**FACE RECOGNITION ATTENDANCE SYSTEM**

**SUBMITTED**

**BY**

**K.SURYA**

**22COS64**

Under the guidance of

**Mrs. Mary Monica. J.**

Submitted to

**The Department of Computer Science (UG)**

**FOR THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE**

**AWARD OF THE DEGREE OF**

**B.Sc., COMPUTER SCIENCE**



**THE AMERICAN COLLEGE**

**(An Autonomous Institution Affiliated to Madurai Kamaraj University)**

**Re-accredited with (3rd cycle) by NAAC with Grade “A+”, CGPA – 3.47 on a 4-Point scale**

**MADURAI - 625002**

**APRIL -2025**

**THE AMERICAN COLLEGE**

**(An Autonomous Institution Affiliated to Madurai Kamaraj University)**

**Re-accredited with (3rd cycle) by NAAC with Grade “A+”, CGPA – 3.47 on a 4-Point scale**

**MADURAI - 625002**

**APRIL - 2025**



**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled “FACE RECOGNITION ATTENDANCE SYSTEM” is a bonafide record of work done by K.SURYA(22COS64) in partial fulfillment of the requirements for the award of the degree of B.Sc., Computer Science of The American College. This work has not been submitted for the award of any degree to the best of our knowledge.

Submitted for viva-voice examination, held on \_\_\_\_\_\_\_\_\_\_\_\_, in the Department of Computer Science, The American College, Madurai.

Internal Guide Head of the Department

**DECLARATION**

I hereby declare that the project entitled “FACE RECOGNITION ATTENDANCE SYSTEM” is a project report of the original work done by myself. This project work is submitted to The American College (Affiliated to Madurai Kamaraj University) in partial fulfillment of the degree of Bachelor of Science in Computer Science during the academic year 2022-2025.

I declare that this project work or any part thereof has not been submitted for getting any degree or diploma from any other university or college.

Place : Madurai

Date :

Signature of the Student

|  |  |  |
| --- | --- | --- |
| S.NO | TITLE | PAGE.NO |
| 1 | **ABSTRACT** | **5,6** |
| 2 | **SYSTEM REQUIREMENT**  Hardware Requirements  Software Requirements | **7-9** |
| 3 | **SYSTEM ANALYSIS**  Existing System  Proposed System | **9-12** |
| 4 | **SYSTEM DESIGN**  Dataflow Diagram  Module Design  Table Design | **12-17** |
| 5 | **MODULE DESCRIPTION** | **17-19** |
| 6 | **SYSTEM TESTING**  Types of Testing | **19-22** |
| 7 | **CONCLUSION** | **23** |
| 8 | **FUTURE ENHANCEMENT** | **24-28** |
| 9  10 | **SCREEN PLATES**  **SOURCE CODES** | **29-32**  **33-40** |
| 11 | **BIBLIOGRAPHY** | **41** |

Project Abstract: Face Recognition Attendance System

The **Face Recognition Attendance System** is an automated solution designed to enhance the efficiency, accuracy, and security of attendance tracking in educational institutions using advanced **facial recognition technology**. Traditional attendance methods, such as manual roll calls, RFID-based systems, and fingerprint scanners, have several drawbacks. These methods are often **time-consuming, error-prone, and susceptible to manipulation**, such as proxy attendance, where one student marks attendance for another. To overcome these challenges, the proposed system leverages cutting-edge machine learning techniques to create a more **reliable, secure, and automated attendance solution**.

This system utilizes **MediaPipe**, an advanced machine learning framework, to **detect and recognize faces in real time**, ensuring fast and accurate attendance marking. The system captures students' facial images through an **IP webcam**, which allows seamless integration with both local and cloud-based databases. The facial recognition process is performed using **MediaPipe’s Face Detection and Face Mesh modules**, which analyze unique facial features and match them against pre-stored data in the system. Unlike traditional biometric systems, which require physical contact or card-based verification, this face recognition approach ensures a **completely contactless** and **hygienic** experience.

A key feature of this system is its ability to **detect and prevent proxy attendance**. By analyzing **unique facial landmarks**, the system can differentiate between individuals and prevent students from using printed photos, mobile screens, or recorded videos to manipulate attendance records. Additionally, the system incorporates **duplicate attempt detection**, which flags **suspicious activities**, such as multiple faces appearing within the camera frame simultaneously or the same student attempting to mark attendance more than once for different individuals.

Once a **valid face is detected and authenticated**, the attendance is **automatically recorded and stored in a CSV file**, which can be later accessed by faculty members for review. The system also **supports subject-wise attendance marking**, allowing students to be recorded present only for the relevant lecture or session. This feature ensures **greater flexibility** and **accuracy** in tracking attendance across multiple subjects and class schedules.

To enhance usability, the **Face Recognition Attendance System** is designed as a **web-based application** with an intuitive and user-friendly interface. This enables faculty members and administrators to access attendance records from anywhere, eliminating the need for physical attendance registers or manual data entry. Furthermore, the system provides **real-time attendance monitoring**, allowing institutions to track student participation instantly.

One of the major advantages of this system is its **scalability and adaptability**. It can be deployed across multiple classrooms, lecture halls, and even large-scale educational campuses without requiring additional infrastructure. The system can also be **customized to integrate with existing Learning Management Systems (LMS)**, enabling a seamless connection between attendance data and academic records.

Beyond educational institutions, the **Face Recognition Attendance System** has potential applications in various fields, including **corporate offices, conferences, training centers, and government institutions**, where automated attendance monitoring can enhance operational efficiency. By leveraging the power of **machine learning and artificial intelligence**, this system not only **eliminates human errors** but also **ensures a secure, efficient, and fraud-resistant attendance management process**.

In conclusion, this system **redefines traditional attendance tracking methods** by integrating **real-time facial recognition with intelligent proxy detection mechanisms**. By offering a **highly secure, efficient, and automated** approach, it significantly **improves classroom discipline, reduces administrative workload, and enhances the overall experience** for both students and faculty members. With its **contactless, scalable, and AI-driven** approach, this system represents a **technological advancement** in attendance management, making it a valuable asset for modern educational institutions.

SYSTEM REQUIRMENTS:

Hardware Requirments:

1. **Processor:**
   * **Minimum:** Intel Core i3 / AMD Ryzen 3
   * **Recommended:** Intel Core i5 or higher / AMD Ryzen 5 or higher
   * A multi-core processor is essential for **efficient real-time face detection and recognition**.
2. **RAM:**
   * **Minimum:** 4GB
   * **Recommended:** 8GB or more
   * Higher RAM ensures **smooth video processing** during face recognition.
3. **Storage:**
   * **Minimum:** 256GB HDD / SSD
   * **Recommended:** SSD (Solid State Drive) for better performance
   * Required for storing **datasets, attendance records, and system files**.
4. **Camera (Webcam - IP or USB):**
   * **Preferred:IP Webcam** (for real-time face capture via mobile or network-based camera).
   * **Alternative:USB Webcam** (a high-resolution camera if an IP webcam is unavailable).
5. **Graphics Processing Unit (GPU) (Optional but Recommended):**
   * A dedicated GPU (e.g., **NVIDIA GTX 1050 or better**) can significantly **improve processing speed**, especially for large datasets.
6. **Internet Connection (For Web-Based System):**
   * Required **if using an IP webcam** or **cloud-based storage**.

**7. Power Supply:**

* + A **stable and reliable** power source to ensure **uninterrupted system operation** during attendance tracking

Software Requirments:

**Operating System:**

* **Windows 10/11 (64-bit) (Recommended)**
* **Linux (Ubuntu 20.04 or higher)** (For better performance and flexibility)

**Programming Language:**

* **Python 3.10.10** (or the latest stable version)

**Libraries & Dependencies:**

* **MediaPipe** – For face detection and recognition
* **OpenCV** – For image processing and webcam integration
* **NumPy** – For numerical operations
* **Pandas** – For handling attendance data in CSV format

**Web Technologies (For Web Interface):**

* **Frontend:** HTML, CSS, JavaScript
* **Backend (if needed):** Flask / FastAPI

**Database (Optional, if using a structured database):**

* **SQLite / MySQL / PostgreSQL** (For storing attendance records instead of CSV)

**Development Environment:**

* **Python IDLE / PyCharm / VS Code** (For coding and debugging)

**Browser Compatibility:**

* **Google Chrome / Mozilla Firefox / Microsoft Edge** (For accessing the web interface)

SYSTEM ANALYSIS:

Existing System:

**Time-Consuming**

* Manual roll calls require the instructor to call out each student’s name individually and wait for their response. This process can be extremely time-consuming, particularly in **large classrooms with hundreds of students**.
* Even in smaller classes, valuable teaching time is wasted on attendance-taking rather than focusing on the actual lesson.

**Human Errors**

* Errors in attendance marking are common in traditional systems. Instructors may accidentally **mark a student as present or absent incorrectly**, leading to discrepancies in attendance records.
* Additionally, paper-based attendance registers are prone to **misplacement, damage, or loss**, making it difficult to retrieve accurate records when needed.

**Proxy Attendance (Fake Attendance)**

* One of the most significant challenges in traditional systems is **proxy attendance**, where a student **answers for an absent classmate** or misuses an ID card or biometric device.
* This unethical practice affects classroom discipline and compromises the **integrity of attendance records**, making it difficult to track actual student participation.

**Dependency on Physical Cards or Devices**

* RFID-based attendance systems require students to carry **ID cards with embedded chips**. If a student forgets, loses, or damages their ID card, they may be **unable to mark their attendance**.
* Replacing lost or damaged ID cards incurs additional costs for both students and institutions, making the system **less efficient and more resource-intensive**.

**Hygiene Concerns (Especially in Post-Pandemic Scenarios)**

* Biometric fingerprint systems require **physical contact** with a shared scanner, leading to hygiene concerns.
* In a **post-pandemic world**, institutions must prioritize **contactless and hygienic** solutions to minimize the risk of spreading infections through frequently touched surfaces.
* Additionally, fingerprint sensors may malfunction due to **dirt, moisture, or damage**, causing inconvenience and requiring frequent maintenance.

Proposed System:

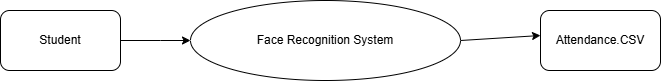
The **proposed system** incorporates the following key features and improvements:

1. **Automated Attendance Marking**
   * The system automatically captures and processes students' faces using an **IP webcam**, eliminating the need for **manual intervention** by teachers.
   * As students enter the classroom, their faces are detected, recognized, and matched against a **pre-stored database** in real time.
   * This significantly **reduces the time spent on attendance**, allowing instructors to **focus more on teaching** rather than administrative tasks.
   * Unlike **ID card scanning or fingerprint-based systems**, students do not need to **carry additional devices** or interact with any external hardware.
2. **Proxy Detection (Prevention of Fake Attendance)**
   * To **ensure that only the actual student is present**, the system **analyzes unique facial landmarks**, which are distinct for every individual.
   * **Advanced anti-spoofing measures** are implemented to detect **suspicious activities**, such as attempts to use **printed photos, digital screens, or recorded videos** for proxy attendance.
   * The system also flags cases where a **single student attempts to mark attendance multiple times** within a short period, preventing fraudulent attendance marking.
   * Unlike traditional methods, where students can **easily answer for absent classmates**, this system ensures **100% authenticity** in attendance records.
3. **Contactless System for Enhanced Hygiene and Convenience**
   * Unlike fingerprint-based biometric systems, which require **physical contact** with a shared scanner, the **Face Recognition Attendance System** is entirely **contactless**.
   * This eliminates hygiene concerns, particularly in **post-pandemic** environments where institutions **prioritize health and safety**.
   * Since there is no need for **physical interaction with hardware**, the system is **more durable, requires less maintenance**, and operates efficiently **without wear and tear**.
4. **Real-Time Processing for Instant Attendance Marking**
   * The system uses **MediaPipe’s Face Detection and Face Mesh modules**, which enable **fast and accurate** face detection and recognition.
   * Attendance is marked **instantly**, ensuring that **large classrooms** with many students can be managed without delays.
   * The use of **optimized deep learning models** ensures that the system performs well **even on devices with moderate hardware specifications**.
   * Since the system captures faces in **real time**, there is no risk of **delayed attendance marking or post-class modifications**, ensuring **accuracy and transparency**.
5. **Web-Based Interface for Accessibility and Ease of Use**
   * The system provides a **user-friendly web interface** that allows administrators and faculty members to **view and manage attendance records** effortlessly.
   * Through the web-based dashboard, users can:
     + **Monitor** real-time attendance updates.
     + **Generate reports** for individual students or entire classes.
     + **Filter attendance records** based on date, subject, or session.
   * Since the system is **web-based**, it can be accessed **from any device**, including laptops, tablets, and mobile phones, offering greater **convenience and flexibility**.
6. **CSV-Based Storage for Efficient Data Management**
   * Attendance data is **stored in CSV format**, making it **easy to manage, analyze, and export** for further processing.
   * CSV-based storage offers several advantages:
     + It is **lightweight and does not require a database** for small-scale implementations.
     + Attendance records can be **easily opened in Excel** or other data processing tools for further analysis.
     + The system remains **fast and efficient**, even with a large number of attendance entries.
   * For institutions requiring **more advanced storage options**, the system can be integrated with **structured databases like MySQL, SQLite, or PostgreSQL** for **scalability and better data organization**.

SYSTEM DESIGN:

Dataflow Diagram:

DFD Level 0 – High-Level System Overview



 The student interacts with the system.

 The system recognizes the face and marks attendance.

 The attendance is stored in the CSV file.

DFD Level 1 – Dataset Creation Process

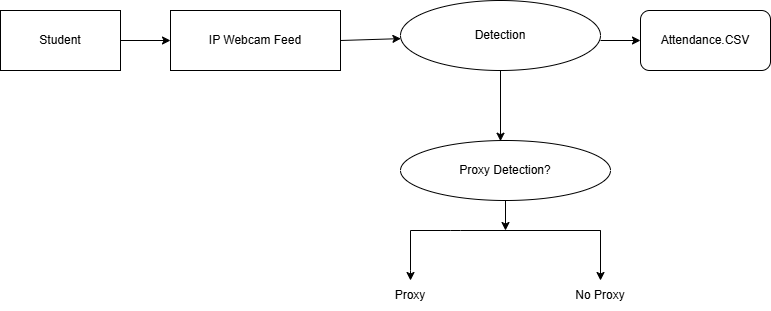
## Untitled Diagram.drawio (5).png

**1 Student provides face image** (via IP Webcam)

2 **Face is detected using Mediapipe**

3 **If valid, the face is saved in the dataset**

DFD Level 1 – Recognition & Attendance Process



**Module Design (Your Existing Program)**

The Face Recognition Attendance System is developed using a modular design approach to ensure clarity, maintainability, and ease of integration. Each module is designed to perform a specific task that contributes to the overall functionality of the system. The major modules in the system are as follows:

**1. Dataset Creation Module**

**Purpose:**  
The purpose of this module is to create a dataset by capturing facial images of students. These images will be used later for training the facial recognition model.

**Tools Used:**

* MediaPipe

**Functionality:**  
This module plays a crucial role in the initial setup of the system. It captures 40 face images for each student using an IP webcam, preferably a mobile device running an IP Webcam app. The images are captured in real-time, ensuring natural lighting and angle variations for better model training. The system allows the entry of the student’s name and roll number, which are then used to label and organize the images. These images are stored in individual folders for each student, which helps in structured dataset management and makes it easier to train the face recognition model later.

**2. Face Recognition Module**

**Purpose:**  
The goal of this module is to detect and recognize student faces from the live camera feed.

**Tools Used:**

* MediaPipe
* face\_recognition (based on dlib and deep learning)

**Functionality:**  
This module is responsible for identifying students by analyzing their facial features in real-time. It uses MediaPipe for initial face detection due to its high speed and efficiency, and face\_recognition for accurate face matching. The system loads the previously created dataset and generates facial encodings for each student. When a new face appears in the webcam feed, it is compared with these encodings. If a match is found, the module retrieves the corresponding student’s name and roll number. This identification process is fast, contactless, and reliable, ensuring that the system can recognize students with minimal delay.

**3. Attendance Marking Module**

**Purpose:**  
This module is designed to mark attendance for students on a subject-wise basis.

**Functionality:**  
After successful recognition, the system checks whether the student has already been marked present for the selected subject during the current session. This prevents duplicate entries and ensures accurate recordkeeping. If the student is not already marked, the system records their name, roll number, subject, date, and timestamp into a CSV file. Each subject has its own CSV file or a dedicated column in a master file. The module ensures that attendance is logged only once per subject per day for each student, thereby maintaining data integrity and avoiding proxy or repeated entries.

**4. Web Interface Module**

**Purpose:**  
This module enables the entire system to operate through a user-friendly web interface, eliminating the need for standalone desktop GUIs.

**Tools Used:**

* Python web framework (e.g., Flask)
* HTML, CSS, and JavaScript for front-end design

**Functionality:**  
The web interface is designed to be simple, intuitive, and functional. When the system starts, it first displays a login screen accessible only to the administrator. Upon successful login, the user is presented with a dashboard containing subject buttons (e.g., Maths, Physics, Chemistry, etc.). After selecting a subject, the system opens a live webcam feed titled “Face Recognition” using the IP webcam. The face recognition process begins automatically, and attendance is marked in real-time without requiring any further user action. This module ensures platform independence and can be accessed from any browser, offering portability and ease of deployment.

**Table Design**

**1. Attendance Table (CSV Format)**

The attendance data for students recognized by the face recognition system is stored in a simple, lightweight **CSV (Comma-Separated Values)** format. This format is ideal for small to medium-sized datasets and is easy to handle using Python for both reading and writing operations. The table is automatically updated whenever a student is recognized and their attendance is marked.

**Table Structure**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| Name | String | Name of the recognized student |
| Timestamp | DateTime | The exact date and time when attendance was marked |

**Description of Fields**

* **Name**:  
  This field stores the full name of the student as recognized by the system through face recognition. The name should match the one provided during the dataset creation phase.
* **Timestamp**:  
  This field logs the date and time when the student's face was detected and recognized successfully. It helps track the exact moment attendance was marked, which is useful for subject-wise or period-wise record keeping.

Example CSV File (attendance.csv)

Name,Timestamp

Aarav Sharma,2025-04-05 09:00:14

Priya Verma,2025-04-05 09:01:22

Rohit Kumar,2025-04-05 09:02:07

This file is updated in real time by the system during classroom sessions. Each new recognized student is appended as a new row, and duplicate entries for the same subject and period are prevented using internal logic in the attendance marking module.

**Module Description**

**1. Dataset Creation Module:**

**Description:**  
The Dataset Creation Module is responsible for capturing facial images of students through an IP webcam. It uses **MediaPipe**, a powerful and lightweight face detection library, to accurately detect and crop faces. Each captured image is saved in a well-organized folder structure to prepare a training dataset for the face recognition module. This step is essential to ensure that the recognition system can reliably identify each student.

**Key Features:**

* Captures approximately **40 face images** per student to improve recognition accuracy.
* Saves images in a structured directory format like dataset/Name/, making it easy to manage individual student datasets.
* Ensures that **only one face** is present in the camera frame during image capture to avoid mislabeling or incorrect training.

**2. Face Recognition Module:**

**Description:**  
This module handles the real-time identification of students using a live camera feed. It integrates **MediaPipe** for precise face detection and uses the **face\_recognition** library, which is based on **dlib**, to perform face encoding and matching. When a student’s face is recognized, the module displays their name and roll number, which are fetched from the pre-encoded data.

**Key Features:**

* Performs **real-time detection and recognition** of faces using efficient algorithms.
* Compares detected faces against the **previously stored encodings** in the system.
* Displays the **recognized student's name** on the screen for easy verification by the admin or teacher.

**3. Attendance Marking Module:**

**Description:**  
Once a student has been recognized, this module is triggered to log their attendance. The system saves the student’s name along with a timestamp into a CSV file. To avoid incorrect entries, the module checks whether the student has already been marked present for that session. This ensures that **attendance is only marked once** per subject or session.

**Key Features:**

* Ensures **unique attendance entry** per student for each session to avoid duplication.
* Automatically saves attendance data in a CSV file named **attendance.csv**.
* Each record in the CSV includes:
  + **Name** of the student
  + **Timestamp** indicating the exact time of recognition

**4. Web Interface Module:**

**Description:**  
This module provides a **user-friendly web interface** to manage the face recognition system. Designed using basic web technologies and a Python backend like **Flask**, it enables the admin to log in and initiate the face recognition process. Once logged in, the interface displays a live feed from the IP webcam under the title **“Face Recognition”**, and attendance marking begins automatically.

**Key Features:**

* Provides an **admin login system** (can be hardcoded or simplified using local validation).
* Runs in the **browser without XAMPP or any database setup**, making it lightweight and portable.
* Integrates **IP webcam feed** for live face recognition within the browser itself.
* After selecting the subject, the system **automatically starts recognition** without requiring further interaction.

**Types of Testing :**

Thorough testing is an essential phase in software development to ensure that the system is functioning as expected, reliable under various conditions, and meets the defined requirements. The Face Recognition Attendance System undergoes multiple types of testing to verify the correctness, performance, security, and usability of the entire system. The testing methods applied in the project are described in detail below:

**1. Functional Testing**

**Purpose:**  
Functional testing verifies that each function of the system operates in accordance with the requirement specifications. It ensures that all features perform as intended without any logical or runtime errors.

**Application in the Project:**  
In the Face Recognition Attendance System, functional testing includes verifying that face detection works accurately using the IP webcam. It ensures that each detected and recognized face corresponds correctly to a student and that the attendance is correctly recorded in the respective CSV file. Additionally, the functionality of the web interface—such as logging in, viewing subject buttons, and starting the recognition session—is tested to ensure seamless operation.

**2. Unit Testing**

**Purpose:**  
Unit testing focuses on testing individual components or functions of the software in isolation. This helps in identifying bugs at an early stage and ensures that each module behaves correctly on its own.

**Application in the Project:**  
Each module of the system is tested independently. For instance, the face detection functionality is tested using MediaPipe to confirm that faces are being accurately located in real-time video frames. Similarly, the recognition module is tested to verify that the correct student name and roll number are returned when a known face is detected. Unit tests are also written for utility functions that handle file operations or CSV data entry.

**3. Integration Testing**

**Purpose:**  
Integration testing checks whether different modules of the application interact properly when combined. This ensures that data is correctly passed and processed between components.

**Application in the Project:**  
This testing ensures that the dataset creation module integrates well with the face recognition module. For example, after images are stored, the recognition module should be able to access and encode them without issues. Furthermore, it verifies that once a student is recognized, the attendance module correctly stores the data without conflicts. Smooth interaction between the recognition module and the attendance logging mechanism is vital for overall functionality.

**4. Performance Testing**

**Purpose:**  
Performance testing evaluates the system’s responsiveness, stability, and speed under different conditions, including real-time usage scenarios.

**Application in the Project:**  
The system is tested to measure how quickly and efficiently it can detect and recognize faces, particularly when multiple students are present in the camera frame. The performance is monitored to ensure that there are no lags or significant delays during face detection and attendance marking. This ensures that the system remains usable even in large classrooms or under network strain when using IP webcams.

**5. Usability Testing**

**Purpose:**  
Usability testing ensures that the system is easy to use, intuitive, and accessible for the target users, such as teachers or administrators.

**Application in the Project:**  
The system’s web interface is tested for ease of navigation. The login process is examined for clarity, and the visibility of subject buttons after logging in is verified. The layout and design of the web interface are evaluated to ensure that any user, regardless of technical expertise, can operate the system without confusion. Clear instructions, error messages, and interface simplicity are key aspects assessed during this testing phase.

**6. Security Testing**

**Purpose:**  
Security testing identifies vulnerabilities and ensures that the system is protected against unauthorized access, data theft, or misuse.

**Application in the Project:**  
In this project, only authorized administrators are permitted to access the attendance system through a secure login interface. This prevents students or unauthorized users from tampering with the data. Additionally, security measures are considered for protecting the stored face image data and attendance records, ensuring they are not exposed to public access or vulnerable directories.

**7. Regression Testing**

**Purpose:**  
Regression testing ensures that any new updates or code modifications do not affect the existing functionalities of the system.

**Application in the Project:**  
As the system evolves, for example by integrating the face\_recognition library alongside MediaPipe, regression testing is conducted to confirm that the previous functionalities—such as face detection and attendance marking—continue to operate without bugs. This testing is vital to maintaining system stability after enhancements or bug fixes.

**8. Acceptance Testing**

**Purpose:**  
Acceptance testing verifies whether the overall system meets the project objectives and client or user expectations. It is usually the final phase of testing before deployment.

**Application in the Project:**  
The Face Recognition Attendance System is run in a simulated or actual classroom environment to determine its effectiveness in a real-use scenario. During this testing, the entire process—from logging in, selecting subjects, recognizing faces, and marking attendance—is observed to validate whether the system performs as required. If the system operates successfully under these conditions, it is deemed ready for deployment.

**Conclusion:**

The **Face Recognition Attendance System** represents a significant advancement over traditional methods of attendance tracking. By integrating **MediaPipe** for efficient and lightweight face detection and **OpenCV** for image processing and webcam interfacing, the system successfully automates the attendance process. It eliminates the need for manual roll calls, ID card scanning, or physical biometric devices. The system is designed to recognize student faces in real time through an **IP webcam**, accurately marking attendance in a **CSV file** with the corresponding name and timestamp.

One of the key strengths of this system lies in its focus on **security and reliability**. Through **security testing**, special attention has been given to the prevention of **proxy attendance**. The system has been tested to ensure that it does not accept attendance from **printed photographs**, **mobile screens**, or **pre-recorded videos**, thereby ensuring that only the legitimate student is recognized and marked present. This adds a critical layer of trust and integrity to the system, especially in academic environments where attendance records impact student assessments.

Additionally, the system is designed to function effectively across various **lighting conditions** and environmental setups, ensuring **consistent recognition performance**. It also prevents **duplicate entries** within the same session, making it ideal for classroom use across multiple periods or subjects. The **real-time face detection** capability ensures that attendance is marked instantly without requiring any manual intervention after the initial setup.

From a usability perspective, the **web interface** provides a **user-friendly platform** that simplifies the process for the administrator. The admin login, subject selection buttons, and automatic webcam integration make the system convenient and efficient for daily use.

In conclusion, this project demonstrates a **secure, efficient, and intelligent attendance solution** that aligns with modern technological needs. As future work, the system can be enhanced further by incorporating **liveness detection** and **anti-spoofing mechanisms**, which would strengthen its resistance to deceptive practices. Features such as integration with cloud storage, database-backed dashboards, and analytics could also be considered to elevate the system to a fully scalable institutional solution.

Future Enhancement:

To ensure the Face Recognition Attendance System evolves with technological advancements and real-world needs, several future enhancements are proposed. These improvements aim to increase security, scalability, user experience, and system reliability, especially when deployed in academic institutions of varying sizes.

1. Liveness Detection

**Problem Addressed**: Preventing spoofing or proxy attendance.

**Description**: A major vulnerability in facial recognition systems is their inability to distinguish between a real person and a photo/video replay. Liveness detection aims to counter this by verifying whether the detected face is that of a live human.

**Implementation Techniques**:

* **Eye Blink Detection**: Track blinking patterns using facial landmarks.
* **Head Movement Detection**: Prompt users to turn their heads or follow a moving object.
* **3D Depth Sensing**: Use stereo cameras or depth sensors (e.g., Intel RealSense) to detect 3D structure.
* **Texture Analysis**: Differentiate between real skin textures and printed images.

**Benefit**: Enhances security by ensuring that only physically present students are marked as present, reducing the risk of manipulation.

2. Cloud-Based Attendance Storage

**Problem Addressed**: Local data storage limits accessibility and scalability.

**Description**: Transitioning from local CSV files to cloud storage enables centralized access, automatic backups, and real-time data synchronization.

**Suggested Technologies**:

* **Firebase**: For real-time database functionality with user authentication.
* **MySQL or PostgreSQL**: For structured, relational data storage.
* **MongoDB**: For scalable, NoSQL-based document storage.

**Benefit**: Facilitates access to attendance data from any device, supports collaboration among faculty, and enables integration with Learning Management Systems (LMS).

3. Multi-Camera Support

**Problem Addressed**: Incomplete student coverage in large classrooms.

**Description**: Implementing support for multiple IP cameras ensures wider visual coverage, capturing students seated at different angles or at the back of large halls.

**Key Features**:

* Camera switching or multi-feed processing.
* Face matching synchronization across cameras.
* Load balancing between camera streams.

**Benefit**: Improves recognition reliability and system scalability across multiple classrooms or larger auditoriums.

4. Improved Face Recognition Model

**Problem Addressed**: Basic models may not perform well in challenging conditions.

**Description**: Upgrading from traditional models to deep learning architectures improves recognition under varying lighting, head poses, and facial occlusions.

**Advanced Models**:

* **FaceNet**: Generates 128-dimensional embeddings with high accuracy.
* **ArcFace**: Uses angular margin loss for better discrimination.
* **Dlib’s ResNet**: Offers a good balance between speed and accuracy.

**Benefit**: Increases recognition performance, especially in real-world classroom environments where lighting and movement are inconsistent.

5. Mobile App Integration

**Problem Addressed**: Lack of mobile accessibility for students and teachers.

**Description**: A companion mobile app can offer teachers and students instant access to attendance data and notifications.

**Features**:

* Real-time attendance view.
* Push notifications for defaulters.
* Graphical attendance insights.
* Attendance submission verification.

**Benefit**: Enhances convenience, encourages student awareness, and supports remote monitoring by faculty and administrators.

6. Auto-Detection of Classroom Start Time

**Problem Addressed**: Manual triggering of the system may be forgotten or misused.

**Description**: Automating the system to start and stop based on class schedules ensures timely and consistent operation.

**Implementation**:

* Sync with institutional timetable.
* Use Python schedulers or CRON jobs.
* Combine with RFID or GPS for room-specific automation.

**Benefit**: Promotes discipline and punctuality, and minimizes manual intervention, increasing the system’s reliability.

7. Improved UI for Subject Selection and Reports

**Problem Addressed**: Basic UI limits functionality and insights.

**Description**: A modern, user-friendly dashboard can significantly enhance the system’s usability for teachers and administrators.

**UI Features**:

* Subject buttons for attendance.
* Filters for date, student, and subject.
* Charts showing attendance trends and defaulters.
* Real-time logs and analytics.

**Benefit**: Provides a comprehensive and visual way to manage attendance records, improving decision-making and engagement.

8. Face Recognition with Mask Detection

**Problem Addressed**: Difficulty in recognizing faces covered by masks.

**Description**: Post-pandemic norms often require students to wear masks, which hinders recognition systems relying on full facial features.

**Solution**:

* Train the model on masked face datasets.
* Use hybrid models that analyze visible features (eyes, forehead).
* Combine with temperature or ID scanning if necessary.

**Benefit**: Maintains accuracy even when masks are worn, ensuring compliance with health guidelines.

9. Attendance Summary Reports

**Problem Addressed**: Manual report generation is time-consuming.

**Description**: Automatically generating downloadable reports in formats like PDF or Excel enhances administrative efficiency.

**Features**:

* Daily, weekly, and monthly summaries.
* Individual student attendance percentages.
* Defaulter lists with color-coded warnings.

**Benefit**: Simplifies auditing, parent-teacher meetings, and record-keeping processes.

10. Automatic Notification System

**Problem Addressed**: Lack of communication with students and guardians.

**Description**: The system can send automated emails or SMS alerts when students fall below attendance thresholds or miss specific sessions.

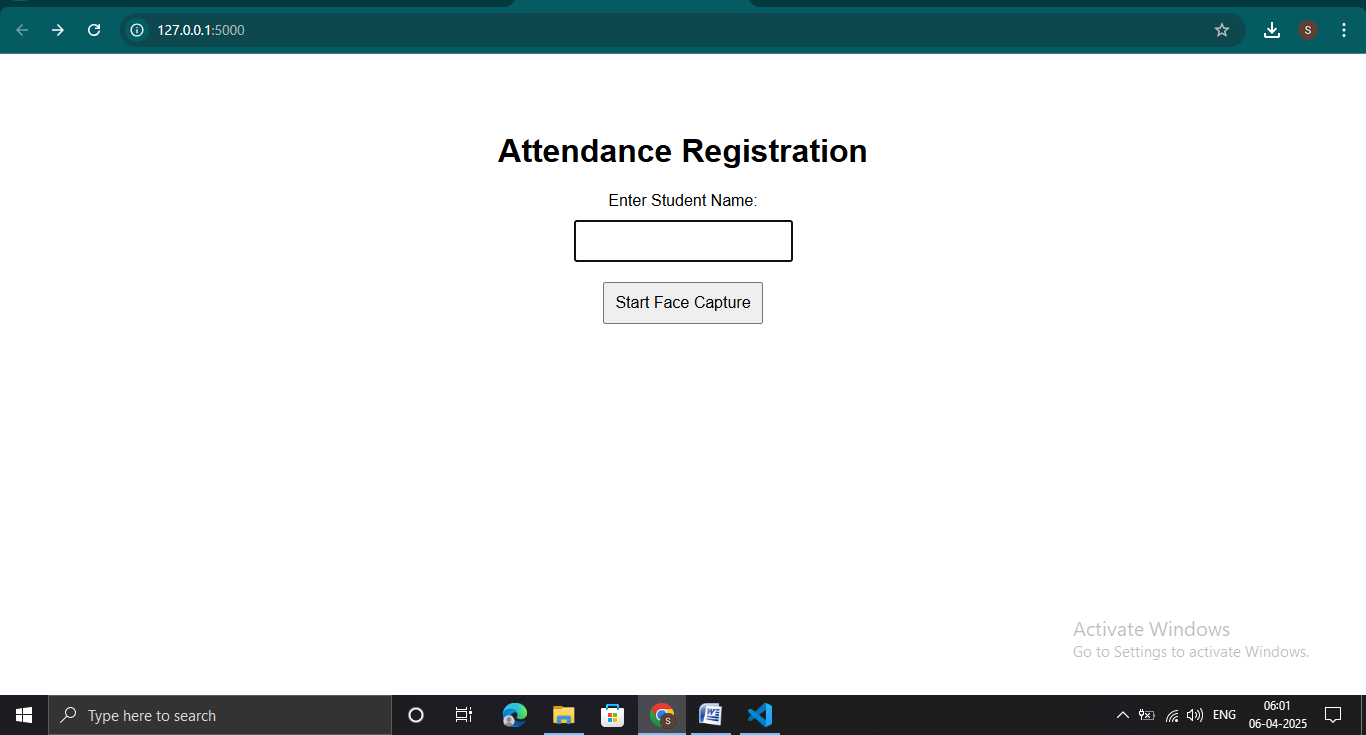
**Notification Triggers**:

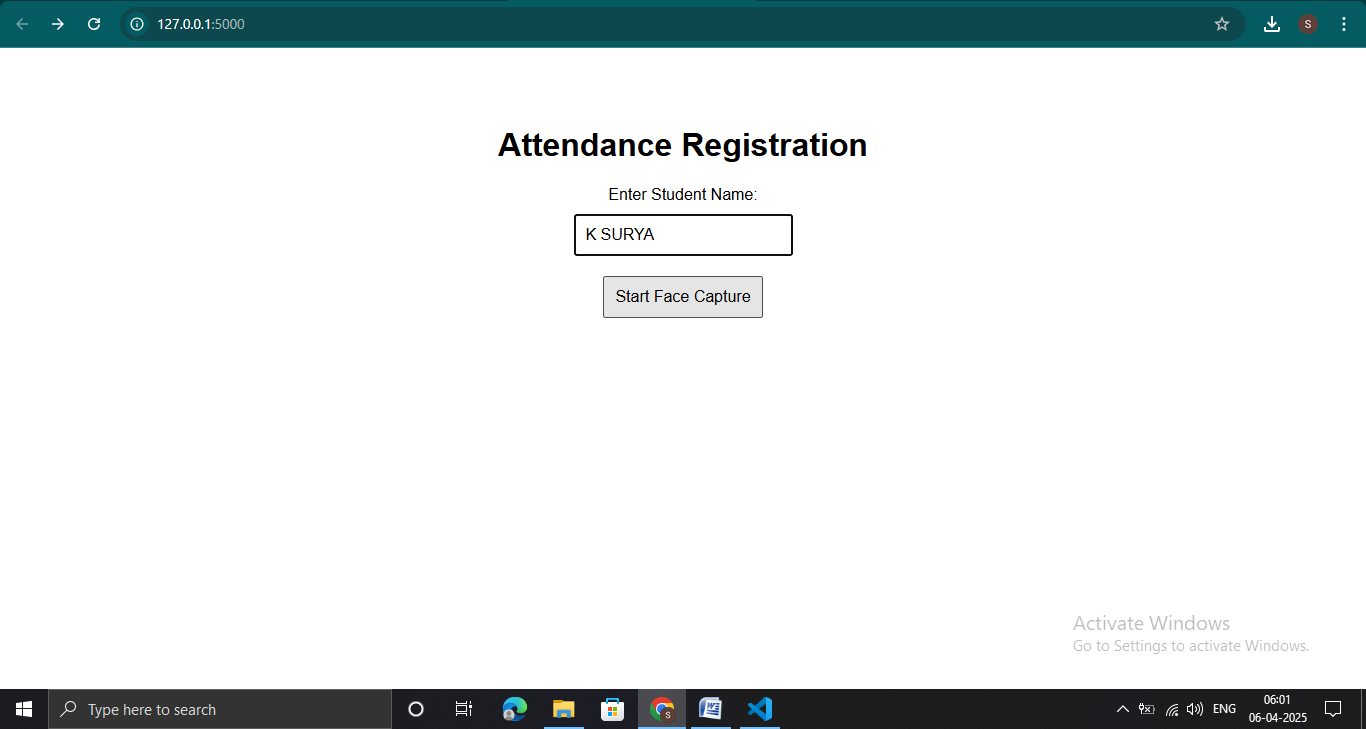
* Attendance below 75%.
* Daily summary for faculty.
* Absence for consecutive days.

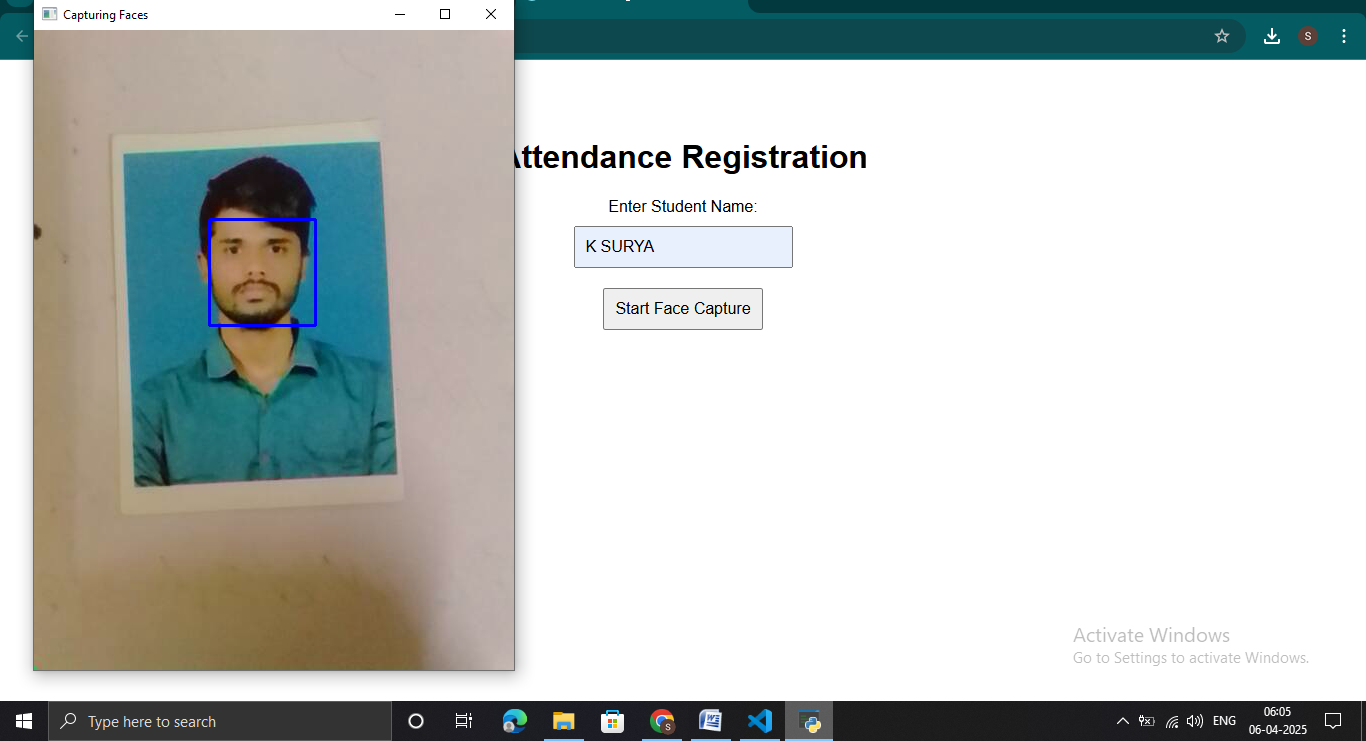
**Benefit**: Improves accountability, keeps students and parents informed, and helps intervene early in cases of absenteeism.

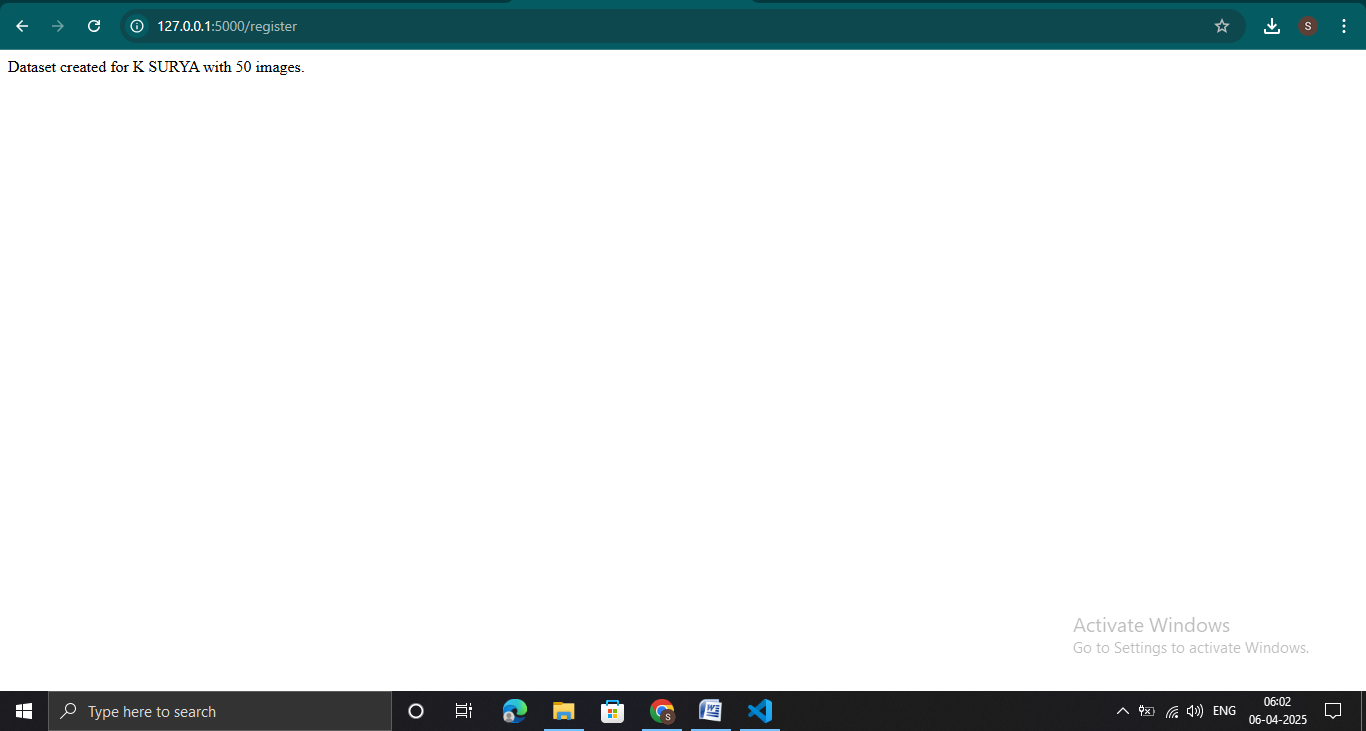
.

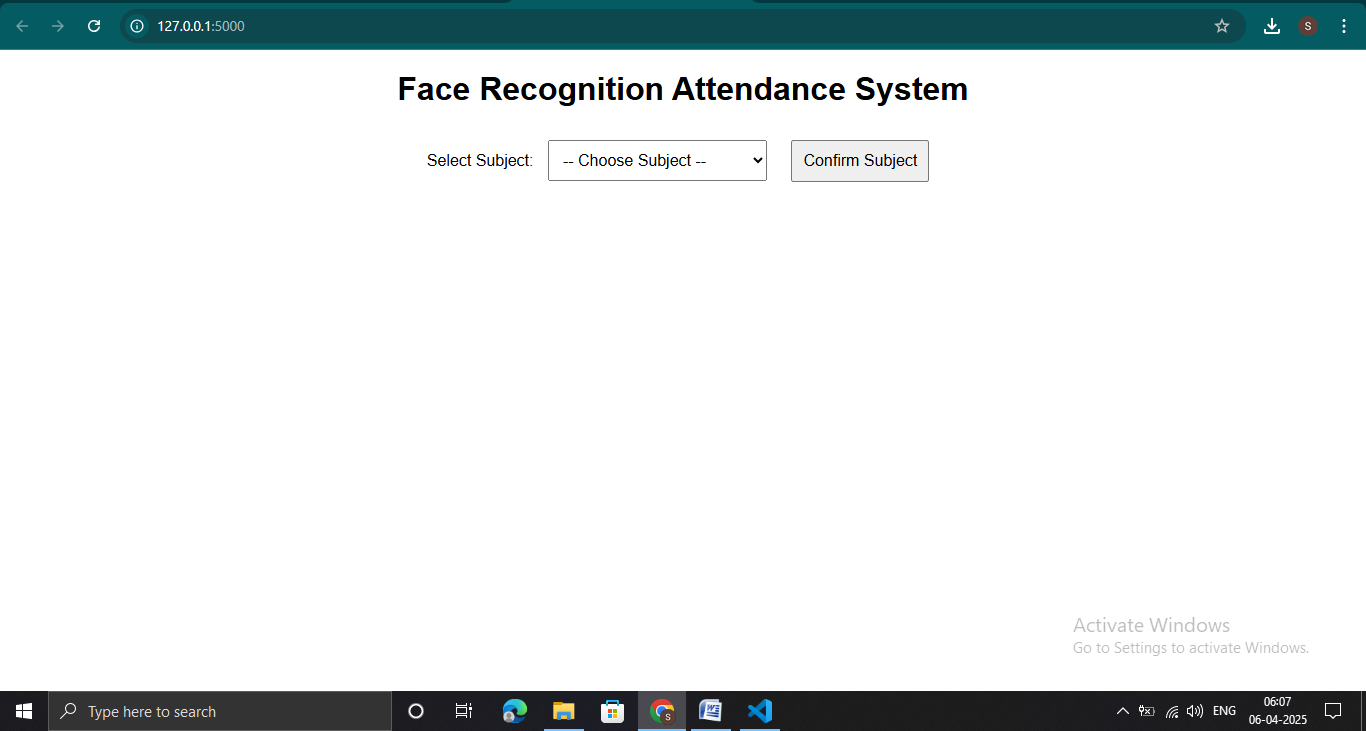
Screenplates

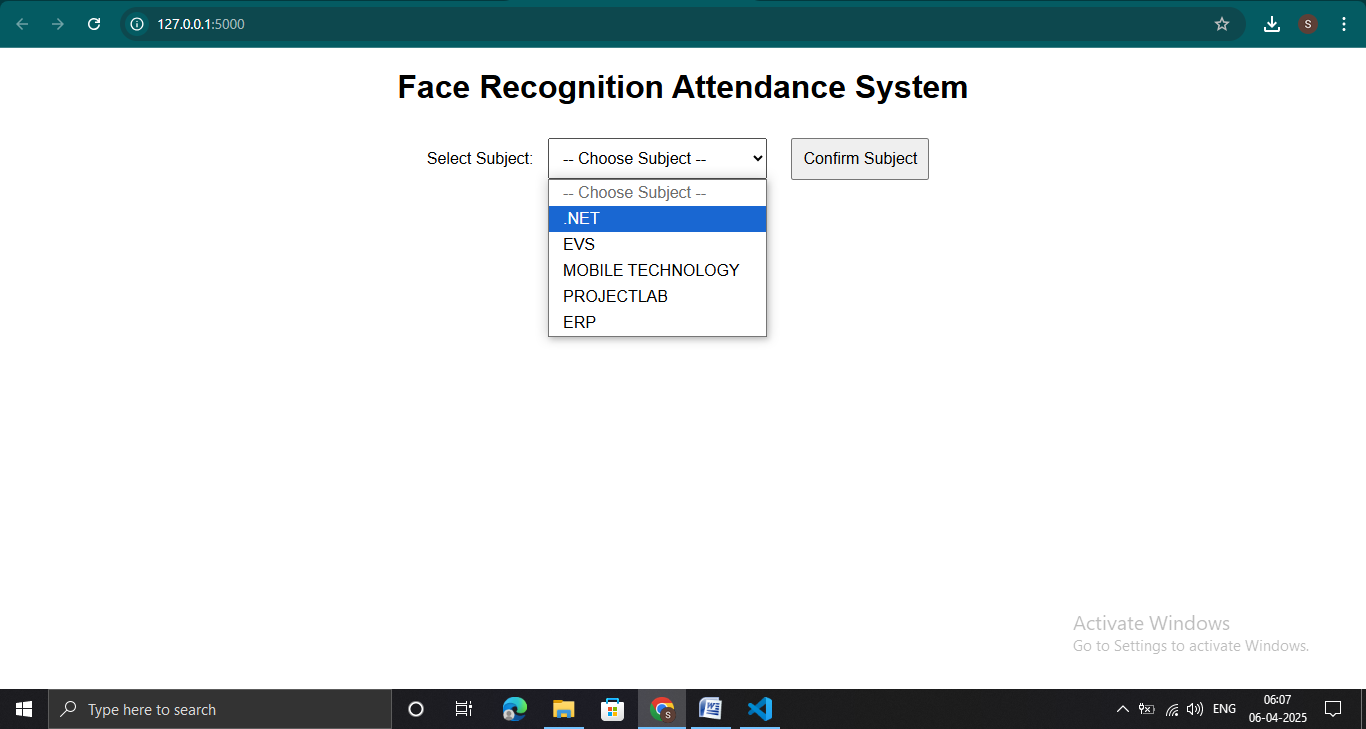


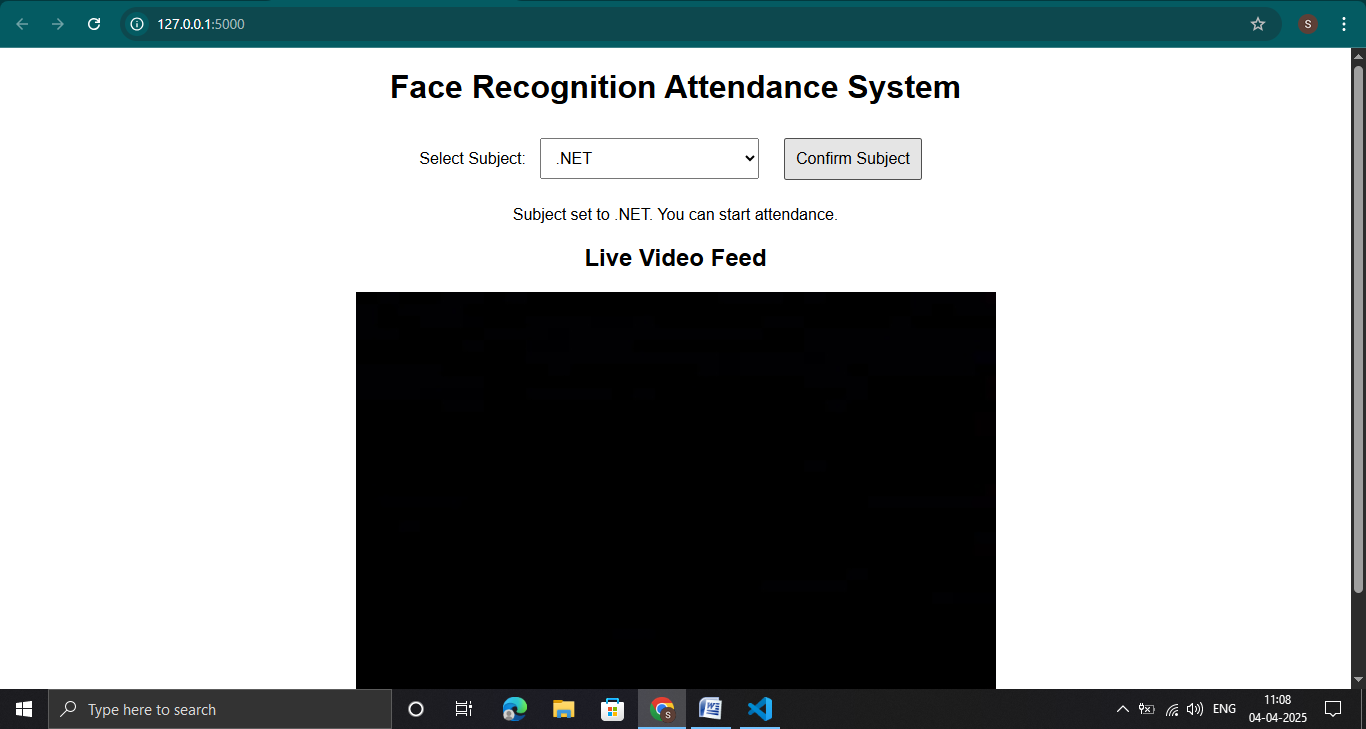












Source Code:

Dataset Creation Code:

Register.Html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Attendance Registration</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            padding-top: 50px;

        }

        input, button {

            font-size: 16px;

            padding: 10px;

            margin: 10px;

        }

    </style>

</head>

<body>

    <h1>Attendance Registration</h1>

    <form action="/register" method="post">

        <label for="name">Enter Student Name:</label><br>

        <input type="text" id="name" name="name" required><br>

        <button type="submit">Start Face Capture</button>

    </form>

</body>

</html>

Register.py:

from flask import Flask, render\_template, request, redirect, url\_for

import cv2

import os

import mediapipe as mp

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

mp\_face\_detection = mp.solutions.face\_detection

face\_detection = mp\_face\_detection.FaceDetection(min\_detection\_confidence=0.5)

# IP Webcam URL

IP\_WEBCAM\_URL = 'http://192.168.189.210:8080/video'

@app.route('/')

def index():

    return render\_template('register.html')

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

def register():

    name = request.form['name']

    if not name:

        return "Name is required", 400

    datasets = 'Dataset'

    path = os.path.join(datasets, name)

    os.makedirs(path, exist\_ok=True)

    webcam = cv2.VideoCapture(IP\_WEBCAM\_URL)

    if not webcam.isOpened():

        return "Could not open webcam. Check URL.", 500

    count = 1

    (width, height) = (130, 100)

    while count <= 50:

        success, frame = webcam.read()

        if not success:

            continue

        rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

        results = face\_detection.process(rgb\_frame)

        if results.detections:

            for detection in results.detections:

                bbox = detection.location\_data.relative\_bounding\_box

                h, w, \_ = frame.shape

                x = int(bbox.xmin \* w)

                y = int(bbox.ymin \* h)

                box\_w = int(bbox.width \* w)

                box\_h = int(bbox.height \* h)

                x, y = max(0, x), max(0, y)

                box\_w = min(w - x, box\_w)

                box\_h = min(h - y, box\_h)

                face = frame[y:y+box\_h, x:x+box\_w]

                if face.size == 0:

                    continue

                face\_gray = cv2.cvtColor(face, cv2.COLOR\_BGR2GRAY)

                face\_resized = cv2.resize(face\_gray, (width, height))

                cv2.imwrite(f"{path}/{count}.png", face\_resized)

                count += 1

                cv2.rectangle(frame, (x, y), (x + box\_w, y + box\_h), (255, 0, 0), 2)

                cv2.imshow('Capturing Faces', frame)

        if cv2.waitKey(10) == 27:

            break

    webcam.release()

    cv2.destroyAllWindows()

    return f"Dataset created for {name} with 50 images."

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

    app.run(debug=True)

Face Recognition Code:

Index.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Face Recognition Attendance</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

        }

        select, button {

            padding: 10px;

            margin: 10px;

            font-size: 16px;

        }

        #video-container {

            display: none;

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <h1>Face Recognition Attendance System</h1>

    <label for="subject">Select Subject:</label>

    <select id="subject">

        <option value="" selected disabled>-- Choose Subject --</option>

        {% for subject in subjects %}

            <option value="{{ subject }}">{{ subject }}</option>

        {% endfor %}

    </select>

    <button onclick="setSubject()">Confirm Subject</button>

    <p id="message"></p>

    <div id="video-container">

        <h2>Live Video Feed</h2>

        <img id="video\_feed" src="" width="640px">

    </div>

    <script>

        function setSubject() {

            let selectedSubject = document.getElementById("subject").value;

            if (!selectedSubject) {

                alert("Please select a subject.");

                return;

            }

            fetch('/select\_subject', {

                method: 'POST',

                headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

                body: 'subject=' + encodeURIComponent(selectedSubject)

            })

            .then(response => response.json())

            .then(data => {

                document.getElementById("message").innerText = data.message;

                document.getElementById("video-container").style.display = "block";

                document.getElementById("video\_feed").src = "/video\_feed";

            })

            .catch(error => console.error('Error:', error));

        }

    </script>

</body>

</html>

Attendance.py

from flask import Flask, render\_template, Response, request, jsonify

import cv2

import os

import numpy as np

import csv

from datetime import datetime

import mediapipe as mp

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

# Initialize Mediapipe Face Detection

mp\_face\_detection = mp.solutions.face\_detection

face\_detection = mp\_face\_detection.FaceDetection(min\_detection\_confidence=0.5)

# Dataset folder

datasets = 'Dataset'

subjects = ['.NET', 'EVS', 'MOBILE TECHNOLOGY', 'PROJECTLAB', 'ERP']

# Global variables

selected\_subject = None

video\_active = False  # Controls whether the video feed starts

# Initialize the recognizer

recognizer = cv2.face.LBPHFaceRecognizer\_create()

def prepare\_training\_data(datasets):

    faces, labels = [], []

    label\_dict = {}

    label\_id = 0

    for subdir in os.listdir(datasets):

        if os.path.isdir(os.path.join(datasets, subdir)):

            label\_dict[label\_id] = subdir

            for filename in os.listdir(os.path.join(datasets, subdir)):

                filepath = os.path.join(datasets, subdir, filename)

                img = cv2.imread(filepath, cv2.IMREAD\_GRAYSCALE)

                faces.append(img)

                labels.append(label\_id)

            label\_id += 1

    return np.array(faces), np.array(labels), label\_dict

faces, labels, label\_dict = prepare\_training\_data(datasets)

recognizer.train(faces, labels)

def mark\_attendance(name):

    if not selected\_subject:

        return "No subject selected!"

    attendance\_file = f'attendance\_{selected\_subject.lower()}.csv'

    if not os.path.isfile(attendance\_file):

        with open(attendance\_file, 'w', newline='') as file:

            writer = csv.writer(file)

            writer.writerow(['Name', 'Time'])

    with open(attendance\_file, 'r', newline='') as file:

        reader = csv.reader(file)

        entries = list(reader)

    if any(row[0] == name for row in entries):

        return f"{name} is already marked for {selected\_subject}."

    with open(attendance\_file, 'a', newline='') as file:

        writer = csv.writer(file)

        current\_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        writer.writerow([name, current\_time])

    return f"Attendance marked for {name} in {selected\_subject}."

def generate\_frames():

    global video\_active

    ip\_webcam\_url = 'http://192.168.189.210:8080/video'

    webcam = None

    while True:

        if not video\_active:  # Only start webcam if subject is selected

            continue

        if webcam is None:

            webcam = cv2.VideoCapture(ip\_webcam\_url)

        success, frame = webcam.read()

        if not success:

            break

        # Resize the frame to a rectangular shape (4:3 aspect ratio)

        frame = cv2.resize(frame, (640, 480))  # Change to (500, 500) for square

        rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

        results = face\_detection.process(rgb\_frame)

        if results.detections:

            for detection in results.detections:

                bboxC = detection.location\_data.relative\_bounding\_box

                h, w, \_ = frame.shape

                x, y, box\_w, box\_h = (int(bboxC.xmin \* w), int(bboxC.ymin \* h), int(bboxC.width \* w), int(bboxC.height \* h))

                x, y = max(0, x), max(0, y)

                box\_w, box\_h = min(w - x, box\_w), min(h - y, box\_h)

                face = frame[y:y+box\_h, x:x+box\_w]

                if face.size == 0:

                    continue

                face\_gray = cv2.cvtColor(face, cv2.COLOR\_BGR2GRAY)

                face\_resized = cv2.resize(face\_gray, (130, 100))

                label, confidence = recognizer.predict(face\_resized)

                name = label\_dict.get(label, "Unknown")

                if confidence < 100:

                    mark\_attendance(name)

                    cv2.putText(frame, name, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 0), 2)

                else:

                    cv2.putText(frame, "Unknown", (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 0, 255), 2)

                cv2.rectangle(frame, (x, y), (x + box\_w, y + box\_h), (255, 0, 0), 2)

        ret, buffer = cv2.imencode('.jpg', frame)

        frame = buffer.tobytes()

        yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/')

def index():

    return render\_template('index.html', subjects=subjects)

@app.route('/video\_feed')

def video\_feed():

    if not video\_active:  # Don't start if no subject is selected

        return jsonify({"error": "Select a subject first!"}), 400

    return Response(generate\_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/select\_subject', methods=['POST'])

def select\_subject():

    global selected\_subject, video\_active

    selected\_subject = request.form['subject']

    video\_active = True  # Enable webcam feed

    return jsonify({'message': f'Subject set to {selected\_subject}. You can start attendance.'})

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

    app.run(debug=True)

Bibliography

 Novitech RD Pvt. Ltd., *Face Recognition Attendance System Source Code*, Retrieved from project training materials and demonstrations during internship sessions.

 Google MediaPipe Documentation, *MediaPipe Solutions Overview*, Retrieved from: <https://mediapipe.dev>

 OpenCV Team, *OpenCV Library for Real-Time Computer Vision*, Retrieved from: <https://opencv.org>

 Face Recognition Library by Adam Geitgey, *Simple Face Recognition API for Python*, GitHub Repository: <https://github.com/ageitgey/face_recognition>

 W3Schools, *HTML, CSS, and JavaScript Tutorials*, Used for designing the web interface. Retrieved from: <https://www.w3schools.com>

 Real Python, *Building Web Apps with Flask*, Tutorial for integrating Python with a web-based interface. Retrieved from: <https://realpython.com>

 GeeksforGeeks, *Introduction to CSV Handling using Pandas in Python*, Retrieved from: <https://www.geeksforgeeks.org>

 Stack Overflow, *Community Discussions on IP Webcam Integration and Python Libraries*, Multiple threads accessed between January–April 2025.