

Face Recognition Based Attendance Management System



Submitted By

Hadia Moosa	2022-CS-119
Momina Rashad	2022-CS-137
Shahneela Iqbal	2022-CS-159

Supervised By

Sir Samyan Qayyum Wahla

Department of Computer Science

University of Engineering and Technology, Lahore

Abstract

This project presents an AI-powered Facial Recognition Attendance System to automate the attendance-taking process in institutions. Traditional attendance management systems often suffer from inefficiencies, errors, and challenges in accurately tracking attendance under varying conditions, such as the presence of masks, glasses, or facial expressions. These conventional methods, whether paper-based or biometric, are increasingly inadequate in meeting the demands for a more reliable, scalable, and adaptable solution. As organizations strive to enhance operational efficiency, the need for an advanced, automated system for attendance tracking has become paramount. This project introduces a Facial Recognition-Based Attendance System designed to address these limitations. By leveraging deep learning algorithms, using OpenCV DNN for face detection with the ResNet-10SSD model and MobileNetV2-based CNN for mask detection, the system ensures high accuracy and efficiency. The backend is built with Flask and Tkinter, while the frontend uses HTML, CSS, and Bootstrap. It effectively handles diverse conditions, such as mask-wearing and varying head orientations, offering a robust solution for real-world applications. The system integrates with a MySQL database to store and manage attendance data, ensuring secure, accessible, and reliable record-keeping. A user-friendly interface, developed using Tkinter, allows both administrators and users to seamlessly register and mark attendance. With its ability to operate efficiently in dynamic environments, this system offers a contactless and scalable solution that enhances accuracy and inclusivity. By catering to diverse user conditions, it ensures a broad application range, from educational institutions to corporate settings, fostering a more efficient and inclusive approach to attendance management.

Contents

1	Introduction	5
1.1	Problem Statement	5
1.2	Objective	5
1.3	Scope	5
2	Project Description	6
2.1	Project Methodology	6
2.2	Project Rationale	6
2.2.1	Operational Inefficiencies	6
2.2.2	Security and Fraud Prevention	6
2.2.3	Adapting to User Needs	6
2.2.4	Scalability and Centralized Management	7
2.2.5	Compliance with Post-Pandemic Standards	7
2.3	Product Features	7
2.3.1	Face Detection	7
2.3.2	Face Recognition	7
2.3.3	Attendance Logging	7
2.3.4	Admin Dashboard	8
2.3.5	User Management	8
3	Technology stack	8
4	Constraints	8
4.1	Camera Quality	8
4.2	Processing Power	8
4.3	Data Privacy Laws	9
5	Use Cases	9
5.1	Admin Login	9
5.2	Admin Signup	9
5.3	Register Student (By Admin)	10
5.4	Student Management (CRUD Operations)	11
5.5	Generate Attendance Report (CSV Format)	12
5.6	Take Attendance with Face Detection	13
6	Functional Requirements	14
6.1	User Authentication	14
6.1.1	Sign Up	14
6.1.2	Log In	14
6.1.3	Log Out	14
6.2	Face Recognition	14
6.2.1	Face Registration	14
6.2.2	Face Recognition for Attendance	14
6.2.3	Face Detection with Mask and Glasses	15
6.3	Attendance Management	15
6.3.1	Automatically Attendance Marking	15

6.3.2	Manual Attendance Entering (Admin)	15
6.4	Reports and Dashboards	15
6.4.1	View Attendance Reports (Admin)	15
6.5	Security Management	15
6.5.1	Role-Based Access Control	15
7	System Architecture	16
7.1	Subsystems and Components	16
7.1.1	User Interface Subsystem	16
7.1.2	Student Management (CRUD) Subsystem	16
7.1.3	Face Recognition for Attendance Subsystem	16
7.1.4	Student Interface Subsystem	17
7.1.5	Design Considerations	17
8	Methodology	17
8.1	Data Collection	17
8.2	Implementation	17
8.2.1	Pre-Processing	17
8.2.2	Overfitting	18
8.2.3	Under Sampling	18
8.2.4	Over Sampling	18
8.2.5	Confusion Matrix	19
9	App Development	19
9.1	UI/UX Design	19
9.2	Front-End Development	20
9.3	Back-End Development	20
9.4	Database Design	20
9.5	API Integration	21
10	Implementation	21
10.1	System Interface	21
10.2	Backend Workflows	24
10.3	Code Overview	25
11	Limitations	28
11.1	Failure to Account for Edge Cases	28
11.2	Insufficient Error Reporting	28
11.3	Ignoring Multi-Device Compatibility	28
11.4	Underestimating the Impact of Poor Data Quality	28
11.5	Neglecting Real-World Environmental Factors	28
12	Results and Discussion	28
12.1	Summary of Results	28
12.2	Real-world Applicability	29
13	Conclusion	29
14	Future Work	30

List of Tables

1	Technology Stack	8
2	Performance of the Facial Recognition-based Attendance System	29

List of Figures

1	Analysis of correlation	19
2	Database design	21
3	First Screen	21
4	Sign Up page	22
5	Login Page	22
6	Admin Dashboard	22
7	Register Student	23
8	View,Update,Delete Student	23
9	View Attendance	23
10	Admin Setting	24
11	Student Dashboard	24

1 Introduction

1.1 Problem Statement

Traditional attendance management systems often suffer from inefficiencies, errors, whether paper-based or biometric, and are increasingly inadequate in meeting the demands. One of the primary challenges is ensuring accurate facial recognition in scenarios where users are wearing masks or glasses, which poses a significant barrier to conventional systems. This issue is further exacerbated by the need to accommodate current health and safety protocols. The purpose of this project is to design an efficient way of marking attendance through a face recognition mechanism.

1.2 Objective

The objective of this project is to automate the attendance-taking process, improving efficiency, accuracy, and user experience. The system aims to achieve the following:

- Automate the attendance-taking process to eliminate manual attendance logging.
- Improve efficiency, accuracy, and user experience through advanced AI and facial recognition technologies.
- Use real-time face detection to automatically identify individuals as soon as they enter a monitored area, with attendance logged instantly.
- Ensure a quick, error-free process, enhancing reliability and minimizing delays.
- Enhance the system's accuracy through deep learning algorithms, ensuring precise recognition under varying conditions.
- Provide a user-friendly interface for both administrators and individuals, enabling simple attendance tracking and access to reports.
- Ensure security and privacy with encrypted facial data storage and compliance with data protection regulations, handling user data responsibly.
- Generate detailed attendance reports to further enhance the system's utility.
- Save time and improve attendance management by offering a more accurate, reliable, and secure solution.
- Provide a hassle-free experience for all users.

1.3 Scope

The system is designed to be a versatile and scalable solution, suitable for a wide range of organizations, including educational institutions (such as schools, colleges, and universities), corporate offices, and healthcare facilities like hospitals. Its primary function is to automate the process of attendance tracking for both staff members and students, significantly reducing the time and effort required for manual attendance management. The application is built to be adaptable, meaning it can be seamlessly integrated into various

organizational structures and workflows. It is capable of supporting multiple use cases and different types of users, such as administrators, teachers, students, and employees, with each user role having specific access and functionality. This customization allows the system to cater to the unique needs of each institution, whether it's tracking student attendance in a classroom, employee presence in an office, or patient attendance in a hospital setting. Moreover, the system will provide flexibility with user role-based customization. Administrators can have full access to manage the entire system, including adding or removing users, monitoring attendance data, generating reports, while students or employees can mark attendance. This role-based access enhances security and ensures that each user has appropriate access to the system based on their responsibilities.

2 Project Description

2.1 Project Methodology

The project will be divided into phases, with each including requirements gathering, design, implementation, testing, and deployment. Regular stakeholder meetings and feedback sessions will ensure that the business objectives and user expectations are aligned. Agile approaches, such as continuous integration and sprint-based development, will improve flexibility and responsiveness to changing needs.

2.2 Project Rationale

The decision to implement an AI-based Face Recognition Attendance System stems from the growing demand for efficient, secure, and automated attendance solutions. The project addresses critical operational challenges while offering scalable and adaptable technology to meet the diverse needs of various institutions. Key motivations includes

2.2.1 Operational Inefficiencies

Traditional attendance systems, whether manual or card-based, are prone to errors, time delays, and fraudulent practices such as proxy attendance. An AI-driven system utilizing face recognition eliminates these inefficiencies by providing a fast, reliable, and contactless method of attendance tracking.

2.2.2 Security and Fraud Prevention

Face recognition technology adds an additional layer of security by ensuring that attendance is marked only for authenticated individuals. This eliminates issues like buddy punching, thereby improving accuracy and accountability.

2.2.3 Adapting to User Needs

Modern organizations require attendance systems that are seamless and easy to use. By integrating AI with real-time face recognition, this system caters to the evolving needs of schools, colleges, universities, corporate offices, and hospitals. It also offers customizable features for different roles, ensuring a versatile and user-friendly experience.

2.2.4 Scalability and Centralized Management

The system is designed to handle operations across single or multiple branches, enabling centralized attendance record management. This scalability ensures that the system can grow with the organization's needs, making it suitable for both small-scale and large-scale deployments.

2.2.5 Compliance with Post-Pandemic Standards

With the emphasis on contactless solutions in a post-pandemic world, the system offers a hygienic alternative to fingerprint-based or manual attendance systems.

2.3 Product Features

2.3.1 Face Detection

Description

Utilizes advanced deep learning algorithms, specifically OpenCV DNN, to detect faces in real-time video streams. This ensures that attendance is recorded quickly and accurately.

Functionality

- Detects faces in live video streams or uploaded images.
- Handles single faces at a time to maintain scalability and accuracy.

2.3.2 Face Recognition

Description

Implements robust recognition capabilities to identify users' faces with high precision. The system remains effective under challenging real-world conditions, including when users wear masks, glasses, or are slightly angled.

Functionality

- Matches detected faces against stored user profiles with high accuracy.
- Supports real-time recognition for seamless attendance marking.
- Accurately identifies users even when their faces are tilted up to 45 degrees left or right.

2.3.3 Attendance Logging

Description

Automatically logs attendance records into a .csv file format, providing a reliable and structured method for data storage.

Functionality

- Each entry includes a user ID, name, timestamp, and attendance count.
- Enables easy retrieval, analysis, and reporting of attendance data.
- Facilitates integration with external systems for extended use cases like payroll or academic performance tracking.

2.3.4 Admin Dashboard

Description

A comprehensive interface designed for administrators to manage user profiles, attendance records, and reports effectively.

Functionality

- Allows administrators to add, update, or remove user profiles.
- Provides access to detailed attendance logs.
- Offers intuitive tools for managing attendance across multiple locations or branches.

2.3.5 User Management

Description

Empowers users to mark their attendance independently.

Functionality

- Users can mark attendance in real-time using facial recognition.
- Provides a personalized dashboard where users can view their attendance history, including dates, timestamps, and attendance count.
- Ensures data accuracy by linking attendance to unique user profiles.

3 Technology stack

Category	Details
Language	Python
Platform	Desktop
Frontend	Streamlit, Flask, Tkinter
Backend	Python-based API
Database	MySQL
IDEs	VS Code

Table 1: Technology Stack

4 Constraints

4.1 Camera Quality

The system will require a high-quality camera to ensure clear and accurate image capture, which is crucial for effective face detection and recognition.

4.2 Processing Power

Deep learning models will be optimized to perform real-time detection, demanding a robust processing capability to maintain high performance during operation.

4.3 Data Privacy Laws

The system will incorporate security measures to protect sensitive information related to admin and students, including encryption protocols and secure data storage practices. Ensuring the confidentiality and integrity of personal data will be a top priority.

5 Use Cases

5.1 Admin Login

Primary Actor: Admin

Preconditions: Admin must be registered in the system and have valid login credentials.

Success Guarantee: Admin will be successfully logged in and will have access to the system's functionalities on the admin dashboard.

Main Success Scenario

- (a) Admin opens the application and navigates to the login page.
- (b) Admin enters credentials, such as username and password.
- (c) System validates the entered credentials.
- (d) If the credentials are valid, the system grants access and redirects the Admin to the dashboard.
- (e) The system displays the message "Logged In successfully".

Extensions (Alternatives)

5.1a Invalid Credentials

Trigger:

- Admin enters incorrect username or password.

System Response:

- The system displays an error message: "Please Re-enter all credentials".
- Admin will re-enter correct credentials and try again.

5.2 Admin Signup

Primary Actor: Admin

Preconditions: Admin has access to create an account.

Success Guarantee: Admin is successfully registered and can now log in.

Main Success Scenario

- (a) Admin navigates to the signup page.
- (b) Admin enters required information such as name, email, and password.
- (c) System checks the validations and creates a new account.
- (d) System displays the message “Account created successfully”.
- (e) System redirects the Admin to the login page.

Extensions (Alternatives)**5.2c Data Validation Failure****Trigger:**

- Admin submits the form with missing or incorrect data (e.g., invalid email format, weak password).

System Response:

- The system shows a failure message due to missing or incorrect data (e.g., invalid email format, weak password, or specific character requirements).
- Admin corrects the data and resubmits the signup form.

5.2d Duplicate Email**Trigger:**

- Admin enters an email address that is already registered in the system.

System Response:

- The system shows an error message: “Email is already registered. Please use a different email”.
- Admin enters a unique email and resubmits the form.

5.3 Register Student (By Admin)

Primary Actor: Admin

Preconditions: Admin must be logged in.

Success Guarantee: The student is successfully registered in the system.

Main Success Scenario

- (a) Admin selects the option “Register Student” from the dashboard.
- (b) Admin fills in the required student details that the student provides.
- (c) Admin uploads the student’s face photo for future recognition.
- (d) System validates and stores the student’s details along with the face photo.
- (e) System confirms the successful registration of the student.

Extensions (Alternatives)

5.3e Data Validation Issues

Trigger:

- Admin submits the form with missing, incorrect, or duplicate information (e.g., missing student ID, duplicate ID).

System Response:

- The system shows an error message: "Please fill all required fields".
- If a duplicate entry is found, the system shows: "Student ID already exists".
- Admin corrects the information and resubmits the form.

5.4 Student Management (CRUD Operations)

Primary Actor: Admin

Preconditions: Admin must be logged in.

Success Guarantee: Admin can update, view, or delete student records.

Main Success Scenario

(1) Admin navigates to the Student Management module:

- **Trigger:** Admin clicks on the "Student Management" button in the system dashboard.
- **System response:** The system displays the available options: Update, View, and Delete.

(2) Admin selects options:

- **A. Update**
 - (i) Admin selects the "Update" option and searches for a student by ID or name.
 - (ii) The system retrieves the student's current details and displays them in an editable form.
 - (iii) Admin edits the fields that need to be updated (e.g., contact details, class, etc.).
 - (iv) Admin then clicks on "Save Changes".
 - (v) The system displays the message "Changes Saved Successfully!".
- **B. Delete**
 - (i) Admin selects "Delete" and searches for a student by name or ID.
 - (ii) The system retrieves the student's details and asks for confirmation before deleting.
 - (iii) Admin confirms the deletion.
 - (iv) The system shows the message "Student deleted successfully" and updates the list of students.

- **C. View**

- (i) Admin selects "View" and searches for a student by ID or name.
- (ii) The system displays the student's details (e.g., name, class, contact, face photo) in a read-only format.
- (iii) Admin can navigate back to the list of students or perform another action like update or delete.

Extensions (Alternatives)

5.4.2Ai, 5.4Bi, 5.4Ci: If a student is not found during search

Trigger:

- Admin tries to search for a student by Student ID or name, but the ID does not exist in the system.

System Response:

- The system shows an error message: "Student not found."
- Admin can re-enter the correct ID or search by name.

5.4.2Aiii: Validation Error

Trigger:

- Admin tries to submit the form with missing or incorrect data (e.g., blank required fields, invalid student ID format).

System Response:

- The system highlights the invalid fields and displays error messages for each (e.g., "Student ID must be unique", "Name cannot be blank").
- Admin corrects the data and re-submits the form.

5.5 Generate Attendance Report (CSV Format)

Primary Actor: Admin

Preconditions: Admin must be logged in.

Success Guarantee: Attendance report is generated and available for download sorted by date.

Main Success Scenario

- (a) Admin navigates to the "Generate Attendance Report" module.
- (b) Admin selects the desired parameters.
- (c) The system compiles the data into a CSV file.
- (d) Admin clicks "Download", and the file is saved on their local machine.

Extensions

5.5b. No Attendance Data Record

Trigger:

- Admin selects a class that contains no attendance data.

System Response:

- The system shows a message: "No attendance data found for the selected parameter."
- Admin can modify the search criteria or exit the module.

5.6 Take Attendance with Face Detection

Primary Actor: Admin, System

Preconditions: Students must be registered with a valid photo in the system.

Success Guarantee: Attendance is recorded for each student based on face detection.

Main Success Scenario

- (a) Admin selects the "Take Attendance" module.
- (b) Students stand in front of the camera.
- (c) The system detects and identifies the student's face.
- (d) The system marks the student as present.
- (e) The system stores the attendance data for future reporting.

Extensions

5.6b. Face Not Recognized

Trigger:

- The system is unable to detect or match the student's face with the stored photo.

System Response:

- The system shows a message: "Face not recognized."
- Admin can either retry face detection or manually mark the student as present.

6 Functional Requirements

6.1 User Authentication

6.1.1 Sign Up

Description: User can create an account by entering email, name, password, and contact number.

Input: Username, email, contact number, and password.

Process: System validates the inputs, checks if the user already exists, and stores the information in the database.

Output: A confirmation message if successful or an error message if any information is incomplete or invalid.

6.1.2 Log In

Description: User can log in to the system by entering their username and password.

Input: Username and password.

Process: System checks if this user is present in the database and verifies the password with the stored password.

Output: If successful, the user will be redirected to their dashboard; otherwise, an error message is shown.

6.1.3 Log Out

Description: User can log out from the system.

Input: Click on the exit button.

Process: System invalidates the current session and redirects the user to the login page.

Output: A log-out message will be shown.

6.2 Face Recognition

6.2.1 Face Registration

Description: User can register their bare and masked face by capturing images through a camera attached to the system.

Input: Images captured from the camera.

Process: System extracts the features of the face using algorithms like OpenCV DNN and stores them in the database.

Output: A confirmation message will be shown.

6.2.2 Face Recognition for Attendance

Description: System recognizes the user from their face and marks their attendance automatically.

Input: Real-time image from the camera.

Process: System matches the real-time image with the images stored in the database and confirms the user's identity.

Output: Attendance is entered, and a confirmation message is shown.

6.2.3 Face Detection with Mask and Glasses

Description: System recognizes the user's face even if they are wearing glasses and masks.

Input: Real-time image from the camera.

Process: System adjusts the algorithm to detect the partial features of the face and matches it with the images in the database.

Output: Attendance is entered, and a confirmation message is shown.

6.3 Attendance Management

6.3.1 Automatically Attendance Marking

Description: System automatically enters attendance by detecting the user's face when the *Take Attendance* module is clicked.

Input: Image captured from the camera.

Process: System matches the real-time image with the images stored in the database for identity confirmation and marks the attendance with the date and time in the database.

Output: A confirmation message is shown if attendance is marked successfully; otherwise, an error message is shown.

6.3.2 Manual Attendance Entering (Admin)

Description: Admin can manually enter user attendance and adjust user records.

Input: User ID, date, and time for attendance entry.

Process: Admin selects the user, inputs the date and time, and updates the attendance record in the database.

Output: A confirmation message is shown if the database is updated successfully; otherwise, an error message is shown.

6.4 Reports and Dashboards

6.4.1 View Attendance Reports (Admin)

Description: Admin can view the attendance report of users from a specific date and ID

Input: Date and ID selection.

Process: System retrieves data for that specific date and ID and generates a report of that record.

Output: A report is shown and available for download.

6.5 Security Management

6.5.1 Role-Based Access Control

Description: System ensures that only authorized users (e.g., admins) can access certain functionalities like manual attendance entry.

Input: Log in as an admin.

Process: System checks if the credentials provided are correct.

Output: Admin dashboard is opened.

7 System Architecture

The face recognition-based student management and attendance system is designed as a modular, distributed system that integrates face detection technology with a comprehensive student management platform. The system is primarily divided into front-end and back-end components, each with specific responsibilities and functionalities. The front-end interacts with users (admin and students) to gather data, such as face images, while the back-end processes these inputs, performs face recognition, manages the student database, and generates attendance reports.

7.1 Subsystems and Components

7.1.1 User Interface Subsystem

The User Interface (UI) is the part of the system through which users interact with the system. In a face recognition-based student management and attendance system, the user interface must be intuitive, user-friendly, and responsive, designed to support different user roles such as Admin and Students for profile views.

Components:

- **Dashboard:** The admin dashboard presents a clear summary of key features such as total students and daily attendance records.
- **Login/Signup Page:** Includes login and signup pages with proper validation and authentication mechanisms. Admins must log in using secure credentials.
- **Student Registration:** Allows the Admin to register students with proper details like name, ID, class, contact number, and upload student face photos for training.

7.1.2 Student Management (CRUD) Subsystem

This subsystem handles CRUD (Create, Read, Update, Delete) operations for students, including adding new students, updating records, and removing students.

Components:

- **Update:** Admins can search for a student by StudentID and edit their necessary information.
- **Delete:** Displays a confirmation message when deleting a student from the system.
- **View:** A page from where admins can view all student details, including photos that are only accessible to admins.
- **Attendance Reports:** A UI that allows the selection of classes to generate attendance reports, including a “Download Report” button.

7.1.3 Face Recognition for Attendance Subsystem

This subsystem provides an interface to display the camera in real-time as the system scans student faces and marks attendance.

Components:

- **Image Capture Module:** Uses the laptop camera to capture student faces.
- **Face Detection Module:** Detects faces in the captured image.
- **Face Recognition Module:** Identifies faces by matching them with stored data.

7.1.4 Student Interface Subsystem

This subsystem provides students access to their profiles, attendance marking without admin intervention.

Components:

- **Profile View:** Allows students to view their profiles.
- **Self-Attendance:** Enables students to mark attendance through face recognition without admin intervention.

7.1.5 Design Considerations

- **Responsive Design:** The UI is designed to work across multiple devices, including desktops.
- **Accessibility:** Includes features for accessibility (e.g., high-contrast mode, screen readers) to accommodate different user needs.

8 Methodology

8.1 Data Collection

For this project, the dataset is collected from Kaggle, containing two categories: **with mask** and **without mask**. In the **with mask** category, all faces with masks are present, while in the **without mask** category, all faces without masks are included.

8.2 Implementation

8.2.1 Pre-Processing

Before training, images in the dataset are pre-processed to ensure uniformity:

- Convert all images to grayscale or normalize them.
- Resize the images to a fixed size (e.g., 128x128 pixels) to ensure compatibility with the recognition model.
- Use data augmentation techniques (e.g., rotation, scaling, and flipping) to increase dataset diversity.

8.2.2 Overfitting

Overfitting occurs when the model learns noise or random patterns in the training data, causing it to perform well on the training set but poorly on unseen data. In this project, overfitting could occur if the model is excessively trained on specific images, making it overly reliant on those features. But overfitting in this project is properly maintained by oversampling as the images with mask were 3725 with mask and 3828 without masks.

8.2.3 Under Sampling

Under sampling reduces the number of samples in overrepresented classes to balance the dataset. For this project, under sampling is not required because we have used the technique of oversampling to handle overfitting.

Pros:

- Balances the dataset, avoiding model bias
- Simplifies Data Storage Requirements

Cons:

- May lead to the loss of valuable information.
- Reduced Model Accuracy for Majority Class

8.2.4 Over Sampling

Over sampling is a technique used to address class imbalance by increasing the number of samples in underrepresented classes. This can be achieved through duplication of existing samples or by generating synthetic samples. In this project we have done oversampling.

Pros:

- **Improves Model Performance for Underrepresented Cases:** By increasing the number of samples for minority classes, the model can better learn and generalize patterns from these classes, leading to improved accuracy and reduced bias.
- **Enhances Dataset Balance:** Ensures that the model does not favor majority classes during training.
- **Prevents Ignoring Minority Classes:** Without oversampling, the model might overlook underrepresented classes, but this technique ensures they are included adequately in the learning process.

Cons:

- **Risk of Overfitting:** If synthetic data closely resembles original data without introducing diversity, the model may overfit to the minority class, performing poorly on unseen data.

- **Increased Training Time:** Larger datasets due to oversampling require more computational resources and training time.
- **Potential Noise Introduction:** If synthetic samples are not generated carefully, they may introduce noise or unrealistic patterns.

8.2.5 Confusion Matrix

A confusion matrix is a tool used to evaluate the performance of a classification model. It provides a summary of prediction results by comparing true labels to predicted labels. Below is a representation of the confusion matrix:

- True Positives (TP): Correctly identified as positive.
- True Negatives (TN): Correctly identified as negative.
- False Positives (FP): Incorrectly identified as positive.
- False Negatives (FN): Incorrectly identified as negative.

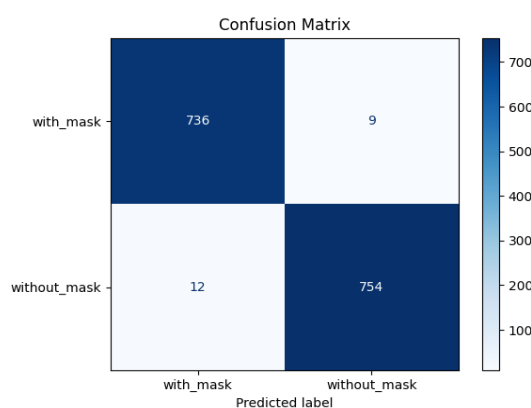


Figure 1: Analysis of correlation

9 App Development

The app development process involved a structured approach to ensure the final application is user-friendly, efficient, and scalable. It incorporated a robust combination of UI/UX design principles, modern front-end frameworks, a reliable back-end system, and a well-structured database. Each component was carefully integrated to create a seamless experience for the users while ensuring all functionalities operate efficiently.

9.1 UI/UX Design

The design was centered around user-friendliness and accessibility. Wireframes were created to map out the app's structure, focusing on intuitive navigation and clear visual hierarchies. Elements such as buttons, forms, and visual cues were positioned strategically to enhance user interaction and minimize confusion. The design adhered to principles of

simplicity and responsiveness, ensuring compatibility with devices of varying screen sizes. The goal was to provide users with a visually appealing interface that simplifies the task of managing attendance and accessing related information.

9.2 Front-End Development

The app utilized **HTML**, **CSS**, and **JavaScript** to create an interactive and responsive user-facing interface:

- **HTML:** Provided the structural foundation of the app.
- **CSS:** Ensured visual styling and layout consistency.
- **JavaScript:** Introduced interactivity, such as dynamic content updates and real-time validation.

These technologies enabled the creation of a responsive design that functions seamlessly across different platforms, offering a rich user experience.

9.3 Back-End Development

The back-end development was handled using **Tkinter**, **Flask**, a lightweight Python framework known for its simplicity and flexibility. The back-end managed the core application logic, including:

- User authentication.
- Attendance management workflows.
- Server-side processing.

It also handled API requests and integrated seamlessly with the database. Flask's modular design allowed for efficient scaling, ensuring the app could handle increasing user demands without performance degradation.

9.4 Database Design

The database relied on **MySQL**, a robust relational database management system. The database was structured with tables for users, attendance records, and face recognition data. Relationships between tables were carefully designed to ensure efficient data retrieval and consistency. For example:

- A **Users** table stored information about users, such as user ID, name, and contact details.
- An **Attendance** table tracked attendance logs linked to specific users.

The database design prioritized fast query execution to support real-time data retrieval during attendance checks, ensuring the system's efficiency and reliability.

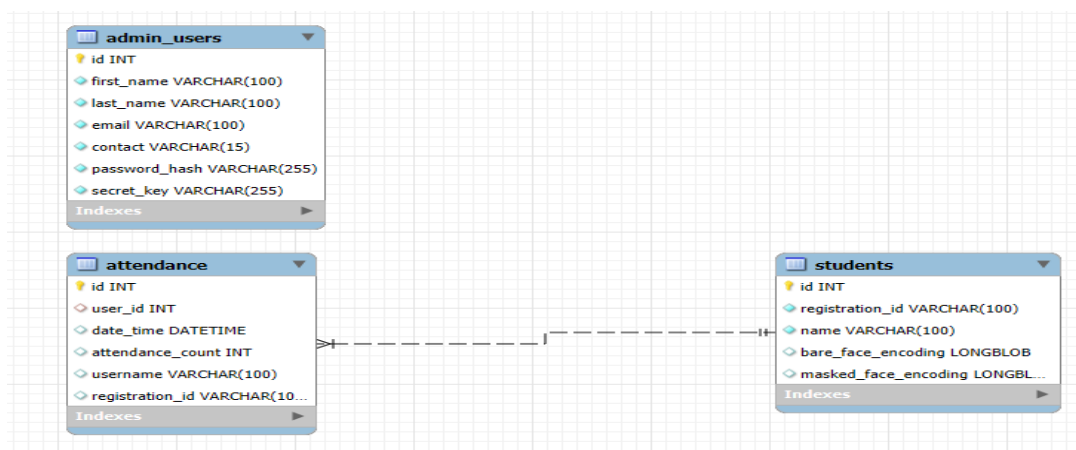


Figure 2: Database design

9.5 API Integration

API integration played a crucial role in enhancing the app's functionality. **Flask APIs** connected the front-end with the back-end and database, enabling smooth communication between components. The APIs facilitated the following:

- Seamless user authentication and session management.
- Real-time attendance tracking and data retrieval.
- Efficient database operations, such as CRUD (Create, Read, Update, Delete).

While third-party APIs for additional functionality were not extensively used in this system, the modular nature of Flask ensures future scalability. For instance, APIs for advanced analytics or integration with external systems, such as school management platforms, can be easily incorporated if needed.

10 Implementation

10.1 System Interface

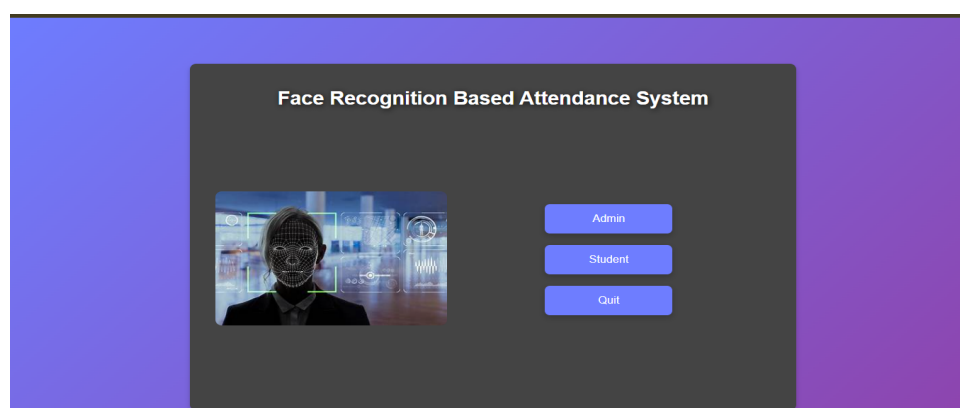


Figure 3: First Screen

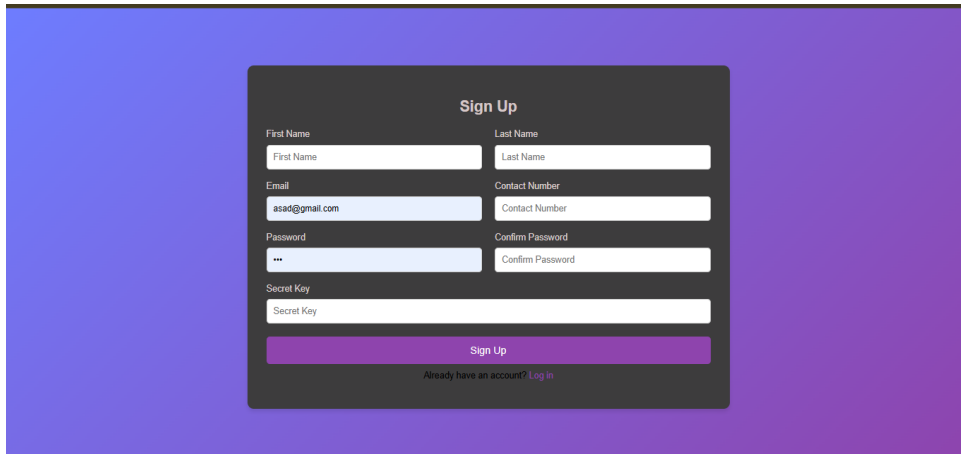
The image shows a 'Sign Up' form centered on a purple gradient background. The form is a dark gray rectangle with white text and input fields. It contains fields for 'First Name', 'Last Name', 'Email' (with 'asad@gmail.com' entered), 'Contact Number', 'Password', 'Confirm Password', and 'Secret Key'. A purple 'Sign Up' button is at the bottom, with a link 'Already have an account? Log in' below it.

Figure 4: Sign Up page

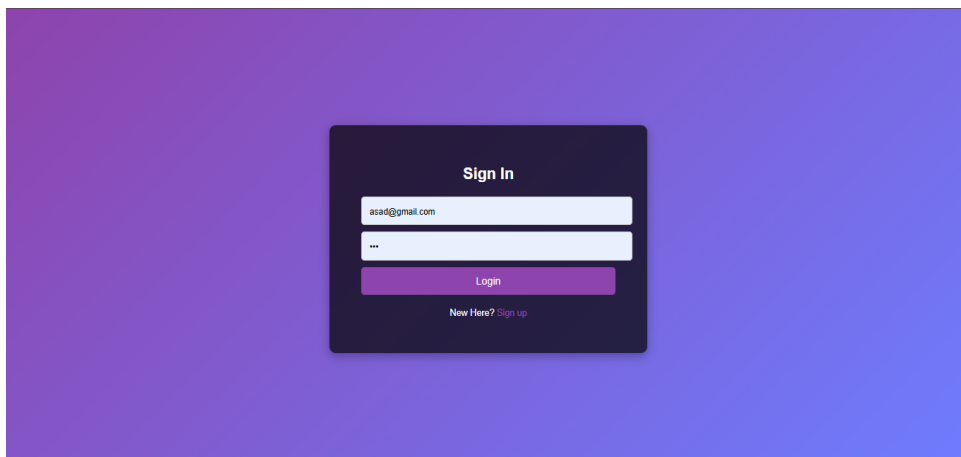
The image shows a 'Sign In' form centered on a purple gradient background. The form is a dark gray rectangle with white text and input fields. It contains fields for 'Email' (with 'asad@gmail.com' entered) and 'Password'. A purple 'Login' button is at the bottom, with a link 'New Here? Sign up' below it.

Figure 5: Login Page

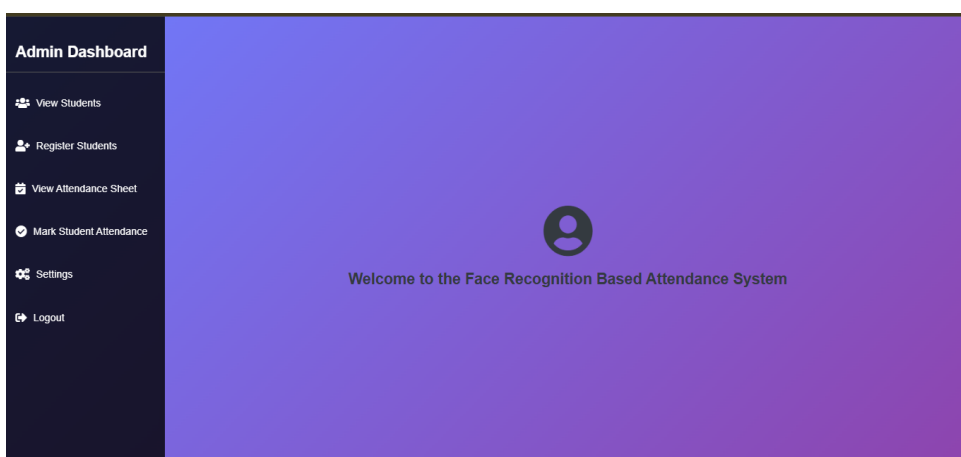
The image shows the 'Admin Dashboard' interface. On the left is a dark sidebar with the title 'Admin Dashboard' and a list of menu items: 'View Students', 'Register Students', 'View Attendance Sheet', 'Mark Student Attendance', 'Settings', and 'Logout'. The main area has a purple gradient background with a large circular profile icon placeholder in the center and the text 'Welcome to the Face Recognition Based Attendance System' below it.

Figure 6: Admin Dashboard

The screenshot shows the 'Admin Dashboard' on the left with a sidebar menu containing: View Students, Register Students, View Attendance Sheet, Mark Student Attendance, Settings, and Logout. The main area has a purple gradient background with a 'Student Registration' form. The form includes a 'Registration ID' field, a 'Name' field, and a 'Register Student' button.

Figure 7: Register Student

The screenshot shows the 'Admin Dashboard' on the left with the same sidebar menu. The main area has a purple gradient background with a 'Student List' table. The table has three columns: Registration ID, Name, and Actions. It contains two rows of student data.

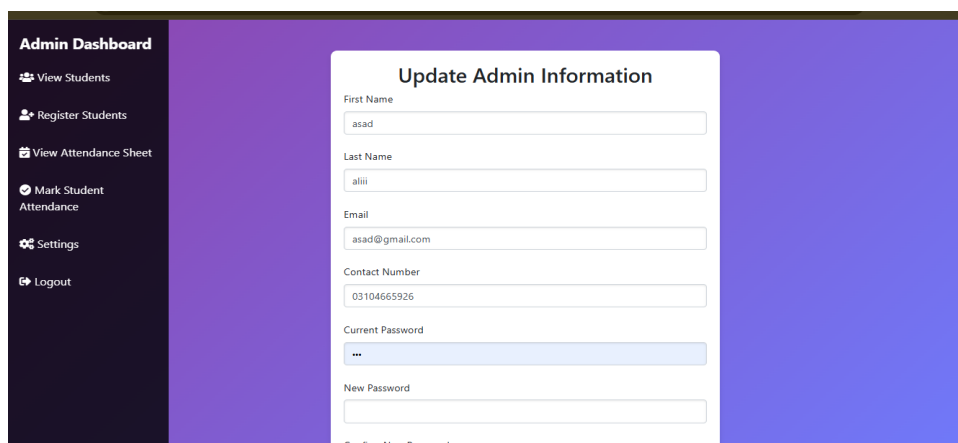
Registration ID	Name	Actions
1	Mominaa Rashad	Update Delete
119	Hadia Moosa	Update Delete

Figure 8: View,Update,Delete Student

The screenshot shows the 'Admin Dashboard' on the left with the same sidebar menu. The main area has a purple gradient background with an 'Attendance Records' section. It includes a search bar with the text 'Search by Registration ID' and a date input field with the text 'mm/dd/yyyy'. There are 'Filter' and 'Download Filtered Records' buttons. Below is a table with four columns: Date & Time, Attendance Count, Username, and Registration ID. It contains six rows of attendance data.

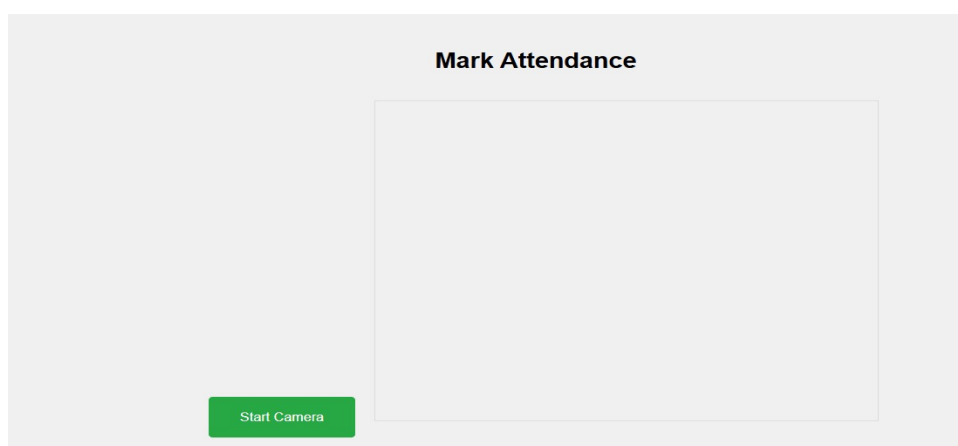
Date & Time	Attendance Count	Username	Registration ID
2024-12-19 11:27:44	1	Momina Rashad	1
2024-12-19 11:27:50	1	Momina Rashad	1
2024-12-19 11:27:55	1	Momina Rashad	1
2024-12-20 15:32:56	1	Hadia Moosa	119
2024-12-20 15:38:08	1	Mominaa Rashad	1

Figure 9: View Attendance



The image shows an 'Admin Dashboard' on the left with a dark sidebar containing links: View Students, Register Students, View Attendance Sheet, Mark Student Attendance, Settings, and Logout. The main area has a purple gradient background and features a white form titled 'Update Admin Information'. The form contains input fields for First Name (filled with 'asad'), Last Name (filled with 'aliii'), Email (filled with 'asad@gmail.com'), Contact Number (filled with '03104665926'), Current Password (masked with '***'), New Password, and Confirm New Password.

Figure 10: Admin Setting



The image shows a 'Mark Attendance' interface with a light gray background. It features a large, empty rectangular box for a camera feed. In the bottom-left corner, there is a green button labeled 'Start Camera'.

Figure 11: Student Dashboard

10.2 Backend Workflows

The backend flow for training data begins with the collection of facial images, captured through a camera for bear face and for mask ones taken from kaggle or uploaded by the admin via the dashboard. These images are organized systematically, with each student's photos stored in separate directories to facilitate identification. The collected data undergoes preprocessing to ensure consistency, including resizing images to a standard resolution, converting them to grayscale or normalized color formats, and applying data augmentation techniques such as rotation and flipping to increase dataset variability. For dataset of mask we use DNN algorithm. Unique facial features are then extracted from each image using advanced recognition techniques like Local Binary Patterns Histograms (LBPH) or deep learning-based methods, which encode these features into numerical vectors known as facial embeddings. These embeddings are used to train a machine learning classifier, such as Support Vector Machines (SVM) or K-Nearest Neighbors (KNN), to map embeddings to the corresponding student identities. The trained classifier and embeddings are securely stored, enabling the system to compare live facial data with stored information for accurate recognition. Additionally, the system allows for periodic re-training to incorporate new students, ensuring long-term accuracy and adaptability of the model.

10.3 Code Overview

```

app.py > download_attendance > generate
1  import numpy as np
2  import cv2
3  import csv
4  import face_recognition
5  from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
6  from tensorflow.keras.preprocessing.image import img_to_array
7  from tensorflow.keras.models import load_model
8  from datetime import datetime
9  from flask import Flask, render_template, redirect, request, flash, url_for, session, jsonify, Response
10 from flask_mysqldb import MySQL
11 from werkzeug.security import generate_password_hash, check_password_hash
12 from functools import wraps
13 import os
14 from io import BytesIO
15 app = Flask(__name__)
16 SECRET_KEY = "Molina0192004."
17 # MySQL configuration
18 app.config['MYSQL_HOST'] = 'localhost'
19 app.config['MYSQL_USER'] = 'root'
20 app.config['MYSQL_PASSWORD'] = 'Molina0192004.' # Update with your actual password
21 app.config['MYSQL_DB'] = 'face_attendance'
22 app.config['SESSION_COOKIE_NAME'] = 'admin_session'
23 app.config['SECRET_KEY'] = SECRET_KEY # Add this line
24

```

```

# Load the pre-trained face detector and mask detector models
prototxt_path = "deploy.prototxt.txt"
model_path = "res10_300x300_ssd_iter_140000.caffemodel"
face_net = cv2.dnn.readNetFromCaffe(prototxt_path, model_path)
mask_net = load_model("mask_detector.model.h5")

# Database connection function (using Flask MySQLdb)
def get_db_connection():
    try:
        conn = mysql.connect # Use the `mysql.connect` method from Flask MySQLdb
        return conn
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None

# Login required decorator
def login_required(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if 'logged_in' not in session:
            return redirect(url_for('login'))
        return func(*args, **kwargs)
    return wrapper

```

```

@app.route('/admin_dashboard')
@login_required
def admin_dashboard():
    return render_template('admin_dashboard.html')

@app.route('/attendance_sheet', methods=['GET'])
def attendance_sheet_page():

    # Fetch attendance records
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT date_time, attendance_count, username, registration_id FROM attendance")
    attendance_records = cursor.fetchall()
    conn.close()

    return render_template('attendance_sheet.html', attendance=attendance_records)

```

```
# Route to download attendance CSV
@app.route('/admin/download_attendance', methods=['GET'])
def download_attendance():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT date_time, attendance_count, username, registration_id FROM attendance")
    attendance_records = cursor.fetchall()

    def generate():
        yield 'Date Time, Attendance Count, Username, Registration ID\n'
        for record in attendance_records:
            yield f'{record[0]}, {record[1]}, {record[2]}, {record[3]}\n'

    return Response(generate(), mimetype='text/csv', headers={"Content-Disposition": "attachment;filename=attendance.csv"})

# Detect Faces function (no changes)
def detect_faces(image):
    blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104.0, 177.0, 123.0))
    face_net.setInput(blob)
    detections = face_net.forward()
    h, w = image.shape[:2]
    face_locations = []

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 3]
        if confidence > 0.5: # Confidence threshold
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            face_locations.append((startY, endX, endY, startX)) # Convert to face_recognition format

    return face_locations
```

```
def detect_and_predict_mask(image):
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300), (104.0, 177.0, 123.0))
    face_net.setInput(blob)
    detections = face_net.forward()

    faces = []
    locs = []
    preds = []

    if detections.shape[2] > 0: # Ensure detections are non-empty
        for i in range(detections.shape[2]):
            confidence = detections[0, 0, i, 3]
            if confidence > 0.5:
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")
                (startX, startY) = (max(0, startX), max(0, startY))
                (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                face = image[startY:endY, startX:endX]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                face = cv2.resize(face, (224, 224))
                face = img_to_array(face)
                face = preprocess_input(face)

                faces.append(face)
                locs.append((startX, startY, endX, endY))

    if len(faces) > 0:
        faces = np.array(faces, dtype="float32")
```

```
def capture_face(mode, cap):
    while True:
        ret, frame = cap.read()
        if not ret:
            return jsonify({"error": "Failed to capture frame."}), 500

        locs, preds = detect_and_predict_mask(frame)
        label = "No Mask" if mode == "bare" else "Mask"
        valid_capture = False

        for (box, pred) in zip(locs, preds):
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred
            current_label = "Mask" if mask > withoutMask else "No Mask"

            if current_label == label:
                valid_capture = True
                face_locations = detect_faces(frame)
                if face_locations:
                    encodings = face_recognition.face_encodings(frame, face_locations)
                    if encodings:
                        cv2.putText(frame, f"Captured {label} face", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0))
                        cv2.imshow(f"Register {mode.capitalize()} Face", frame)
                        if cv2.waitKey(1) & 0xFF == ord('q'):
                            return encodings[0] # Only capture when the correct face is detected

        if not valid_capture:
            instruction = f"Please {'remove' if mode == 'bare' else 'wear'} your mask to proceed."
            cv2.putText(frame, instruction, (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 255), 2)
```

```
@app.route('/register', methods=['GET', 'POST'])
def register_user():
    if request.method == 'GET':
        return render_template('register.html')

    if request.method == 'POST':
        name = request.form.get("name")
        registration_id = request.form.get("registration_id")

        if not name or not registration_id:
            return jsonify({"error": "Name and Registration ID cannot be empty."}), 400

        cap = cv2.VideoCapture(0)

        bare_face_encoding = capture_face("bare", cap)
        if bare_face_encoding is None:
            return jsonify({"error": "Failed to capture bare face."}), 500

        print("Please wear a mask and capture your masked face.")
        masked_face_encoding = capture_face("masked", cap)
        if masked_face_encoding is None:
            return jsonify({"error": "Failed to capture masked face."}), 500

        # Ensure the face encodings are valid arrays
        if bare_face_encoding is None or masked_face_encoding is None:
            return jsonify({"error": "Invalid face encodings captured."}), 500

        conn = get_db_connection()
        if conn:
            try:
                with conn.cursor() as cursor:
```

```

@app.route('/mark_attendance', methods=['GET'])
def mark_attendance_page():
    return render_template('mark_attendance.html')
@app.route('/mark_attendance', methods=['POST'])
def start_attendance_marking():
    # Check if 'image' is in the request files
    if 'image' not in request.files:
        return jsonify({"error": "No image found in request."}), 400

    # Get the image file from the request
    image_file = request.files['image']
    if not image_file:
        return jsonify({"error": "No image found in request."}), 400

    # Read the image file into a NumPy array
    nparr = np.frombuffer(image_file.read(), np.uint8)
    frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

    # Process the frame to detect and match faces (you can add your face recognition logic here)
    attendance_marked = False
    unregistered_user = False

    locs, preds = detect_and_predict_mask(frame)

    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        current_label = "Mask" if mask > withoutMask else "No Mask"

```

```

# Login route
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        try:
            cursor = mysql.connection.cursor() # Correct method for cursor
            cursor.execute("SELECT id, first_name, last_name, email, password_hash FROM admin_users WHERE user = %s" % email)
            user = cursor.fetchone()
            cursor.close()

            if user:
                if check_password_hash(user[4], password):
                    session['logged_in'] = True
                    session['admin_id'] = user[0] # Save admin ID in session
                    return jsonify({'success': True, 'admin_id': user[0]}) # Return admin ID for local storage
                else:
                    flash("Invalid email or password", "danger")
            else:
                flash("Invalid email or password", "danger")

            return redirect(url_for('login'))

        except Exception as e:
            flash(f"Error: {e}", "danger")
            return redirect(url_for('login'))

    return render_template('login.html')

```

```

# Home route
@app.route('/')
def home():
    return render_template('home.html')

# Signup route
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        first_name = request.form.get('first_name')
        last_name = request.form.get('last_name')
        email = request.form.get('email')
        contact = request.form.get('contact')
        password = request.form.get('password')
        confirm_password = request.form.get('confirm_password')
        secret_key = request.form.get('secret_key')

        if password != confirm_password:
            flash("Passwords do not match!", "danger")
            return redirect(url_for('signup'))

        if secret_key != SECRET_KEY:
            flash("Invalid secret key.", "danger")
            return redirect(url_for('signup'))

        password_hash = generate_password_hash(password)

        try:
            cursor = mysql.connection.cursor()
            cursor.execute("""

```

```

# Logout route
@app.route('/logout', methods=['POST'])
def logout():
    session.clear()
    return redirect(url_for('login'))

# Route to view all students
@app.route('/admin/students')
def view_students():
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM students")
    students = cursor.fetchall()
    return render_template('students.html', students=students)

# Route to delete a student
@app.route('/admin/delete_student/<int:id>', methods=['GET'])
def delete_student(id):
    cursor = mysql.connection.cursor()
    cursor.execute("DELETE FROM students WHERE id = %s", [id])
    mysql.connection.commit()
    return redirect(url_for('view_students'))

# Route to update student information
@app.route('/admin/update_student/<int:id>', methods=['GET', 'POST'])
def update_student(id):
    cursor = mysql.connection.cursor()

    if request.method == 'GET':
        cursor.execute("SELECT * FROM students WHERE id = %s", [id])
        student = cursor.fetchone()
        return render_template('update_student.html', student=student)

```

11 Limitations

11.1 Failure to Account for Edge Cases

Face recognition systems may struggle in low-light conditions or when students have covered faces. These scenarios must be considered to improve system performance.

11.2 Insufficient Error Reporting

The system may fail without providing informative error messages. To enhance usability, every failure in face recognition (e.g., inability to detect a face, network failure) should provide meaningful feedback to users, enabling them to take corrective action.

11.3 Ignoring Multi-Device Compatibility

Although the system is primarily designed for use on laptops with built-in cameras, administrators may require access on tablets or mobile phones. Without multi-device support and adaptability to different screen sizes, the system's usability may be limited.

11.4 Underestimating the Impact of Poor Data Quality

The face recognition system relies heavily on the quality of input data. High-quality, well-lit face images are essential for accurate recognition. Poor-quality images can significantly affect the system's effectiveness, and these requirements must be explicitly stated.

11.5 Neglecting Real-World Environmental Factors

Environmental conditions such as lighting, camera angles, and classroom setups can significantly impact system performance. Failure to account for these factors in the system design may lead to inaccuracies in attendance tracking.

12 Results and Discussion

12.1 Summary of Results

The Face Recognition Attendance System demonstrates substantial improvements over traditional attendance management methods by leveraging state-of-the-art facial recognition technology. Key achievements include:

- Efficient facial data capture, comparison with stored encodings, and attendance marking with high accuracy.
- Adaptability to recognize both bare and masked faces, addressing post-pandemic requirements.
- Significant reduction in time compared to manual methods, with minimal errors in attendance recording.

The system offers a reliable and automated solution for managing attendance records.

12.2 Real-world Applicability

The system has extensive applicability in schools, universities, offices, and other organizations where attendance tracking is critical. By automating the process, it eliminates common issues such as proxy attendance and human error in manual records. Specific applications include:

- Educational institutions: Accurate tracking of student attendance aids in compliance with regulatory requirements.
- Workplaces: Enhances employee monitoring and attendance reporting, improving productivity and resource management.
- Healthcare facilities and public spaces: Adaptability for masked face detection makes it particularly relevant in environments where hygiene is a priority.

This project addresses the growing demand for secure and efficient attendance systems by providing a scalable, real-time solution to improve accountability and prevent fraudulent practices.

13 Conclusion

The Face Recognition Attendance System provides an efficient, secure, and contactless solution for modern attendance management. By leveraging OpenCV DNN, the `face_recognition` library, and MySQL, the system successfully implements facial detection and authentication while streamlining the attendance process. Despite challenges such as lighting variations and masked face recognition, the system has proven its potential to replace traditional methods. Its scalability, accuracy, and integration with biometric technology highlight its suitability for real-world applications, setting a strong foundation for future advancements.

Predicted	No Mask	Mask	Glasses	Angle 45° Right	Angle 45° Left	Correct Person Identified
No Mask	TP	FN	FN	FN	FN	FN
Mask	FN	TP	FN	FN	FN	FN
Glasses	FN	FN	TP	FN	FN	FN
Angle 45° Right	FN	FN	FN	TP	FN	FN
Angle 45° Left	FN	FN	FN	FN	TP	FN
Correct Person Identified	FN	FN	FN	FN	FN	TP

Table 2: Performance of the Facial Recognition-based Attendance System

Table 2 illustrates the performance of the facial recognition-based attendance system under various conditions, such as no mask, mask, glasses, and angled views (45° left

or right). Each scenario evaluates the system's accuracy, with "TP" (True Positive) indicating successful recognition and "FN" (False Negative) representing failures. The system performs well in straightforward conditions, like identifying uncovered faces (No Mask), but struggles in scenarios like masks, glasses, and angled views, as evident from the frequent "FN" entries. These challenges highlight the system's sensitivity to occlusions and non-frontal angles, which are common in real-world environments. The "Correct Person Identified" row reflects the system's overall ability to match identities, which is critical for reliable attendance tracking.

14 Future Work

Future improvements to the system include:

- Adopting advanced deep learning models such as YOLO or Faster R-CNN for enhanced accuracy and reliability, especially for masked face recognition.
- Expanding functionality through multi-modal biometrics, cloud integration, and real-time reporting to increase scalability and usability.
- Implementing security upgrades such as encrypted data handling.
- Adding cross-platform support to optimize the system for use across various devices and operating systems.

With these advancements, the system can evolve into a robust, scalable solution for modern attendance management.

References

- [1] P. Pattnaik and K. K. Mohanty, “Ai-based techniques for real-time face recognition-based attendance system-a comparative study,” in *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2020, pp. 1034–1039.
- [2] K. N. Reddy, T. Alekhya, T. Sushma Manjula, and R. Krishnappa, “Ai-based attendance monitoring system,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 2S, pp. 592–597, 2019.
- [3] S. Abirami, S. Jyothikamalesh, M. Sowmiya, S. Abirami, S. A. L. Mary, and C. Jayasudha, “Ai-based attendance tracking system using real-time facial recognition,” in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*. IEEE, 2022, pp. 1330–1335.
- [4] S. K. Arora, P. Behki, G. Batar, V. Tiwari, and S. Jindal, “Face detection attendance system in artificial intelligence,” in *Advances in AI for Biomedical Instrumentation, Electronics and Computing*. CRC Press, 2024, pp. 491–496.
- [5] S. K. Shakya and R. Misra, “Face recognition attendance system, smart learning, college enquiry using ai chat-bot.”
- [6] B.-T. Nguyen-Tat, M.-Q. Bui, and V. M. Ngo, “Automating attendance management in human resources: A design science approach using computer vision and facial recognition,” *International Journal of Information Management Data Insights*, vol. 4, no. 2, p. 100253, 2024.
- [7] K. M. M. Uddin, A. Chakraborty, M. A. Hadi, M. A. Uddin, and S. K. Dey, “Artificial intelligence based real-time attendance system using face recognition,” in *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*. IEEE, 2021, pp. 1–6.
- [8] D. D. Nguyen, X. H. Nguyen, T. T. Than, and M. S. Nguyen, “Automated attendance system in the classroom using artificial intelligence and internet of things technology,” in *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2021, pp. 531–536.
- [9] J. RajaSekhar, S. Sakhamuri, A. Dhruva Teja, and T. Siva Sai Bhargav, “Automatic attendance management system using ai and deep convolutional neural network,” in *Embracing Machines and Humanity Through Cognitive Computing and IoT*. Springer, 2023, pp. 67–76.
- [10] P. B. Jha, A. Basnet, B. Pokhrel, B. Pokhrel, G. K. Thakur, and S. Chhetri, “An automated attendance system using facial detection and recognition technology,” *Apex Journal of Business and Management*, vol. 1, no. 1, pp. 103–120, 2023.
- [11] A. K. Shukla, A. Shukla, and R. Singh, “Automatic attendance system based on cnn-lstm and face recognition,” *International Journal of Information Technology*, vol. 16, no. 3, pp. 1293–1301, 2024.

- [12] M. H. M. Kamil, N. Zaini, L. Mazalan, and A. H. Ahamad, "Online attendance system based on facial recognition with face mask detection," *Multimedia Tools and Applications*, vol. 82, no. 22, pp. 34 437–34 457, 2023.
- [13] A. S. Bist, W. Febriani, C. Lukita, S. Kosasi, and U. Rahardja, "Design of face recognition attendx for recording student attendance data based on artificial intelligence technology," *Solid State Technology*, vol. 63, no. 2s, pp. 4505–4518, 2020.
- [14] A. S. Rafika, M. Hardini, A. Y. Ardianto, D. Supriyanti *et al.*, "Face recognition based artificial intelligence with attendx technology for student attendance," in *2022 international conference on science and technology (ICOSTECH)*. IEEE, 2022, pp. 1–7.
- [15] R. S. Kumar, A. Rajendran, V. Amrutha, and G. T. Raghu, "Deep learning model for face mask based attendance system in the era of the covid-19 pandemic," in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2021, pp. 1741–1746.
- [16] A. Aljohnai, R. E. Al-Mamlook, and N. Alharbe, "Ai-based attendance management system using image processing techniques during covid-19 pandemic."
- [17] A. Ahila, A. Poonguzhali, M. Deepa, A. Arthy, R. Saravanakumar *et al.*, "Artificial intelligence for realtime face recognition attendance using college classrooms and buses," in *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, vol. 1. IEEE, 2024, pp. 1–6.
- [18] H. Kale, K. Aswar, and D. Y. M. K. Yadav, "Attendance marking using face detection," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 417–424.
- [19] M. Ali, A. Diwan, and D. Kumar, "Attendance system optimization through deep learning face recognition," *International Journal of Computing and Digital Systems*, vol. 15, no. 1, pp. 1527–1540, 2024.
- [20] S. K. Tiwari, V. Fande, G. N. Patil, K. Kalbande, and D. M. Khanapurkar, "A novel approach for attendance monitoring system with face mask detection," in *AIP Conference Proceedings*, vol. 2424, no. 1. AIP Publishing, 2022.