ID and Name in the respective fields to initiate your registration process.

2.Capture Your Images:

Click on the "Take Images" button. A pop-up window will appear, allowing you to capture your images for the face recognition system.



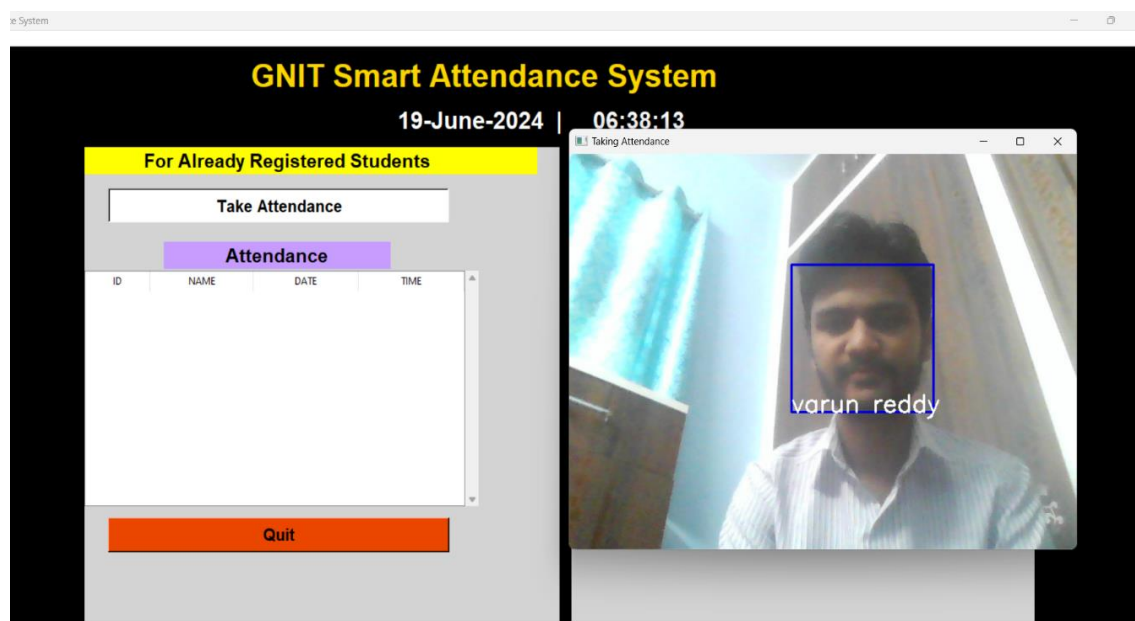
Comprehensive Student Registration Log (CSV)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SERIAL NO.		ID		NAME								
2													
3	1		1		varun								
4													
5													
6													
7													

a CSV file named StudentDetails used for storing student registration information in the GNIT Smart Attendance System.

For already registered students

1. On the main interface, you will see a section titled "For Already Registered Students."
2. Click on the "Take Attendance" button. This will activate the camera feed on the right side of the interface.
3. The system will automatically detect and recognize your face. Once recognized, your name will appear below the bounding box around your face in the live feed.



Comprehensive Student Registration Log (CSV)

The screenshot shows an Excel spreadsheet titled 'Attendance_18-06-2024 - Excel'. The spreadsheet contains a CSV log of student registrations. The columns are labeled 'Id', 'Name', 'Date', and 'Time'. The data row shows Id: 1, Name: varun, Date: 18-06-2024, and Time: 17:57:40.

Id	Name	Date	Time
1	varun	18-06-2024	17:57:40

a CSV file named attendance used for storing student registration information in the GNIT Smart Attendance System

overall view:

The screenshot displays the GNIT Smart Attendance System interface. The title bar indicates the application is 'Attendance System' with a 'Help' button. The main header shows the date and time: '19-June-2024 | 06:36:49'.

The interface is divided into two main sections:

- For Already Registered Students:** This section features a green 'Take Attendance' button. Below it is a table titled 'Attendance' with columns for ID, NAME, DATE, and TIME. The table contains one entry for 'varun reddy' with ID '22831A05E', dated '19-06-2024' at '06:36:33'. A red 'Quit' button is located at the bottom of this section.
- For New Student Registrations:** This section includes an 'Enter ID' field with the value '22831A05E4' and a 'Clear' button. Below it is an 'Enter name' field with the value 'varun reddy' and a 'Clear' button. A green message box states 'Profile Saved Successfully'. Further down are blue buttons for 'Take Images' and 'Save Profile'. At the bottom, a green box displays 'Total Registrations till now : 1'.

ID	NAME	DATE	TIME
22831A05E	varun reddy	19-06-2024	06:36:33
1	varun reddy	19-06-2024	06:28:52

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

The Python GUI integrated attendance system using face recognition represents a significant advancement in student attendance tracking. It combines OpenCV for facial recognition, Tkinter for the GUI, and CSV, Numpy, Pandas, and datetime for data handling.

This system offers accurate attendance tracking by leveraging OpenCV's facial recognition capabilities, reducing the risk of proxy attendance. The user-friendly GUI simplifies attendance marking and management, with features like password-protected registration and live updates.

Using CSV files for data storage ensures organized and accessible records. Daily attendance is recorded with timestamps, providing a comprehensive attendance history. Password protection enhances system security, preventing unauthorized access.

In conclusion, this project demonstrates the effectiveness of combining technologies to create innovative solutions. It improves upon traditional attendance methods, offering a reliable and user-friendly alternative for educational institutions.

8.2 FUTURE ENHANCEMENT

The Python GUI integrated attendance system using face recognition is a powerful tool for efficient attendance management. Future enhancements could include integrating cloud storage and processing for centralized data management and improved scalability.

Developing a mobile application would increase accessibility, enabling remote attendance marking and real-time notifications.

Enhancing facial recognition with advanced deep learning models and optimizing for real-time performance can boost accuracy. Security can be improved with multi-factor authentication and data encryption.

Adding analytics tools would provide valuable insights, and integrating with Learning Management Systems can streamline administrative tasks. Scalability can be achieved through load balancing and database optimization.

Improvements in user experience could include customizable interfaces and real-time notifications. Additionally, incorporating other biometric modalities like fingerprint or iris recognition could enhance system flexibility.

REFERENCES

- i. Bhise et al. (2015) proposed an attendance system using NFC technology and an embedded camera on mobile devices for face validation. (PDF available)
- ii. SenthamilSelvi et al. (2014) developed a face recognition-based attendance marking system for employees, ensuring secure attendance tracking. (PDF available, ISSN: 2321-0869 (O) 2454-4698 (P), Volume-3, Issue-8, August 2015)
- iii. Kumar Yadav et al. (2015) implemented a fingerprint-based attendance system using microcontrollers and LabView for database management. (PDF available, Vol 11, Issue 5, May/2020, ISSN NO:0377-9254)
- iv. Hussain et al. (2014) introduced an RFID-based student attendance system, enabling attendance tracking through a web portal. (PDF available)