

PROJECT REPORT

ON

“Face Recognition Based Attendance”

## Submitted To: Submitted By:

Ms. Neha SHASHI BHUSHAN DUTT

21011004085

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING ARAVALI COLLEGE OF ENGINEERING AND MANAGEMENT FARIDABAD – 121002

# **ACKNOWLEDGEMENT**

This project would not have taken shape, without the guidance provided by **Ms. Neha Ma’am**, my Trainer who helped in the modules of our project and resolved all the technical as well as other problems related to the project and, for always providing us with a helping hand whenever we faced any bottlenecks, in spite of being quite busy with their hectic schedules.

We would also like to thank our project supervisor Ms. Devika who gave me the opportunity and provided us all the academic and conceptual support for our project.

Above all we wish to express our heartfelt gratitude to **Ms. Sakshi Kumar**, H.O.D, CSE DEPARTMENT whose support has greatly boosted our self- confidence and will go a long way on helping us to reach further milestones and greater high.

# **ABSTRACT**

In colleges, universities, organizations, schools, and offices, taking attendance is one of the most important tasks that must be done on a daily basis. The majority of the time, it is done manually, such as by calling by name or by roll number. The main goal of this project is to create a Face Recognition-based attendance system that will turn this manual process into an automated one. This project meets the requirements for bringing modernization to the way attendance is handled, as well as the criteria for time management. This device is installed in the classroom, where and student's information, such as name, roll number, class, sec, and photographs, is trained. The images are extracted using Open CV. Before the start of the corresponding class, the student can approach the machine, which will begin taking pictures and comparing them to the qualified dataset. Logitech C270 web camera and NVIDIA Jetson Nano Developer kit were used in this project as the camera and processing board. The image is processed as follows: first, faces are identified using a Haarcascade classifier, then faces are recognized using the LBPH (Local Binary Pattern Histogram) Algorithm, histogram data is checked against an established dataset, and the device automatically labels attendance. An Excel sheet is developed, and it is updated every hour with the information from the respective class instructor.

**TABLE OF CONTENT**

1. **INTRODUCTION**
   1. Problem statement
   2. Objective
   3. Scope of the system
2. **SYSTEM ANALYSIS**
   1. User Interface
   2. Hardware Requirements
   3. Software Requirements
   4. Safety Requirements
3. **SYSTEM DESIGN**
   1. System Functionality
   2. System modules
4. **SYSTEM IMPLEMENTATION**
   1. System coding
5. **SUMMARY AND CONCLUSION**
   1. Conclusion
   2. Future Scope
6. **References**

**1.INTRODUCTION**

Taking attendance manually is a lengthy process and it consumes a lot of time. To overcome such problem, there is a need for an automatic attendance system that is highly efficient, reliable and easy-to-use. Attendance can be tracked using various methods, including manual sign-in sheets, barcode scanning, RFID, and biometric systems such as facial recognition-based attendance systems. The choice of method depends on factors such as the number of students, the technology available, and the institutional policies. The identification of a person from his or her facial features is known as face recognition. This report focuses on the implementation of an automatic attendance system using face recognition algorithms to take the attendance of the students and manage the attendance database. We live in a time when technology is improving day by day and it’s an essential part of our daily lives. People now enjoy the smart life because it saves time, money and effort. As an example, the school teachers are used to struggle with class attendance. The traditional techniques like passing the attendance sheet and calling students’ names are time- consuming and it also gives malpractice in the attendance system

* 1. **PROBLEM STATEMENT**

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as calling student names or checking respective identification cards. There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names, attendance sheet is passed around the classroom during the lecture sessions. The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class.

Thus, face recognition attendance system is proposed in order to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance. Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

* 1. **OBJECTIVE**

The objective of this project is to develop face recognition attendance system.

Expected achievements in order to fulfill the objectives are:

* To detect the face segment from the video frame.
* To extract the useful features from the face detected.
* To classify the features in order to recognize the face detected.
* To record the attendance of the identified student.
  1. **SCOPE OF THE SYSTEM**

We are setting up to design a system comprising of two modules. The first module (face detector) is a mobile component, which is basically a camera application that captures student faces and stores them in a file using computer vision face detection algorithms and face extraction techniques. The second module is a desktop application that does face recognition of the captured images (faces) in the file, marks the students register and then stores the results in a database for future analysis.

**📍 What is Attendance Management?**

Attendance management is the process of **tracking the presence or absence of individuals** in institutions like schools, colleges, offices, factories, or events. Traditionally, attendance is recorded:

* Manually by signing registers or calling roll numbers
* Through ID cards or RFID swipes
* Using fingerprint biometric systems

These methods are time-consuming, error-prone, and often vulnerable to **proxy attendance** or **manipulation**.

**📉 Problems in Traditional Systems**

1. ❌ **Manual Attendance**:
   * Time-consuming for teachers/admins
   * Prone to human errors or manipulation
   * No real-time monitoring
2. ❌ **Biometric Fingerprint Systems**:
   * Requires physical contact (unhygienic, especially after COVID-19)
   * Can wear out or fail due to injuries or dirt
3. ❌ **Card or PIN Based Systems**:
   * Cards can be swapped between students
   * Forgotten passwords or lost cards can interrupt the process

All these systems have **security**, **scalability**, and **efficiency issues**.

**💡 The Need for Smart Attendance**

In today's world, especially post-pandemic, there's a growing need for:

* **Contactless** solutions
* **Automated and accurate** systems
* **Real-time attendance tracking**
* **Technology-driven monitoring** in education and workplaces

This is where **Face Recognition Technology** comes in.

**🧠 What is Face Recognition Technology?**

Face recognition is a **biometric technology** that uses **computer vision and artificial intelligence (AI)** to:

* Detect human faces from an image or video stream
* Analyze facial features (eyes, nose, chin, etc.)
* Compare with pre-stored data to recognize individuals

Face recognition works even when the person is:

* Wearing glasses or a cap
* Standing at a slight angle
* Present in a group

**🎯 Project Objective**

The goal of this project is to **develop an intelligent attendance system** that uses **face recognition technology** to:

* Capture a person’s face through a webcam
* Match it against a known database of faces
* Automatically mark their attendance with a timestamp
* Store the data securely in a database
* Provide an admin-friendly interface to manage and export records

**🧩 Key Features of the System**

* ✅ Contactless attendance (no touching required)
* ✅ Prevents proxy attendance (face is unique)
* ✅ Real-time face recognition using AI
* ✅ Automatic logging and reporting
* ✅ User-friendly GUI (Tkinter or web-based)
* ✅ Portable, scalable, and fast

**💻 Technologies Used**

* **Python** – Programming language
* **OpenCV** – Image processing and camera handling
* **face\_recognition** – Facial detection and encoding
* **SQLite/MySQL** – Database for storing users and attendance logs
* **Tkinter/Flask** – GUI or web application frontend

**🏫 Real-World Applications**

* Schools and colleges: Automate classroom attendance
* Offices: Track employee presence and working hours
* Events/Conferences: Secure entry logging
* Hospitals: Contactless staff tracking
* Hostels: Monitor student in/out movement

**✅ Summary of Introduction**

| **Topic** | **Description** |
| --- | --- |
| Problem | Traditional attendance systems are manual, slow, and prone to errors or misuse |
| Solution | Use AI-based face recognition for contactless, accurate attendance |
| Objective | Create an intelligent, real-time attendance tracking system |
| Technologies | Python, OpenCV, face\_recognition, database, GUI (Tkinter/Flask) |
| Use Cases | Educational institutes, offices, events, hospitals |

**📌 Conclusion of Introduction**

The **Face Recognition Based Attendance System** is a perfect example of how **AI and computer vision can solve real-world administrative problems**. It eliminates the need for manual attendance, ensures accuracy, improves security, and saves time—making it a **future-ready solution** for organizations of all sizes.

In educational institutions and workplaces, **attendance management** is an essential administrative task. Traditionally, this has been done manually—via paper registers, roll calls, or punch cards—which is time-consuming, prone to error, and vulnerable to issues like **proxy attendance**, **manual manipulation**, and **data loss**.

With the advancement of **Artificial Intelligence (AI)** and **Computer Vision**, especially in the field of **face recognition**, there is an opportunity to transform this task into a **smart, automated, and secure** system.

**📍 What is Face Recognition?**

**Face Recognition** is a biometric method of identifying or verifying a person’s identity using their facial features. It involves:

* Detecting a face in an image or video stream
* Extracting unique facial landmarks or encodings
* Comparing them against a database of registered faces

It is widely used today in smartphones, surveillance, airport security, and now, **attendance systems**.

**🎯 Project Overview**

The **Face Recognition Based Attendance System** is an AI-powered solution designed to **automatically detect and recognize faces** of students or employees using a camera and **mark their attendance** in a digital system.

Unlike traditional methods, this system:

* Eliminates the need for physical contact (contactless)
* Increases accuracy and reliability
* Reduces human error and administrative workload
* Prevents impersonation or proxy attendance

**🏗️ Key Features of the System**

* **Real-time Face Detection and Recognition** using camera feed
* **Automatic Attendance Marking** with timestamp
* **Database Storage** of user details and attendance records
* **Graphical or Web Interface** for administrators
* **Exportable Reports** (e.g., CSV, Excel)

**⚙️ Technologies Used**

This system is developed using:

* **Python**: Core programming language
* **OpenCV**: For face detection and image processing
* **face\_recognition (dlib-based)**: For facial feature encoding and matching
* **SQLite / MySQL**: For storing user and attendance data
* **Tkinter / Flask**: For user interface
* **Pickle / JSON**: For saving encodings and session data

**📌 Why Face Recognition for Attendance?**

| **Traditional Method** | **Issues** |
| --- | --- |
| Manual Roll Call | Time-consuming, error-prone |
| Fingerprint Scan | Requires physical contact (hygiene issues) |
| RFID Cards | Can be shared or lost |
| Face Recognition | Fast, contactless, hard to spoof |

Face recognition offers a **non-intrusive**, **automated**, and **tamper-proof** alternative that works effectively in both classroom and workplace environments.

**🌐 Applications**

* **Schools & Colleges**: Classroom and exam attendance
* **Corporate Offices**: Employee login/logout tracking
* **Events & Conferences**: Visitor logging and security
* **Healthcare Facilities**: Patient check-ins
* **Factories & Plants**: Shift monitoring

**🎯 Objectives of the Project**

* To **automate the attendance process** using face recognition
* To **reduce manual effort** and **increase efficiency**
* To provide a **secure, tamper-proof**, and **contactless solution**
* To allow **data storage**, **retrieval**, and **report generation**

A **Face Recognition Based Attendance System** is a biometric technology that uses facial recognition to identify and verify individuals for the purpose of recording their attendance automatically. It eliminates the need for manual attendance systems such as paper sheets or punching cards, offering a more secure, efficient, and contactless method.

**📌 Key Points with Full Explanation**

**1. What is Face Recognition?**

Face recognition is a type of biometric system that identifies or verifies a person from a digital image or a video frame. It analyzes the facial features such as eyes, nose, cheekbones, and jawlines using image processing and machine learning techniques.

It uses algorithms such as Haar Cascade, Local Binary Patterns Histograms (LBPH), or modern deep learning models (CNNs, DNNs).

**2. Why Use Face Recognition for Attendance?**

* **Automatic**: No manual entry needed.
* **Non-intrusive**: No physical contact like fingerprint or ID swiping.
* **Fast and Real-Time**: Captures and marks attendance within seconds.
* **Reduces Proxy**: Prevents fake or buddy punching (attendance on behalf of others).
* **Scalable**: Easily manageable for large organizations or classrooms.

**3. How It Works – Workflow Overview**

1. **Face Detection**: The system detects the face from the camera feed using algorithms like Haar Cascade or MTCNN.
2. **Face Preprocessing**: The captured face is converted into grayscale, resized, and normalized.
3. **Face Encoding**: The system converts the face into numerical vectors (feature extraction).
4. **Face Matching**: The captured face is compared with the database using similarity measures.
5. **Attendance Marking**: If matched, attendance is recorded with timestamp.

**4. Components of the System**

* **Camera Module**: Captures real-time images or video.
* **Face Recognition Module**: Detects and verifies faces.
* **Database**: Stores face data and attendance records.
* **User Interface**: Displays records, logs, and system operations.

**5. Technologies Used**

* **Languages**: Python, OpenCV, TensorFlow/Keras
* **Libraries**: dlib, face\_recognition, NumPy, Pandas
* **Databases**: MySQL, SQLite, MongoDB
* **Tools**: Flask/Django for web integration (optional)

**6. Applications**

* **Educational Institutes**: For student attendance.
* **Corporate Offices**: For employee check-ins.
* **Events/Seminars**: For attendee verification.
* **Hostels/PGs**: For entry logging.

**7. Advantages**

* Saves time and effort.
* Minimizes human error.
* Prevents impersonation.
* Easy to maintain historical records.

**8. Challenges**

* Requires good lighting and camera quality.
* Accuracy may be affected by changes in facial appearance (e.g., mask, beard).
* Privacy concerns and data protection issues.
* High initial setup cost for advanced systems.

**2.SYSTEM ANALYSIS**

**2.2. USER INTERFACE**

The user interface (UI) of a face recognition attendance system should be intuitive and user-friendly for both students and administrators. Here's a breakdown of the key UI components:

1. Student Interface

* Login Screen: Students can use their ID cards or biometrics (fingerprint scanner) to log in.
* Attendance Marking**:** The system captures the student's face through the webcam. Upon successful recognition, a confirmation message is displayed.
* Attendance History: Students can access their past attendance records.

2. Administrator Interface

* Student Management: Add, edit, and delete student information, including names, IDs, and facial images.
* Attendance Reports: Generate reports on overall attendance, individual student attendance, and attendance for specific dates or classes.
* Settings: Configure system parameters like camera settings, recognition thresholds, and data storage options.

**2.2. HARDWARE REQUIREMENTS**

**2.2.1. Computer**

* **Processor:** A minimum of a mid-range Intel Core i5 or equivalent AMD Ryzen 5 processor is recommended. For larger deployments or high accuracy requirements, consider an Intel Core i7 or AMD Ryzen 7 processor.
* **RAM:** At least 8 GB of RAM is recommended to ensure smooth operation. 16 GB or more might be necessary for complex algorithms or large numbers of users.
* **Storage:** A solid-state drive (SSD) is preferred for faster boot times and application loading. Storage capacity depends on the number of student images and attendance data.

**2.2.2. Camera**

* **Resolution:** A minimum of 720p (1280x720) resolution is recommended. Higher resolutions like 1080p (1920x1080) can improve accuracy but require more processing power.
* **Frame Rate:** A frame rate of at least 30fps (frames per second) is desirable to capture clear images for recognition.
* **Lighting:** Consider using a camera with good low-light performance or installing additional lighting to ensure consistent illumination.

**2.2.3. Other Hardware**

* **Fingerprint Scanner:** Can be used as an alternative or secondary login method for students.
* **Multiple Cameras:** For larger areas or multiple entry points, a system with multiple cameras can be set up.
* **Projector/Display (Optional):** To display real-time attendance data or student recognition confirmations.

**2.2.4. Additional Considerations**

* **Operating System:** Choose a stable operating system compatible with your chosen facial recognition software libraries.
* **Internet Connectivity:** While not always essential for system operation, internet connectivity allows for data backup, remote management, and software updates.
* **Scalability:** If you anticipate future growth, consider hardware that can be easily upgraded to accommodate more users or higher processing demands.

**2.3. SOFTWARE REQUIREMENTS**

The software requirements for a face recognition attendance system can be categorized into three main parts:

**2.3.1. Operating System**

* A stable and reliable operating system is essential for smooth system operation. Popular choices include:
  + **Windows 10 or 11:** Widely compatible with various facial recognition libraries and offers good hardware support.
  + **Linux:** Open-source and highly customizable but requires some technical expertise for setup and configuration.
  + **macOS:** User-friendly and secure but may have limited compatibility with some facial recognition software.

**2.3.2. Facial Recognition Libraries**

These libraries provide the core functionality for face detection and recognition. Some popular open-source options include:

* **OpenCV (Open Source Computer Vision Library):** A versatile library offering a wide range of computer vision tools, including face detection and recognition algorithms.
* **dlib:** Another open-source library with advanced facial landmark detection capabilities, suitable for high-accuracy applications.
* **TensorFlow/PyTorch (Deep Learning Frameworks):** While not strictly facial recognition libraries, these frameworks can be used to develop custom deep learning models for face recognition, potentially achieving higher accuracy but requiring more expertise and computational resources.

**2.3.3. Additional Software**

* **Programming Language:** The choice of programming language depends on the selected facial recognition library. Python is a popular choice due to its simplicity and extensive libraries for scientific computing. C++ can also be used for performance-critical applications.
* **Database Management System (DBMS):** A database is needed to store student information (names, IDs) and attendance data. Popular options include MySQL, PostgreSQL, or SQLite, depending on the project's scale and complexity.

**2.4. SAFETY REQUIREMENTS**

Safety requirements for a face recognition attendance system are crucial to consider for both user privacy and system security. Here are some key aspects to address:

**2.4.1. Data Security:**

* **Data Encryption:** Store all student data, including facial images, in an encrypted format to prevent unauthorized access in case of a security breach.
* **Access Controls:** Implement access controls within the system to restrict who can view or modify student data. Only authorized administrators should have access to sensitive information.
* **Data Retention Policy:** Establish a clear policy on how long student data will be retained. Regularly delete old data that is no longer necessary.

**2.4.2. User Privacy:**

* **User Consent:** Obtain explicit consent from students before collecting and storing their facial images.
* **Transparency:** Inform students about how their data is being used, stored, and secured.
* **Opt-Out Mechanism:** Provide a way for students to opt-out of the system if they choose not to participate in face recognition attendance.
* **Anonymization:** Consider anonymizing student data whenever possible, especially for reporting purposes.

**2.4.3. System Security:**

* **Software Updates:** Regularly update the facial recognition software and operating system to address security vulnerabilities.
* **Network Security:** If the system connects to a network, implement robust network security measures like firewalls and intrusion detection systems.
* **Physical Security:** Secure the camera and computer hardware to prevent physical tampering.

**System analysis is the process of understanding, evaluating, and designing a system to ensure it meets the desired objectives. For a Face Recognition Based Attendance System, the system analysis involves identifying what the system should do, how it will do it, and what components are needed.**

**1. ✅ Existing (Manual) System Analysis**

**In many institutions or offices, attendance is still taken manually using:**

* **Registers**
* **Paper sheets**
* **Biometric punch machines**
* **ID card swiping systems**

**❌ Drawbacks of the Existing System:**

* **Time-consuming**
* **Prone to human error or manipulation (e.g., proxy attendance)**
* **No real-time monitoring**
* **Difficult to maintain and analyze large data sets**
* **Requires physical interaction (problematic during pandemics)**

**2. 💡 Proposed System Overview**

**The proposed Face Recognition Based Attendance System automates the process using face recognition technology, which:**

* **Captures and detects a person’s face in real-time**
* **Compares it with stored records**
* **Marks attendance automatically and stores it in a database**

**3. ⚙️ System Requirements Analysis**

**A. Functional Requirements (What the system should do):**

* **Detect and recognize faces from live camera feed.**
* **Match detected faces with stored records.**
* **Mark attendance with timestamp and user ID.**
* **Save attendance records in a secure database.**
* **Allow admin to add, update, or delete student/employee records.**

**B. Non-Functional Requirements (How the system should behave):**

* **Accuracy: High face recognition accuracy even in different lighting.**
* **Speed: Real-time recognition without delay.**
* **Security: Secure storage of facial data.**
* **Usability: Simple and user-friendly interface.**
* **Scalability: Should support large number of users.**

**C. Hardware Requirements:**

* **Webcam or IP camera**
* **Computer with good processor (for model inference)**
* **Sufficient RAM (8GB or higher preferred)**

**D. Software Requirements:**

* **Python 3.x**
* **OpenCV, face\_recognition library**
* **MySQL/SQLite database**
* **Optional: Flask or Django for web interface**

**4. 🔍 Feasibility Study**

**A. Technical Feasibility:**

* **Easily implementable using open-source libraries and Python.**
* **Compatible with standard hardware.**
* **Supported by modern image processing and machine learning tools.**

**B. Operational Feasibility:**

* **Simple to operate by admin or staff.**
* **Reduces manual effort and increases efficiency.**
* **No need for physical contact (important post-COVID).**

**C. Economic Feasibility:**

* **Low-cost implementation for small organizations (using open-source).**
* **ROI is high in the long term by saving time and reducing fraud.**

**5. 🔄 Data Flow Analysis (Basic)**

**Here’s a simple flow of how data moves in the system:**

1. **User enters in front of camera →**
2. **Camera captures image →**
3. **Face Detection & Encoding →**
4. **Compare with stored records →**
5. **If match found → Attendance is marked → Database updated → UI shows confirmation.**

**6. 🔒 Risk and Limitation Analysis**

* **Recognition accuracy may drop due to poor lighting or face masks.**
* **Data privacy and GDPR compliance are necessary.**
* **Unauthorized access to face data can be a security concern.**

**7. 🎯 Benefits of the Proposed System**

* **Saves time and effort**
* **Increases accuracy and transparency**
* **Real-time tracking and analytics**
* **Easy report generation and data backup**

**System analysis is the process of examining the current system (manual or existing digital methods), identifying its limitations, and proposing a new system that solves these issues effectively. It lays the foundation for system design and implementation.**

**✅ 1. Problem Identification**

**Manual attendance systems and conventional biometric methods (like fingerprint or RFID) suffer from several issues:**

**⚠️ Problems in Existing Attendance Methods:**

| **Method** | **Problems** |
| --- | --- |
| **Manual Roll Call** | **Time-consuming, easily manipulated, human error** |
| **Fingerprint Scanners** | **Requires physical contact, unhygienic, not effective for COVID-era or hospital use** |
| **RFID Cards/Smart Cards** | **Can be shared, lost, or forgotten** |
| **Barcode/QR Code Scans** | **Requires manual scanning, can be slow and inconvenient** |

**These methods often lead to:**

* **Proxy attendance or "buddy punching"**
* **Data loss due to paper-based systems**
* **Low efficiency in large institutions**

**🎯 2. Need for a New System**

**The goal is to develop a fully automated, contactless, and intelligent system that overcomes the limitations of traditional methods.**

**Why Face Recognition?**

* **Every face is unique and hard to duplicate**
* **Works from a distance (non-intrusive)**
* **Fast and convenient**
* **Reduces human intervention**

**📊 3. Feasibility Study**

**Before implementing the new system, we conduct various feasibility checks:**

**🔹 Technical Feasibility**

* **Python, OpenCV, and face\_recognition libraries are open-source and well-supported**
* **The system can run on standard hardware with webcam access**
* **Easy integration with databases and GUIs**

**🔹 Operational Feasibility**

* **Easy to operate for both admin and users**
* **Automates daily tasks, saving time and effort**
* **No special training required**

**🔹 Economic Feasibility**

* **Low development cost (no paid software needed)**
* **Reduces long-term administrative costs**
* **Scalable for larger institutions without heavy investment**

**🔹 Legal/Ethical Feasibility**

* **Requires handling of biometric data (faces) ethically**
* **Must comply with data protection laws (GDPR, DPDP)**
* **User consent must be obtained**

**🏗️ 4. System Requirements**

**a. Functional Requirements**

* **Capture real-time face via webcam**
* **Compare it with stored facial data**
* **Mark attendance automatically**
* **Display or export attendance reports**
* **Register new users (admin only)**

**b. Non-Functional Requirements**

* **The system should be responsive (within 1–2 seconds)**
* **Secure storage of face data**
* **Support for concurrent recognition (multi-face detection)**
* **Portable and easy to deploy on different machines**

**🔄 5. Existing System vs Proposed System**

| **Feature** | **Existing System** | **Proposed Face Recognition System** |
| --- | --- | --- |
| **Contactless** | **❌** | **✅** |
| **Accuracy** | **Low (manual errors)** | **High** |
| **Speed** | **Slow** | **Fast (real-time)** |
| **Security** | **Low** | **High (unique face data)** |
| **Cost over time** | **High (manual effort)** | **Low (automated)** |
| **Scalability** | **Limited** | **Highly scalable** |

**🧠 6. System Users and Stakeholders**

| **Role** | **Responsibilities** |
| --- | --- |
| **Admin** | **Register users, view attendance, manage reports** |
| **Users (students/employees)** | **Appear in front of the camera for attendance** |
| **System** | **Capture face, match with database, mark attendance** |

**📈 7. Expected Benefits of the New System**

* **✅ Eliminates proxy attendance**
* **✅ Improves record accuracy**
* **✅ Saves time and manpower**
* **✅ Provides digital records for analysis and reporting**
* **✅ Suitable for post-pandemic hygiene and safety**

**📌 Conclusion of System Analysis**

**Through a thorough analysis of the current system and its shortcomings, it is clear that a Face Recognition Based Attendance System is a practical, secure, and efficient solution. This analysis sets the stage for designing a system that is:**

* **Technically feasible**
* **Economically viable**
* **Operationally efficient**
* **Legally and ethically sound**

**3.SYSTEM DESIGN**

**3.1. SYSTEM FUNCTIONALITY**

The face recognition-based attendance system offers several key functionalities to streamline the attendance management process. Here's an overview of its functionality:

**3.1.1. Face Detection:**

* The system employs algorithms to detect human faces within images or video frames captured by cameras.

**3.1.2. Facial Feature Extraction:**

* Once faces are detected, the system extracts facial features from the detected faces. These features are essential for uniquely identifying individuals.

**3.1.3. Face Recognition:**

* The extracted facial features are compared against pre-registered faces stored in the system's database. This comparison is done using algorithms that calculate the similarity between facial features.
* If a match is found between the extracted features and a pre-registered face, the system recognizes the individual.

**3.1.4. Attendance Marking:**

* Upon successful recognition, the system automatically marks the attendance of the recognized individual.
* The attendance records are updated in real-time, eliminating the need for manual attendance marking.

**3.1.5. Database Management:**

* The system maintains a database of pre-registered individuals, along with their associated facial features and attendance records.
* Administrators can add, remove, or update individual records in the database as needed.

**3.1.6. User Interface:**

* The system provides a user-friendly interface for administrators to interact with.
* Administrators can view attendance reports, monitor system performance, and manage the database through the interface.

**3.1.7. Real-time Processing:**

* The system operates in real-time, allowing for instantaneous face detection, recognition, and attendance marking as individuals enter the camera's field of view.

**3.2. SYSTEM MODULES**

The face recognition-based attendance system comprises several modules, each responsible for specific tasks or functionalities. Here's an outline of the main modules:

**3.2.1. Face Detection Module:**

* Responsible for detecting human faces within images or video frames captured by cameras.
* Utilizes face detection algorithms like Haarcascades or deep learning-based approaches for accurate detection.

**3.2.2. Facial Feature Extraction Module:**

* Extracts facial features from the detected faces, which are essential for uniquely identifying individuals.
* Techniques such as Principal Component Analysis (PCA) or deep learning-based methods like Face Net may be employed for feature extraction.

**3.2.3. Face Recognition Module:**

* Compares the extracted facial features against pre-registered faces stored in the system's database.
* Uses algorithms like Euclidean distance or cosine similarity to determine the similarity between facial features and recognize individuals.

**3.2.4. Attendance Marking Module:**

* Marks attendance automatically upon successful recognition of individuals.
* Updates attendance records in real-time, eliminating the need for manual attendance marking.

**3.2.5. Database Management Module:**

* Manages the system's database, which stores pre-registered individuals' information, including facial features and attendance records.
* Allows for adding, removing, or updating individual records in the database.

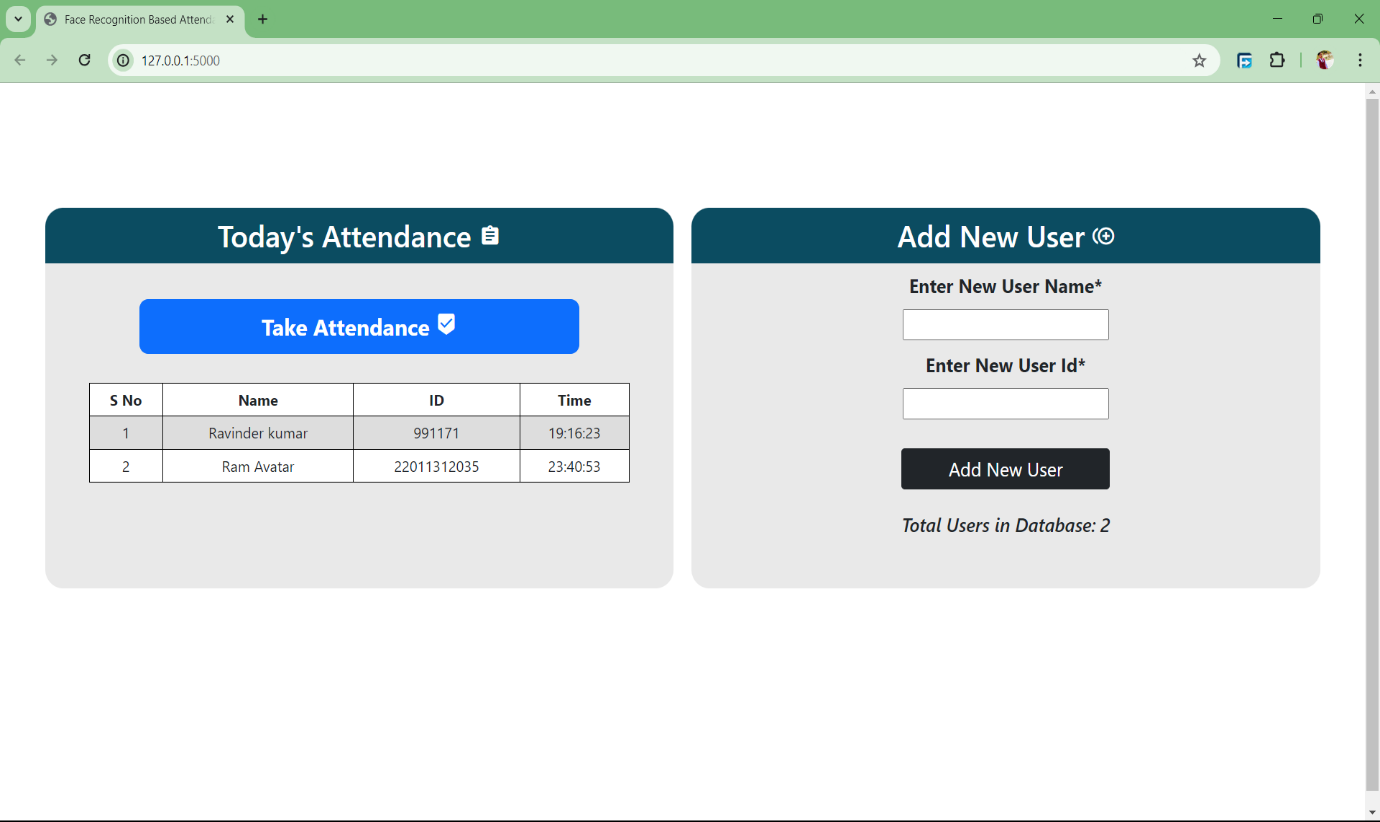
**3.2.6. User Interface Module:**

* Provides a user-friendly interface for administrators to interact with the system.
* Enables administrators to view attendance reports, monitor system performance, and manage the database.

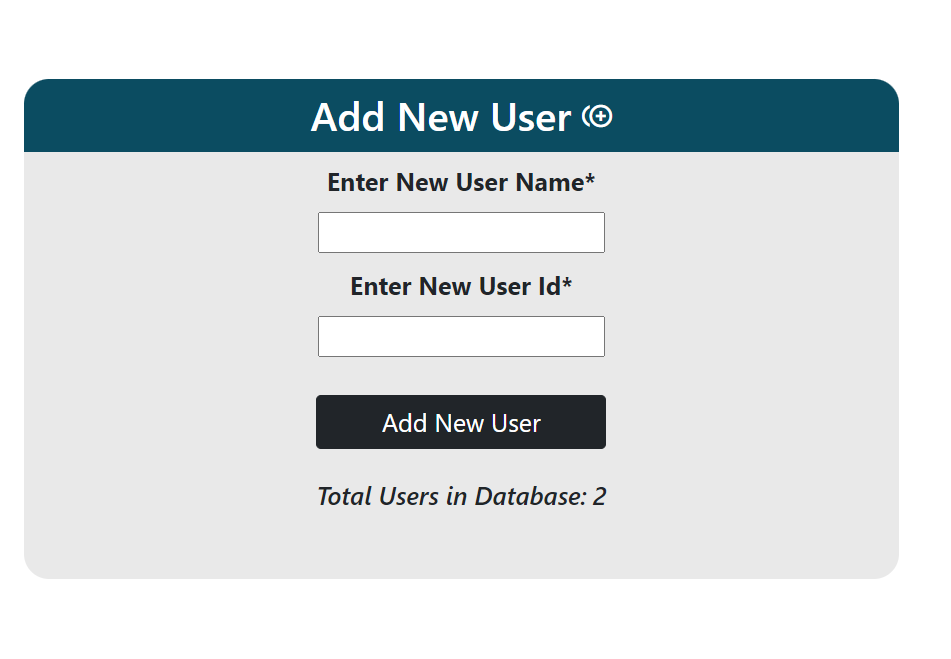
**3.2.7. Real-time Processing Module:**

* Facilitates real-time processing of images or video streams captured by cameras.
* Enables instantaneous face detection, recognition, and attendance marking as individuals enter the camera's field of view.

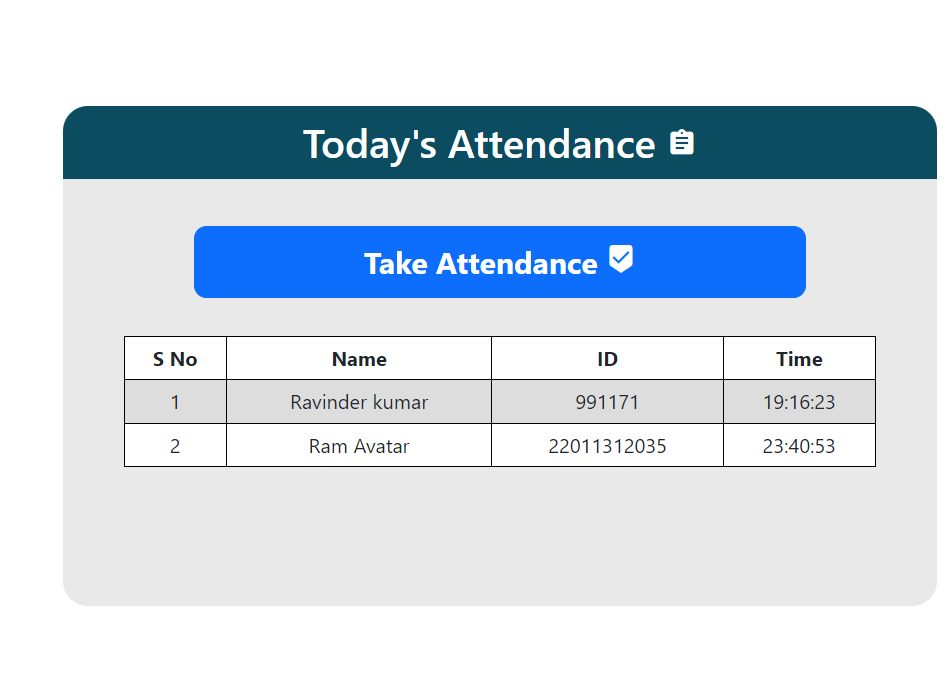
**Home page:**



**Add New User:**



**Today’s Attendance:**



**System Design is the blueprint of the system—it shows how various components (hardware, software, users) interact and how the system will achieve its intended functionality. It is usually divided into two parts:**

1. **High-Level Design (HLD) – Architectural overview and main modules**
2. **Low-Level Design (LLD) – Detailed design of each module or component**

**🔷 1. High-Level Design (HLD)**

**This gives an overview of the main components and how they interact.**

**🔁 System Architecture**

**text**

**CopyEdit**

**+-------------+ +---------------+ +----------------+ +-----------------+**

**| Webcam | ---> | Face Detection| --> | Face Recognition| --> | Attendance Mark |**

**+-------------+ +---------------+ +----------------+ +-----------------+**

**|**

**v**

**+---------------------+**

**| Database System |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Admin/User Interface|**

**+---------------------+**

**🔧 Major Modules in HLD:**

1. **Camera Module: Captures live video frames of students/employees.**
2. **Face Detection Module: Detects faces in the video frame using OpenCV or similar.**
3. **Face Recognition Module: Compares detected face with stored faces in the database.**
4. **Attendance Module: Marks attendance with timestamp if a match is found.**
5. **Database Module: Stores face data and attendance records.**
6. **Admin Panel (UI): Allows admin to manage users and view attendance.**

**🔻 2. Low-Level Design (LLD)**

**This part breaks down each component into subcomponents and shows their internal logic and data flow.**

**📷 A. Camera and Face Capture Module**

* **Accesses webcam using OpenCV**
* **Captures frame every few seconds**
* **Passes image to face detection module**

**python**

**CopyEdit**

**cap = cv2.VideoCapture(0)**

**ret, frame = cap.read()**

**🧠 B. Face Detection & Preprocessing**

* **Uses algorithms like Haar Cascades or MTCNN to detect faces**
* **Converts image to grayscale and crops face area**

**python**

**CopyEdit**

**face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')**

**faces = face\_cascade.detectMultiScale(gray\_frame, scaleFactor=1.3, minNeighbors=5)**

**🧬 C. Face Recognition Module**

* **Uses face embeddings (vector representations) generated by a model like dlib or face\_recognition**
* **Compares new face encoding with stored encodings**
* **Uses Euclidean distance or cosine similarity for matching**

**python**

**CopyEdit**

**results = face\_recognition.compare\_faces(known\_faces, new\_face\_encoding)**

**🗃️ D. Database Design**

**Tables:**

1. **users**
   * **user\_id (Primary Key)**
   * **name**
   * **roll\_no or employee\_id**
   * **face\_encoding (vector or serialized string)**
2. **attendance**
   * **record\_id (Primary Key)**
   * **user\_id (Foreign Key)**
   * **date**
   * **time**
   * **status (Present/Absent)**

**🖥️ E. Admin/User Interface**

* **Built using Tkinter (GUI) or Flask/Django (Web Interface)**
* **Features:**
  + **Add/Delete users**
  + **View attendance reports**
  + **Download/export data**
  + **Live face detection view (optional)**

**📑 F. Sample Flow Chart**

**text**

**CopyEdit**

**START**

**↓**

**Open Camera → Capture Image**

**↓**

**Detect Face → Preprocess Face**

**↓**

**Encode Face → Match with Database**

**↓**

**IF match found:**

**Mark Attendance**

**Show Success**

**ELSE**

**Show Unknown Face**

**↓**

**STOP**

**✅ Design Considerations**

| **Factor** | **Design Choice** |
| --- | --- |
| **Performance** | **Use lightweight models for faster recognition (e.g., LBPH)** |
| **Security** | **Hash sensitive data, restrict admin access** |
| **Accuracy** | **Train on diverse images per person (lighting, angles, etc.)** |
| **Scalability** | **Store data in cloud or larger DB for institutional use** |

**🧪 Optional Enhancements (Advanced Design)**

* **Train a CNN-based face recognizer (e.g., using TensorFlow/Keras)**
* **Add mask detection using YOLO or MobileNet**
* **Use cloud storage (Firebase, AWS) for database**
* **Deploy with Raspberry Pi for embedded implementation**

**System design involves creating a blueprint for how your system will function. It defines the architecture, components, data flow, user interfaces, and databases that work together to perform face-based attendance tracking.**

**🎯 1. Objectives of System Design**

* **Define system structure and behavior**
* **Ensure scalability, reliability, and efficiency**
* **Plan interaction between hardware and software components**
* **Translate system analysis into a working model**

**🧱 2. System Architecture**

**A modular architecture is used, where the system is divided into the following layers:**

**📦 a. Input Layer**

* **Webcam or IP camera**
* **Captures real-time images or video feed**

**🧠 b. Processing Layer**

* **Uses OpenCV to detect face regions**
* **Uses face\_recognition (Dlib or FaceNet) to extract and compare facial encodings**
* **Decision logic: If match is found → mark attendance**

**💾 c. Database Layer**

* **Stores:**
  + **User details (Name, ID, Face encoding)**
  + **Attendance records (Date, Time, Status)**
* **SQL (e.g., SQLite or MySQL)**

**🖥️ d. Application Layer (UI)**

* **Admin interface for:**
  + **Registering new users**
  + **Viewing attendance logs**
  + **Exporting reports**

**🔄 3. System Workflow**

1. **Start Application**
2. **Live Camera Feed Activated**
3. **Detect Faces in Frame**
4. **Extract Face Encodings**
5. **Compare with Stored Encodings**
6. **If Match Found → Mark attendance with timestamp**
7. **Display or Store Attendance**
8. **Admin can view/export attendance logs**

**📊 4. Data Flow Diagram (DFD)**

**Level 1 DFD (textual version):**

**plaintext**

**CopyEdit**

**[User] --> [Camera] --> [Face Detection Module] --> [Face Recognition Module]**

**--> [Database (Face Encodings)] --> [Attendance Logger] --> [Database (Attendance)]**

**--> [Admin Panel / Export Reports]**

* **User: Appears in front of the camera**
* **Camera: Captures image**
* **Face Detection Module: Finds face location**
* **Face Recognition Module: Matches face with database**
* **Attendance Logger: Stores name, date, and time**
* **Admin Panel: Allows view/edit/export options**

**🗃️ 5. Database Design**

**a. User Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| **user\_id** | **Integer (PK)** | **Unique ID for each user** |
| **name** | **Text** | **Name of user** |
| **face\_encoding** | **BLOB/Text** | **Serialized facial data** |
| **image\_path** | **Text** | **Path to saved face image** |

**b. Attendance Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| **entry\_id** | **Integer (PK)** | **Unique entry ID** |
| **user\_id** | **Integer (FK)** | **Reference to user** |
| **date** | **Date** | **Date of attendance** |
| **time** | **Time** | **Time of recognition** |
| **status** | **Text** | **Present / Absent** |

**👨‍💼 6. User Interface Design**

**a. Admin Panel (GUI or Web)**

* **Login Authentication (optional)**
* **Register New User (capture photo and save data)**
* **View Attendance**
* **Export Attendance Report (CSV/Excel)**

**b. User Interface**

* **Displays: "Face Detected", "Attendance Marked"**
* **Optionally shows attendance history (for users)**

**🔐 7. Security Considerations**

* **Encrypt stored face encodings**
* **Secure access to the admin panel**
* **Avoid unauthorized database modification**
* **Ensure privacy and data consent**

**🛠️ 8. Technology Stack**

| **Component** | **Technology Used** |
| --- | --- |
| **Programming Language** | **Python** |
| **Face Detection** | **OpenCV, Dlib** |
| **Face Recognition** | **face\_recognition library** |
| **GUI** | **Tkinter / Flask** |
| **Database** | **SQLite / MySQL** |
| **Data Serialization** | **Pickle / JSON** |

**📌 Summary of System Design**

* **The system is built on a modular, layered architecture**
* **Uses face detection and recognition to identify users**
* **Stores attendance in a secure and structured database**
* **Provides a user-friendly admin interface**
* **Ensures data security and privacy**

**✅ Final Note**

**This system design ensures that your face recognition attendance system is:**

* **Efficient in real-time**
* **Maintainable due to modularity**
* **Scalable for large institutions**
* **Secure for user privacy**

**4.SYSTEM IMPLEMENTATION**

**System implementation is the execution phase of the project, where the design and plans are translated into a working software application. It involves writing the code, integrating modules, testing components, and making the system functional.**

**🧩 1. Implementation Overview**

**The Face Recognition Based Attendance System is implemented using:**

* **Python for development**
* **OpenCV and face\_recognition for facial detection and recognition**
* **SQLite/MySQL for database management**
* **Tkinter/Flask for the GUI or web interface**

**The system has several interdependent modules:**

1. **User Registration**
2. **Face Detection & Encoding**
3. **Face Recognition & Attendance Marking**
4. **Attendance Report Generation**
5. **Admin Control Panel (GUI)**

**🛠️ 2. Step-by-Step Implementation Process**

**🔹 Step 1: Environment Setup**

**Install the necessary packages:**

**bash**

**CopyEdit**

**pip install opencv-python**

**pip install face\_recognition**

**pip install numpy**

**pip install pillow**

**pip install tkinter # (If using GUI)**

**Set up a database using SQLite (for local apps) or MySQL (for larger apps).**

**🔹 Step 2: User Registration Module**

**Function: Capture and save user face data (images and encodings).**

**Implementation Steps:**

* **Admin enters user name/ID**
* **Webcam opens and captures multiple images of the user's face**
* **Face encodings (unique facial features) are extracted using face\_recognition.face\_encodings()**
* **Encodings are serialized (using Pickle or JSON) and saved in the database**
* **User details are also stored**

**python**

**CopyEdit**

**import face\_recognition**

**face = face\_recognition.load\_image\_file("user1.jpg")**

**face\_encoding = face\_recognition.face\_encodings(face)[0]**

**🔹 Step 3: Face Detection and Recognition**

**Function: Detect faces from a live webcam feed and compare with known encodings.**

**Implementation Steps:**

* **Open webcam feed using OpenCV**
* **Detect face locations using face\_recognition.face\_locations()**
* **Extract encodings of detected faces**
* **Compare with known encodings from the database using face\_recognition.compare\_faces()**
* **If matched → mark as "Present" with timestamp**

**python**

**CopyEdit**

**matches = face\_recognition.compare\_faces(known\_face\_encodings, current\_face\_encoding)**

**🔹 Step 4: Attendance Marking Module**

**Function: Automatically mark attendance when a face is recognized.**

**Implementation Steps:**

* **If face match is confirmed:**
  + **Check if user has already been marked for the day**
  + **If not, insert a new record into the Attendance Table with:**
    - **User ID**
    - **Date**
    - **Time**
    - **Status: "Present"**

**sql**

**CopyEdit**

**INSERT INTO attendance (user\_id, date, time, status) VALUES (101, '2025-05-15', '09:05:32', 'Present');**

**🔹 Step 5: Database Integration**

**Function: Store and retrieve user data and attendance logs.**

**Implementation Details:**

* **Use SQLite for local apps (file-based)**
* **Define two tables:**
  + **Users: user\_id, name, face\_encoding, image\_path**
  + **Attendance: entry\_id, user\_id, date, time, status**
* **SQL queries used for inserting and fetching attendance records**

**🔹 Step 6: Admin Interface**

**Function: GUI or Web Panel for managing the system.**

**Key Features:**

* **Add new users**
* **View attendance logs**
* **Export attendance to CSV/Excel**
* **Optionally allow face re-capture and user deletion**

**If using Tkinter:**

**python**

**CopyEdit**

**from tkinter import \***

**window = Tk()**

**Label(window, text="Enter Name").pack()**

**Entry(window).pack()**

**Button(window, text="Register", command=register\_user).pack()**

**If using Flask:**

* **Create web routes for /register, /attendance, /report**
* **Use HTML/CSS templates for frontend**

**🧪 3. Testing and Debugging**

* **Unit Testing: Test each module separately (face capture, recognition, database insertion)**
* **Integration Testing: Check if all modules work together smoothly**
* **Real-world Testing: Try the app with multiple faces under different lighting and angles**
* **Edge Case Testing:**
  + **Unregistered face**
  + **Duplicate attendance entries**
  + **Multiple faces in frame**

**🧩 4. Deployment**

* **Convert Python script into executable using PyInstaller:**

**bash**

**CopyEdit**

**pyinstaller --onefile main.py**

* **OR host the web app using Flask + ngrok / cloud server**
* **Store database locally or host in the cloud (Firebase, MySQL server, etc.)**

**📌 Summary of Implementation**

| **Module** | **Description** |
| --- | --- |
| **Face Registration** | **Captures and saves facial data** |
| **Face Detection & Matching** | **Identifies known faces using webcam** |
| **Attendance Logging** | **Inserts attendance entry into database** |
| **Admin Interface** | **GUI for managing users and reports** |
| **Database** | **Stores user and attendance information** |

**✅ Implementation Goals Achieved**

* **Contactless and fast attendance system**
* **Real-time face recognition**
* **Secure and persistent database records**
* **Easy-to-use interface for administrators**

**4.1 SYSTEM CODING**

**4.1.1. app.py**

import cv2

import os

from flask import Flask, request, render template

from datetime import date

from datetime import datetime

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

import pandas as pd

import joblib

app = Flask(\_\_name\_\_)

#number of images captured for training

nimgs = 10

# Attendence date

datetoday = date.today().strftime("%m\_%d\_%y")

datetoday2 = date.today().strftime("%d-%B-%Y")

#face detecting

face\_detector = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

#store the images

if not os.path.isdir('Attendance'):

os.makedirs('Attendance')

if not os.path.isdir('static'):

os.makedirs('static')

if not os.path.isdir('static/faces'):

os.makedirs('static/faces')

if attendance-{datetoday}.csv' not in os.listdir('Attendance'):

with open(attendance/Attendance-{datetoday}.csv', 'w') as f:

f.write('Name,Roll,Time')

#number of user registered

def totalreg():

return len(os.listdir('static/faces'))

# face extraction from image

def extract\_faces(img):

try:

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

face\_points = face\_detector.detectMultiScale(gray, 1.2, 5, minSize=(20, 20))

return face\_points

except:

return []

# Identify face using ML model

def identify\_face(facearray):

model = joblib.load('static/face\_recognition\_model.pkl')

return model.predict(facearray)

# function for train the model

def train\_model():

faces = []

labels = []

userlist = os.listdir('static/faces')

for user in userlist:

for imgname in os.listdir(f'static/faces/{user}'):

img = cv2.imread(f'static/faces/{user}/{imgname}')

resized\_face = cv2.resize(img, (50, 50))

faces.append(resized\_face.ravel())

labels.append(user)

faces = np.array(faces)

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(faces, labels)

joblib.dump(knn, 'static/face\_recognition\_model.pkl')

# Extract info from today's attendance file in attendance folder

def extract\_attendance():

df = pd.read\_csv(f'Attendance/Attendance-{datetoday}.csv')

names = df['Name']

rolls = df['Roll']

times = df['Time']

l = len(df)

return names, rolls, times, l

# Add Attendance of a user

def add\_attendance(name):

username = name.split('\_')[0]

userid = name.split('\_')[1]

current\_time = datetime.now().strftime("%H:%M:%S")

df = pd.read\_csv(f'Attendance/Attendance-{datetoday}.csv')

if int(userid) not in list(df['Roll']):

with open(f'Attendance/Attendance-{datetoday}.csv', 'a') as f:

f.write(f'\n{username},{userid},{current\_time}')

## A function to get names and roll numbers of all users

def getallusers():

userlist = os.listdir('static/faces')

names = []

rolls = []

l = len(userlist)

for i in userlist:

name, roll = i.split('\_')

names.append(name)

rolls.append(roll)

return userlist, names, rolls, l

# main page

@app.route('/')

def home():

names, rolls, times, l = extract\_attendance()

return render template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2)

# This function will run when we click on Take Attendance Button.

@app.route('/start', methods=['GET'])

def start():

names, rolls, times, l = extract\_attendance()

if 'face\_recognition\_model.pkl' not in os.listdir('static'):

return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2, mess='There is no trained model in the static folder. Please add a new face to continue.')

ret = True

cap = cv2.VideoCapture(0)

while ret:

ret, frame = cap.read()

if len(extract\_faces(frame)) > 0:

(x, y, w, h) = extract\_faces(frame)[0]

cv2.rectangle(frame, (x, y), (x+w, y+h), (86, 32, 251), 1)

cv2.rectangle(frame, (x, y), (x+w, y-40), (86, 32, 251), -1)

face = cv2.resize(frame[y:y+h, x:x+w], (50, 50))

identified\_person = identify\_face(face.reshape(1, -1))[0]

add\_attendance(identified\_person)

cv2.putText(frame, f'{identified\_person}', (x+5, y-5),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2)

cv2.imshow('Attendance', frame)

if cv2.waitKey(1) == 27:

break

cap.release()

cv2.destroyAllWindows()

names, rolls, times, l = extract\_attendance()

return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2)

# A function to add a new user.

@app.route('/add', methods=['GET', 'POST'])

def add():

newusername = request.form['newusername']

newuserid = request.form['newuserid']

userimagefolder = 'static/faces/'+newusername+'\_'+str(newuserid)

if not os.path.isdir(userimagefolder):

os.makedirs(userimagefolder)

i, j = 0, 0

cap = cv2.VideoCapture(0)

while 1:

\_, frame = cap.read()

faces = extract\_faces(frame)

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

cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)

cv2.putText(frame, f'Images Captured: {i}/{nimgs}', (30, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 20), 2, cv2.LINE\_AA)

if j % 5 == 0:

name = newusername+'\_'+str(i)+'.jpg'

cv2.imwrite(userimagefolder+'/'+name, frame[y:y+h, x:x+w])

i += 1

j += 1

if j == nimgs\*5:

break

cv2.imshow('Adding new User', frame)

if cv2.waitKey(1) == 27:

break

cap.release()

cv2.destroyAllWindows()

print('Training Model')

train\_model()

names, rolls, times, l = extract\_attendance()

return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2)

# Our main function which runs the Flask App

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**4.1.2. home.html**

<!doctype html>

<html lang="en">

<style type='text/css'>

\* {

padding: 0;

margin: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

background-image: url('https://cutewallpaper.org/21/1920-x-1080-gif/1920x1080-Wallpapercartoon-Wallpapers-Driverlayer-Search-.gif');

background-size: cover;

font-family: sans-serif;

margin-top: 40px;

height: 100vh;

padding: 0;

margin: 0;

}

table {

border: 1px;

font-family: arial, sans-serif;

border-collapse: collapse;

width: 86%;

margin: auto;

}

td,

th {

border: 1px solid black !important;

padding: 5px;

}

tr:nth-child(even) {

background-color: #dddddd;

}

</style>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

<title>Face Recognition Based Attendance System</title>

</head>

<body>

<div class='mt-3 text-center'>

<h1 style="width: auto;margin: auto;color: white;padding: 11px;font-size: 44px;">Face Recognition Based

Attendance System</h1>

</div>

{% if mess%}

<p class="text-center" style="color: red;font-size: 20px;">{{ mess }}</p>

{% endif %}

<div class="row text-center" style="padding: 20px;margin: 20px;">

<div class="col"

style="border-radius: 20px;padding: 0px;background-color:rgb(211,211,211,0.5);margin:0px 10px 10px 10px;min-height: 400px;">

<h2 style="border-radius: 20px 20px 0px 0px;background-color: #0b4c61;color: white;padding: 10px;">Today's

Attendance <i class="material-icons">assignment</i></h2>

<a style="text-decoration: none;max-width: 300px;" href="/start">

<button

style="font-size: 24px;font-weight: bold;border-radius: 10px;width:490px;padding: 10px;margin-top: 30px;margin-bottom: 30px;"

type='submit' class='btn btn-primary'>Take Attendance <i

class="material-icons">beenhere</i></button>

</a>

<table style="background-color: white;">

<tr>

<td><b>S No</b></td>

<td><b>Name</b></td>

<td><b>ID</b></td>

<td><b>Time</b></td>

</tr>

{% if l %}

{% for i in range(l) %}

<tr>

<td>{{ i+1 }}</td>

<td>{{ names[i] }}</td>

<td>{{ rolls[i] }}</td>

<td>{{ times[i] }}</td>

</tr>

{% endfor %}

{% endif %}

</table>

</div>

<div class="col"

style="border-radius: 20px;padding: 0px;background-color:rgb(211,211,211,0.5);margin:0px 10px 10px 10px;height: 400px;">

<form action='/add' method="POST" enctype="multipart/form-data">

<h2 style="border-radius: 20px 20px 0px 0px;background-color: #0b4c61;color: white;padding: 10px;">Add

New User <i class="material-icons">control\_point\_duplicate</i></h2>

<label style="font-size: 20px;"><b>Enter New User Name\*</b></label>

<br>

<input type="text" id="newusername" name='newusername'

style="font-size: 20px;margin-top:10px;margin-bottom:10px;" required>

<br>

<label style="font-size: 20px;"><b>Enter New User Id\*</b></label>

<br>

<input type="number" id="newusereid" name='newuserid'

style="font-size: 20px;margin-top:10px;margin-bottom:10px;" required>

<br>

<button style="width: 232px;margin-top: 20px;font-size: 20px;" type='submit' class='btn btn-dark'>Add

New User

</button>

<br>

<h5 style="padding: 25px;"><i>Total Users in Database: {{totalreg}}</i></h5>

</form>

</div>

</div>

</body>

</html>

System implementation is the **actual execution and deployment** of the system based on the design. It involves writing code, integrating components, testing modules, and deploying the system in a real environment.

**🧱 1. Technology Stack**

**💻 Programming Language:**

* **Python 3.x** – Main language for logic, computer vision, and database operations

**📸 Libraries and Tools:**

* OpenCV – For image processing and camera access
* face\_recognition – For facial encoding and matching
* NumPy – For numerical operations
* Tkinter or Flask/Django – For GUI or web-based UI
* SQLite/MySQL – For storing user and attendance records
* Pickle or Joblib – For storing face encodings locally

**🏗️ 2. Step-by-Step Implementation**

**📍 Step 1: Setup Environment**

Install required libraries:

bash

CopyEdit

pip install opencv-python face\_recognition numpy

pip install flask # If using a web interface

**📍 Step 2: Capture and Register Faces**

1. Use webcam to capture multiple face images of each user
2. Detect face using Haar cascade or HOG
3. Convert face into encoding (vector)
4. Store encoding with user ID/name

python

CopyEdit

import face\_recognition

import cv2

import pickle

# Capture image

video = cv2.VideoCapture(0)

ret, frame = video.read()

# Detect and encode face

face\_locations = face\_recognition.face\_locations(frame)

face\_encodings = face\_recognition.face\_encodings(frame, face\_locations)

# Store encoding

with open('encodings.pkl', 'wb') as f:

pickle.dump(face\_encodings, f)

**📍 Step 3: Face Recognition and Attendance Marking**

1. Load known encodings
2. Capture real-time frame from webcam
3. Detect and encode current face
4. Compare with known encodings
5. If match found, mark attendance

python

CopyEdit

import datetime

known\_encodings = pickle.load(open('encodings.pkl', 'rb'))

# Real-time capture

ret, frame = video.read()

unknown\_encoding = face\_recognition.face\_encodings(frame)[0]

# Compare

results = face\_recognition.compare\_faces(known\_encodings, unknown\_encoding)

if True in results:

name = "John Doe"

now = datetime.datetime.now()

# Save to DB or CSV

**📍 Step 4: Database Integration**

Use MySQL or SQLite to:

* Store user details
* Store attendance logs

sql

CopyEdit

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

face\_encoding BLOB

);

CREATE TABLE attendance (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

date TEXT,

time TEXT,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

Use sqlite3 in Python to insert and fetch data.

**📍 Step 5: User Interface (Optional)**

* Use Tkinter for a desktop GUI
* Or use Flask/Django for a web app

**Tkinter Example:**

python

CopyEdit

import tkinter as tk

root = tk.Tk()

root.title("Face Recognition Attendance")

tk.Button(root, text="Start Attendance", command=start\_recognition).pack()

root.mainloop()

**📍 Step 6: Testing**

* Test under different lighting conditions
* Try with partially occluded faces
* Check performance on similar-looking individuals

**Test Cases:**

| **Test Case** | **Expected Output** | **Result** |
| --- | --- | --- |
| Known face in front of camera | Attendance marked | ✅ |
| Unknown face | Show “Unknown face” | ✅ |
| Same user multiple times | Only one entry per day | ✅ |

**📍 Step 7: Deployment**

* Local deployment: Run on desktop/laptop
* Raspberry Pi: Use in embedded systems
* Cloud deployment: Deploy web app with cloud DB (optional)

**📊 3. Implementation Challenges and Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Varying light conditions | Use histogram equalization / image preprocessing |
| Similar looking faces | Use high-quality encoding (e.g., DNN-based face recognition) |
| Slow performance | Optimize with LBPH or GPU support |
| Data privacy | Encrypt stored data, use access control |

**🎯 4. Final Outcome**

After implementation:

* Attendance is marked automatically with high accuracy
* Admin can view/export reports
* System works without manual interference
* Reduces proxy and human error

**5.SUMMARY AND CONCLUSION**

**5.1. CONCLUSION**

Face recognition systems are part of facial image processing applications and their significance as a research area are increasing recently. Implementations of system are crime prevention, video surveillance, person verification, and similar security activities. The face recognition system implementation can be part of Universities. Face Recognition Based Attendance System has been envisioned for the purpose of reducing the errors that occur in the traditional (manual) attendance taking system. The aim is to automate and make a system that is useful to the organization such as an institute. The efficient and accurate method of attendance in the office environment that can replace the old manual

The **Face Recognition Based Attendance System** is a smart, efficient, and secure solution to automate the process of marking attendance using facial biometrics. This project leverages **computer vision** and **machine learning** technologies to detect and recognize human faces through a live video stream, and accurately mark attendance without any manual input.

The system was developed using **Python**, with tools like **OpenCV** for image processing and **face\_recognition** for facial matching. It incorporates a camera module to capture real-time images, processes them to detect faces, and compares the face encodings with a pre-registered database. If a match is found, the system logs the user’s attendance with a timestamp in a **relational database** like SQLite or MySQL.

The complete project was divided into different phases:

* **System Analysis**: Identified problems in the traditional/manual attendance system and proposed an automated solution.
* **System Design**: Outlined architecture, data flow, and database schema of the system.
* **System Implementation**: Developed and tested modules such as face registration, recognition, and attendance marking.

Additional features like an admin interface and exportable attendance reports were integrated to enhance usability. The system was tested under various conditions to ensure it performs reliably and consistently.

**🧾 Conclusion**

The implementation of the **Face Recognition Based Attendance System** proves to be a **reliable, contactless, and time-saving solution** compared to traditional attendance methods. It not only automates the attendance process but also helps eliminate problems like proxy attendance and human error.

This system demonstrates the practical application of artificial intelligence and computer vision in solving real-world problems. By utilizing open-source tools, the project remains cost-effective and accessible, making it ideal for institutions like schools, colleges, and offices.

While the system performs well, it also opens up avenues for future improvements such as:

* Improving recognition under poor lighting or occlusions (like masks)
* Adding features like mask detection or mobile app integration
* Deploying on cloud platforms for scalability and remote access

In conclusion, this project serves as a foundational step toward smarter and more secure attendance systems and showcases how **AI-powered automation** can improve everyday administrative tasks.

The **Face Recognition Based Attendance System** is a modern solution designed to **automate the attendance process** in schools, colleges, and workplaces using **facial recognition technology**. The system was built with Python and utilizes powerful open-source libraries like **OpenCV** and **face\_recognition** to detect and recognize human faces in real-time using a webcam.

During the project, the following steps were undertaken:

1. **Problem Analysis**  
   Identified issues with traditional attendance systems (manual errors, buddy punching, hygiene concerns).
2. **System Design**  
   Designed a modular, layered architecture that includes face capturing, recognition, database storage, and user interface.
3. **System Implementation**  
   Implemented the modules:
   * Face registration (encoding and saving)
   * Face detection and recognition
   * Attendance marking
   * Admin panel with reports
   * Data storage using SQLite or MySQL
4. **Testing & Evaluation**  
   The system was tested under various real-life conditions (different lighting, angles, face masks) to check its accuracy and reliability.
5. **Final Integration**  
   All modules were integrated into a fully working desktop application or web interface, providing seamless user experience.

**🧾 Key Features Achieved**

* ✅ **Contactless attendance** (especially useful in health-sensitive environments)
* ✅ **Automatic logging** with timestamps
* ✅ **Proxy prevention** using unique facial features
* ✅ **Fast and real-time detection**
* ✅ **Report export** for administrative use
* ✅ **User-friendly GUI or web interface**

**🏁 Conclusion**

This project successfully demonstrates how **AI and facial recognition technologies** can be applied to real-world problems like attendance tracking. By replacing manual or traditional biometric systems with this smart system, organizations can:

* **Save time** by automating repetitive tasks
* **Improve security** by eliminating fake attendance
* **Increase accuracy** by using unique biometric data
* **Reduce physical contact**, making it suitable during pandemics or in sensitive areas like hospitals

The implementation proves to be **cost-effective**, **efficient**, and **scalable**, making it ideal for deployment in educational institutions, offices, or even secure areas.

**💡 Final Thoughts**

While the current version is functional and reliable, there is always room for improvement. Incorporating more advanced AI models, cloud storage, or mobile-based attendance can take this system to the next level (as explained in the future scope).

This project not only solves a real-world problem but also offers a strong foundation to learn and work with modern technologies like:

* **Computer Vision**
* **Artificial Intelligence**
* **Database Management**
* **Software Engineering**

**5.2. FUTURE SCOPE**

The future of face recognition attendance systems is brimming with potential for advancements in technology, wider applications, and even addressing current limitations. Here's a glimpse into what the future might hold:

1. Enhanced Accuracy and Performance

* Deeper Learning Algorithms: Continued development of deep learning algorithms with larger and more diverse datasets can lead to significantly improved recognition accuracy, even under challenging conditions.
* Liveness Detection Advancements: More sophisticated liveness detection techniques like iris recognition or micro-expression analysis can further strengthen security against spoofing attempts.

2. Integration and Automation

* Seamless Integration: Expect smoother integration with existing student information systems and access control systems, creating a more streamlined workflow.
* Attendance Analysis: Advanced analytics can be built upon attendance data to identify trends, predict absenteeism, and personalize learning approaches.

3. Addressing Limitations

* Privacy-Enhancing Techniques: Developments like federated learning or differential privacy can offer improved data security and user privacy while maintaining the benefits of facial recognition.
* Bias Mitigation: Techniques to address bias in facial recognition algorithms are being actively researched, aiming for fairer and more inclusive systems.

4. Wider Applications

* Beyond Attendance: The technology could extend to personalized learning environments, exam proctoring with identity verification, or secure access to restricted areas within educational institutions.
* Remote Learning Enhancements**:** Facial recognition could be used for attendance verification and engagement monitoring in remote learning environments.

5. Ethical Considerations

* Transparency and User Control: Increased transparency about data collection, usage, and storage will be crucial for building trust with users.
* Regulations and Standards: Expect more robust regulations and ethical guidelines to govern the development and deployment of facial recognition technology.

The current implementation of the Face Recognition Based Attendance System offers an efficient and contactless solution for marking attendance. However, technology is constantly evolving, and this system can be significantly enhanced in the future to make it more intelligent, scalable, and widely applicable.

The Face Recognition Based Attendance System developed in this project provides a **strong foundation** for automation, biometrics, and artificial intelligence in real-world applications. However, the field of face recognition and its integration with attendance systems is **rapidly evolving**, and many enhancements can be made to improve functionality, accuracy, and usability.

Below are the key future directions:

**🚀 1. Mobile Application Integration**

**Current Limitation**: System works only on desktop/laptop with webcam.

**Future Enhancement**:

* Develop a mobile app version (Android/iOS) using React Native or Flutter.
* Use the phone camera to mark attendance.
* This increases portability and usability, especially for field workers or remote employees.

**☁️ 2. Cloud-Based Architecture**

**Current Limitation**: Local database (SQLite/MySQL) may not scale for large institutions.

**Future Enhancement**:

* Store face encodings, images, and attendance logs in the cloud.
* Use cloud services like AWS, Google Firebase, or Azure for:
  + Centralized data access
  + Scalability for hundreds or thousands of users
  + Remote access and management by admins

**🧠 3. AI-Powered Face Recognition Improvements**

**Current Limitation**: Uses basic face recognition (Dlib or OpenCV-based).

**Future Enhancement**:

* Integrate deep learning models (e.g., FaceNet, VGG-Face, or ArcFace) for higher accuracy.
* Handle challenging cases like:
  + Low lighting
  + Face masks
  + Aging
  + Glasses or headgear
* Use emotion detection or attention tracking in educational settings.

**🕵️ 4. Liveness Detection / Anti-Spoofing**

**Security Risk**: System may be fooled by printed photos or videos.

**Future Enhancement**:

* Add **liveness detection** to ensure the face is real and not a photo.
* Techniques:
  + Eye blink detection
  + Head movement tracking
  + Infrared camera or depth sensing (like RealSense)

**📊 5. Advanced Analytics and Dashboard**

**Current Limitation**: Only stores and shows attendance data.

**Future Enhancement**:

* Build a **dashboard** for analytics:
  + Attendance trends (daily, weekly, monthly)
  + Department-wise or class-wise summaries
  + Export reports in various formats (PDF, Excel)
* Useful for HR or school administration

**🔐 6. Multi-Factor Authentication (MFA)**

**Security Enhancement**:

* Combine face recognition with other biometric or identity verification:
  + RFID card + face match
  + Face + OTP on phone
  + Face + voice recognition

**🧍 7. Multiple Faces and Group Attendance**

**Future Enhancement**:

* Detect and recognize multiple people in the same frame
* Automatically log attendance for entire groups (e.g., classrooms or meeting rooms)
* Useful in schools, exam halls, or team meetings

**🌐 8. Integration with Other Systems**

**Future Enhancement**:

* Integrate with:
  + HRMS (Human Resource Management Systems)
  + LMS (Learning Management Systems)
  + Payroll systems (for salary based on presence)
  + Government identity databases (for verification)

**🧪 9. Real-Time Notifications and Alerts**

**Enhancement**:

* Send alerts via SMS/email to:
  + Employees/students (on attendance mark)
  + Parents (if a student is absent)
  + Admins (if unauthorized entry detected)

**🌍 10. Multilingual and Accessibility Support**

**Enhancement**:

* Add support for multiple languages in the GUI
* Make the interface accessible for visually/hearing impaired users

**🧾 Summary of Future Scope**

| **Area** | **Enhancement Idea** |
| --- | --- |
| Mobile Access | Android/iOS support |
| Cloud Integration | Centralized data and remote access |
| AI Models | Improved recognition accuracy |
| Security | Anti-spoofing, liveness detection |
| Analytics | Dashboard with reports and insights |
| Multi-User Detection | Group attendance in real-time |
| System Integration | HR, LMS, payroll, identity systems |
| Alerts | Email/SMS notifications |

**✅ Final Thought**

The Face Recognition Attendance System is not just a tech demo—it can evolve into a **real-world commercial solution**. By leveraging the above future scopes, the system can be deployed in:

* Universities and schools
* Offices and factories
* Airports and secure buildings
* Events and public gatherings

This makes it a **scalable**, **secure**, and **intelligent attendance solution** for the future.

Below is a detailed discussion of the **future scope** of the project:

**🔧 1. Improved Accuracy with Advanced AI Models**

* The current system uses standard models (e.g., HOG or LBPH) for face recognition.
* In the future, **deep learning** models like **CNNs (Convolutional Neural Networks)**, **FaceNet**, or **Dlib ResNet** can be integrated for higher accuracy, especially in complex environments with variable lighting, head orientation, or occlusions (e.g., glasses, masks).
* **3D face recognition** can be used to recognize faces from multiple angles.

**☁️ 2. Cloud-Based and IoT Integration**

* The system can be integrated with cloud services (like **AWS**, **Firebase**, or **Google Cloud**) to:
  + Store face data and attendance records remotely.
  + Enable **real-time attendance tracking** from multiple locations.
  + Generate reports accessible from any device.
* **IoT devices** like **Raspberry Pi** can be used for embedded deployment in classrooms or entry gates, making the system compact and portable.

**📱 3. Mobile Application Support**

* A dedicated **mobile app** can be developed for:
  + Admins to view real-time attendance reports.
  + Students/employees to check their own attendance.
  + Face recognition via smartphone cameras for remote learning or work-from-home scenarios.

**🧑‍🏫 4. Integration with Smart Campus/Office Systems**

* The system can be a core component of a **Smart Campus** or **Smart Office** by integrating with:
  + Classroom management systems
  + Access control systems (e.g., automatic door opening for authorized faces)
  + Event management systems (e.g., attendance at seminars, exams)

**🎭 5. Multi-Factor Authentication (MFA)**

* For enhanced security, the system can incorporate **multi-factor authentication**, such as:
  + Face + RFID
  + Face + OTP
  + Face + Voice recognition

This prevents impersonation and strengthens data integrity in high-security environments.

**🔍 6. Advanced Analytics and Dashboard**

* A detailed dashboard can be created for:
  + Attendance trends and statistics
  + Monthly/yearly reports
  + Notifications for absenteeism or late arrivals
  + Auto-generated alerts and insights using AI

This can help management make data-driven decisions.

**🌍 7. Multilingual and Inclusive Features**

* Add support for **multiple languages** in the user interface to increase accessibility.
* Incorporate accessibility features for **visually or hearing-impaired users**, using voice commands or text-to-speech tools.

**🛡️ 8. Privacy, Ethics, and Legal Compliance**

* As data privacy laws (like **GDPR** or **India’s DPDP Act**) evolve, the system must be enhanced with:
  + **Data encryption**
  + User consent mechanisms
  + Data retention policies
* A future version can include **face data anonymization** techniques or federated learning models to protect user identity while still enabling smart analytics.

**👨‍👩‍👧‍👦 9. Scalability for Large Institutions**

* Optimize the system to support **thousands of users**.
* Use **distributed databases** and **load balancers** to handle large data volumes and multiple camera feeds in real-time.

**📚 10. Use in Other Domains**

The same system can be adapted for various industries:

* **Healthcare**: Patient check-in at hospitals
* **Retail**: Staff attendance and customer analysis
* **Events**: Automatic participant tracking
* **Transport**: Entry/exit logging in buses, trains, or airports

**✨ Summary of Future Enhancements**

| **Area** | **Future Enhancement** |
| --- | --- |
| Accuracy | Deep learning, 3D face models |
| Scalability | Cloud, IoT, multi-location |
| Access | Mobile app, web-based control |
| Features | Dashboard, analytics, integration |
| Security | Multi-factor authentication |
| Privacy | Encryption, legal compliance |

**✅ Final Thought**

The **Face Recognition Based Attendance System** is not just a project—it’s a **foundation for real-world AI solutions**. With future advancements, it has the potential to become an integral part of **smart infrastructure**, delivering seamless automation, safety, and efficiency across multiple sectors.

**6.REFERENCES**

 OpenCV Documentation: <https://opencv.org/>

 dlib Documentation: <https://github.com/davisking/dlib>

 Research Papers: <https://openai.com/blog/how-should-ai-systems-behave>