**Face Recognition Based Attendance System Using Raspberry Pi**



*A Mini Project report submitted to*

# JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

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

# BACHELOR OF TECHNOLOGY

**in**

# ARTIFICIAL INTELLIGENCE & DATA SCIENCE

*Submitted by*

**E Hasini (23HR1A3016)**

**G Nikitha (23HR1A3017) G Ranjith Kumar (23HR1A3018) G Sasidhar (23HR1A3019)**

**G Sailaja (23HR1A3021)**

**Poola Akhil (24HR5A3007)**

*Under the esteemed guidance of*

**Mr. Mahesh Kumar M MTech.,(Ph.D)**

**Associate Professor**

#### DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE MOTHER THERESA INSTITUTE OF ENGINEERING & TECHNOLOGY

**Melumoi (P &V), Palamaner-517408.**

**Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu-515002 NAAC Accredited and An ISO 9001:2015 Certified Institution**

**2023-2027**

## MOTHER THERESA INSTITUTE OF ENGINEERING AND TECHNOLOGY

**AN ISO 9001:2015 CERTIFIED INSTITUTION**

**(Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu) Melumoi (P&V), Palamaner-517 408, Chittoor (Dist), A.P**

Certificate

**Face Recognition Based Attendance System Using Raspberry Pi**

*is the bonafide work done and Submitted by*

E Hasini (23HR1A3016)

G Nikitha (23HR1A3017) G Ranjith Kumar (23HR1A3018) G Sasidhar (23HR1A3019)

G Sailaja (23HR1A3021)

Poola Akhil (24HR5A3007)

*in the Department of Artificial Intelligence & Data Science, Mother Theresa Institute of Engineering and Technology, Palamaner affiliated to JNTUA, Ananthapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Artificial Intelligence & Data Science during 2025-26.*

# Submitted on:

**Internal Guide HoD**

Mr. Mahesh Kumar M, MTech.,(Ph.D) Mr. K. Lokesh, M.Tech., (Ph.D)

Associate Professor Associate Professor

**Internal Examiner External Examiner**

## ACKNOWLEDGEMENTS

Any achievement, be it scholastic or otherwise does not depend solely on the individual effort but on the guidance, encouragement and cooperation of intellectuals, elders and friends. We would like to take this opportunity to thank them all.

We feel ourselves honored for placing our warm salutation to **THE MANAGEMENT** Mother Theresa Institute of Engineering and Technology, Palamaner, which gave us the opportunity to obtain a strong base in B. Tech and profound knowledge. We express our sincere thanks to **Dr. M. LAKSHMIKANTHA REDDY, M.Tech, Ph.D.** Our beloved Principal for his encouragement and suggestions during our course of study. With deep sense of gratitude we acknowledge **Mr. K. Lokesh, M.Tech., (Ph.D)**

Head of the Department, Artificial Intelligence & Data Science, for his valuable support and help in processing our Socially Relevant Project.

We also express sincere thanks to our Project Coordinator Mahesh Kumar M., M.Tech (Ph.D) Associate Professor, in Department of Artificial Intelligence & Data Science, for encouraging us in doing this Socially Relevant Project.

We whole heartedly express gratitude and regard to our guide A V e n k a t a r a m a n a . , M.Tech (Ph.D) Assistant Professor in Department of Artificial Intelligence & Data Science, for constant support and encouragement.

Finally, we would like to express our sincere thanks to all the **Faculty Members** of AI&DS Department, Lab Technicians, Friends & Family members, who all have motivated and helped us to do this project.

**E Hasini (23HR1A3016)**

**G Nikitha (23HR1A3017) G Ranjith Kumar (23HR1A3018) G Sasidhar (23HR1A3019)**

**G Sailaja (23HR1A3021)**

**Poola Akhil (24HR5A3007)**

## ABSTRACT

This project aims to develop an automated attendance system using Face Recognition with Raspberry Pi. The system captures images using a camera module, detects faces, and matches them with stored records to mark attendance using OpenCV. The system ensures accurate recognition and eliminates the need for manual attendance. The system is designed to streamline the attendance process by providing two levels of identity verification—first through RFID tags, and second via facial recognition using OpenCV. This dual approach ensures greater reliability, prevents proxy attendance, and enhances overall system security. Attendance is marked automatically for future access, making the process efficient, secure, and contactless.

Once both authentication steps are successfully completed, the system automatically records the individual’s attendance along with the timestamp and uploads the data to database. The use of Flask API eliminates the need for local database management and enhances scalability. The system reduces human error, prevents proxy attendance, and supports contactless identification, making it ideal for deployment in schools, colleges, offices, and labs.

**Keywords:**

***Face Recognition, Raspberry Pi, OpenCV, Attendance System, RFID, Webcam, Flask API.***

## TABLE OF CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chapter No.** | **Chapter Name** | | | **Page No.** |
|  | **Title** | | |  |
|  | **Certificate** | | |  |
|  | **Acknowledgement** | | |  |
|  | **Abstract** | | |  |
|  | **List of Contents** | | |  |
| **1** | **Introduction** | | | **1-6** |
|  | 1.1 | Motivation | |
|  | 1.2 | Objectives of the Project | |
|  | 1.3 | Significance of the project | |
|  | 1.4 | Technological Overview | |
|  | 1.5 | Real World Applications | |
|  | 1.6 | Research Relevance and Innovation | |
| **2** | **Literature Survey** | | | **7-13** |
|  | 2.1 | Summary of existing articles | |
|  | 2.2 | Objective | |
|  | 2.3 | Scope of the project | |
|  | 2.4 | Raspberry Pi in IOT | |
|  | 2.5 | Components used | |
| **3** | **Problem Statement And Objectives** | | | **14-16** |
|  | 3.1 | Problem Statement | |
|  | 3.2 | Objectives | |
| **4** | **System Requirement And Architecture** | | | **17-18** |
|  | 4.1 | System Requirement | |
|  | 4.2 | System Architecture | |
| **5** | **Methodology And Design** | | | **19-21** |
|  | 5.1 | | Methodology |
|  | 5.2 | | System Design |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5.3 | | Working Principle |  |
| **6** | **Implementation** | | | **22-26** |
|  | 6.1 | | Implementation |
|  | 6.2 | | Workflow Overview |
| **7** | **Results and Discussion** | | | **27-30** |
|  | 7.1 | | Testing Parameters |
|  | 7.2 | | Face Recognition Accuracy |
|  | 7.3 | | SystemRobustness in various conditions |
|  | 7.4 | | Security And Error Handling |
|  | 7.5 | | User Feedback & Usability |
|  | 7.6 | | Discussionand future scope |
|  | 7.7 | | Real time Face Detection Challenges |
|  | 7.8 | | DiscussionAnd Improvements |
| **8** | **Comparative Analysis and Validation** | | | **31-32** |
|  | 8.1 | | SystemArchitecture Overview |
|  | 8.2 | | Comparative Feature Table |
|  | 8.3 | | Testing |
|  | 8.4 | | Maintenance and Reliability |
|  | 8.5 | | Future Enhancements |
|  | 8.6 | | System Validation Security& Potential Improvements |
| **9** | **Conclusion & future work** | | | **33-34** |
|  | 9.1 | Conclusion | |
|  | 9.2 | Future Work | |
| **10** | **Reference** | | | **35** |
| **11** | **Appendices** | | | **36-40** |

**ABBREVIATIONS**

**Pi** – Raspberry Pi

**LED** – Light Emitting Diode

**GPIO** – General Purpose Input/Output

**Pi Camera** – Raspberry Pi Camera Module

**Open CV** – Open Source Computer Vision Library **API** – Application Programming Interface **Numpy** – (Numeric python- library name)

**IoT** – Internet of Things **RGB** – Red Green Blue **USB** – Universal Serial Bus

**RFID** – Radio Frequency Identification

**MFRC522** – (Model number of RFID reader; not an abbreviation)

**Pi OS** – Raspberry Pi Operating System **LBPH** – Local Binary Patterns Histograms **SD** – Secure Digital

**RPI** – Raspberry Pi

# CHAPTER 1

**INTRODUCTION**

### Motivation

In the era of rapid digital evolution, automation and smart technology are revolutionizing traditional practices across every sector. Attendance management, a crucial activity in educational institutions and corporate environments, is one such area undergoing transformation. Manual attendance systems, although widely used, are proving inadequate in today's fast-paced world. These outdated methods are not only time-consuming but also highly prone to inaccuracies, manipulation, and fraudulent behaviour like proxy attendance. With growing organizational demands, there is a pressing need for more reliable, streamlined solutions. Technological advancements now offer smarter, faster alternatives for monitoring attendance. These innovations aim to reduce manual effort while ensuring accuracy and transparency.

Technology is now deeply integrated into our everyday lives, simplifying routine tasks and boosting productivity. In schools and colleges, ensuring accurate attendance records is vital for tracking student participation and academic progress. Likewise, in corporate settings, proper attendance helps maintain discipline, manage payrolls, and monitor employee productivity. Traditional methods such as roll calls, paper registers, or signing sheets are outdated and inefficient for modern needs. Moreover, in large institutions or companies, managing and verifying manual attendance records becomes increasingly difficult. There is also a risk of manipulation, making the system less trustworthy. Hence, the adoption of automated attendance systems has become not just an option but a necessity.

Automated systems that use biometric verification, RFID tags, and computer vision have gained significant traction for attendance tracking. Among them, face recognition stands out due to its contactless nature and high reliability. RFID, on the other hand, enables fast scanning of identity cards without the need for physical contact or manual verification. When these technologies are combined, they create a powerful system that eliminates the drawbacks of traditional methods. Integrating them with compact hardware like Raspberry Pi makes the solution both affordable and scalable.

This approach also allows for customization and real-time data logging. Such systems

offer enhanced security by ensuring only authorized individuals are recognized and logged.

Our proposed solution is a Face Recognition and RFID-Based Attendance System Using Raspberry Pi, designed to address the limitations of existing methods. The system captures the facial features of individuals using a camera and cross-verifies them with stored data for accurate identification. Simultaneously, users can scan their RFID cards to log their entry or exit. The Raspberry Pi acts as the central processing unit, managing inputs from both the face recognition module and the RFID reader.

It logs attendance data locally and can optionally send alerts via messaging platforms. The system also features an LCD display and a buzzer to provide immediate feedback to the user. By combining hardware and software seamlessly, the system delivers a complete, reliable attendance management solution.

This project not only enhances efficiency and accuracy but also promotes contactless and secure attendance logging. It is especially useful in situations where hygiene and social distancing are priorities, such as during health emergencies. Furthermore, the system can be expanded to include cloud storage, data analytics, or even machine learning for behaviour analysis and predictive reporting. Educational institutions and businesses alike can benefit from this innovation by saving time, reducing fraud, and increasing productivity. With further improvements, it can also support multi-location tracking, voice feedback, and mobile integration. Ultimately, this smart attendance system represents the future of attendance tracking. It reflects how automation and intelligent technologies can simplify and secure essential administrative tasks.

### Objectives of the Project

The primary goal of this project is to develop a Face Recognition-Based Attendance System using Raspberry Pi that overcomes the limitations of current attendance systems and offers a secure, reliable, and intelligent solution. The specific objectives are outlined below:

#### To Develop a Contactless and Reliable Attendance System

Design a system that leverages face recognition and RFID technology to record attendance accurately without physical interaction.

#### To Enhance Accuracy and Eliminate Proxy Attendance

Utilize facial recognition and unique RFID verification to prevent manipulation or fraudulent attendance, such as proxy entries.

#### To Integrate Raspberry Pi as a Central Controller

Employ Raspberry Pi to coordinate and process data from the face recognition module, RFID reader, buzzer, and LCD display efficiently.

#### To Enable Real-Time Attendance Logging

Store and update attendance data in real-time, ensuring immediate feedback and accurate record-keeping.

#### To Provide Instant Feedback to Users

Use visual (LCD display) and audio (buzzer) outputs to confirm successful attendance logging for each user.

#### To Improve Hygiene and Support Social Distancing

Promote a contactless system that is especially beneficial in health-sensitive environments like schools or workplaces during pandemics.

#### To Ensure Data Security and Accessibility

Securely store attendance data locally with the potential for cloud integration to facilitate centralized data access and reporting.

#### To Enable Future Scalability and Feature Expansion

Build a modular system that can be expanded with additional features like mobile app integration, cloud analytics, or multi-location tracking.

### Significance of the Project

The Face Recognition Attendance System using Raspberry Pi 3B+ with RFID represents a major step toward automating identity verification and attendance logging in a secure, contactless, and real-time manner. In traditional attendance systems, methods like manual registers or fingerprint scanners often lead to delays, inaccuracies, and hygiene concerns.

This project addresses all of these issues by combining two robust verification mechanisms: RFID card scanning and AI-powered face recognition. Together, they ensure that attendance is logged only when both the card and the facial identity match

a pre-registered user, significantly reducing the risk of impersonation or buddy punching. By using Raspberry Pi 3B+ as the core processing unit, the system is both cost-effective and compact, making it suitable for small to medium-scale institutions without the need for expensive infrastructure. It supports real-time decision-making using input from the RFID module and camera, and the outputs (like buzzer alerts and LCD messages) provide immediate feedback to users—informing them whether access was granted or denied. This not only improves usability but also fosters transparency and accountability in access control.

### Technological Overview

The Face Recognition Attendance System using Raspberry Pi 3B+ and RFID integrates various modern technologies including embedded systems, computer vision, IoT communication, and automation. It offers a cost-effective and efficient solution to automate attendance with enhanced security and real-time monitoring.

This system leverages both hardware and software components to work collaboratively and reliably in real-world applications.

#### Hardware Technologies

* + - * **Raspberry Pi 3B+**

Acts as the brain of the system. It interfaces with all peripherals like camera, RFID, buzzer, and LCD. Chosen for its compact size, GPIO support, and Linux-based OS compatibility.

#### MFRC522 RFID Reader

Used to identify users via RFID cards. Ensures only registered cardholders can proceed to face recognition.

#### Camera Module (USB)

Captures real-time images of the user's face for recognition. Supports OpenCV and face encoding through Python.

#### Buzzer

Provides audio feedback to indicate success or failure of authentication.

#### LCD Display (16x2)

Displays real-time messages such as “Access Granted”, “Face Not Recognized”, etc., improving usability.

### Software Technologies

* + - * Python 3

The core programming language used due to its extensive library support and Raspberry Pi compatibility.

#### Face\_recognition Library

Numpy library is used to detect and compare faces with high accuracy.

#### OpenCV

Handles image capture, processing, and enhancements like resizing and color space conversion.

#### Flask (Micro Web Framework)

Enables API-based communication between the Raspberry Pi and a web server or dashboard.

#### Telegram Bot API

Sends live images and attendance alerts directly to authorized users or groups via a Telegram bot.

#### RPi.GPIO

Used to control the buzzer, LCD, and interface with GPIO-connected devices.

#### Data Handling and Communication

* + - * **CSV File Logging**

Attendance entries are stored locally in .csv format with name, date, and time.

#### Flask REST API

Sends attendance data (name, time, UID) to a web server or cloud endpoint in real time.

#### Telegram Messaging

Captured face image of the attendee is sent via a bot for real-time monitoring and verification.

#### Real-world Applications

* **Educational Institutions (Schools, Colleges, Universities)**

Prevents proxy attendance, automates entry logs for students and staff, and integrates with digital dashboards for record keeping.

#### Corporate Offices & IT Companies

Automates employee check-in/check-out processes, improves HR attendance accuracy, and enhances workplace security.

#### Research Labs & High-Security Facilities

Controls access to critical lab equipment or data centers using multi-level authentication (RFID + Face).

#### Government Offices & Municipal Corporations

Tracks daily staff presence while providing photo-verified logs for audit and administrative records.

#### Examination Centers & Test Halls

Verifies that only the registered candidate attends the exam; helps eliminate impersonation and unfair practices.

#### Factories and Manufacturing Units

Provides shift-based attendance records and ensures only trained/authorized personnel operate machinery.

#### Banks and Financial Institutions

Adds an extra layer of verification for internal staff access or visitor entry to high-security rooms. .

#### Temples, Mosques & Religious Institutions

Monitors authorized staff or volunteers entering donation rooms or archives.

#### Airports & Railway Control Rooms

Enhances internal staff verification in restricted areas like control towers or engine rooms.

# CHAPTER-2 LITERATURE SURVEY

### Summary of Existing Articles

1. In 2019, Patel et al introduced a facial recognition attendance system using the Haar Cascade Classifier and Local Binary Pattern Histogram (LBPH) algorithm implemented on a Raspberry Pi 3. The authors developed a real-time face detection and recognition module capable of marking attendance automatically. The system proved to be cost- effective and suitable for small to medium-scale classroom environments with decent accuracy under stable lighting conditions.
2. In 2020, Sharma et al proposed an enhanced face recognition-based attendance monitoring system using Raspberry Pi integrated with Python and OpenCV. The system utilized the Dlib and face\_recognition libraries for improved detection accuracy. The paper emphasized system performance under different lighting scenarios and head poses. The results indicated that facial recognition was a reliable alternative to biometric or RFID systems in terms of user convenience and accuracy.

### Objective

The objective of this project is to design and implement an automated attendance system using face recognition technology powered by a Raspberry Pi. The system aims to accurately identify and verify individuals' identities through facial recognition and record their attendance in real time, thereby reducing manual errors, preventing proxy attendance, and improving efficiency in managing attendance records in educational institutions or workplaces.

This project also seeks to leverage the portability and cost-effectiveness of the Raspberry Pi to create a compact, standalone solution that can operate with minimal hardware requirements. By integrating a camera module, facial recognition algorithms, and a user- friendly interface, the system ensures secure and seamless attendance marking. Additionally, the project aims to explore the use of open-source libraries and tools to make the solution scalable, customizable, and accessible for various real-world applications.

### Scope of the Project

The project focuses on developing a reliable and efficient face recognition-based attendance system using Raspberry Pi as the core processing unit. The system is intended

for use in educational institutions, offices, and other organizations where regular attendance tracking is necessary. It includes capturing facial images through a camera, processing them using facial recognition algorithms, and storing attendance records in a database. The scope also covers real-time face detection, recognition accuracy in various lighting conditions, and secure storage of data. The system is designed to reduce human intervention, eliminate proxy attendance, and enhance administrative efficiency.

Additionally, the project explores the use of lightweight and open-source technologies to ensure affordability and scalability. Future expansions may include features like remote access, integration with cloud storage, SMS/email notifications, and multi-device synchronization to extend its usability and functionality in larger institutions or multiple locations.

#### Significance

The face recognition-based attendance system using Raspberry Pi offers a modern and efficient alternative to traditional attendance methods such as manual sign- ins or RFID cards. Its significance lies in its ability to automate the entire attendance process with high accuracy and minimal human intervention. By using facial recognition technology, the system eliminates issues like buddy punching or proxy attendance, ensuring that attendance records are reliable and tamper-proof. This enhances accountability and discipline in educational and professional environments.

Furthermore, the use of Raspberry Pi makes the system cost-effective, compact, and energy-efficient, making it accessible even for small institutions or organizations with limited budgets. The project also promotes the adoption of innovative technologies like artificial intelligence and computer vision at a grassroots level, encouraging smarter and more secure systems for daily operations. It demonstrates how affordable hardware and open-source software can be combined to solve real-world problems effectively.

#### Face Recognition Algorithms

**Open CV:**

OpenCV (Open Source Computer Vision Library) is an open-source

computer vision and machine learning software library. It contains over 2,500 optimized algorithms for real-time image processing tasks such as object detection, face recognition, motion tracking, etc.

Role of OpenCV in Face Recognition Attendance System

In a face recognition-based attendance system, OpenCV is primarily used for:

* + - * Capturing Real-Time Video from the camera module (Pi Camera or USB webcam).
      * Face Detection using pre-trained classifiers (e.g., Haar Cascades or HOG).
      * Image Preprocessing, such as resizing, converting to grayscale, etc.
      * Drawing bounding boxes around detected faces.
      * Interfacing with face recognition libraries for encoding and matching.

### Raspberry Pi in IOT

The development of a face recognition-based attendance system using Raspberry Pi holds significant importance in today's digital and technology-driven world. Traditional attendance methods, such as manual roll-calling or card-based systems, are time-consuming, prone to human error, and vulnerable to manipulation, such as proxy attendance. This project introduces an innovative, efficient, and secure approach to automate the attendance process using facial recognition, with several impactful benefits:

#### Automation and Time Efficiency

By automating the attendance process, the system drastically reduces the time required to take attendance in schools, colleges, or workplaces. This allows teachers and administrators to focus on more productive activities, increasing overall efficiency.

#### Improved Accuracy and Reliability

Unlike manual or card-based systems that are susceptible to mistakes or misuse, facial recognition technology provides a high level of accuracy in identifying individuals. It reduces the chances of errors or proxy attendance, ensuring that attendance records reflect the true presence of individuals.

#### Cost-Effective and Scalable Solution

Using Raspberry Pi—a low-cost, credit-card-sized computer—makes this system affordable and accessible for institutions with limited budgets. The compact and portable nature of the Raspberry Pi also allows the system to be easily deployed and scaled to different environments and applications.

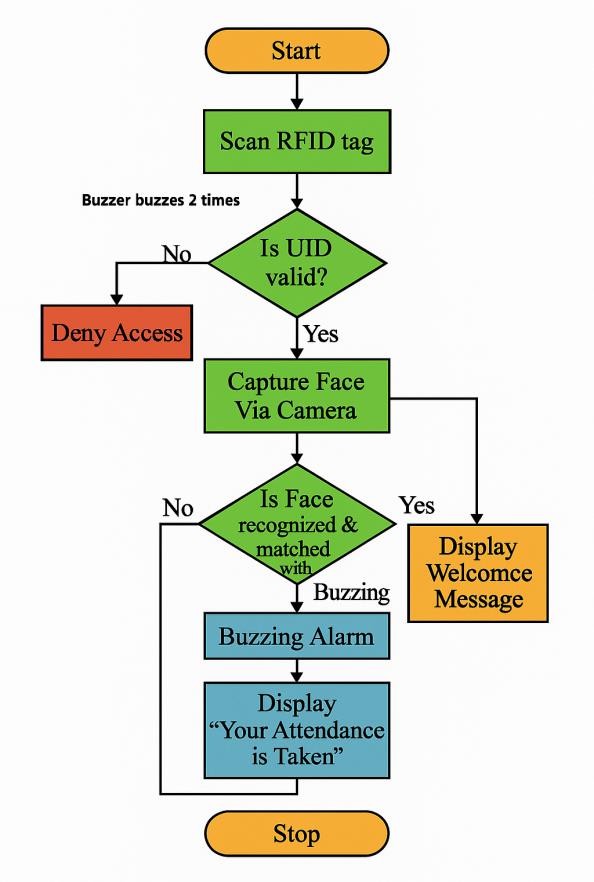
#### Contactless and Hygienic Operation

Especially in the wake of global health concerns like the COVID-19 pandemic, contactless technologies have become essential. This system eliminates the need for physical interaction with biometric devices or paper registers, promoting hygiene and safety.

#### Data Management and Record Keeping

The system stores attendance data in a structured and searchable format, allowing for easy retrieval, monitoring, and reporting. This improves administrative efficiency and supports better decision-making through data analysis

### Workflow

****

***Figure 2.4.1: System Workflow***

### Components Used

#### Webcam (USB camera)

In this project, the webcam plays a crucial role as the primary input device for capturing facial images. It is connected to the Raspberry Pi and is responsible for continuously scanning and detecting faces in real time. Once a face is detected, the webcam captures the image, which is then processed by the face recognition algorithm to identify the individual. The quality and resolution of the webcam significantly impact the accuracy of recognition, especially under varying lighting conditions or angles. A high- definition USB webcam is typically used to ensure clear image capture, which is essential for effective face detection and matching. The use of a webcam makes the system user-

friendly, non-intrusive, and suitable for real-time attendance monitoring without requiring physical interaction.



***Figure 2.5.1: WebCam***

#### Raspberry Pi 3b Model

The Raspberry Pi 3B also features 40 GPIO (General Purpose Input/Output) pins, which open up possibilities for hardware expansion like buzzers, LEDs, or fingerprint sensors if multi-factor authentication is needed in future upgrades.

In this project, the Raspberry Pi 3B is responsible for:

* + - * Capturing live video feed from the webcam
      * Detecting and recognizing faces using computer vision libraries
      * Recording attendance data and timestamps
      * Optionally storing data locally or sending it to a remote server

Its compact size, low power consumption, and community support make it an ideal platform for implementing a smart attendance system that is both portable and scalable.



***Figure 2.5.2: Raspberry Pi***

#### Raspberry Pi Charger

A stable and reliable power supply is essential for the effective operation of the Raspberry Pi 3 Model B and all connected peripherals in this face recognition-based attendance system. The Raspberry Pi 3B requires a 5V micro-USB power supply with a current rating of at least 2.5A to function optimally. This ensures that the board receives enough power to run the processor, maintain Wi-Fi connectivity, and support connected devices such as the webcam, keyboard, or USB drives without interruptions or unexpected shutdowns.

***Figure 2.5.3: USB Power Cable***

#### Memory Card

****

***Figure 2.5.4: Memory card***

This project aims to design and implement a smart attendance system that uses face recognition technology to automate attendance tracking, built using a Raspberry Pi 3B. The goal is to enhance efficiency and accuracy in classrooms, offices, and institutions by replacing traditional roll calls and biometric systems with a contactless, AI-powered solution.

A USB webcam or Raspberry Pi camera module is used to capture real-time images of individuals. These images are processed using OpenCV and the face recognition library to detect and identify authorized faces. When a recognized face is detected, the system records the individual's name, date, and timestamp in a local database such as a cloud- based platform. If the face is not recognized, it is either ignored or logged as an unknown

entry for further review.

The Raspberry Pi 3B acts as the central processing unit, handling image processing, face matching, database updates, and optional web integration. A microSD card stores the operating system, trained face data, and attendance records. The device is powered by a 4 USB 2 ports power supply, ensuring stable and reliable performance.

This IoT-based attendance system is cost-effective, low-maintenance, and scalable. It reduces manual effort, prevents proxy attendance, and supports remote monitoring via a web interface. With simple Python scripts, opencv for face recognition, and database integration, this project demonstrates how IoT technologies can be used to build intelligent automation systems for real-world applications.

# CHAPTER-3

**PROBLEM STATEMENT AND OBJECTIVES**

### Problem Statement

The existing attendance systems fall short in several aspects. Traditional manual systems—where students or employees mark their attendance by signing a register or responding to a roll call— are inefficient and easily manipulated. These systems are highly time-intensive, especially in institutions with a large number of individuals. Moreover, they are vulnerable to human errors, missing entries, and proxy attendance, where one person marks attendance on behalf of another.

To address some of these issues, organizations have adopted basic automated systems such as RFID card readers, barcode scanners, and biometric fingerprint scanners. While these offer some level of automation, they also have significant limitations. RFID-based systems do not authenticate identity beyond verifying the presence of a card; if a person gives their card to someone else, the system cannot distinguish between the two. Fingerprint systems, although more secure, are not always hygienic and can suffer from poor accuracy due to dirt, moisture, or injuries on fingers. Moreover, most of these systems operate in isolation without integration into centralized cloud based platforms.

This makes real-time data monitoring, analytics, and report generation difficult. In a post-pandemic world, contactless technologies have become essential for safety and hygiene, further underlining the inadequacy of traditional touch-based biometric systems. Hence, the challenge lies in designing a smart, secure, and contactless attendance system that ensures high accuracy, prevents proxy attendance, offers real- time data access, and is scalable and cost-effective. This project tackles this challenge by leveraging facial recognition as a second layer of authentication alongside RFID, powered by a Raspberry Pi, and integrating the system with cloud storage for real-time record keeping.

### Objectives

The primary goal of this project is to develop a Face Recognition-Based Attendance System using Raspberry Pi that overcomes the limitations of current attendance systems and offers a secure, reliable, and intelligent solution. The specific objectives are outlined below:

#### To Design and Develop a Dual Authentication Attendance System

The system will utilize two levels of authentication:

* + - First, each user will scan an RFID card that contains a unique ID.
    - Second, the system will capture the user’s face using a connected camera and match it against stored images using facial recognition algorithms (OpenCV and numpy libraries).

Only if both RFID and facial data match will the system consider the identity verified and mark attendance. This dual-layer authentication is aimed at preventing fraudulent entries and enhancing system security.

#### To Implement a Contactless and User-Friendly Interface

The system should be completely contactless to maintain hygiene and user convenience. It will provide real-time feedback to users through:

* + - LCD displays showing authentication status,
    - LED indicators for visual feedback (e.g., green for success, red for failure),
    - Buzzers for audible which is not confirm attendance.

This ensures users are aware of their attendance status immediately and the system remains intuitive to operate.

#### To Use Low-Cost, Energy-Efficient Hardware

Raspberry Pi is selected as the hardware platform due to its:

* + - Low cost and compact form.
    - Sufficient processing power for face recognition tasks.
    - GPIO support for connecting RFID readers, LCDs, buzzers, and LEDs.
    - Compatibility with Python, an accessible and powerful programming language.

This makes the system highly portable, energy-efficient, and ideal for small-to-medium scale institutions with limited budgets.

#### To Automate Attendance Recording and Reporting

Once a user is successfully authenticated, the system will:

* + - Automatically record the attendance along with the timestamp.
    - Save it to the database in real-time.

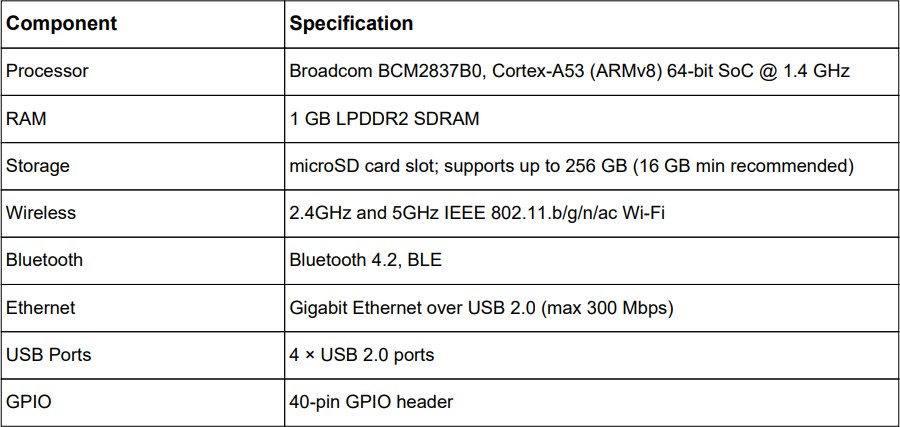
This automation reduces the burden on faculty or administrators, eliminates data entry errors, and ensures quick report generation when needed.

# CHAPTER-4

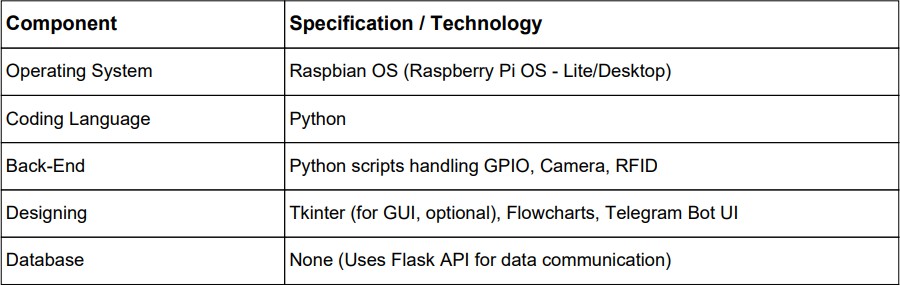
**SYSTEM REQUIREMENT AND ARCHITECTURE**

### System requirement

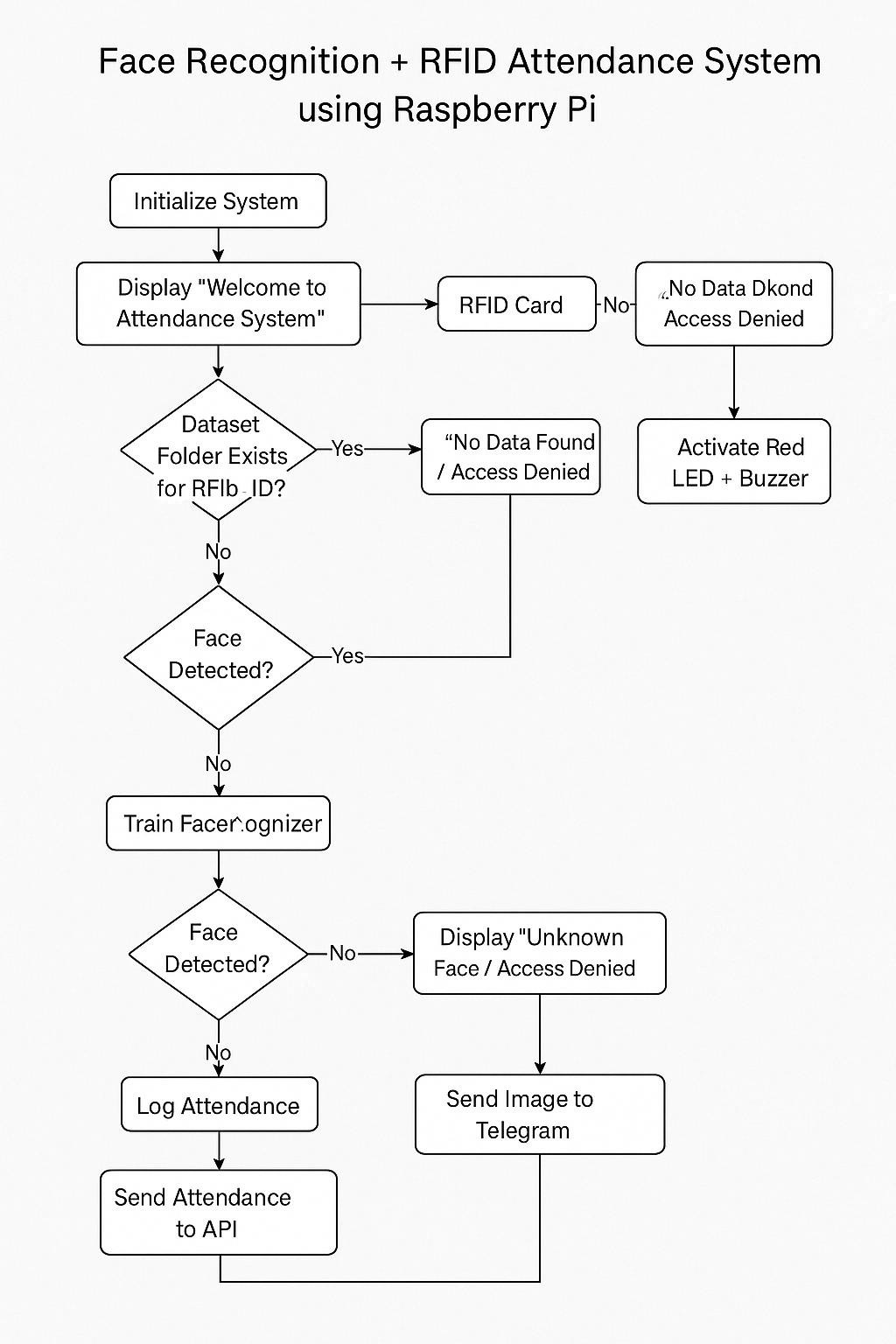
* + 1. **Hardware Components**

****

* + 1. **Software Components**

****

### System Architecture Block Diagram



***Fig: System Architecture***

# CHAPTER-5

**METHODOLOGY AND DESIGN**

### Methodology

The methodology for the Face Recognition Attendance System using Raspberry Pi 3B+ and RFID is designed to ensure secure, real-time, and contactless attendance verification. The system follows a modular and sequential process, combining hardware integration with intelligent software logic for effective performance

#### Requirement Analysis

* + - * Identified the key components: RFID reader, camera module, buzzer, LCD display, and Raspberry Pi 3B+.
      * Defined the problem: eliminate proxy attendance and improve security with double authentication.
      * Selected appropriate libraries: *face\_recognition, OpenCV, Flask, and telepot*.

#### System Design

* + - * Created a dual-verification logic:
        + ○ Step 1: Verify user with RFID card.
        + ○ Step 2: Validate face using stored encodings.
      * Integrated buzzer and LCD for user feedback.
      * Designed a Telegram bot to send real-time photo alerts.
      * Used Flask API for remote logging and web integration.

#### Data Collection and Preprocessing

* Collected face images of authorized users.
* Encoded each face using the face\_recognition library and stored them locally.
* Stored authorized RFID UIDs in a lookup list or database.

#### Implementation

* Programmed Raspberry Pi GPIO to interface with RFID, camera, buzzer, and LCD.
* Developed Python scripts to:
  + Read and validate RFID.
  + Activate camera upon valid RFID scan.
  + Match the captured face with stored encodings.
  + Store attendance in a CSV file and send data to Flask API.
  + Send user image to Telegram bot.

#### Evaluation and Refinement

* Fine-tuned timing, face recognition threshold, and buzzer duration.
* Improved response time and accuracy of face detection under different lighting conditions.
* Ensured system resets properly after each entry.

#### Deployment

* The complete system is deployed on Raspberry Pi 3B+ with all peripherals connected.
* Scripts are configured to auto-start on boot for real-time use.

### 5.2.3. Design Considerations

* **Security**: Double verification using RFID and facial identity prevents misuse.
* **Efficiency**: Camera activates only after RFID validation to save resources.
* **Modularity**: Each part (RFID, face recognition, logging, bot) functions independently.
* **Portability**: System is lightweight and easy to deploy using Raspberry Pi.
* **Scalability**: Can support additional users and new APIs with minimal modification.

### Hardware Architecture

* + - * **Processor** – Runs face recognition algorithms and controls connected devices.
      * **RAM** – Supports smooth execution of scripts and image processing.
      * **Storage** – Stores OS, face data, Python scripts, and logs.
      * **Wireless** – Sends attendance data and connects to Telegram API.
      * **Bluetooth** – Optional: for wireless communication with external modules.
      * **Ethernet** – Used for stable network connection if Wi-Fi is unavailable.
      * **USB Ports** – Connects camera, RFID reader, and other peripherals.
      * **GPIO** – Interfaces with buzzer, LCD, and RFID module.

### Software Architecture

* + - * **Operating System** – Hosts all scripts and manages system functions.
      * **Coding Language** – Python handles face recognition and hardware control.
      * **Back-End** – Executes logic for face matching and attendance marking.
      * **Designing** – Telegram Bot UI used for attendance alerts.
      * **Database** – Flask API used to send data; no internal database needed.

# CHAPTER-6 IMPLEMENTATION

### Implementation

The implementation phase involved setting up both the hardware and software components in a step-by-step manner. Special emphasis was placed on accurate data collection and efficient model preparation for reliable face recognition under varying real-world conditions.

#### Data Collection

* + - * Face images of all authorized users were captured using the Raspberry Pi camera module.
      * Each user’s face was captured from **multiple angles and under different lighting conditions** to improve recognition accuracy.
      * The captured images were stored in a folder structure where each subfolder was named after the user (e.g., /faces/Shubham/, /faces/Sneha/).
      * RFID UIDs were also collected and mapped to the corresponding face image folders manually or via script.
      * **Telegram user IDs** were also noted for sending image alerts through the bot.

#### Face Encoding and Model Preparation

* The face\_recognition Python library was used to convert face images into

**128-dimensional feature vectors** (face encodings).

* Each encoding was paired with a corresponding **user name and UID**, forming a training dataset in memory.
* This dataset was stored in a Python dictionary or .pkl file to load quickly during runtime.
* The Raspberry Pi performs **real-time face matching** using **Euclidean distance**

between live camera face encodings and the pre-trained face data.

#### Integration with RFID

* The MFRC522 RFID module was programmed to continuously scan for RFID tags.
* If the UID matched a stored value, the face recognition process was triggered.
* If the UID was invalid, the system gave an immediate alert (buzzer twice) and rejected access.

#### Real-Time Execution

Once both RFID and face recognition passed, the system:

* Logged the attendance to a **CSV file**
* Sent the data to a **Flask API** (for live dashboard display)
* Sent the captured face image to the **Telegram bot**

**Tools & Libraries Used**

|  |  |
| --- | --- |
| Component | Purpose |
| OpenCV | Camera handling and image capture |
| MFRC522 | RFID scanning and UID extraction |
| Flask | Attendance API integration |
| telepot / python-telegram-bot | Telegram image notification |
| RPi.GPIO | Controls buzzer and LCD modules via GPIO pins |

### Workflow Overview

To successfully implement the **Face Recognition Attendance System using Raspberry Pi 3B+ with RFID**, both the hardware and software environments were configured carefully to ensure real-time processing, GPIO control, and secure data transmission.

#### Hardware Setup

The hardware components are interconnected through the GPIO pins of the Raspberry Pi 3B+. Below is the setup flow:

#### Raspberry Pi 3B+

* + Acts as the central controller.
  + All peripheral components (camera, RFID, buzzer, LCD) are connected to it.

#### Camera Module

* + Connected via CSI interface or USB.
  + Used to capture the face of the person after RFID verification.

#### MFRC522 RFID Reader

* + Connected via SPI to GPIO pins.
  + Continuously scans for RFID tags/cards.
  + Detects and extracts UID to verify authorization.

#### LCD Display (16x2)

* + Connected using 4-bit mode via GPIO.
  + Displays messages like “Access Granted”, “Face Not Recognized”.

#### Buzzer

* + Connected to GPIO pin.
  + Used for access feedback:
    - 1 buzz → success
    - 2 buzzes → failure

#### Power Supply

* + 5V/3A adapter powers the Raspberry Pi and all connected modules.

#### Software Setup

The software environment includes operating system configuration, Python libraries, and service setup.

#### Operating System

* + **Raspbian OS** installed on a **16GB or higher microSD card**.
  + SSH and camera enabled via Raspberry Pi Configuration.

#### Python Environment

* **Python 3** installed as the core development language.

Key libraries installed using pip:

pip install opencv-python numpy flask telepot xlwt pandas

#### Local Face Encoding Script

* A script is used to load images from folders and generate.encodings using the face\_recognition library.

#### RFID Reader Script

* Python script constantly reads UID values from MFRC522 module.

#### Main Attendance Script

* Integrates RFID reading, camera capture, face comparison, CSV logging, buzzer & LCD control, Telegram image sending, and Flask API request.

#### User Interface and Notifications

Integration with a small 16x2 LCD to display status messages. Display shows "Access Granted" or "Access Denied".

#### Notifications:

Email notification using SMTP if an unauthorized face is detected. Logging of access attempts saved locally in a text file with timestamps. Enhancements: Integration with IOT dashboards for remote monitoring.

### Testing and Troubleshooting

#### Testing Conditions:

Tested under various lighting conditions (daylight, dim light). Verified system performance with known and unknown faces. **Common Issues:**

Low-quality images in poor lighting affected accuracy. Servo motor jitter due to unstable power.

#### Future Improvements:

Add support for mask detection.

Use fingerprint or RFID as a backup method. Connect to mobile app for live alerts and control.

# CHAPTER 7

**RESULTS AND DISCUSSION**

### Testing Parameters

The testing parameters for the **Face Recognition Attendance System using Raspberry Pi 3B+ and RFID** were carefully selected to evaluate the accuracy, performance, reliability, and real-world responsiveness of the system. Each parameter focuses on a critical aspect of functionality to ensure that the system meets expected operational standards.

#### Accuracy

* **Definition**: Measures how accurately the system matches the scanned face and RFID tag with the correct registered user.
* **Goal**: Achieve >95% accuracy in face recognition with correctly matched RFID.
* **Method**: Multiple test attempts with known users under various lighting and angles.

#### Response Time

* **Definition**: The time taken from RFID scan to attendance logging and feedback.
* **Expected Range**: Less than 3 seconds from scan to system response.
* **Measurement**: Stopwatch timing during trials from RFID placement to buzzer/response.

#### False Acceptance Rate (FAR)

* **Definition**: Rate at which unauthorized users are mistakenly accepted by the system.
* **Goal**: Maintain a FAR close to 0%.
* **Test Condition**: Present unknown faces or invalid RFID cards to test rejection mechanism.

#### False Rejection Rate (FRR)

* **Definition**: Rate at which valid users are mistakenly denied access.
* **Goal**: Keep FRR below 5%.
* **Test Condition**: Vary lighting and face orientation to test robustness.

#### Real-Time Feedback Validation

* **Definition**: Checks whether the buzzer, LCD, and Telegram bot give appropriate responses for each scenario.
* **Goal**: Accurate, instant feedback on access status (e.g., 1 buzz for success, 2 buzzes for failure).

#### Logging Integrity

* **Definition**: Ensures that attendance entries (name, date, time) are correctly stored in the CSV file and sent via API.
* **Verification**: Cross-validate CSV logs and web API database records.

#### Telegram Notification

* **Definition**: Validates that the captured face image is sent to the Telegram bot only when a face is successfully verified.
* **Test**: Check the bot chat history after each valid entry.

#### Component Reliability

* **Definition**: Measures the stable operation of hardware like the camera, RFID reader, LCD, and buzzer over repeated tests.
* **Test**: Continuous operation over extended hours without system failure.

### 7.8 Discussion and Improvement

1. Incorporate Aliveness Detection Mechanisms

Techniques such as blink detection, motion analysis, or depth sensing can help distinguish real faces from spoofing attempts.

1. Mobile App Integration

Developing a companion mobile app can enable real-time user notifications, remote control, and system alerts.

1. Cloud-Based Storage and Access

Shifting user data to cloud platforms allows for scalable storage, centralized database access, and remote management of attendance logs.

1. Include Power Backup System

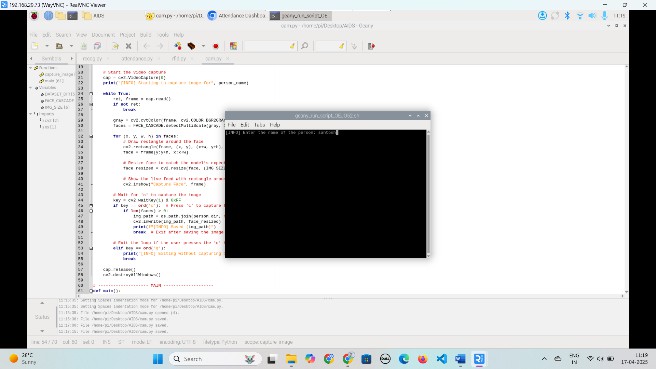
Adding an uninterrupted power supply (UPS) or battery backup ensures the system remains functional during power outages.

1. Optimize Database Search Algorithms

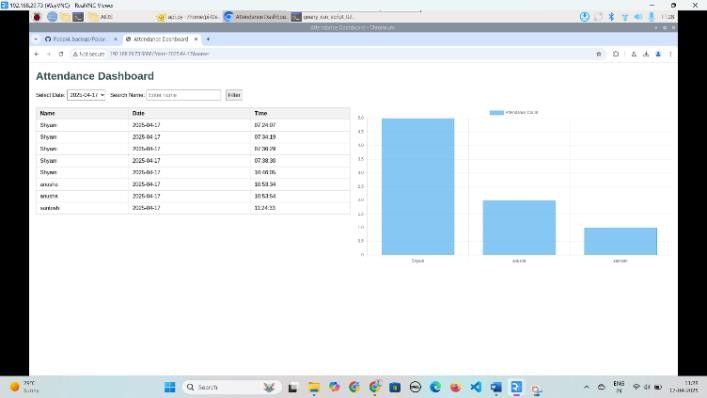
Enhancing the facial recognition algorithm for quicker comparisons can improve real- time performance with larger datasets.



***Figure: RFID Connectivity Figure: Hardware Setup***



***Figure: Data Storage***

******

***Figure: Data displayed in cloud***

# CHAPTER 8

**COMPARATIVE ANALYSIS AND VALIDATION**

### System Architecture Overview

#### Core Components

Raspberry Pi – Acts as the main controller for processing and coordination.

Camera Module – Captures live images for face recognition.

RFID Reader – Verifies ID cards and triggers face matching process.

LCD Display (I2C) – Displays status messages like "Access Granted".

Buzzer – Gives audio feedback on recognition success or failure.

Telegram API – Sends attendance notifications to the admin.

Flask API – Handles attendance data communication (optional backend).

#### Work Flow

**RFID Scan** – Student scans RFID card.

**Face Detection** – Camera captures the face of the student.

**Face Recognition** – Compares captured image with stored face data.

**Attendance Marking** – If matched, attendance is recorded.

**Notification & Output** – LCD shows status, buzzer gives feedback, and a Telegram message is sent to the admin.

#### Modules Integrated:

* RFID Reader Module – Detects and reads the UID of RFID cards.
* Camera Module – Captures the user’s face for verification.
* Face Recognition Module – Compares captured image with registered faces.
* Buzzer Module – Provides alerts for access status (success or failure).
* Flask API Logger – Logs attendance data to a server or database.

#### 8.3.3 Validation Checking: Objective:

Validation checking ensures that the inputs provided to the system are accurate, complete, and meet the required format or conditions. It helps prevent unauthorized access, invalid operations, or incorrect data logging in the Face Recognition Attendance System using Raspberry Pi 3B+ and RFID.

#### Key Validation Checks in the System:

1. **RFID Card Validation**
   * **Validation Rule**: UID must match a pre-registered list of valid RFID cards.
   * **Check Type**: Format check and lookup in UID database or list.

#### System Behaviour:

If the UID is found → Proceed to camera capture.

If the UID is not found → Deny access and buzz twice.

* + **Result**: Prevents unauthorized card usage.

#### Face Recognition Match

* + **Validation Rule**: The captured face must match a known face encoding.
  + **Check Type**: Pattern recognition using face\_recognition.compare\_faces().

#### System Behaviour:

If face matches → Attendance is logged.

If face does not match → Access denied, buzzer alert.

* + **Result**: Ensures correct user identification after RFID scan.

#### Duplicate Attendance Prevention

* + **Validation Rule**: A user should not be able to mark attendance more than once in the same session.
  + **Check Type**: Cross-check with the previous entries before logging.

#### System Behaviour:

If already marked → Skips re-entry.

If new entry → Logs name and timestamp.

* + **Result**: Prevents data redundancy.

# CHAPTER 9 CONCLUSION and FUTURE WORK

### Conclusion

The successful implementation of a Face Recognition-Based Attendance System using Raspberry Pi and RFID showcases the powerful impact of combining artificial intelligence with embedded systems. This innovative solution overcomes the limitations of traditional attendance methods by introducing a fast, secure, and contactless alternative that minimizes human error and prevents misuse. The use of low- cost, open-source tools makes the system not only accessible but also easily scalable. Its flexibility and adaptability make it well-suited for institutions of all sizes, setting the stage for further advancements in smart automation and digital transformation.

This project proves that intelligent automation can solve everyday administrative challenges with elegance and efficiency. By employing dual-layer authentication through facial recognition and RFID technology, the system ensures accurate, real-time attendance logging. Cloud integration supports centralized data access and management, while the Raspberry Pi provides an affordable yet powerful platform for edge computing. The system’s contactless nature, combined with its reliability and ease of deployment, makes it ideal for use in classrooms, labs, offices, and even public facilities. It is a practical, scalable solution that blends hardware and software innovation to enhance institutional productivity.

### Future work

#### LivenessDetectionIntegration

To prevent spoofing using photos or videos, the system can be enhanced with liveness detection techniques such as blink detection, head movements, or depth sensing using IR cameras.

#### MobileAppIntegration

Develop a companion mobile app for students, employees, or administrators to view attendance records, receive notifications, and manage user profiles in real-time.

#### Cloud-Based Real-Time Analytics

Implement dashboards using platforms like Firebase, AWS, or Google Cloud for real- time attendance tracking, analytics, and reporting, accessible from any device.

#### Camera Integration

Especially relevant post-pandemic, integrating thermal sensors can allow simultaneous temperature checks with attendance marking.

#### Multi-Camera and Multi-Room Support

Extend the system to support multiple cameras across different rooms or buildings for centralized attendance monitoring.

#### Self-Enrollment

Build an interface that allows users to self-register their face and RFID tag through a guided process using a touchscreen and camera.

#### Automated Alerts and Escalations

Configure the system to automatically notify authorities or guardians when a user is marked absent for a specified number of days.

#### RFID card Personalization

Include dynamic data on RFID cards such as role, permissions, or daily schedule, enabling role-based access control within the attendance system.

# CHAPTER 10 REFERENCES

1. Raspberry Pi Official Documentation https:/[/www.raspberrypi.org/documentation/](http://www.raspberrypi.org/documentation/)

– Guides and technical docs for setting up and programming the Raspberry Pi.

1. MFRC522 RFID Reader Library for Python https://pypi.org/project/mfrc522/

– Python package used to interface the RFID reader with the Raspberry Pi.

1. Face Recognition Library (by Ageitgey) https://github.com/ageitgey/face\_recognition

– Powerful Python library for real-time face recognition using dlib and OpenCV.

1. RPi.GPIO Documentation

https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

– Used for controlling buzzer, RGB LED, and other GPIO devices.

1. I2C LCD with Raspberry Pi Tutorial [https://www.circuitbasics.com/raspberry-pi-i2c-lcd-set-up-and-](https://www.circuitbasics.com/raspberry-pi-i2c-lcd-set-up-and-programming/)

[programming/](https://www.circuitbasics.com/raspberry-pi-i2c-lcd-set-up-and-programming/)

– Tutorial for interfacing 16x2 I2C LCD with Raspberry Pi using Python.

1. OpenCV (cv2) Python Library https://docs.opencv.org/ –

Open-source computer vision library used for capturing and processing camera

input.

1. 7ROBOLABS - How To Use RFID With Raspberry Pi , Face Recognition on Raspberry Pi
2. Python CSV Module Documentation https://docs.python.org/3/library/csv.html

– Used for attendance log storage in CSV format.

1. Stack Overflow & Raspberry Pi Forums https:/[/www.raspberrypi.org/forums/](http://www.raspberrypi.org/forums/)

# CHAPTER 11

**APPENDICES**

# Appendix A: Face Data Collection and Training

### Data Collection(01Facedataset.py)

import cv2 import os

def collect\_face\_data(name): save\_path = f"dataset/{name}"

os.makedirs(save\_path, exist\_ok=True) cap = cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

print("[INFO] Press 's' to save face, 'q' to quit.") while True:

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

break

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

for (x, y, w, h) in faces:

face\_img = frame[y:y+h, x:x+w] cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,0), 2) cv2.imshow("Face Capture", frame)

key = cv2.waitKey(1) if key == ord('s'):

face\_resized = cv2.resize(face\_img, (160, 160)) cv2.imwrite(f"{save\_path}/face.jpg", face\_resized) print(f"[INFO] Face saved at {save\_path}/face.jpg") cap.release()

cv2.destroyAllWindows() return

elif key == ord('q'): cap.release() cv2.destroyAllWindows() return

collect\_face\_data("abcd")

### Train the Model (02\_face\_training.py)

import cv2

import numpy as np import os

import json

import tflite\_runtime.interpreter as tflite def preprocess\_face(face\_img):

face = cv2.resize(face\_img, (160, 160))

face = face.astype('float32')

mean, std = face.mean(), face.std() face = (face - mean) / std

return np.expand\_dims(face, axis=0) def get\_embedding(interpreter, face\_img):

input\_details = interpreter.get\_input\_details() output\_details = interpreter.get\_output\_details() preprocessed = preprocess\_face(face\_img)

interpreter.set\_tensor(input\_details[0]['index'], preprocessed) interpreter.invoke()

embedding = interpreter.get\_tensor(output\_details[0]['index']) return embedding[0].tolist()

interpreter = tflite.Interpreter(model\_path="facenet.tflite") interpreter.allocate\_tensors()

embeddings = {}

for name in os.listdir("dataset"): path = f"dataset/{name}/face.jpg" if os.path.exists(path):

img = cv2.imread(path)

embedding = get\_embedding(interpreter, img) embeddings[name] = embedding

with open("face\_embeddings.json", "w") as f: json.dump(embeddings, f)

print("[INFO] Face embeddings saved to face\_embeddings.json")

**Appendix B: Face Recognition System with RFID and Telegram**

**Integration.**

**B.1: Face Data Collection & Embedding Generation**

import cv2 import os

def collect\_images(name): save\_path = f'dataset/{name}'

os.makedirs(save\_path, exist\_ok=True) cap = cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

count = 0

print("Collecting images. Press 'q' to quit.") while count < 20:

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

break

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

for (x, y, w, h) in faces:

face = frame[y:y+h, x:x+w]

face\_path = os.path.join(save\_path, f"{count}.jpg")

cv2.imwrite(face\_path, face) count += 1

print(f"Saved {face\_path}") if count >= 20:

break cv2.imshow("Collecting", frame)

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

cap.release() cv2.destroyAllWindows()

name = input("Enter the person's name: ") collect\_images(name)

### B.2: Generate Embeddings for Each User

import cv2

import numpy as np import os

import tflite\_runtime.interpreter as tflite import pickle

interpreter = tflite.Interpreter(model\_path="facenet.tflite") interpreter.allocate\_tensors()

input\_details = interpreter.get\_input\_details() output\_details = interpreter.get\_output\_details() def preprocess\_face(face\_img):

face = cv2.resize(face\_img, (160, 160))

face = face.astype('float32')

mean, std = face.mean(), face.std() face = (face - mean) / std

return np.expand\_dims(face, axis=0) def get\_embedding(face\_img):

preprocessed = preprocess\_face(face\_img) interpreter.set\_tensor(input\_details[0]['index'], preprocessed) interpreter.invoke()

return interpreter.get\_tensor(output\_details[0]['index'])[0] embeddings = {}

for name in os.listdir('dataset'): folder = f'dataset/{name}'

face\_files = [f for f in os.listdir(folder) if f.endswith(".jpg")] if not face\_files:

continue embs = []

for file in face\_files:

img = cv2.imread(os.path.join(folder, file)) emb = get\_embedding(img) embs.append(emb)

mean\_embedding = np.mean(embs, axis=0) embeddings[name] = mean\_embedding

with open("embeddings.pkl", "wb") as f: pickle.dump(embeddings, f)

print("[INFO] Embeddings generated and saved.")

### Appendix C: Simple Face Recognition Example

def load\_known\_faces(): known\_embeddings = {}

for name in os.listdir("dataset"): path = f"dataset/{name}/face.jpg" if os.path.exists(path):

img = cv2.imread(path) embedding = get\_embedding(img)

known\_embeddings[name] = embedding return known\_embeddings

known\_faces = load\_known\_faces()

### Appendix D: RFID Authentication Code

import RPi.GPIO as GPIO

from mfrc522 import SimpleMFRC522 import json

import os

reader = SimpleMFRC522() json\_file = 'rfid\_map.json'

try:

print("Place your RFID tag near the reader...")

id, \_ = reader.read() # We're not using the text from the tag name = input("Enter name for this RFID: ")

print(f"ID: {id}")

print(f"Name: {name}")

# Load existing JSON data if it exists if os.path.exists(json\_file):

with open(json\_file, 'r') as file: data = json.load(file)

else:

data = []

# Append new record data.append({"id": id, "name": name})

# Save back to JSON file

with open(json\_file, 'w') as file:

json.dump(data, file, indent=4) # fixed this line print("Data saved to rfid\_map.json")

finally:

GPIO.cleanup()