**TRIBHUVAN UNIVERSITY**



**Faculty of the Institute of Science and Technology**

A Project Report On

“**Face Recognition Attendance System Using CNN**”

For

7th semester Mid Defense Report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Information Technology

**Submitted By:**

Jeeswan Bajracharya(29203/078)

Krisha Tuladhar(29208/078)

Suyesh Rimal(29229/078)

**Submitted to:**

Department of Computer Science and Information Technology

Asian College of Higher Studies (ACHS College)

Ekantakuna, Lalitpur

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project be prepared under my supervision by Jeeswan Bajracharya [29203], Krisha Tuladhar [29208], and Suyesh Rimal [29229] entitled “Face Recognition Attendance System” in partial fulfillment of the requirements for the Project work be processed for evaluation.

.................................

Supervisor,

Mr. Bidur Sapkota

# LETTER OF APPROVAL

This is to certify that this project is prepared by Jeeswan Bajracharya, Krisha Tuladhar, and Suyesh Rimal entitled “Face Recognition Attendance System” in partial fulfillment of the requirements for the degree of Bachelor in Computer Science and Information Technology. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| Signature of Supervisor  Mr. Bidur Sapkota  ................................. | Signature of Coordinator  Mr. Pranaya Nakarmi  ................................. |
| Signature of Internal Examiner  Mr. Janak k lal  ................................. | Signature of External Examiner  ................................. |

# ACKNOWLEDGEMENT

The completion of this report would not have been possible without the valuable support, guidance, and contributions from various individuals and organizations. We would like to express our heartfelt gratitude to the following:

We are grateful to ACHS College for providing us with the ICT infrastructure and a friendly environment that was essential to the success of our project.

We would also like to thank Mr. Bidur Sapkota, Supervisor, for providing the necessary feedback and guidance in the development of this project. He has been a great motivator and helper, encouraging us to keep moving ahead on the project.

Last but not least, our sincere thanks go to everyone who has helped, directly or indirectly, in bringing this project to completion.

With Respect,

Jeeswan Bajracharya

Krisha Tuladhar

Suyesh Rimal

# ABSTRACT

Taking attendance manually has long been a cumbersome, time-consuming, and error-prone task, particularly in environments with large groups such as classrooms, laboratories, or corporate offices. The Face Recognition Attendance System is a real-time solution developed to automate and streamline attendance management using facial recognition technology. The system captures live video input from a webcam, detects faces using OpenCV’s Haar Cascade Classifier, ensuring that multiple individuals can be identified in real time. Each detected face is then processed through a custom-trained Convolutional Neural Network (CNN) developed with TensorFlow and Keras. The backend, implemented with Flask, supports user management and live attendance tracking. Attendance data is stored in a database, ensuring easy integration with dashboards and reporting systems. Its fast, efficient, and non-intrusive design makes it ideal for educational institutions, laboratories, and corporate workplaces. By combining reliable facial recognition with an intuitive backend, the system reduces administrative workload, minimizes errors, eliminates the need for physical contact, and highlights how AI can improve operational efficiency while creating safer, more convenient, and more modern environments for students, staff, and employees.

***Keywords:*** *Face Recognition, Attendance Automation, OpenCV, Haar Cascade, CNN, TensorFlow, Keras, Flask*

# Table of Contents

[SUPERVISOR’S RECOMMENDATION i](#_Toc207030692)

[LETTER OF APPROVAL ii](#_Toc207030693)

[ACKNOWLEDGEMENT iii](#_Toc207030694)

[ABSTRACT iv](#_Toc207030695)

[Table of Contents v](#_Toc207030696)

[List of Figures: vii](#_Toc207030697)

[List of Tables: viii](#_Toc207030698)

[Chapter 1: Introduction 1](#_Toc207030699)

[1.1 Introduction 1](#_Toc207030700)

[1.2 Problem Statement 1](#_Toc207030701)

[1.3 Objectives 2](#_Toc207030702)

[1.4 Scope and Limitations 2](#_Toc207030703)

[1.4.1 Scope 2](#_Toc207030704)

[1.4.2 Limitation 2](#_Toc207030705)

[1.5 Development Methodology 3](#_Toc207030706)

[1.6 Report Organization 3](#_Toc207030707)

[Chapter 2: Background Study and Literature Review 5](#_Toc207030708)

[2.1 Background Study 5](#_Toc207030709)

[2.2 Literature Review 5](#_Toc207030710)

[Chapter 3: System Analysis 7](#_Toc207030711)

[3.1 System Analysis 7](#_Toc207030712)

[3.1.1 Requirement Analysis 7](#_Toc207030713)

[3.1.2 Feasibility Study 9](#_Toc207030714)

[3.1.3 Analysis 11](#_Toc207030715)

[Chapter 4: System Design 14](#_Toc207030716)

[4.1 Design 14](#_Toc207030717)

[4.1.1 Database Design 14](#_Toc207030718)

[4.1.2 Form and Report Design 16](#_Toc207030719)

[4.1.3 Interface and Dialog Design 16](#_Toc207030720)

[4.2 Algorithm Details 18](#_Toc207030721)

[Chapter 5: Implementation and Testing 26](#_Toc207030722)

[5.1 Implementation 26](#_Toc207030723)

[5.1.1 Tools Used 26](#_Toc207030724)

[5.1.2 Implementation Details of Modules 26](#_Toc207030725)

[5.2 Testing 28](#_Toc207030726)

[5.2.1 Test Cases of Unit Testing 28](#_Toc207030727)

[5.2.2 Test Cases of System Testing 29](#_Toc207030728)

[5.3 Result Analysis 30](#_Toc207030729)

[Chapter 6: Conclusion and Future Recommendations 33](#_Toc207030730)

[6.1 Conclusion 33](#_Toc207030731)

[6.2 Future Recommendation 33](#_Toc207030732)

[References 35](#_Toc207030733)

[Appendices 36](#_Toc207030734)

# List of Figures:

[Figure 1.5.1: Waterfall model 3](#_Toc207030783)

[Figure 3.1.1.1: Use Case Diagram 8](#_Toc207030784)

[Figure 3.1.2.1: Schedule Gantt Chart 10](#_Toc207030785)

[Figure 3.1.2.2: ER Diagram 11](#_Toc207030786)

[Figure 3.1.2.3: Level 0 DFD 12](#_Toc207030787)

[Figure 3.1.2.4: Level 1 DFD 13](#_Toc207030788)

[Figure 4.1.1.1: Database Schema 14](#_Toc207030789)

[Figure 4.1.2.1: Admin Login Form Interface 16](#_Toc207030790)

[Figure 4.1.2.2: Registration Page 17](#_Toc207030791)

[Figure 4.1.2.3: Facial Image Capture 17](#_Toc207030792)

[Figure 4.1.2.4: Face Detection and Recognition 18](#_Toc207030793)

[Figure 4.2.1: Haar-like features [8] 20](#_Toc207030794)

[Figure 4.6: Integral image Concept[10] 21](#_Toc207030795)

[Figure 4.9: CNN Architecture [9] 22](#_Toc207030796)

[Figure 5.3.1: Model Loss Over Epochs 30](#_Toc207030797)

[Figure 5.3.2: Model Accuracy Over Epochs 31](#_Toc207030798)

# List of Tables:

[Table 3.1.1.1:Use Case Description 8](#_Toc207030239)

[Table 4.1.1.1: Database Table 15](#_Toc207030240)

[Table 5.2.1.1: Unit Testing 28](#_Toc207030241)

[Table 5.2.2.1: System Testing 29](#_Toc207030242)

[Table 5.3.1: Model Summary 31](#_Toc207030243)

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

Monitoring student attendance in schools and colleges can be a slow process. Using paper sheets or roll calling are examples of traditional methods that take up class time and can result in issues like fake attendance. With busy schedules and shorter class hours, there’s a need for a better, faster way to handle this process. Furthermore, keeping accurate records is made more difficult by manual attendance, which can lead to issues for administrators and students alike.

The web-based Face Recognition Attendance System simplifies and automates attendance. Students just need to register their faces once through a simple interface. After that, students need to stand in front of the camera and start the process themselves. After identifying their face, the system automatically logs their attendance. A CNN-based face recognition system with Haar Cascade Classifier for face detection, Flask for backend, and OpenCV for video processing are some of the tools used in its construction.

Additionally, accessing and reviewing attendance history is made simple by safely storing all records in a database. This system saves teachers a great deal of time, lowers error, and stops cheating. It’s a modern and hygienic method of taking attendance because it does not require physical contact.

The system can handle large classes quickly and reliably, helping schools run more smoothly and letting teachers focus on teaching instead of paperwork. CNN-based feature extraction guarantees dependable recognition accuracy, while a Haar Cascade Classifier-based method is employed for precise face detection. This approach offers a quick, cheap, and dependable solution for contemporary classrooms by lowering errors and increasing proxy attendance.

## 1.2 Problem Statement

Manual attendance systems in educational institutions often involve students physically signing attendance sheets or verbally responding to roll calls. This process is time-consuming and disrupts the flow of class, reducing valuable teaching time. Manual attendance is prone to errors and fake practices, such as proxy attendance, where one student signs in on behalf of another, compromising the integrity of attendance records.

These challenges motivate the development of an automated attendance system using face recognition technology. This system aims to eliminate the burden on both students and lecturers by automating attendance marking. It ensures accuracy and reliability by verifying the presence of each student based on their unique facial features, reducing the possibility of fake attendance entries.

Furthermore, lecturers are relieved from the slow task of counting and verifying attendance repeatedly, enabling them to focus fully on teaching and classroom management. The automated system improves data integrity, simplifies attendance tracking, and reduces administrative workload.

## 1.3 Objectives

This system records student attendance using real-time face recognition, offering a smooth and precise web-based solution.

The major objectives of the Face Recognition Attendance System are as follows:

* To take attendance using a web-based face recognition system.
* To provide educators with an interface for monitoring and managing attendance records of students in real time.

## 1.4 Scope and Limitations

### 1.4.1 Scope

* Operates the attendance marking process using face recognition.
* Provides a web interface for student registration and attendance monitoring.
* Stores attendance logs in a database for further analysis and reporting.
* Can be deployed in classrooms, seminars, or corporate training environments.

### 1.4.2 Limitation

* The system needs a clear camera view, so things like poor lighting or a low-quality camera can mess with detection.
* Haar Cascade may not handle extreme angles as accurately as advanced detectors.
* Requires students to stand still in front of the camera for accurate recognition.
* Limited to one-to-one face matching at a time (multi-face recognition in parallel is not implemented in the current scope).

### 1.5 Development Methodology

The development of the Face Recognition Attendance System follows the Waterfall Model, which is a step-by-step approach. Each stage is completed before moving on to the next one. This model was suitable for our project because its requirements were well defined from the beginning, which reduced ambiguity and made planning straightforward.

In the requirement stage, we listed out the core features, which are login for admin, face recognition for marking attendance, and storing records. Then, in the developmental stage, we built a face recognition model, set up a backend to handle data and attendance logic, and created the frontend for a smooth user experience.

Once the system was built, the testing stage started, where we checked both the accuracy and functionality of the system. After confirming that the system is working a intended, we deployed it as a web application so that admins and students could use it in real time. At last, for the maintenance stage, we plan to keep improving the model’s accuracy, fix bugs, and add small features as needed.

The Waterfall model works effectively for this project because it gives a clear path to follow. Instead of jumping between stages, we could finish one stage properly and then move to the next. This helped us stay organized, learn each step of the process, and end up with a reliable attendance system that actually matches the requirements we set at the beginning.

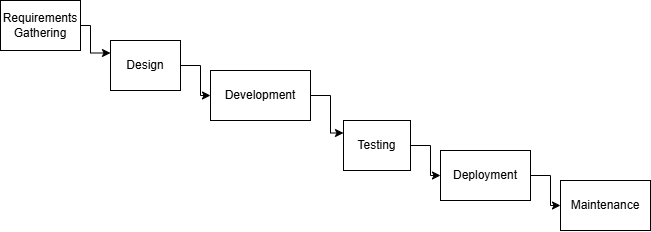


Figure 1.5.1: Waterfall model of Face Recognition Attendance System

## 1.6 Report Organization

The report is structured as follows:

**Chapter 1: Introduction**

This chapter gives a clear overview of the project. It defines the core problem, time-consuming manual attendance, and outlines the project's objectives to address these issues. The scope, limitations, and methodology are also given during development.

**Chapter 2: Background Study and Literature Review**

This chapter dives into the key theories that support the project, such as computer vision, facial recognition algorithms, and machine learning models. It includes the technologies used, including Flask, OpenCV, TensorFlow/Keras, and explores similar existing systems to show why this project is useful. Comparisons help identify gaps in existing solutions and establish how this system improves upon them.

**Chapter 3: System Analysis**

This chapter includes a detailed requirement analysis, covering both functional and non-functional aspects needed for the system to operate efficiently. A feasibility study is conducted to assess the technical, operational, and economic viability of the project. System modeling tools like Data Flow Diagrams (DFDs) and ER models are used to predict data movement and system behavior.

**Chapter 4: System Design**

Here, the system's architecture is laid out, showing how different components of face detection, recognition, and attendance logging interact with each other. The chapter also presents the user interface design and form design.

**Chapter 5: Implementation and Testing**

This chapter breaks down the actual coding and integration of different modules, including the face detection, face recognition, user management, and attendance logging features. It describes the tools and frameworks used, such as Python, Flask, OpenCV, and a database. Comprehensive testing, including unit and system testing, is discussed. The test cases, expected outcomes, and results are also mentioned.

**Chapter 6: Conclusion and Future Recommendations**

The final chapter provides a summary of the work accomplished throughout the development. It also highlights the key lessons learned during the project journey.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

The Face Recognition Attendance System is a real-time web-based application designed to operate the attendance process using facial recognition. It uses a webcam to detect and recognize faces, then logs attendance securely in a database, reducing human error and eliminating proxy attendance.

The system is built using Flask for the backend, providing a simple and lightweight web framework. For live video capture and face detection, it integrates OpenCV, using Haar Cascade classifiers to identify faces in real-time. Once a face is detected, a convolutional neural network built with TensorFlow and Keras recognizes and differentiates individuals based on their unique facial features. Attendance data is saved in the database for easy and reliable access in the future.

The system comes with a variety of important features designed to make attendance tracking simple and efficient. It enables face-based attendance, where users are identified and logged automatically through webcam input. Admin users can manage the system by registering, unregistering, or deleting users via a dashboard. Recognized individuals are displayed in real time on the video feed with their name and ID. For reporting and analytics, administrators can search attendance records by date or user ID.

The backend database is divided into three main components. The Student table stores all the details of students, including their registration status, so the system easily differentiates between registered and unregistered individuals. The Attendance table is responsible for recording each instance a student is recognized, capturing their student ID, name, section, and the corresponding timestamp to maintain accurate attendance logs. Lastly, the Admin table holds the IDs and names of system administrators, ensuring secure access and management of the platform. This structure supports effective data organization and facilitates reliable attendance tracking and user management.

## 2.2 Literature Review

Various studies have demonstrated the effectiveness of face recognition-based attendance systems in addressing the limitations of traditional methods. A system utilizing Convolutional Neural Network (CNN) cascades for both face detection and recognition achieved over 95% accuracy, even with small datasets, making it ideal for real-time classroom environments [1].

Another approach implemented using Python, OpenCV, and LBPH (Local Binary Pattern Histogram) features proved not only cost-effective but also highly resilient to proxy attendance, providing a hygienic and fully contactless alternative to conventional methods [2].

Conventional biometric systems, such as fingerprint scanners and QR codes, although effective, require physical interaction or additional hardware, making them less scalable in modern classrooms. This has shifted the focus to fully automated face recognition systems, which rely on CNNs for accurate identification. In one such system, Haar Cascade classifiers were used for detecting faces, followed by LBPH for recognition, with attendance data stored in the cloud for real-time access [3].

A more advanced solution leveraged deep CNN-based image processing combined with data augmentation techniques to improve recognition accuracy. Despite operating on a small dataset, the system achieved an impressive 95.02% accuracy, showing real-world applicability and seamless integration potential [4].

In the online learning context, another framework recognized multiple faces from lecture video frames. By using VGG16, CNN, and Haar features, the system could accurately identify faces with different poses and expressions, although further optimization is required for real-time deployment [5].

One hybrid approach combined HOG (Histogram of Oriented Gradients) for detection, CNN for feature extraction, and SVM (Support Vector Machine) for classification. This system reported 99.75% accuracy, successfully identifying and counting students in real-time imagery, further proving the viability of AI-based attendance over manual tracking [6].

Finally, Bavaskar [7] presented a Haar Cascade-based real-time face recognition attendance system that handled variations in lighting, facial expressions, and accessories like glasses. This system also tackled challenges like database integration, performance optimization, and privacy protection through data encryption, showcasing a well-rounded, production-ready solution.

# CHAPTER 3: SYSTEM ANALYSIS

## 3.1 System Analysis

The system analysis phase focuses on understanding and specifying the functional and non-functional requirements of the Face Recognition Attendance System. It also evaluates feasibility aspects and models the system through diagrams to visualize its processes.

### 3.1.1 Requirement Analysis

Requirement analysis focuses on determining the needs of a system. The purpose of this analysis is to ensure that the software or system meets the desired goals. This system enables users to register their faces, mark their attendance, and view the attendance log as well.

1. **Functional Requirement**

Functional requirements define the core functions the system must perform to meet its purpose.

* User Face Registration: Users should be able to register their face data through the system, which will be used for future identification and attendance tracking.
* Real-Time Face Recognition: The system must detect and recognize registered users in real-time using a live camera feed.
* View Attendance Records: Administrators should be able to view daily attendance records.

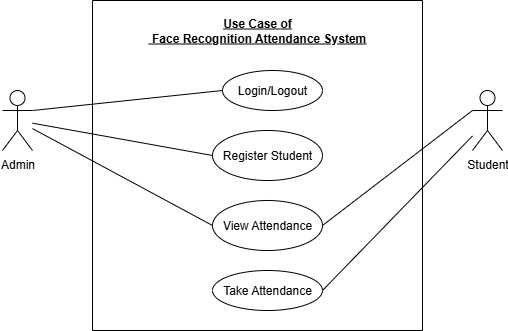


Figure 3.1.1.1: Use Case Diagram of Face Recognition Attendance System

Table 3.1.1.1:Use Case Description

|  |  |  |
| --- | --- | --- |
| **Actor** | **Use Case** | **Description** |
| Admin | Log In | Authenticates admin access to system functionalities. |
| Admin | Register user | Allows the admin to input student details and capture face images for training. |
| Admin | View Attendance | Lets the admin access and filter attendance records by date or student. |
| Student | Mark Attendance | The system recognizes the student’s face and logs their attendance. |
| Student | View Attendance | Lets the students access attendance records by date or student. |

1. **Non-functional Requirement**

Non-functional requirements define the quality and performance aspects of the system, ensuring it works efficiently and reliably.

* **Performance:** The system detects and recognizes faces in real-time using Haar Cascade for detection and a CNN model for recognition, ensuring both speed and accuracy.
* **Security:** The system prevents duplicate attendance through session-based control and secures facial data in the database with admin-only registration access.
* **Usability:** A simple Flask-based web interface where administrators manage attendance and students interact with the system.
* **Portability:** The system runs on Windows PCs with webcams and uses lightweight tools like OpenCV and Keras, requiring no special hardware.

### 3.1.2 Feasibility Study

The purpose of the feasibility study is to consider all the aspects of the proposed project and determine its success. In this chapter, we determine if we have the right technology, financial resources, and time required to complete the project and whether the project will be completed in time.

* **Technical**

The system is technically feasible, using accessible tools like Python, Flask, OpenCV, TensorFlow, and Keras. For face detection, the system utilizes Haar Cascade Classifier, and for face recognition, it employs a Convolutional Neural Network (CNN) model trained with TensorFlow and Keras. Registered faces are converted into numerical feature vectors. Then the system compares these with real-time webcam input to identify individuals. The setup requires only a standard computer and webcam, making it straightforward to implement in educational or organizational settings.

* **Operational**

Our system uses a Flask web interface that lets users register, track attendance, and manage data easily. This ensures that the system is practical and operationally feasible for daily use by both administrators and users.

* **Economic**

The system uses open-source tools and libraries, reducing development costs significantly. It runs on standard computers without the need for high-end hardware or paid licenses. This cost-effective approach makes the solution affordable for schools, colleges, and organizations, making the system economically feasible.

To validate this further, a finantial analysis was conducted

* **Schedule**

The estimated duration of the project is around 2 to 3 months. The project will take about 2 to 3 months to finish. We have planned all the work and have the resources needed. By checking progress often and fixing problems quickly, we can complete the project on time. Therefore, the project is feasible in terms of schedule.

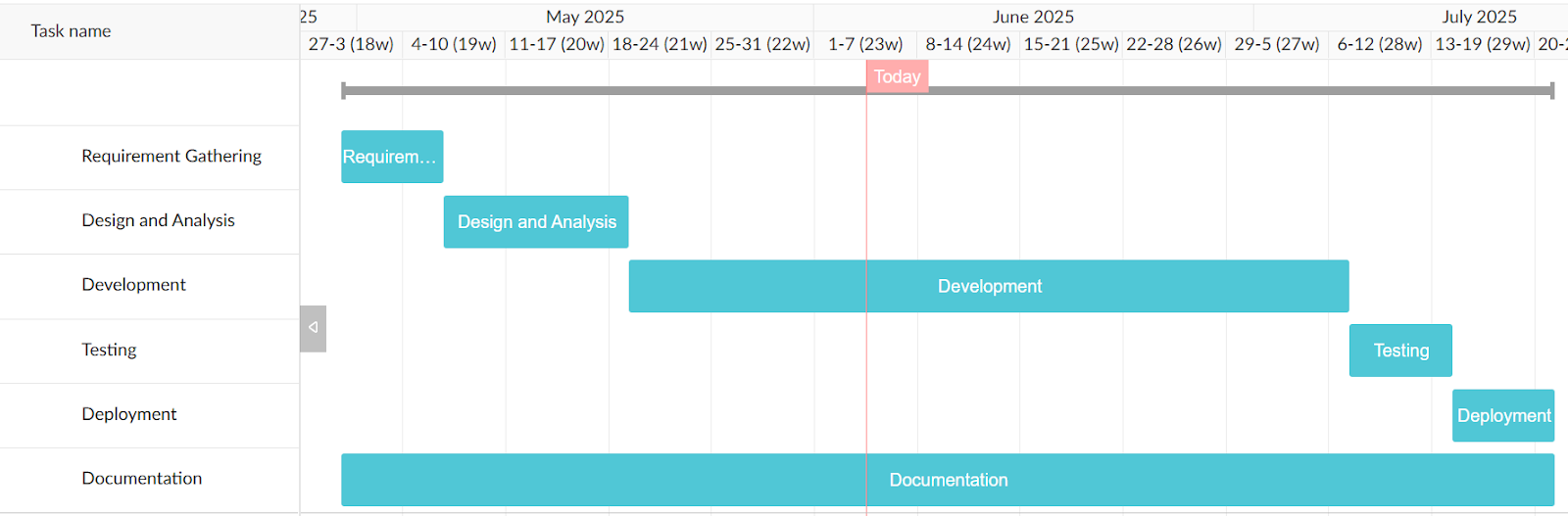


Figure 3.1.2.1: Schedule Gantt Chart

The Gantt chart illustrates a system development plan set to finish in 3.5 months, within the target of 3.2 months.

Requirement Gathering: 7 days

Design and Analysis: 13 days (includes database design of 3 days)

Development: 49 days

Testing: 7 days

Deployment: 7 days

Documentation: Ongoing throughout all phases

The timeline is well-structured and achievable.

### 3.1.3 Analysis

We chose a structured analysis approach because the system follows a clear, step-by-step process: detecting faces, recognizing them, and then marking attendance.

To better understand how the system works and how the data flows, we used:

Entity Relationship Diagram (ERD):  
This helped us plan the database structure by showing how student data and attendance records are connected.

The following ER diagram illustrates this process:

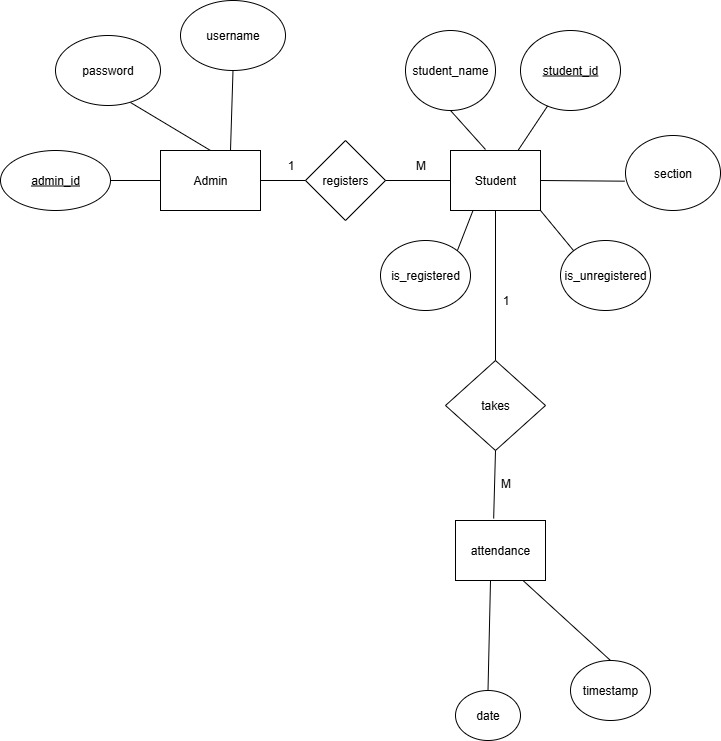


Figure 3.1.2.2: ER Diagram

This figure shows the data structure of the system using an Entity-Relationship Diagram. It highlights entities like Student and Attendance, along with their key attributes such as student ID, name, and date. The relationship between these entities enables the tracking of each student's attendance.

**Data Flow Diagram (DFD):**  
During the analysis phase, we use both Level 0 and Level 1 Data Flow Diagrams (DFDs) to better understand how data flows through the system. The Level 0 DFD provides a high-level overview of how users interact with the system. The Level 1 DFD breaks this down further into detailed processes like image capture, face detection, recognition, and attendance logging. These diagrams help us define the system’s structure and internal data movement clearly before proceeding to the design phase.

The following DFD diagrams illustrate these processes:

**Level 0 DFD:**

The Level 0 Data Flow Diagram (DFD) for the Face Recognition Attendance System presents a simplified, top-level view of the system. At its core is a central process that handles user registration, face recognition, and attendance marking. The user interacts with the system by registering their face and later by having their attendance marked through facial recognition. Data flows between users and the system, and attendance information is stored in a dedicated log file. This diagram effectively illustrates the system’s primary functions and its interaction with external entities.

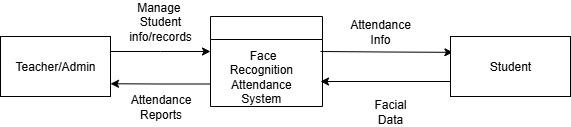


Figure 3.1.2.3: Level 0 DFD

**Level 1 DFD**

The Level 1 Data Flow Diagram (DFD) provides a more detailed view of the system’s internal processes. When an admin logs into the system (Process 1.0 – Login System), their credentials are verified using the stored admin data. Once authenticated, the admin can register a student by providing personal and facial data (Process 1.1 – Student Registration), which is stored in the student data repository (D1). After registration, the system automatically triggers Process 1.2 – Train Recognition Model, where the captured facial data is used to update the face recognition model. Students can then scan their faces through Process 1.3 – Student Attendance, which uses the trained model to recognize the student and record their attendance. The recognized data, including timestamps and student identity, is stored in the attendance record (D2). This diagram clearly illustrates the flow of data and how each component interacts to support automated attendance tracking.

A diagram of a student data system

AI-generated content may be incorrect.

Figure 3.1.2.4: Level 1 DFD

# CHAPTER 4: SYSTEM DESIGN

## 4.1 Design

The system design phase for the Face Recognition Attendance System outlines the structure and components necessary to develop the complete solution. It includes the database design, form and report design and interface and dialog design. This chapter explains how the Face Recognition Attendance System works, from capturing face images to recognizing students and recording attendance.

### 4.1.1 Database Design

The system uses a traditional relational database schema in design and the implementation follows normalized structures. The database schema is derived from the ER diagram and ensures normalization up to Third Normal Form (3NF).



Figure 4.1.1.1: Database Schema

**Table Structure:**

The following files represent the system's data storage:

Table 4.1.1.1: Database Table

|  |  |
| --- | --- |
| Table | Purpose |
| Admin | Stores details of system administrators (login credentials) |
| Student | Stores details of all students registered in the system |
| Attendance | Stores attendance logs of students for each date |

**ER Diagram to Logical Relations**

From the ER diagram, the entities are transformed into the following relations:

Admin(admin\_id, username, password)

Student(student\_id,student\_name, section, is\_registered, is\_unregistered, admin\_id)

Attendance(attendance\_id,student\_id,date,timestamp)

Relationships:

Registers/Manages → One Admin manages many Students (Admin.admin\_id → Student.admin\_id).

Takes → One Student can have many Attendance records (Student.student\_id → Attendance.student\_id).

Normalization: All relations follow relational normalization principles up to Third Normal Form (3NF):

1NF: Each field contains atomic data (e.g., student\_name, section, timestamp).

2NF: All non-key attributes depend on the full primary key. Example: in Attendance, date and timestamp depend only on attendance\_id.

3NF: No transitive dependencies exist. Attributes like username and password depend only on admin\_id, not through another field.

Final Normalized Relations (3NF):

Admin(admin\_id PK, username, password)

Student(student\_id PK, student\_name, section, is\_registered, is\_unregistered, admin\_id FK)

Attendance(attendance\_id PK, student\_id FK, date, timestamp)

This structure ensures minimal redundancy, maintains data integrity, and supports efficient attendance tracking.

### 4.1.2 Form and Report Design

Admin Login Form:  
The system features an admin login form where administrators enter their username and password to access the system securely. This form includes input validation to prevent empty or invalid credentials.

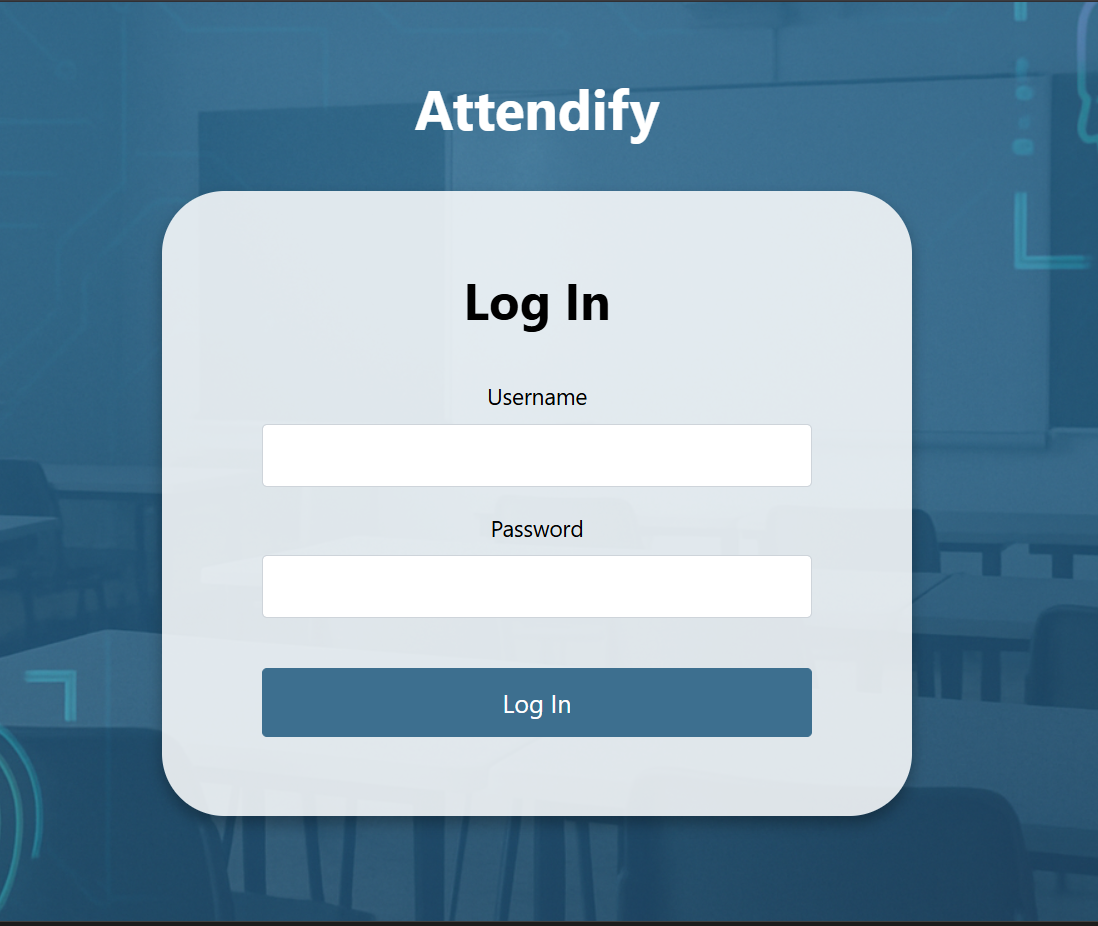


Figure 4.1.2.1: Admin Login Form Interface

### 4.1.3 Interface and Dialog Design

The Face Recognition Attendance System provides a web-based Flask interface to enable:

Face Registration Page:  
 Allows administrators or users to upload face images and register their details.

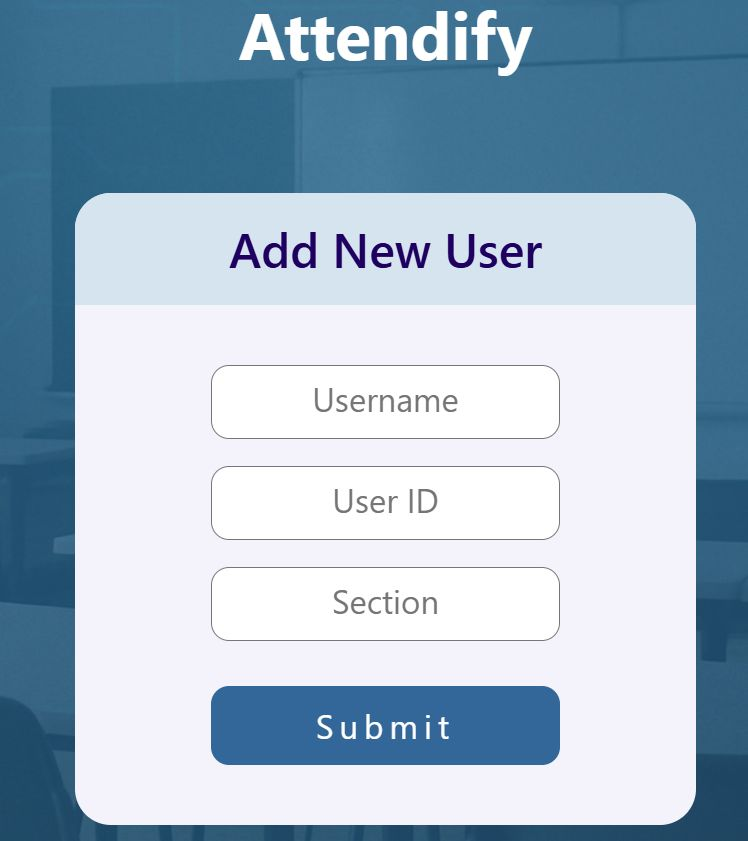


Figure 4.1.2.2: Registration Page

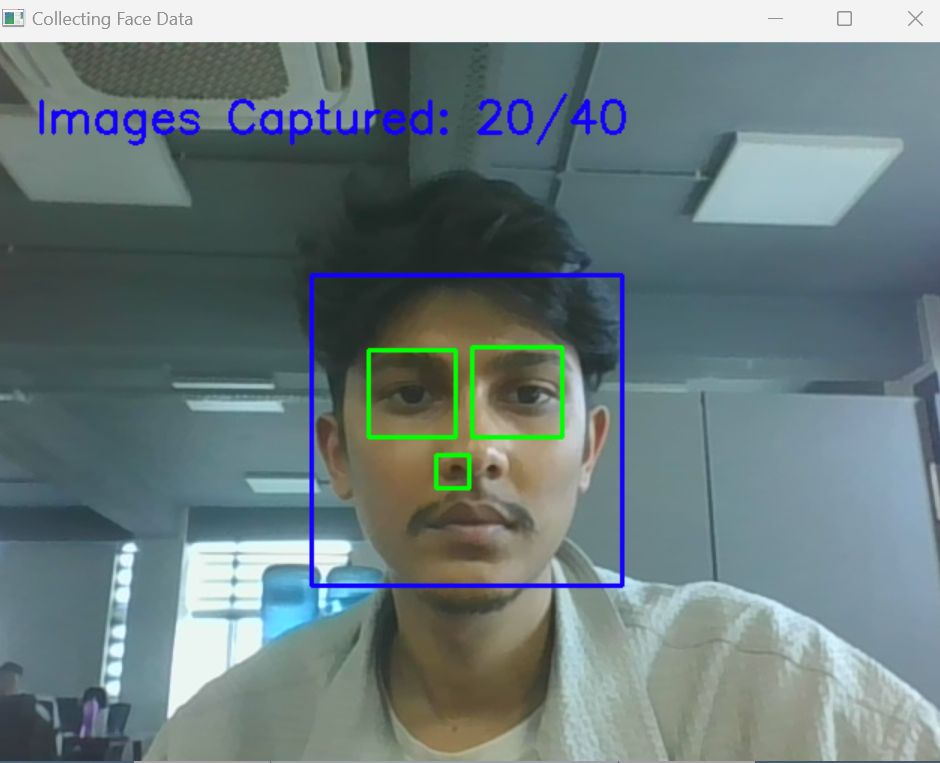


Figure 4.1.2.3: Facial Image Capture

Attendance Marking Page:  
 Activates the webcam, detects faces in real-time, recognizes registered users, and marks attendance automatically.

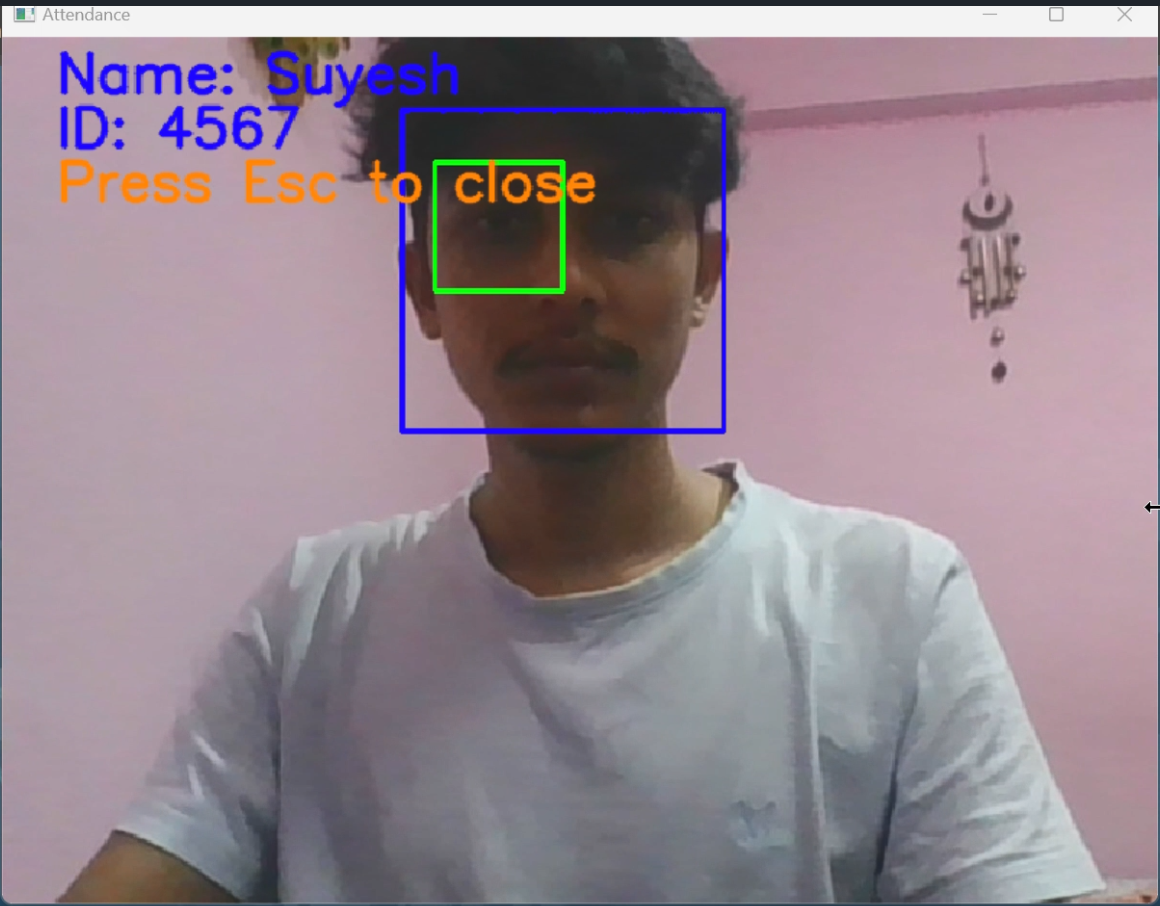


Figure 4.1.2.4: Face Detection and Recognition

Attendance Records Page: Allows administrators to view, filter, and export attendance logs.

The user interface is designed to be simple and user-friendly, so users with minimal technical experience can operate the system easily.

## 4.2 Algorithm Details

In this project, a two-stage pipeline was implemented for face recognition. The first stage uses a Haar Cascade Classifier for detecting faces within images, and the second stage uses a Convolutional Neural Network (CNN) for classifying the detected faces into respective identities. Here, both components are explained in detail:

**Face Detection using Haar Cascade Classifier**

The Haarcascade Frontal Face Detection Algorithm is a machine learning-based object detection method that identifies human faces from digital images or video streams. It is based on the Viola–Jones object detection framework, introduced in 2001 by Paul Viola and Michael Jones, which is known for its high detection accuracy and real-time performance.

The process consists of four primary stages:

**Haar Feature Selection**

Face detection using HaarCascade begins with extracting features known as Haar-like features. These features resemble convolution filters used in image processing and are sensitive to patterns such as edges and lines. A few notable observations include:

Eye regions tend to be darker than the upper cheeks.

The nose bridge is brighter than the eyes.

The spatial arrangement of facial elements is relatively consistent across individuals.

Haar features are defined as combinations of black and white rectangles. For each feature, the algorithm subtracts the sum of pixels under the black region from the white region to quantify contrast and detect facial patterns.

In the Face Recognition Attendance System, these features allow the program to quickly find areas that look like faces in images or live webcam frames. Only these areas are then passed on for recognition.

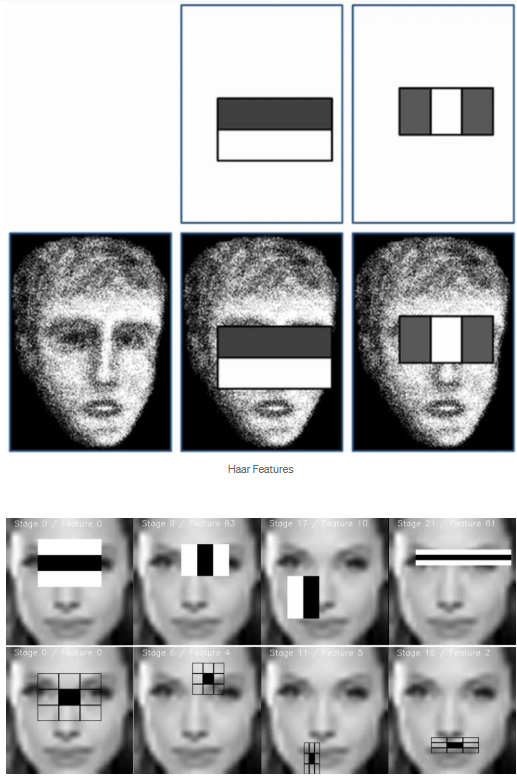


Figure 4.2.1: Haar-like features [8]

**Integral Image**

To accelerate the computation of Haar-like features, the algorithm uses an Integral Image (also called a Summed Area Table). This structure allows for fast and efficient calculation of the sum of pixel values within any rectangular region in the image.

Definition:

Let i(x,y) represent the original image pixel value at position (x,y), and ii(x,y) be the integral image at the same point.

The integral image at point (x,y) is defined as:

This means that each point in the integral image contains the sum of all pixels above and to the left of it (inclusive) in the original image.

Efficient Computation (using recurrence):

To compute the integral image efficiently in a single pass, we use the following recurrence:

Where:

s(x,y) is the cumulative row sum at position (x,y)(x, y)(x,y)

Base conditions: s(x,−1)=0, ii(−1,y)=0

In practice, this lets the Face Recognition Attendance System calculate Haar features very quickly, so it can detect faces in real time using a webcam.

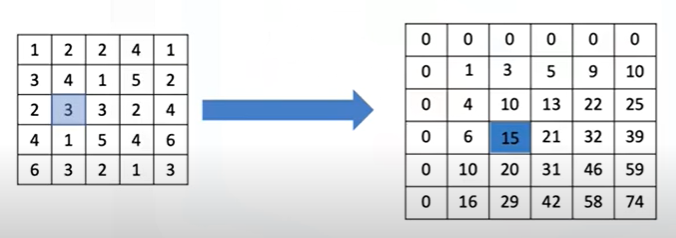


Figure 4.6: Integral image Concept[10]

**AdaBoost Training**

Given that a 24×24 pixel detection window can generate over 160,000 Haar features, evaluating all of them would be computationally infeasible. Therefore, the algorithm employs AdaBoost, a machine learning technique that selects the most informative features and combines them into a strong classifier.

The Face Recognition Attendance System uses a pre-trained Haar Cascade file (haarcascade\_frontalface\_default.xml) that already contains these selected features. This means the system can detect faces efficiently without having to train a model from scratch.

**Cascading Classifier**

The final phase involves constructing a cascade of classifiers, which filter out negative windows quickly and apply more complex computations only to likely face regions.

Each stage in the cascade is a strong classifier trained to detect faces.

If a region passes the first stage, it proceeds to the next; otherwise, it is rejected.

This approach significantly improves speed and efficiency, especially when processing real-time video.

The cascade ensures that only regions most likely to contain faces are processed through all classifier stages, making the detection system both accurate and performant.

**CNN-Based Face Recognition Architecture**

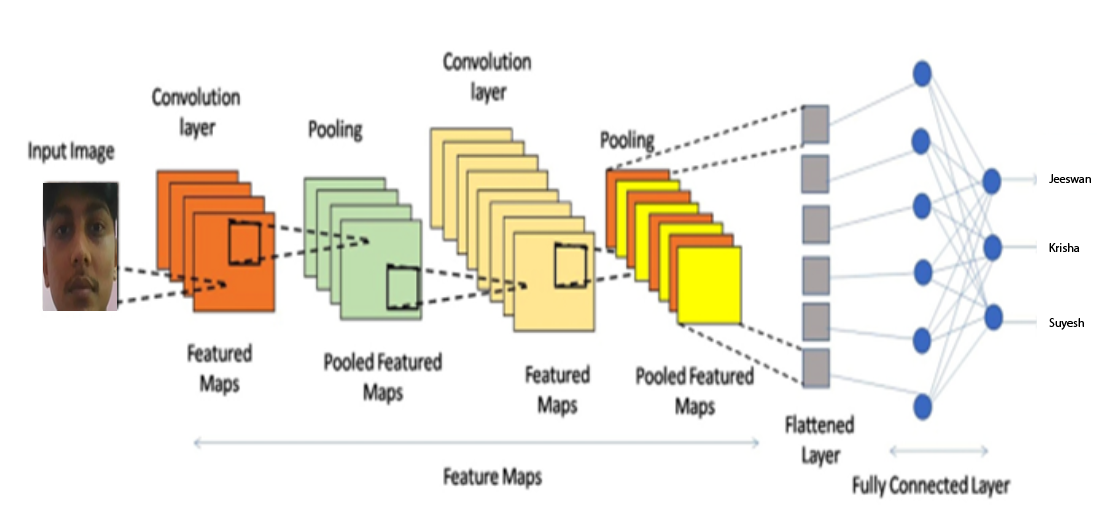


Figure 4.9: CNN Architecture [9]

In this project, we developed a deep learning-based facial recognition system from scratch using a Convolutional Neural Network (CNN). The architecture was carefully designed to extract and learn visual features from input images, with various components serving specific roles in learning, generalization, and prediction. Below is a detailed explanation of each layer and concept integrated into the model:

**Input Layer**

The input layer is the first point of contact between the raw image data and the model. For this system, color face images were used, represented by three channels—Red, Green, and Blue (RGB). Each image was resized to 224×224×3 dimensions to match the model’s expected input shape. This layer passes the image tensor forward without applying any transformation.

**Convolutional Layers (Conv2D)**

Convolutional layers are the core building blocks of CNNs. They apply a series of learnable filters (kernels) that slide over the image, capturing spatial hierarchies and local patterns such as edges, textures, and facial features (eyes, nose, mouth).

Each Conv2D layer in the architecture applies 3×3 filters with 'same' padding to maintain the spatial resolution. The number of filters increases progressively (16 → 32 → 64) to allow the network to learn more complex features at deeper levels.

Mathematically, the convolution operation is defined as:

where I is the input image and K is the filter.

**Activation Function (ReLU)**

After each convolutional operation, the Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity into the model. This enables the network to learn complex and non-linear relationships between features.

The ReLU function is defined as:

It replaces all negative values in the feature maps with zero, maintaining

only positive activations.

**Dropout Layers**

Dropout is a regularization technique used to prevent overfitting. During training, dropout randomly disables a fraction of neurons in a layer, forcing the network to learn more robust features.

In this model:

Dropout is applied after each convolutional block (rates: 0.25, 0.25, 0.3) and after the first dense layer (rate: 0.5).

This ensures that the network does not rely on any single neuron and improves generalization on unseen data.

**Pooling Layers (MaxPooling2D)**

Pooling layers reduce the spatial dimensions of the feature maps, lowering computational complexity and extracting dominant features. This model uses 2×2 Max Pooling, which retains only the maximum value in each window:

Where

F is the feature map. Pooling makes the model translation-invariant to small shifts in the input.

Flatten Layer

After the final convolution and pooling operations, the multi-dimensional feature maps are flattened into a one-dimensional vector. This flattened representation serves as the input to the fully connected (dense) layers and captures the learned spatial features in a linear format.

**Dense Layers and Softmax Output**

A dense (fully connected) layer with 64 units and ReLU activation processes the flattened features to combine them into a more abstract representation. The final output layer is a Dense layer with softmax activation, producing probability distributions over the number of classes (equal to train\_data.num\_classes).

The softmax function is defined as:

Where:

zi​ is the input to the softmax unit.

K is the number of classes,

σ(zi)is the predicted probability for class i.

**Loss Function**

The model uses Categorical Crossentropy with label smoothing (0.1) to compute the error between predicted and true class labels. Label smoothing helps prevent the model from becoming overconfident by slightly reducing the target probabilities, which leads to better generalization.

**Optimizer**

The Adam optimizer with a learning rate of 1e-4 was used for training. Adam combines the advantages of AdaGrad and RMSprop and is well-suited for training deep neural networks with sparse gradients and noisy data.

Metrics

Model performance was monitored using accuracy, which calculates the proportion of correctly classified samples over the total number of predictions:

**Regularization Techniques**

To combat overfitting and improve model generalization, the following regularization methods were employed:

Dropout at multiple layers (0.25 to 0.5)

Label smoothing in the loss function

**Callbacks**

Training was managed using the EarlyStopping callback:

It monitored validation loss and stopped training if no improvement was seen for 6 consecutive epochs.

restore\_best\_weights=True ensured the model reverted to the best-performing weights.

# CHAPTER 5: IMPLEMENTATION AND TESTING

## 5.1 Implementation

The implementation phase focuses on developing the Face Recognition Attendance System using the planned designs and methodologies. The project is developed as a web application that uses real-time face recognition technology to identify students automatically.

### 5.1.1 Tools Used

Programming Languages: Python(for backend logic and face recognition algorithms), HTML/CSS/JavaScript (for frontend interface)

Frameworks: Flask (web backend framework)

Libraries: OpenCV (for video capture and image processing), TensorFlow/Keras (for CNN-based face recognition model)

Others: Haar Cascade classifier for face detection

### 5.1.2 Implementation Details of Modules

The Face Recognition Attendance System is organized into several core modules, each responsible for specific functionalities related to attendance management:

User Module  
Manages user registration, face data capture, and basic student information.

registeruser(): Admin registers a student by name, roll, and section.

unregisteruser(): Moves users from the unregistered to registered state by capturing their face images.

deleteregistereduser(): Deletes registered face data and user info from the system (face folder and database).

deleteunregistereduser(): Deletes unregistered face data and user info from the system (face folder and database).

register\_user\_list(): Displays a list of all registered users with attendance info.

unregistered\_user\_list(): Displays users who haven't been registered yet.

**Face Detection Module**  
Detects faces in each frame captured from the live webcam feed.

extract\_faces\_and\_eyes(img): Uses Haar Cascade or similar model to detect faces and eyes , returning the bounding box coordinates (x, y, w, h).

Code inside attendancebtn(): Calls extract\_faces\_and\_eyes() on each video frame using OpenCV cv2.VideoCapture().

**Face Recognition Module**  
Encodes facial features into embeddings and compares them with known users.

load\_data(): Loads images and labels from the static/faces/ directory for training.

create\_model(): Creates the CNN architecture used for face recognition:

Conv2D → MaxPooling → Flatten → Dense → Softmax

train\_model() (app.py): Calls functions from the notebook to:

Load face data

Encode labels

Train the CNN model

Save as face\_recognition\_model.h5

identify\_face(face\_img) (app.py): Loads the model, predicts the class of the incoming face, and decodes it into format name$roll$section.

Attendance Module  
Manages attendance marking and logs.

attendancebtn(): Core function to start webcam, capture frames, detect faces, identify them, and call add\_attendance().

add\_attendance(name): Appends attendance record with name, roll, section, date, and time to database.

extract\_attendance(): Reads attendance logs from file (or DB) and returns data to be shown in the template.

totalreg(): Counts how many registered users are available (folders inside static/faces).

User Interface Module  
Facilitates interaction between the user and the system via web templates.

LogInForm.html: Login form (handled in login route).

RegisterUserList.html, UnregisterUserList.html: Displays lists of registered and unregistered users.

Attendance.html: Real-time attendance view with OpenCV camera preview and results.

AttendanceList.html: Admin view for attendance record listing.

## 5.2 Testing

### 5.2.1 Test Cases of Unit Testing

Table 5.2.1.1: Unit Testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Module | Function Name | Test Case Description | Expected Outcome | Actual Outcome | Status |
| TC01 | User Module | registeruser() | Register a student via the admin interface | Student details saved, face folder created | As expected | Pass |
| TC02 | Face Detection | extract\_faces() | Detect faces from camera input | Face bounding box coordinates returned | As expected | Pass |
| TC03 | Face Recognition | train\_model() | Train the CNN model on all user faces | face\_recognition\_model.h5 created | As expected | Pass |
| TC04 | Face Recognition | identify\_face() | Identify a student from a face image | Correct name-roll-section extracted | As expected | Pass |
| TC05 | Attendance Module | add\_attendance() | Mark attendance after recognition | Entry written to the database. | As  expected | Pass |
| TC06 | Attendance Module | extract\_attendance() | Read the attendance log for display | Returns complete attendance data | As  expected | Pass |
| TC07 | UI Module | attendancebtn() | Trigger attendance via webcam | Opens camera, displays real-time results | As expected | Pass |
| TC08 | Face Recognition | create\_model() | Create a CNN model architecture | Model layers are correctly compiled | As expected | Pass |
| TC09 | User Module | unregisteruser() | Capture face images for a new user | 50 face images saved in the user folder | As expected | Pass |

### 5.2.2 Test Cases of System Testing

Table 5.2.2.1: System Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Scenario | Description | Expected Result | Actual Result (if tested) | Status |
| Single User  Recognition | Test the system's ability to recognize and log attendance  of a student in a single session | The recognized students are accurately logged in the database. | As expected | Pass |
| Error Handling | Show behavior when an unregistered face or no face is shown | Display appropriate message, no crash | As expected | Pass |
| Duplicate Attendance | Ensure a student can't mark attendance multiple times in one session | Only the first match is logged; subsequent ones are ignored | As expected | Pass |

## 5.3 Result Analysis

The facial recognition models were evaluated using both deep learning and traditional machine learning approaches, comparing performance over epochs and under challenging conditions like reduced data and noise. The following figures provide insights into model training dynamics and final classification performance.

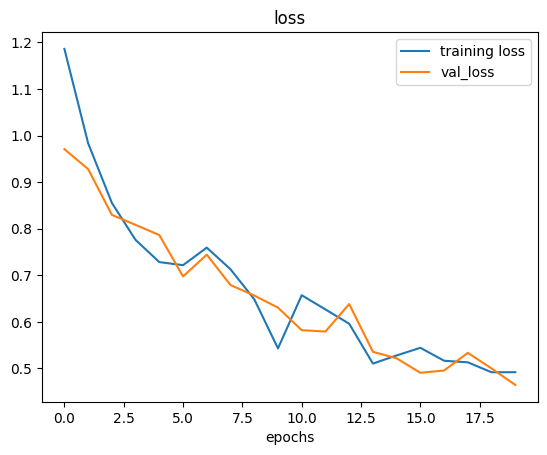


Figure 5.3.1: Model Loss Over Epochs

This plot shows the training and validation loss over 20 epochs. A clear downward trend is observed in both curves, indicating that the model is effectively learning and reducing error during both training and validation.

The validation loss closely follows the training loss, without significant divergence, suggesting minimal overfitting.

Loss values stabilize toward the later epochs, confirming that the model converged.

This indicates good generalization on unseen validation data, a critical indicator of model robustness.

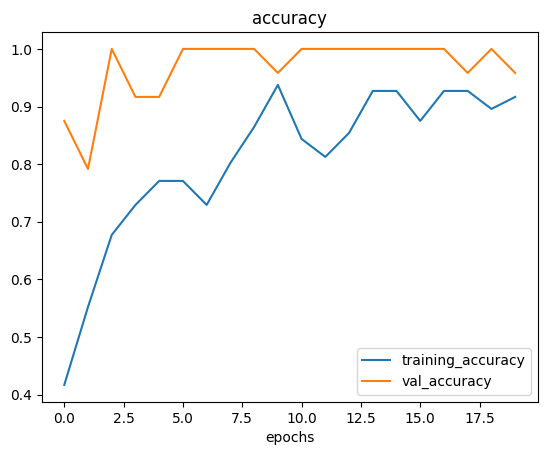


Figure 5.3.2: Model Accuracy Over Epochs

This figure presents the accuracy progression for both training and validation datasets across epochs.

The training accuracy gradually increases and stabilizes around 90%, which implies that the model learned meaningful patterns from the training images.

The validation accuracy reaches nearly 100%, significantly higher than the training accuracy. This could be due to:

A small validation set makes it easier to memorize.

Possible data leakage, or the validation data being too similar to the training set.

Despite these possibilities, the accuracy trend shows that the model can effectively distinguish features relevant to the classes.

Table 5.3.1: Model Summary

|  |  |  |
| --- | --- | --- |
| Layer (type) | Output Shape | Param # |
| conv2d (Conv2D) | (None, 224, 224, 16) | 448 |
| max\_pooling2d(MaxPooling2D) | (None, 112, 112, 16) | 0 |
| dropout (Dropout) | (None, 112, 112, 16) | 0 |
| conv2d\_1 (Conv2D) | (None, 112, 112, 32) | 4, 640 |
| max\_pooling2d\_1(MaxPooling2D) | (None, 56, 56, 32) | 0 |
| dropout\_1 (Dropout) | (None, 56, 56, 32) | 0 |
| conv2d\_2 (Conv2D) | (None, 56, 56, 64) | 18, 496 |
| max\_pooling2d\_2 (MaxPooling2D) | (None, 28, 28, 64) | 0 |
| dropout\_2 (Dropout) | (None, 28, 28, 64) | 0 |
| flatten (Flatten) | (None, 50176) | 0 |
| dense (Dense) | (None, 64) | 3, 211, 328 |
| dropout\_3 (Dropout) | (None, 64) | 0 |
| dense\_1 (Dense) | (None, 3) | 195 |

# CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATION

## 6.1 Conclusion

The face recognition attendance system was successfully developed and tested with all major functions working as expected. It manages the full process of attendance, beginning with face registration, followed by real-time detection and recognition, and ending with automated attendance marking and proper record storage. Each step was implemented to reduce manual work and improve accuracy.

The web application allows students to register their faces and provides administrators with tools to view and manage attendance records. In the backend, Flask is used for application control, while OpenCV handles image processing. A CNN-based model is applied for extracting facial features and recognizing individuals reliably.

The system is designed to operate on standard hardware without requiring costly or specialized devices. Testing showed that it can record attendance efficiently, minimize errors, and prevent duplicate entries. By saving classroom time and reducing manual effort, it provides a dependable way to handle attendance in educational institutions.

## 6.2 Future Recommendation

The system works well as it is, but there are a few ways it could be improved in the future:

Better Face Detection: Using stronger face detection models like RetinaFace could help detect faces more accurately, even in poor lighting or when faces are partially covered.

CCTV Camera Integration: Connecting the system to existing classroom CCTV cameras would allow attendance to be recorded automatically without needing a separate webcam.

Multi-Face Support: Upgrading the system to recognize multiple students at the same time would save time in larger classes.

Mobile Access: A mobile-friendly version would let teachers monitor attendance and students check their records directly from their phones.

These improvements would make the system more practical, flexible, and easier to use in everyday classroom situations.

# REFRENCES

[1]T. Sahay, S. Srinivas, Y. Agarwal, V. Sudarshan, and M. Madhura, “FACE RECOGNITION ATTENDANCE SYSTEM,” International Journal of Research Publication and Reviews Journal homepage: www.ijrpr.com, vol. 3, pp. 1111–1119, 2022.

[2]M. Rajput, “Face-Recognition Attendance Management System,” vol. 8, p. 492, Accessed: Jul. 20, 2025.

[3]R. Bairagi, R. Ahmed, S. A. Tisha, M. S. Sarder, M. S. Islam, and M. A. Islam, "A Real-time Face Recognition Smart Attendance System with Haar Cascade Classifiers," Third International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2021

[4]M. Arsenovic, S. Sladojevic, A. Anderla and D. Stefanovic, "FaceTime — Deep learning based face recognition attendance system," 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2017

|  |
| --- |
| [5]Goyal, Abhiyank, et al. "Online Attendance Management System Based on Face Recognition Using CNN." 2nd International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS 2021), Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems-ICICNIS. 2021. |

[6]Ahmed, M., et al. "An intelligent attendance system based on convolutional neural networks for real-time student face identification." Journal of Engineering Science and Technology 17.5 (2022)

[7]Bavaskar, Vaibhav. Face Recognition Attendance System. Diss. Doctoral dissertation, Sant Gadge Baba Amravati University, Amravati, 2024.

[8]V. HARIRAMANI, “Face Identification using Haar cascade classifier,” GEEKY BAWA, Oct. 07, 2020.

[9]N. Shahriar, “What is Convolutional Neural Network — CNN (Deep Learning),” Medium, Feb. 01, 2023.

[10]Alexander Otchenashenko, “02 object detection with haar cascade classifier,” YouTube, Dec. 18, 2021.

‌

# APPENDICES

