Face Recognition Attendance System Using Support Vector Machine (SVM)

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE AWARD OF THE
DEGREE OF **BACHELOR OF TECHNOLOGY**
IN **INFORMATION TECHNOLOGY**
OF THE ANNA UNIVERSITY

MINI PROJECT WORK

Submitted by

LOGESHWARAN VV
71772118125
SEDHU RAM P
71772118138
SRIDHAR E
71772118145
TAMIZHARASU M
71772118L10

2024

Under the Guidance of
Dr. R. DEVI, M.Tech., Ph.D.,

DEPARTMENT OF INFORMATION TECHNOLOGY
GOVERNMENT COLLEGE OF TECHNOLOGY
(An Autonomous Institution affiliated to Anna University)
COIMBATORE - 641 013

DEPARTMENT OF INFORMATION TECHNOLOGY
GOVERNMENT COLLEGE OF TECHNOLOGY
(An Autonomous Institution affiliated to Anna University)
COIMBATORE - 641 013

MINI PROJECT WORK

DECEMBER 2024

This is to certify that this project work entitled

FACE RECOGNITION ATTENDANCE SYSTEM USING SUPPORT VECTOR MACHINE (SVM)

is the bonafide record of project work done by

LOGESHWARAN VV
71772118125
SEDHU RAM P
71772118138
SRIDHAR E
71772118145
TAMIZHARASU M
71772118L10

of B.TECH INFORMATION TECHNOLOGY during the year 2024 – 2025

______________________________	______________________________
Project Guide	Head of the Department
DR. R. DEVI M.TECH., Ph.D.,	**DR. S. RATHI M.E., Ph.D.,**

Submitted for the Project Viva-Voce examination held on _______________

____________________________	______________________________
Internal Examiner	External Examiner

ACKNOWLEDGEMENT

Great achievements are not possible without standing on the shoulders of giants. Without the active involvement of the following experts this project would not have been a reality.

We express our sincere gratitude to **Dr. K. Manonmani, M.E., Ph.D.,** Principal, Government College of Technology, Coimbatore for providing us all facilities that we needed for the completion of this project.

We whole-heartedly express our thankfulness and gratitude to **Dr. S. Rathi, M.E., Ph.D.,** Professor and Head of the Department of Information Technology, Government College of Technology, for helping us to successfully carry out this project.

Our thankfulness and gratitude to our respectable project guide **Dr. R. Devi, M.Tech., Ph.D.,** Assistant Professor who has been an immense help through the various phases of the project. With her potent ideas and excellent guidance, we were able to comprehend the essential aspects involved.

We would like to thank our faculty advisor **Dr. M. Blessy Queen Mary, M.E., Ph.D.,** Assistant Professor for her continuous support and encouragement throughout this project.

We extend our sincere thanks to the staff members of Information Technology department, **Dr. C. Aswini, M.E., Ph.D.,** Assistant Professor, **Dr. T. Suguna, M.Tech., Ph.D.,** Assistant Professor, **Dr. S. Gladson Oliver, M.Tech., Ph.D.,** Assistant Professor, **Prof. M. Jeyanthi, M.Tech.,** Assistant Professor, **Dr. R. Malavika, M.Tech., Ph.D.,** Assistant Professor, for rendering their help for the completion of this project. We also thank all our friends for their cooperation and suggestions towards the successful completion of this project.

SYNOPSIS

The **Face Recognition Attendance System** is a modern solution designed to automate attendance tracking using facial recognition technology. Traditional attendance systems, such as manual registers or RFID-based solutions, are often prone to errors, inefficiencies, and manipulation. This project addresses these challenges by leveraging the power of machine learning and computer vision to provide a secure, efficient, and user-friendly method for recording attendance.

The system begins with a dataset creation module, where users' facial images are captured and labeled with identifying details like name and roll number. These images are processed using Tensorflow.js and OpenCV and Dlib to extract facial embeddings, which represent the unique features of a person's face. These embeddings are then used to train a machine learning model capable of recognizing individuals with high accuracy. During operation, a live video feed or webcam stream is analyzed in real time to identify registered users, and their attendance is logged with a timestamp into a secure database.

A web-based interface, built using Python's Flask framework, provides an intuitive platform for administrators to manage the system. Features include options to view real-time attendance, access attendance records, and export data in CSV(Comma Separated Value) format. Additionally, functionalities like training the recognition model and preprocessing embeddings are seamlessly integrated into the dashboard. The solution is designed to be scalable, making it suitable for educational institutions, offices, or any organization that requires reliable attendance tracking.

The project emphasizes usability, security, and accuracy, employing advanced tools and techniques in machine learning, computer vision, and web development. It significantly reduces manual effort, prevents fraudulent attendance marking, and ensures a streamlined process for managing attendance records.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM	SUPPORT VECTOR MACHINE
HOG	HISTOGRAM OF GRADIENTS
LBPH	LOCAL BINARY PATTERN HISTOGRAM
CNN	CONVOLUTIONAL NEURAL NETWORKS
MLP	MULTILAYER PERCEPTRON
GPU	GRAPHICAL PROCESSING UNIT
OPENCV	OPEN SOURCE COMPUTER VISION
KNN	K-NEAREST NEIGHBOURS
LSTM	LONG SHORT-TERM MEMORY
RNN	RECURRENT NEURAL NETWORKS
GA	GENETIC ALGORITHM
MTCNN	MULTI-TASK CASCADED CONVOLUTIONAL NEURAL NETWORKS
ResNet	RESIDUAL NEURAL NETWORK
YOLO	YOU ONLY LOOK ONCE
LDA	LINEAR DESCRIMINAT ANALYSIS
PIL	PYTHON IMAGING LIBRARY

CHAPTER 1
INTRODUCTION

1.1 DESCRIPTION

This project aims to automate the process of attendance marking in educational and professional environments through face recognition technology. The system leverages the power of deep learning and machine learning techniques to accurately identify individuals and mark their attendance in real-time, providing a reliable and secure alternative to traditional attendance systems. Face recognition is becoming an essential tool in various IoT-based applications, from smart classrooms to corporate offices. By eliminating manual checks and reducing human errors, this system offers an efficient solution to automate attendance, minimize fraud, and ensure that only authorized individuals are marked present. The face recognition model in this project uses Dlib model and Support Vector Machine (SVM) for facial feature extraction and classification. Dlib provides robust facial landmark detection and recognition capabilities, while SVM is employed as a classifier to accurately identify the registered faces and mark attendance. These models analyze real-time video feed captured through webcams, enabling quick recognition and attendance marking. The project is built using the Flask web framework for the backend, facilitating smooth communication between the frontend and the machine learning models. The frontend is developed using Hyper Text Markup Language (HTML),Cascading Style Sheets (CSS) and Java Script (JS) providing an intuitive interface for users to interact with the system. This includes functionalities such as face registration, viewing attendance, and generating attendance reports.

1.2 PROBLEM STATEMENT

In educational institutions, workplaces, and various organizations, tracking attendance is a critical task for monitoring participation and ensuring accountability. Traditional methods, such as manual attendance registers or ID card-based systems, are labor-intensive, prone to errors and manipulation. These methods often lead to inefficiencies, including inaccurate records, buddy punching, and unauthorized proxy attendance. Additionally, manually processing attendance data for analysis or reporting purposes requires significant effort and is susceptible to human error.The increasing demand for automation and security in everyday processes highlights the need for a more efficient and reliable attendance tracking system. Existing biometric systems, such as fingerprint or Radio-Frequency Identification(RFID) based solutions, require physical interaction with devices, which raises hygiene concerns, especially in scenarios like pandemics. Furthermore, these systems can be costly to maintain and lack the flexibility of modern, contactless solutions.

The challenge lies in designing a system that is accurate, contactless, user-friendly, and capable of integrating seamlessly into existing workflows. A robust solution must address issues of scalability, real-time processing, and secure storage of attendance data while ensuring minimal user intervention. The proposed **Face Recognition Attendance System** seeks to resolve these challenges by employing advanced facial recognition technology, which provides a fast, non-intrusive, and secure method of attendance tracking.

1.3 EXISTING SYSTEM

- Manual Attendance Marking
- Biometric Attendance Marking
- RFID based Attendance Marking
- QR code based Attendance Marking
- OTP based Attendance Marking

1.4 ADVANTAGES AND DISADVANTAGES OF EXISTING SYSTEM

Table 1.1

Attendance System	Advantages	Disadvantages
Manual Attendance Marking	**Simplicity**: No need for special equipment, only pen and paper or simple software. **Customizable**: Easily adapted to various attendance policies.	**Error-prone**: High risk of human error or tampering (proxy attendance). **Inefficient**: Difficult to track and analyze for large attendance data.
Biometric Attendance Marking		
Fingerprint-based	**Highly Accurate**: Unique fingerprint ensures authenticity. **Fast**: Quick scanning process.	**Hygiene Issues**: Frequent contact with devices can lead to hygiene concerns. **False Negatives**: Dirty or wet fingers can cause false rejections. **Expensive**: Requires specialized equipment for scanning and software integration. **Proxy Attendance**: Making fake fingerprint leads to false entries.
Facial Recognition	**Contactless**: No physical contact required, making it hygienic. **Convenient**: Fast and automatic marking once the face is detected. **High Accuracy**: Advanced algorithms can detect faces even in different lighting conditions.	**Environmental Limitations**: Poor lighting or extreme angles may affect recognition. **Privacy Concerns**: Storing facial data raises privacy and security issues.

		False Positives/Negatives: Can have difficulty distinguishing between similar faces or detecting masks/hats.
Iris/Retina Scan	**Highly Secure**: Iris or retina patterns are more unique than fingerprints. **Contactless**: Eliminates hygiene concerns.	**Expensive**: Requires advanced, costly hardware. **User Discomfort**: Some users may find it uncomfortable to scan their eyes.
RFID-based Attendance		
Card Swipe	**Low Cost**: Simple technology that's widely available. **Easy to Implement**: Simple setup process. **Reliable**: Can handle a large number of users with minimal error.	**Lost or Damaged Cards**: Users may lose their RFID cards, and they can be easily damaged. **Proxy Attendance**: Cards can be shared, leading to buddy punching. **Maintenance**: Readers may require regular maintenance.
QR Code-based Attendance	**Easy to Implement**: Users only need a smartphone and a QR code scanner. **Contactless**: No physical contact, making it hygienic. **Low Cost**: No need for specialized hardware except a smartphone camera.	**Dependence on Smartphones**: Users without smartphones are excluded. **Cheating**: QR codes can be shared or replicated. **Slower for Large Groups**: Scanning QR codes for large groups can take time.

OTP-based Attendance	**Secure**: Each login is authenticated with a one-time password. **No Specialized Equipment**: Users only need access to a phone or email.	**Manual Entry**: Requires users to enter an OTP, which can be slower than other methods. **Dependence on Connectivity**: Requires network access to receive OTPs. **Potential Delays**: Delay in OTP delivery can slow down attendance marking.

1.5 PROPOSED SYSTEM

The **Face Recognition Attendance System** aims to revolutionize the traditional attendance process by integrating facial recognition technology to ensure accuracy, efficiency, and a seamless user experience. The system is designed to automate attendance tracking by leveraging computer vision and machine learning, making it suitable for educational institutions, workplaces, and other environments requiring attendance management.

The system's core functionality begins with a **dataset creation phase**, where individuals register by capturing multiple facial images using a webcam. These images are processed and stored in a structured format. Using **Dlib's pre-trained models**, facial features are extracted as embeddings, representing unique facial characteristics. These embeddings, along with individual details, are saved in pickle file for further processing.

After dataset creation, the embeddings are preprocessed, and a machine learning model is trained using a **Support Vector Machine (SVM) classifier**. The SVM classifier ensures accurate and reliable facial recognition by learning the relationships between the facial embeddings and their associated identities. This training phase equips the system to recognize individuals with high precision in real-world scenarios.

In real-time attendance mode, the system captures video streams via a webcam, detects faces, and extracts facial features. The trained model then compares these features to identify the person. Once identified, the system checks if their attendance for the day has already been marked. If not, it logs their attendance with a timestamp in a **MySQL database**. This ensures secure, accurate, and automated attendance recording while minimizing manual errors and the risk of proxy attendance.

The system also provides several features to manage and monitor attendance data:

- **Attendance Logs**: Administrators can view real-time attendance data, displaying names and timestamps.
- **Export Functionality**: Attendance records can be exported as **CSV files** for external reporting or analysis.
- **Registered Student Details**: The system allows administrators to view all registered individuals.
- **Web Interface**: A responsive and interactive user interface, built using **Flask** and **Bootstrap**, enhances usability and accessibility.

Technologies Used

The system is developed using the following technologies:

- **Python**: For backend processing and machine learning tasks.
- **Dlib**: For face detection and the generation of 128-dimensional facial embeddings.
- **TensorFlow.js:** is a JavaScript library that enables machine learning and deep learning directly in the browser or on Node.js. For face detection, TensorFlow.js allows real-time processing and eliminates the need for server-side computation, making it lightweight and efficient.
- **OpenCV**: For real-time image processing.
- **Flask Framework**: To manage the web application, routes, and templates.
- **Bootstrap**: For creating a responsive and user-friendly interface.

- **MySQL**: A relational database to securely store registered users and attendance logs.
- **SVM Classifier**: A robust machine learning algorithm used for face classification.
- **CSV Handling**: To facilitate exporting attendance data in a structured format.

The proposed system is designed to enhance the efficiency of attendance tracking processes while maintaining accuracy and ease of use.

CHAPTER 2

LITERATURE SURVEY

2.1 Online attendance system based on facial recognition with face mask detection[1]

2.1.1 Description:

The paper presents an online system for recording attendance based on facial recognition and face mask detection. The system is accessible through any web browser, eliminating the need for users to install specific software.

- **System Components**

  Server Application: This component is responsible for training the facial recognition model, processing uploaded images, and storing attendance information in a database.

  Client-Side Application: This component is the web interface that allows users to capture selfie images via webcam and upload them to the server for processing.

- **System Workflow**

  Model Training: The server application trains a facial recognition model using two datasets: original facial images of users without masks and synthetically generated images of the same users wearing masks. This allows the system to recognise users with or without masks.

  Image Capture: Users open the system's web interface in their browser and use their webcam to capture a selfie image.

  Image Upload: The user uploads the captured selfie image to the server by clicking a button on the web interface.

  Image Processing: The server receives the uploaded image and activates a Python script to perform facial recognition and face mask detection.

Attendance Recording: Based on the facial recognition results, the system records the user's attendance in a database along with the time of submission and mask-wearing status.

Output Display: The server sends the processed image back to the user's browser, displaying the identified user's name and mask-wearing status on the image.

- **Face Recognition and Mask Detection**

  The system utilises Python code for facial recognition and mask detection. Initially, two separate Python scripts were used: one for facial recognition and another for mask detection. However, this approach resulted in long processing times. To optimise the process, the researchers implemented a single Python script trained using both original and synthetic datasets, enabling simultaneous facial recognition and mask detection with a shorter processing time.

- **Accuracy and Optimisation**

  The accuracy of the system was calculated at 81.8% for face recognition and 80% for face mask detection.

2.1.2 Merits:

- **Accessibility and User-friendliness**: The system is designed as a web-based application, making it accessible to users through any web browser without requiring any specific software installation. This enhances the system's user-friendliness and convenience for both administrators and attendees.
- **Real-time Attendance Recording and Automation:** The system automates the process of attendance recording in real time, capturing and storing attendance data as users upload their selfie images. This eliminates manual processes, reducing errors and saving time for administrators.
- **Integrated Face Mask Detection for Safety Compliance:** The system incorporates face mask detection, adding a layer of safety compliance

monitoring. This is particularly relevant in the context of the COVID-19 pandemic, helping institutions and workplaces enforce safety protocols.

2.1.3 Demerits:

- **Dependence on Internet Connectivity**: The system relies on internet connectivity for users to access the web interface and upload images to the server. In situations with unreliable internet access, the system's functionality may be limited.
- **Potential for Image Quality Issues**: The accuracy of facial recognition and mask detection depends on the quality of uploaded selfie images. Factors such as lighting, camera angle, and image resolution can affect the system's performance.
- **Privacy Concerns and Data Security**: The system's use of facial recognition technology raises privacy concerns regarding the collection, storage, and potential misuse of facial images. Robust data security measures are crucial to protect sensitive user information.

2.2 Face Recognition Based Attendance System Using Histogram of Oriented Gradients and Linear Support Vector Machine[2]

2.2.1 Description:

The system aims to address the limitations of traditional manual attendance methods, such as prone to errors and allowing proxy attendance. The system works by capturing a group image, detecting and recognizing individual faces within that image, and then automatically marking attendance in a database.

This is achieved through four main steps:

- **Dataset creation**: Images of individuals' faces are stored with proper labels, creating a reference dataset.
- **Information entry**: Details of each individual, such as ID, name, class, branch and department, are entered and stored in a SQL database.

- **Group image capture and comparison**: A group image is taken, and the system detects and compares each face with the stored dataset. If a match is found, the individual's details are recorded.
- **Attendance marking**: Based on the recognized faces, the system marks attendance in the database with relevant details, including date and time.
- **Key Techniques and Technologies**

  HOG (Histogram of Oriented Gradients) and SVM (Support Vector Machine): These algorithms are used for face recognition, offering high accuracy. The system employs HOG features as input data and an RBF (Radial Basis Function) kernel in the SVM model.

  Python and Tkinter: The system is developed using Python, and Tkinter is used to create the GUI (Graphical User Interface) for user interaction.

  OpenCV and face_recognition libraries: These libraries are crucial for image processing tasks, including face detection and recognition.

- **Accuracy and Results**

  The system achieved an impressive accuracy of approximately 95% in various trials, demonstrating its effectiveness in recognizing faces and marking attendance.

2.2.2 Merits:

- **Increased Efficiency and Accuracy:** The system automates the process of marking and tracking attendance, eliminating the error-prone nature of traditional manual methods.
- **Reduced Human Error and Proxy Attendance:** The use of facial recognition technology minimizes the possibility of human errors in marking attendance, such as missed entries or incorrect recordings. The system's reliance on unique facial features makes it difficult for individuals to mark attendance for others, preventing proxy attendance.
- **Real-Time Monitoring and Data Security:** The system provides real-time attendance data that can be accessed and monitored easily. This allows for

timely intervention in case of absenteeism or other attendance-related issues. The attendance data is securely stored in a SQL database, ensuring its safety and integrity.

- **Cost-Effective and Scalable Solution:** Compared to other biometric systems like iris or fingerprint recognition, the camera-based facial recognition system reduces the complexity and cost of hardware infrastructure.

2.2.3 Demerits:

- **Vulnerability to Spoofing Attacks:** The system's security can be compromised through spoofing attacks, where individuals may attempt to deceive the system using photographs or masks.
- **Dependence on Environmental Conditions**: Facial recognition accuracy can be affected by variations in lighting, camera angle, and image quality.

2.3 Student attendance with face recognition (LBPH or CNN)[3]

2.3.1 Description:

The paper presents a systematic literature review examining the effectiveness of Convolutional Neural Networks (CNN) and Local Binary Pattern Histogram (LBPH) algorithms for implementing face recognition in university attendance systems. The authors aim to identify the most suitable algorithm for minimizing errors in attendance recording, taking into account accuracy, stability, and external factors.

Algorithms Compared:

- **Traditional Algorithms**: These algorithms require manual feature extraction from images. The review specifically compares **LBPH**, **Eigenface**, and **Fisherface** within this category.678
- **Deep Learning Algorithms**: These algorithms automatically learn features from data. The review highlights **CNN** as a representative of this category, comparing it with **Support Vector Machines (SVM)** and **Multilayer Perceptron (MLP)**.

Findings and Discussion

The review reveals that **CNN generally outperforms LBPH and other algorithms in terms of accuracy and stability**.

Accuracy:

CNN consistently achieved higher accuracy rates compared to LBPH. One study reported a **99% accuracy rate for CNN** using a dataset of 1050 images, while the **highest accuracy for LBPH was 92%** with a dataset of 165 images.

Accuracy for traditional algorithms (LBPH, Eigenface, Fisherface) varied significantly. LBPH generally performed better, but the review suggests its accuracy can be influenced by external factors like lighting, distance, and head coverings.

Stability:

CNN demonstrated greater stability in accuracy even when influenced by external factors like facial position. While the accuracy of LBPH fluctuated more noticeably with changes in factors such as lighting and distance, CNN appeared more robust.

External Factors:

Both CNN and LBPH are susceptible to external factors like lighting, distance, and face position, but **CNN generally shows greater resilience**.

Position significantly affected the accuracy of both CNN and LBPH, particularly in cases of sloping or downward-facing positions.

2.3.2 Merits:

- **High Accuracy Potential, Especially with CNN:**The systematic literature review demonstrates that both CNN and LBPH can achieve reasonably high accuracy rates for face recognition, with CNN consistently outperforming LBPH and other traditional algorithms like Eigenface and Fisherface. Several cited studies report impressive accuracy rates exceeding 90% for both CNN and LBPH. One study using a CNN-based system with a dataset of 1050 images even reached 99% accuracy.These findings suggest that face recognition, particularly using CNN, holds significant promise for accurate and reliable automated attendance tracking.

- **Accessibility and Ease of Deployment with Web-Based Systems:**The online attendance system presented in one of the sources showcases the feasibility of developing web-based face recognition systems that are accessible through standard web browsers. This approach eliminates the need for users to download and install specific software, simplifying deployment and accessibility for both administrators and students.
- **CNN's Ability to Learn and Adapt:**One of the key advantages of CNNs is their ability to automatically learn and extract features from images, eliminating the need for manual feature engineering.As the CNN is exposed to more data during training, it can refine its feature representation and improve its ability to generalise to new faces and variations in appearance, leading to higher accuracy over time.

2.3.3 Demerits:

- **Impact of External Factors on Accuracy:** Both CNN and LBPH exhibit susceptibility to external factors that can degrade accuracy. The research cited in the systematic review indicates that lighting conditions, distance from the camera, head coverings (like hats or scarves), and even face position (straight, sloping, looking down) significantly influence the performance of both algorithms.
- **Computational Demands of CNN:**Deep learning algorithms, particularly CNN, often require substantial computational resources, including powerful Graphical Processing Unit(GPU)s and ample memory. This computational intensity can pose challenges for deployment, especially on resource-constrained devices or in settings with limited processing power.

2.4 Face Recognition Based Attendance System Using Real Time Data[4]

2.4.1 Description:

The paper, titled "**Face Recognition Based Attendance System Using Real Time Data**" investigates the development and evaluation of a facial recognition system designed for tracking attendance in real-time.They propose that face recognition technology offers a viable solution for automating attendance processes, leading to enhanced accuracy and streamlined management.The paper outlines the system's architecture, detailing the process of dataset preparation, model training, and performance evaluation. It emphasises the system's efficacy in real-world scenarios.

- **System Methodology:** The system is built using the Python programming language and leverages the OpenCV library for facial recognition and image processing. The system captures an individual's face using a camera. The facial recognition algorithm extracts facial features from the captured image and compares them against a database of known faces to determine the person's identity. Attendance data, including the individual's name and time of attendance, is stored in a database.
- **Comparison with Existing Research:** The paper reviews existing research on attendance systems, including a study that employed NFC technology and embedded cameras on mobile devices for attendance tracking. It highlights the limitations of such systems, including their reliance on personal mobile phones and the potential for unauthorised Near Field Communication(NFC) tag use. Another study mentioned in the paper utilised the Viola-Jones technique for face detection and Local Binary Patterns Histograms for recognition, storing attendance information in a SQLite database. The authors differentiate their system by using a simpler approach based on the OpenCV algorithm in Python and storing data in an XML database.
- **Algorithm Selection:** The paper selects the K-Nearest Neighbours (KNN) algorithm for image recognition. It explains that KNN is suitable for both classification and regression tasks, determining similarity between new and

existing data points. It clarifies that KNN does not rely on assumptions about the underlying data, making it a non-parametric algorithm. It describes KNN as a "lazy learner" algorithm that stores the dataset and acts upon it during classification rather than learning from a training set.

- **System Design:** It describes a two-phase process: student enrolment and attendance marking. During enrolment, the system captures facial images using a webcam and extracts features using the Haar cascade algorithm. The student's details and images are stored in the database as a NumPy array. The attendance marking phase uses a webcam to capture images, recognise faces using the KNN algorithm, and update the attendance records in the database.
- **GUI Development:** The paper mentions the development of a graphical user interface (GUI) for user interaction. The GUI provides options for student registration, attendance marking, and viewing attendance records.
- **Results and Advantages:** The paper states that the developed system has been successfully tested and is accurate, reliable, and efficient in processing a large number of individuals. It highlights the system's ability to eliminate physical contact, reduce proxy attendance, and provide real-time attendance data tracking. The paper lists several advantages of the system, including increased accuracy, improved efficiency, enhanced security, real-time data availability, and improved record-keeping.

2.4.2 Merits:

- **Increased Accuracy:** The system utilises biometric identification, minimising errors and proxy attendance. Unlike manual roll calls or card-based systems, facial recognition relies on unique facial features, making it difficult for individuals to mark attendance for others.
- **Improved Efficiency:** The system processes attendance rapidly, handling a large number of individuals efficiently. Automated face recognition eliminates the time-consuming process of manual attendance recording.
- **Enhanced Security:** Biometric identification enhances security by making it harder for unauthorised individuals to gain access or manipulate

attendance records. The reliance on unique facial features for authentication strengthens security measures.

- **Real-time Data:** The system provides real-time attendance tracking, allowing immediate access to attendance data and enabling timely identification of patterns and intervention if required. This feature is beneficial for monitoring student attendance and ensuring timely action.

2.4.3 Demerits:

- **Privacy Concerns:** The use of facial recognition technology raises privacy concerns related to the collection, storage, and potential misuse of personal biometric data. The paper acknowledges these concerns as a challenge that needs to be addressed.
- **Accuracy of Facial Recognition Technology:** The accuracy of facial recognition technology can be affected by factors such as lighting conditions, image quality, and facial expressions, potentially leading to inaccurate attendance recording. While the authors claim the system is accurate, they acknowledge the inherent limitations of facial recognition technology.

2.5 Automatic attendance system based on CNN–LSTM and face recognition[5]

2.5.1 Description:

This research paper, titled "**Automatic attendance system based on CNN–LSTM and face recognition**", proposes a novel approach to automated attendance tracking using a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs). The paper focuses on the development and evaluation of a face recognition system that leverages the strengths of CNNs for feature extraction and LSTMs for handling temporal dependencies in attendance data.

- **Rationale for the Proposed System:** The authors argue that while biometric techniques offer higher security and accuracy compared to nonbiometric methods, existing biometric systems can be time-consuming due to the processing time required for individual identification. They propose the use of CNNs and LSTMs to address these limitations.

- **CNNs for Feature Extraction:** The paper explains that CNNs excel at extracting complex features from images, making them suitable for face recognition tasks. CNNs use convolutional layers to learn spatial hierarchies of features, which are then used for classification.
- **LSTMs for Handling Temporal Dependencies:** The paper highlights the ability of LSTMs to capture long-term dependencies in sequential data. LSTMs are a type of Recurrent Neural Network (RNN) specifically designed to address the vanishing gradient problem that can occur in traditional RNNs when processing long sequences. LSTMs are able to learn and remember patterns over extended periods, making them well-suited for handling the temporal nature of attendance data.
- **Data Augmentation:** The authors emphasise the use of data augmentation techniques to enhance the system's performance and generalizability. Data augmentation involves creating variations of the original training data, such as rotated, flipped, or zoomed images, to increase the diversity of the dataset and reduce overfitting.
- **System Architecture and Processes:** The paper details the various stages involved in the system's operation, including dataset creation, face detection, feature extraction using CNNs, classification using LSTMs, and attendance marking. The authors also describe the application module, encompassing teacher and student interfaces for data input, attendance verification, and report generation.
- **Experimental Results and Performance Evaluation:** The paper presents the results of experiments conducted to evaluate the system's accuracy and efficiency. The authors report a high face recognition accuracy of 99.82%, outperforming several existing methods discussed in the literature review. They also compare the system's runtime with other techniques, demonstrating its efficiency in processing attendance data.

2.5.2 Merits:

- **Robust Feature Extraction and Classification:** The source demonstrates the effective use of convolutional layers, pooling layers, and fully connected layers in the CNN architecture to process and classify facial images. The integration of LSTMs further enhances the system's capability by capturing long-term dependencies in sequential data. This combination is particularly relevant for attendance tracking, where the system needs to identify students over time and maintain accurate records.
- **Data Augmentation for Improved Generalisability:** The source acknowledges the importance of data augmentation in mitigating the risk of overfitting and improving the model's ability to generalise to unseen data. It employs techniques such as rotation, zoom, shear, blurring, noise addition, and horizontal flipping to create variations of the original images.

2.5.3 Demerits:

- **Limited Scope of Dataset and Generalisability:** The source does not provide in-depth information about the dataset used for training and evaluating the system. It mentions using 2800 facial images for the dataset creation process, but it lacks details regarding the dataset's diversity, demographics, and potential biases. The limited scope of the dataset raises concerns about the system's generalisability to real-world scenarios with varying lighting conditions, facial expressions, and ethnicities.
- **Lack of Detail on Real-Time Implementation Challenges:** While the paper describes the system's architecture and processes, it does not adequately address the technical challenges associated with real-time implementation.

2.6 Development of an Improved Convolutional Neural Network for an Automated Face-Based University Attendance System[6]

2.6.1 Description:

The research paper explores the use of a Convolutional Neural Network (CNN) enhanced with a Genetic Algorithm (GA) for creating a more accurate and efficient automated attendance system.They propose that face recognition technology offers a solution to these issues, highlighting its ease of use and non-intrusive nature.The paper focuses on improving the performance of CNNs using GAs to address challenges such as high computational costs and determining optimal model parameters. This approach aims to enhance the accuracy and efficiency of face recognition for attendance tracking.

- **Data Acquisition and Preprocessing:** The study involved collecting facial images of students from Ajayi University, Oyo, Nigeria, using a digital camera. These images underwent preprocessing steps, including conversion to grayscale, histogram equalization for normalization, thinning to remove extraneous pixels, and data augmentation techniques.
- **Feature Extraction and Classification:** The researchers employed a deep learning approach using CNNs combined with GAs for feature extraction and classification. CNNs are designed to automatically learn relevant features from images, eliminating the need for manual feature selection. The integration of GAs aims to optimise the CNN's parameters (weights) for improved performance.
- **Training and Evaluation:** The collected dataset was split into training and testing sets. The system was trained using the training set, and its performance was evaluated using the testing set. The researchers used various evaluation metrics, including recognition accuracy, False Acceptance Rate (FAR), False Rejection Rate (FRR), and computation time.
- **Results and Conclusions:** The study demonstrated that the CNN-GA model achieved a **higher recognition accuracy of 96.49%** compared to

the standard CNN model's accuracy of 92.54%.The CNN-GA model also exhibited a **shorter recognition time** than the CNN model.

2.6.2 Merits:

- **Addresses Limitations of Traditional Attendance Systems:** The paper effectively identifies the shortcomings of traditional attendance methods, such as being inaccurate, and prone to manipulation. It argues convincingly for the adoption of face recognition technology as a more efficient and reliable solution.
- **Novel Approach Using CNN-GA:** The research proposes a novel approach by combining CNNs with GAs to improve the performance of the attendance system. This hybrid approach leverages the strengths of both techniques. CNNs excel at automatically learning features from images, while GAs optimise the CNN's parameters for improved accuracy and efficiency.

2.6.3 Demerits:

- **Limited Dataset Details and Generalisability**: While the paper describes the data acquisition process, it lacks specific details about the dataset's characteristics, such as size, diversity in terms of demographics and facial features, and potential biases. This lack of information raises concerns about the generalisability of the system's performance to real-world scenarios with varying lighting conditions, facial expressions, and ethnicities.
- **Lack of Implementation Details and Scalability Discussion:** The paper focuses on the technical aspects of the CNN-GA model but provides limited information about practical implementation details. It does not discuss hardware and software requirements, integration with existing infrastructure, or the scalability of the system for larger deployments. Addressing these practical considerations would enhance the paper's relevance for real-world applications.

2.7 Face Recognition-Based Automatic Attendance System in a Smart Classroom[7]

2.7.1 Description:

This article presents a **facial recognition-based attendance system** developed for a smart classroom setting. The researchers highlight the limitations of traditional attendance methods, such as paper-based systems and RFID cards, citing issues like time consumption, potential for fraud, and security risks. They argue that biometric systems, particularly facial recognition, offer a more efficient and secure alternative. The article discusses several face recognition algorithms, including:**Viola-Jones**,**HOG (Histogram of Oriented Gradients),LBPH (Local Binary Patterns Histograms)**,**Eigenfaces, Fisherfaces,SIFT (Scale-Invariant Feature Transform)**,**ORB (Oriented FAST and Rotated BRIEF)**,**Gabor wavelets**,**MTCNN**,**VGG19**,**Facenet and Arcface**,**SSD (Single-Shot MultiBox Detector)**,**ResNet-34**,**YOLO (You Only Look Once).**The researchers developed a system using **YOLOv7** for face detection and recognition, combined with **CLAHE (Contrast Limited Adaptive Histogram Equalisation)** for image enhancement.A dataset of **2170 images from 31 students** at Mustansiriyah University was collected for training and testing.The system achieved an accuracy rate of **100%**.

The article provides a detailed explanation of the system's methodology:

- **Image Enhancement**: CLAHE is used to adjust the illumination of training images, improving the robustness of the face recognition algorithm under varying lighting conditions.
- **Face Detection**: The system uses YOLOv7 to identify and enclose human faces in bounding boxes within images or videos.
- **Feature Extraction**: Features from the detected faces are extracted and stored in a database for comparison with features from test images. The study utilised LBPH, Dlib’s face encoding, and the YOLOv7 algorithm.
- **Dataset Preparation**: Student information and images are collected and stored in a MySQL database.
- **Marking Attendance**: During attendance registration, the system captures a snapshot from a live video feed, processes it in real-time, and identifies

the students present. The system retrieves student information from the database and records it in a CSV file.

The researchers compared their YOLOv7-based system with two other methods: **HOG + Dlib**: Achieved an accuracy of 94%, **Viola-Jones + LBPH**: Achieved an accuracy of 91%.The results demonstrated that the **YOLOv7 method outperformed the other two methods** in terms of accuracy.

2.7.2 Merits:

- **Effective Use of State-of-the-art Algorithm**: The paper leverages the YOLOv7 algorithm, a leading object detection model known for its speed and accuracy. The choice of this advanced algorithm contributes significantly to the system's high performance.
- **Image Enhancement Technique:** The use of CLAHE for image enhancement addresses the challenge of varying lighting conditions in real-world settings. This preprocessing step helps improve the robustness of the facial recognition algorithm by enhancing image contrast and detail.
- **Combination of Algorithms for Feature Extraction:** The study employs a combination of feature extraction algorithms, including LBPH, Dlib's face encoding, and YOLOv7 itself7. This multi-algorithm approach likely contributes to the system's ability to capture diverse facial features and achieve high accuracy.

2.7.3 Demerits:

- **Limited Dataset Diversity**: While the dataset consists of 2170 images, it's collected from only 31 students. A larger dataset with greater diversity in terms of age, ethnicity, and facial features would enhance the system's generalizability and reduce potential biases.
- **Lack of Robustness to Occlusions:** The paper acknowledges the potential impact of accessories and other occlusions on face detection accuracy. Further research is needed to address this limitation and develop techniques that can reliably handle partially obscured faces.

- **Computational Cost of Training:** The training process for YOLOv7, a deep neural network, can be computationally expensive and time-consuming. While the training is a one-time process, it requires significant computational resources, which may limit the scalability of the system, particularly when dealing with larger datasets.

2.8 Facial Recognition-Based Attendance System[8]

2.8.1 Description:

This paper presents a facial recognition-based attendance system designed to automate and improve the efficiency of traditional attendance-taking methods in classrooms. The authors highlight the shortcomings of existing methods, such as manual roll calls and biometric systems, which are time-consuming, prone to errors, and vulnerable to manipulation.

The proposed system uses a combination of technologies, including:

- **Haar Cascade algorithm for face detection:** This algorithm is known for its efficiency in detecting faces within images.
- **OpenCV for image processing:** OpenCV is a widely used open-source library for computer vision tasks.
- **Dlib for facial landmark detection and feature extraction:** Dlib is a C++ toolkit that provides tools for building machine learning and computer vision applications.
- **Pandas for data manipulation and analysis:** Pandas is a Python library that simplifies data handling and analysis.
- **MySQL for database management:** MySQL is a popular open-source relational database management system.

The system captures video footage of the classroom, automatically detects and recognizes students' faces, and records their attendance in an Excel spreadsheet.To enhance accuracy, the system is tested under various conditions, including changes in lighting, head movements, and camera-to-student distance.

The paper also explores different facial recognition algorithms, including:

- **Eigenfaces:** This algorithm uses statistical methods to extract features from images and represent faces as a combination of principal components.
- **Linear Discriminant Analysis (LDA) (Fisherfaces):** This algorithm, similar to Eigenfaces, uses linear discriminant analysis to find the most discriminative features for distinguishing between faces of different individuals.
- **Local Binary Pattern Histograms (LBPH):** This algorithm describes the local texture patterns in facial images, which can be used for recognition.

2.8.2 Merits:

- **Combines Multiple Established Technologies:** The system effectively integrates several proven technologies:Haar Cascade algorithm for robust and efficient face detection.OpenCV for versatile image and video processing, including grayscale conversion, face detection, and image annotation.Dlib for advanced facial landmark detection and feature extraction, enhancing recognition accuracy.Pandas for efficient data manipulation and analysis, streamlining the handling of attendance data.MySQL for reliable storage and management of student information and attendance records.
- **Algorithm Diversity and Exploration:** The paper explores a range of facial recognition algorithms, including Eigenfaces, Linear Discriminant Analysis (LDA or Fisherfaces), and Local Binary Pattern Histograms (LBPH)11. This demonstrates a comprehensive understanding of different approaches to facial recognition.
- **Real-Time Processing Capability:** The system captures video in real-time, automatically detecting and recognizing faces as they appear in the classroom. This real-time functionality is crucial for practical implementation and eliminates the need for batch processing.

2.8.3 Demerits:

Limited Information on Dataset and Training:

- **Dataset size and diversity**: While mentioning student registration and image storage in a dataset, specifics about the dataset used for training the algorithms are lacking. A larger and more diverse dataset is crucial for generalizability and mitigating biases.
- **Training process and parameter tuning**: Information on the training methodology, hyperparameter optimization, and validation techniques is absent. Transparency in these areas is essential for reproducibility and assessing the system's performance.

Potential for False Positives/Negatives: While the excerpt mentions testing under different conditions, it doesn't provide specific performance metrics like accuracy, precision, recall, or F1-score. Quantifying these metrics is essential for evaluating the system's effectiveness and understanding the likelihood of misclassifications.

2.9 Facial Recognition Based Attendance System Using OpenCV[9]

2.9.1 Description:

This paper presents a system for automating student attendance using facial recognition technology. The system aims to replace inefficient manual attendance methods like roll calls, which are time-consuming and prone to errors. The authors highlight the shortcomings of these manual methods, emphasizing the need for a more efficient and reliable solution.

The proposed system leverages computer vision techniques and algorithms to automate the attendance process. Here's a breakdown of its key components and functionalities:

- **Face Detection:** The system employs the Haar Cascade algorithm to accurately detect student faces within images captured by a camera in the classroom.
- **Feature Extraction and Recognition:** The paper explores several established facial recognition algorithms:

- **Eigenfaces:** Uses statistical analysis to represent faces as combinations of principal components, capturing essential facial features.
- **Fisherfaces (Linear Discriminant Analysis - LDA):** Similar to Eigenfaces, but focuses on finding the most discriminative features to distinguish between individuals.
- **Local Binary Pattern Histograms (LBPH):** Analyses local texture patterns in facial images for effective recognition.
- **Attendance Marking and Reporting:** Once a student's face is detected and recognised, the system automatically marks their attendance and generates reports, typically in an Excel spreadsheet.

Modules and Libraries Used:

- **OpenCV:** A powerful open-source library for computer vision tasks, used for image processing, face detection, and integration with other algorithms.
- **Dlib:** A C++ toolkit providing tools for machine learning and computer vision, including facial landmark detection and feature extraction.
- **Tkinter:** A standard Python interface for creating graphical user interfaces, allowing for user interaction with the system.
- **PIL (Python Imaging Library) and Pillow:** Libraries for image processing and manipulation in Python.
- **Time and Datetime Modules:** Python modules for handling time and date operations, essential for recording accurate timestamps for attendance.
- **NumPy:** A fundamental library for scientific computing in Python, offering powerful array operations and mathematical functions.
- **Pandas:** A Python library for data analysis and manipulation, used for handling attendance data efficiently.
- **MySQL:** A popular open-source relational database management system used for storing student information and attendance records.

2.9.2 Merits:

- **Employs Efficient and Robust Algorithms:** The system utilises the Haar Cascade algorithm, a proven and efficient method for face detection. The sources explain that this algorithm, trained on a vast dataset of images with and without faces, can effectively identify faces in real-time video footage.

This real-time processing capability is crucial for a practical and seamless attendance system.

- **Explores Multiple Facial Recognition Algorithms:** The paper doesn't limit itself to a single facial recognition approach. It explores various algorithms, including Eigenfaces, Fisherfaces (LDA), and Local Binary Pattern Histograms (LBPH)178. This exploration demonstrates the authors' awareness of different techniques and their potential benefits and limitations in various contexts.

2.9.3 Demerits:

- **Limited Information on Dataset and Training**: A crucial aspect of any machine learning system is the dataset used for training. However, the sources lack details about the dataset employed in this system6716. Information about the size, diversity, and collection process of the dataset is essential for assessing the system's robustness, potential biases, and generalizability. Similarly, the sources provide limited information about the training methodology and hyperparameter tuning.
- **Privacy Concerns:** Collecting and storing facial data raises significant privacy concerns18. The sources don't address these concerns or discuss data security measures, consent protocols, or data retention policies.

CHAPTER 3

PROJECT DESIGN

3.1 PROJECT DESIGN

3.1.1 OVER ALL – ARCHITECTURE

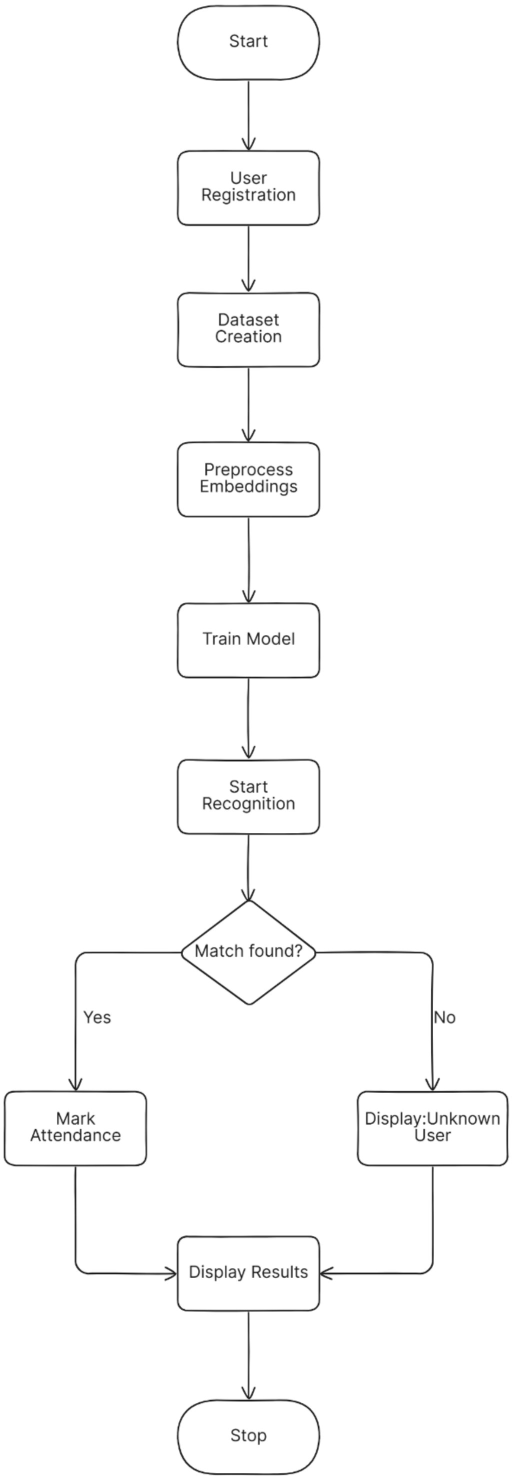


Figure 3.1 Overall Architecture

3.1.2 DATASET CREATION

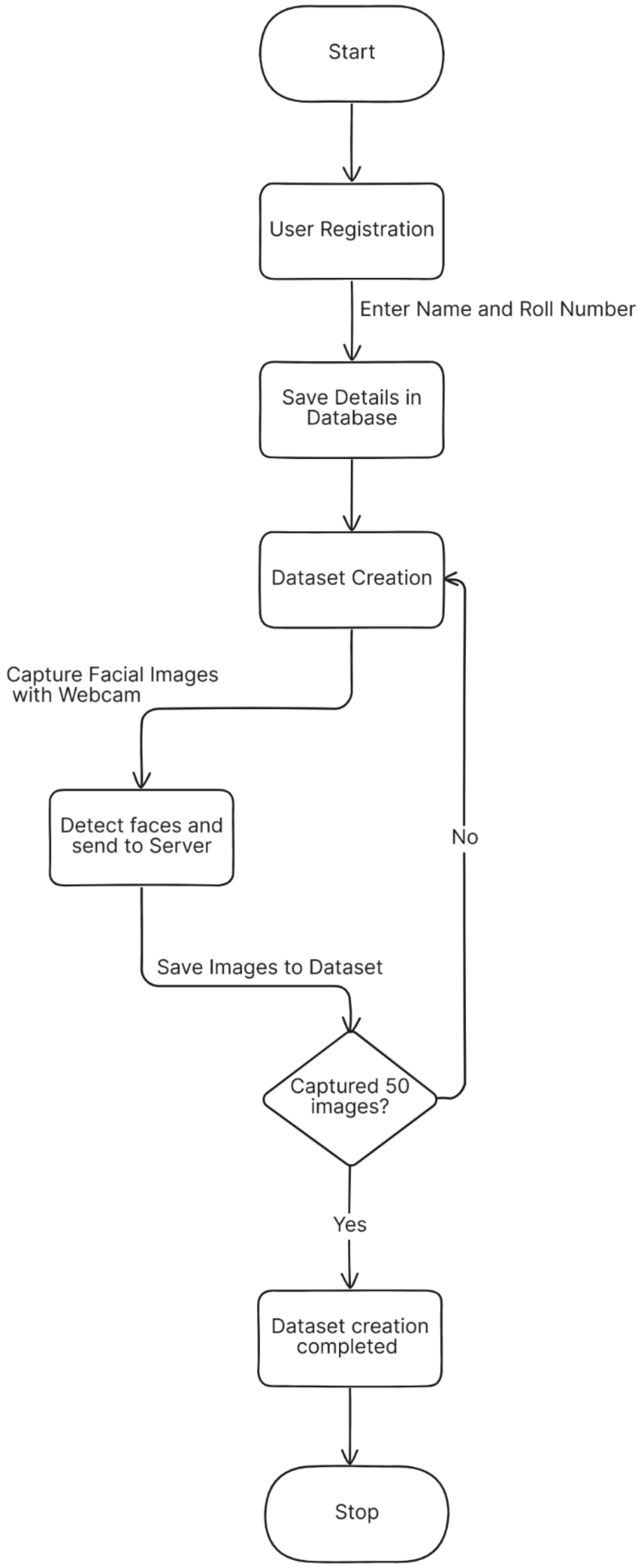


Figure 3.2 Dataset Creation

3.1.3 PREPROCESS IMAGE TO EMBEDDINGS

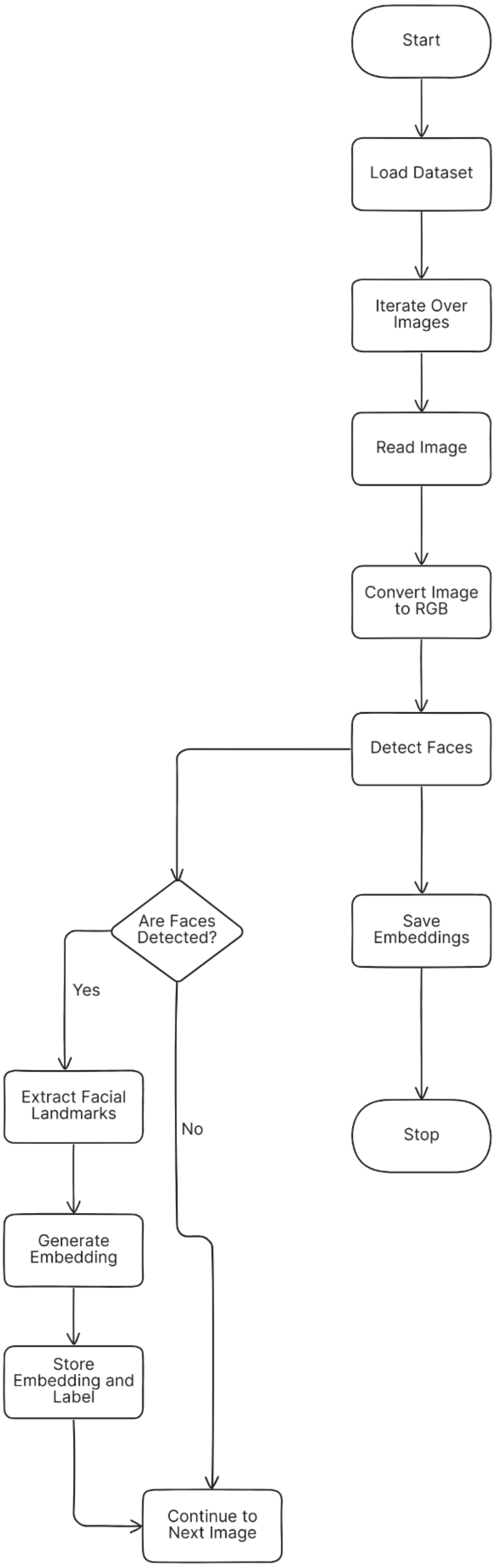


Figure 3.3 Preprocess image to embedding

3.1.4 SVM ARCHITECTURE

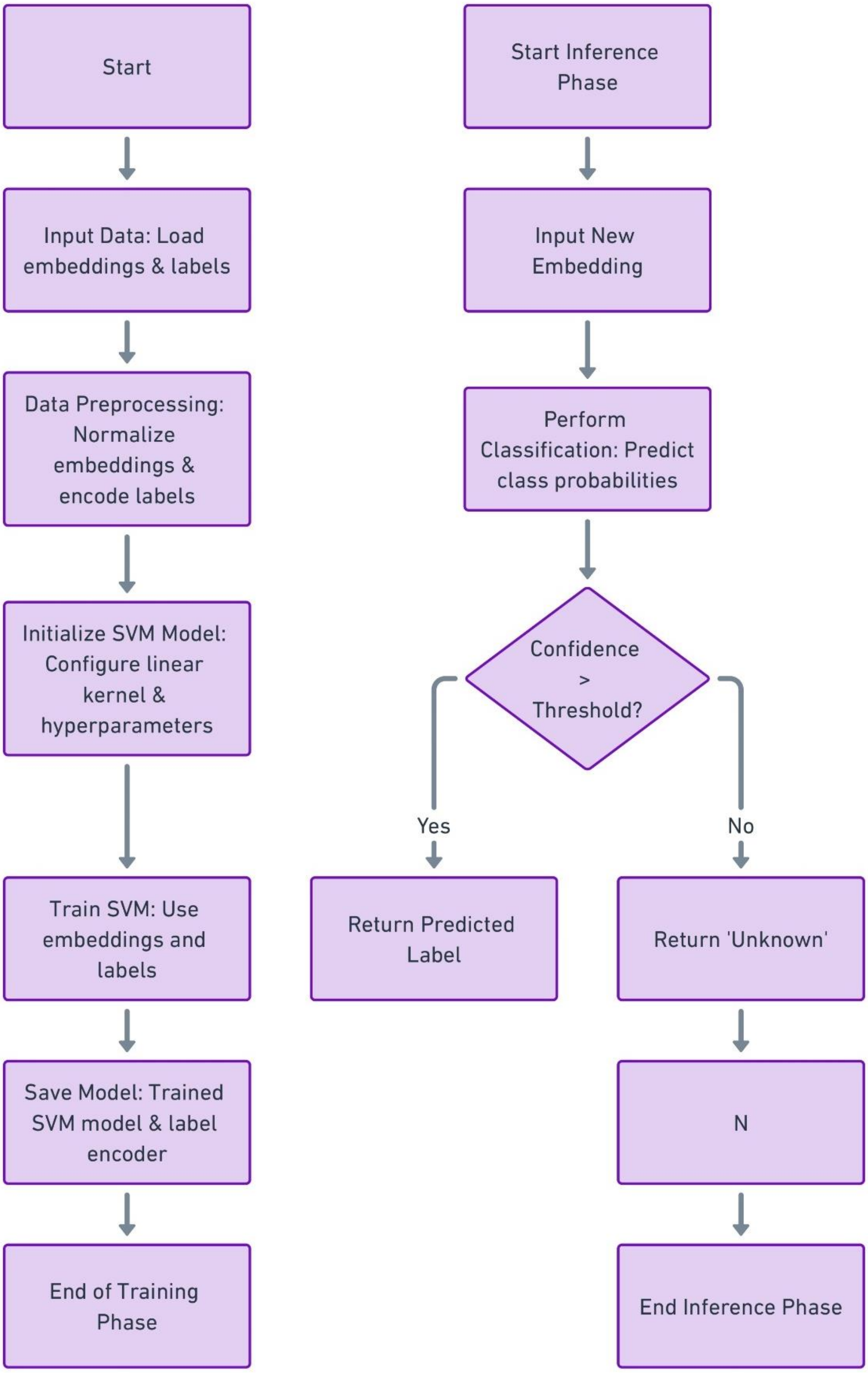


Figure 3.4 SVM Architecture

3.1.5 FACE RECOGNITION AND ATTENDANCE LOGGING

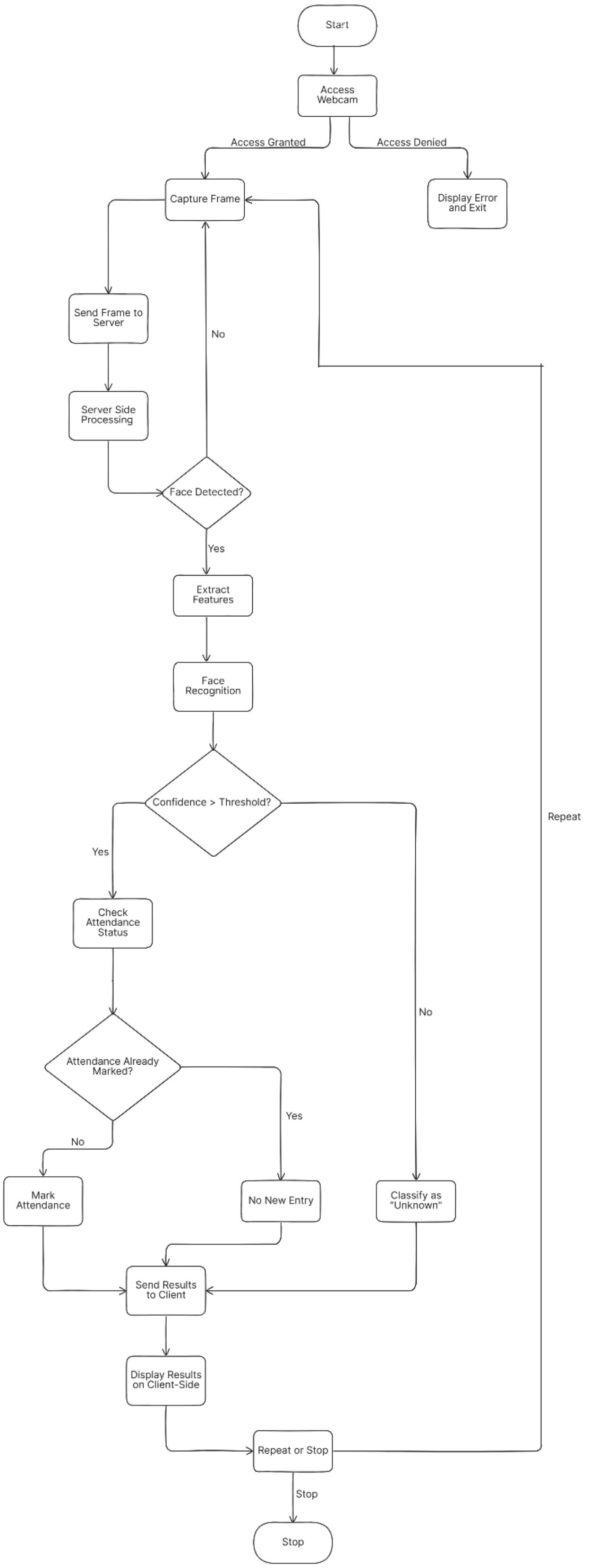


Figure 3.5 Face recognition and Attendance Logging

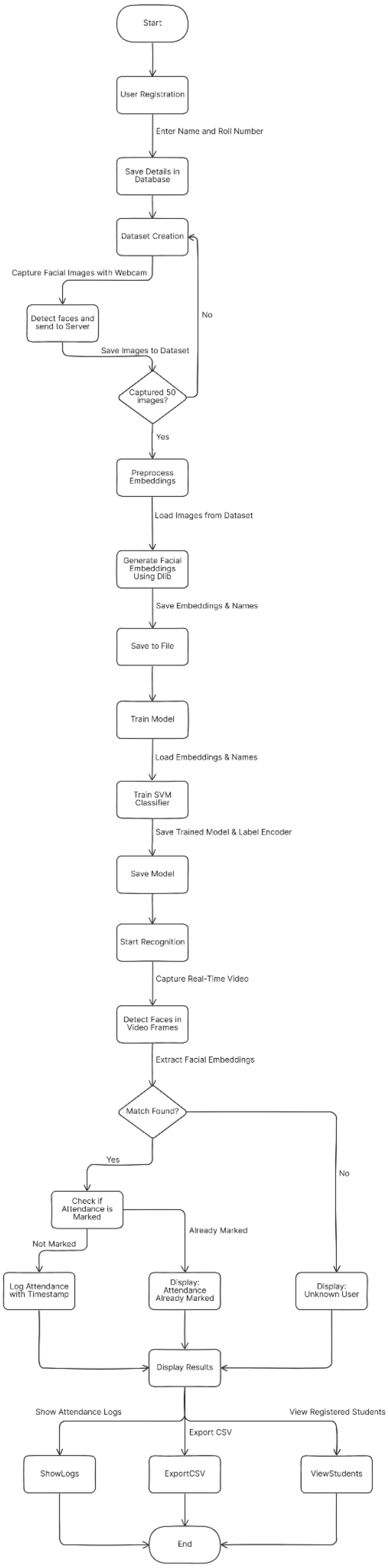


Figure 3.6 Face recognition Attendance System

3.2 MODULE DESCRIPTION

3.2.1 USER INTERFACE MODULE

- **Purpose:** The starting point of the application, acting as a navigation hub for all the functionalities.
- **Description:**
  - The home page provides an intuitive and user-friendly interface, allowing users to navigate through various features such as dataset creation, preprocessing embeddings, training the model, and recognizing faces.
  - Each functionality is displayed as a card with an icon and description for easy identification.
  - Designed using Bootstrap for responsiveness and visual appeal.
  - Accessible actions include:
    - **Create Dataset**: For registering new students.
    - **Preprocess Embeddings**: For preparing the dataset for training.
    - **Train Model**: For training the system to recognize faces.
    - **Start Recognition**: For real-time face recognition.
    - **View Attendance**: To check attendance records.
    - **View Registered Students**: To see all registered students.
    - **Export Attendance**: To download attendance as a CSV file.

3.2.2 CREATE DATASET MODULE

- **Purpose**: Enables registration of new students by capturing their facial images.
- **Description**:
  - This module provides a form where the user inputs their name and roll number.
  - Upon submission, the frontend captures multiple images of the user via the webcam to create a dataset.
  - The images are stored in a designated server directory, categorized by the user's unique roll number.

- **Key Features**:
  - Ensures that all required fields (name and roll number) are filled before submission.
  - Captures images in a systematic format for easy labelling.
  - Uses webcam integration for real-time image capture.
- **Technologies Used**:
  - HTML, Bootstrap and JS for the frontend and Tensorflow.js for face detection.
  - Flask or similar backend frameworks for processing data and OpenCV for processing images.

3.2.3 PREPROCESS EMBEDDINGS MODULE

- **Purpose**: Processes the collected facial images to generate embeddings, a prerequisite for model training.
- **Description**:
  - This module prepares the dataset by using a pre-trained deep learning model (**Dlib**) to extract numerical embeddings from the captured images.
  - These embeddings represent the unique features of each individual's face and are saved for model training.
- **Key Features**:
  - Automates the conversion of raw images to embeddings.
  - Ensures embeddings are labeled correctly for training.
- **Backend Functionality**:
  - Reads images from the dataset directory.
  - Extracts embeddings using a face detection and feature extraction library.
  - Saves embeddings in a structured format (pickle file).

3.2.4 TRAIN MODEL MODULE

- **Purpose**: Trains a machine learning classifier using the preprocessed embeddings.

- **Description**:
  - Uses the embeddings generated in the preprocessing step to train a classification model **Support Vector Machine (SVM)**.
  - The trained model is capable of distinguishing between individuals in the dataset.
- **Key Features**:
  - Automates the training process.
  - Saves the trained model for real-time recognition.
- **Backend Functionality**:
  - Loads embeddings and their labels.
  - Splits the data into training and testing sets for evaluation.
  - Trains the classifier and evaluates its performance (e.g., accuracy or precision).

3.2.5 FACE RECOGNITION MODULE

- **Purpose**: Identifies registered users in real-time using a live webcam feed and records their attendance.
- **Description**:
  - Displays a live webcam feed on the page.
  - Detects faces in the video stream and matches them with the registered users.
  - Recognized faces are marked as "present," and their attendance is recorded with a timestamp.
- **Key Features**:
  - Real-time recognition with visual feedback.
  - Automatic recording of attendance upon successful recognition.
- **Frontend Functionality**:
  - Detect faces in the webcam using blazeface model.
  - Captures images and send to backend for processing.

- **Backend Functionality**:
  - Uses the trained model to predict the identity of detected faces.
  - Records attendance in the database or a file.

3.2.6 ATTENDANCE RECORDS MODULE

- **Purpose**: Provides a tabular view of attendance records for all users.
- **Description**:
  - Displays the list of all attendance entries with the user's name and timestamp.
  - Features a scrollable, responsive table for easy viewing of large datasets.
  - Allows users to navigate back to the home page for further actions.
- **Backend Functionality**:
  - Queries the attendance database or file for records.
  - Renders the data dynamically using Flask and Jinja2 templates.

3.2.7 REGISTERED STUDENTS MODULE

- **Purpose**: Displays a list of all students registered in the system.
- **Description**:
  - Provides a table with student details, including serial number, name, roll number, and registration date.
  - Ensures users can view all individuals whose datasets have been created.
  - Features a "Back to Home" button for navigation.
- **Backend Functionality**:
  - Fetches and displays student data from the database or a directory structure.

3.2.8 EXPORT ATTENDANCE MODULE

- **Purpose**: Allows exporting of attendance data as a CSV file for reporting or integration.
- **Description**:
  - Features a "Download CSV" button to export attendance data.
  - Generates a structured CSV file with columns for Name, Roll Number, and Timestamp.
- **Backend Functionality**:
  - Reads attendance records from the database.
  - Formats the data into CSV format using libraries like **Pandas** or **CSV**.
  - Provides the file for download.

CHAPTER 4

SYSTEM SPECIFICATION, IMPLEMENTATION AND RESULTS

4.1 SYSTEM SPECIFICATION

4.1.1 SOFTWARE REQUIREMENTS

Operating System	:	Windows 11.
Coding Language	:	Python, HTML, CSS,JS, MySQL.
Frameworks	:	Flask.

4.1.2 HARDWARE REQUIREMENTS

CPU	:	12th Gen Intel(R) Core(TM) i5-12450H 2.00 GHz
RAM	:	8.00 GB
SYSTEM TYPE	:	64-bit operating system, x64-based processor

4.2 SOFTWARE DESCRIPTION:

4.2.1 ABOUT PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.

4.2.2 ABOUT PYTHON FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

- Lightweight and minimalistic
- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- Extensible and modular

4.2.3 ABOUT HTML

HTML (HyperText Markup Language) is the standard markup language for creating and structuring web content in a browser. It uses tags to define elements like text, images, links, and forms, providing semantic meaning to content. HTML works with CSS for styling and JavaScript for interactivity, enabling the creation of multimedia-rich, structured web pages. Browsers interpret HTML tags to render web content but do not display the tags themselves.

4.2.4 ABOUT CSS

CSS (Cascading Style Sheets) is a style sheet language used to control the presentation and layout of web documents written in HTML or XML. It separates content from design, allowing for flexible styling of elements like colors, fonts, and layouts. CSS enables shared styling across multiple web pages via external .css files, reducing complexity and improving load speed. It is a core technology of the web, alongside HTML and JavaScript.

4.2.5 ABOUT JS

JavaScript is a versatile, lightweight programming language primarily used to create interactive and dynamic content on websites. It allows developers to implement features like animations, form validations, and dynamic updates without reloading the page. JavaScript runs in the browser, supports object-oriented, functional, and event-driven programming, and is a core technology of web development alongside HTML and CSS. Additionally, it can be used on the server-side with platforms like Node.js.

4.2.6 ABOUT MYSQL

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for managing and organizing data. It supports features like data storage, retrieval, and security, making it ideal for web applications and enterprise-level software. MySQL is cross-platform, highly scalable, and integrates seamlessly with programming languages like PHP, Python, and Java. It allows efficient data handling through indexing, queries, and transactions. Known for its reliability and performance, MySQL is widely used in applications like e-commerce, social media, and content management systems.

4.2.7 ABOUT TENSORFLOW.JS

TensorFlow.js is an open-source library developed by Google that enables machine learning (ML) directly in the browser or on Node.js. It is a JavaScript implementation of the TensorFlow library, designed to make machine learning accessible to web developers without requiring expertise in Python or backend technologies. In TensorFlow.js, BlazeFace is available as a pre-trained model, allowing developers to easily integrate face detection into web applications.

4.2.8 ABOUT Dlib

Dlib is an open-source machine learning library written in C++ with Python bindings, primarily focused on computer vision and image processing tasks. It provides robust tools for facial landmark detection, face recognition, object detection, and image manipulation. Dlib includes pre-trained models for facial recognition, along with support for training custom models using various machine learning algorithms. Known for its high performance, Dlib is widely used in applications involving real-time face tracking, object detection, and even pose estimation.

dlib_face_recognition_resnet_model_v1.dat is a pre-trained model in dlib used for face recognition, utilizing a deep ResNet architecture for accurate facial feature extraction and comparison.

shape_predictor_68_face_landmarks.dat is another pre-trained model in dlib that detects 68 facial landmarks, providing essential points for facial alignment and feature tracking. Both models are commonly used in facial recognition systems for identifying and tracking faces in images and video. They offer high accuracy and efficiency for real-time applications.

4.3 IMPLEMENTATION

4.3.1 LIBRARIES USED

1. Flask:

A lightweight web framework used to build web applications in Python. Provides tools for routing, templating, and handling HTTP requests and responses.

2. request (from Flask):

Used to handle incoming HTTP requests, including form data, JSON payloads, and query parameters.

3. render_template (from Flask):

Renders HTML templates with dynamic data using Jinja2 templating engine.

4. redirect and url_for (from Flask):

Redirects users to a different route or URL, url_for dynamically generates URLs for the specified function.

5. flash (from Flask):

Displays one-time messages to users (e.g., success or error notifications).

6. Response (from Flask):

Customizes HTTP responses sent to the client.

7. jsonify (from Flask):

Converts Python dictionaries to JSON responses.

8. send_file (from Flask):

Sends files (like images, CSVs, etc.) to the client for download or display.

9. os:

Interacts with the operating system for file and directory manipulation.

10. cv2 (OpenCV):

An open-source library for computer vision tasks like image processing, video capture, and face detection.

11. numpy:

A library for numerical computations and handling large, multi-dimensional arrays and matrices.

12. base64:

Encodes and decodes binary data to Base64 format, often used for transferring image data in web applications.

13. datetime:

Handles date and time operations, such as timestamps for attendance records.

14. LabelEncoder (from scikit-learn):

Converts categorical labels (e.g., names) into numerical values for machine learning models.

15. SVC (Support Vector Classifier, from scikit-learn):

A supervised machine learning model used for classification tasks, such as recognizing faces.

16. pickle:

Serializes and deserializes Python objects, like saving and loading trained models.

17. paths (from imutils):

Simplifies file path handling, particularly for traversing directories of image files.

18. dlib:

A machine learning library used for tasks like face detection, feature extraction, and object recognition.

19. logging:

Used to log messages for debugging, tracking errors, or general information during the execution of the application.

20. mysql.connector:

A Python driver for connecting and interacting with MySQL databases.

21. csv:

Handles operations related to CSV (Comma-Separated Values) files, such as reading and writing attendance or dataset files.

4.3.2 FRONDEND

4.3.2.1 USER INTERFACE MODULE

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
System</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.5.2/css/bootstrap.min.css">

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs
/font-awesome/5.15.1/css/all.min.css">

    <style>

        .card {

            transition: transform 0.3s ease-
in-out, box-shadow 0.3s ease-in-out; }

        .card:hover {

            transform: scale(1.05);

            box-shadow: 0px 4px 15px rgba(0,
0, 0, 0.2);

        }

        .navbar-brand {

            font-size: 1.5rem;

            font-weight: bold;

        }

    </style>

</head>

<body>

    <!-- Navigation Bar -->

    <nav class="navbar navbar-expand-lg
navbar-dark bg-dark">

        <a class="navbar-brand"
href="/">Face Recognition System</a>

    </nav>

    <!-- Main Container -->

    <div class="container mt-5">

        <h1 class="text-center mb-4">Face
Recognition Attendance System</h1>

        <div class="row">

            <!-- Create Dataset -->

            <div class="col-md-4 mb-3">

                <div class="card shadow">

                    <div class="card-body
text-center">

                        <i class="fas fa-
camera fa-3x mb-3 text-primary"></i>

                        <h5 class="card-
title">Create Dataset</h5>

                        <a
href="/create_dataset" class="btn btn-
primary btn-block">Start</a>

                    </div>

                </div>

            </div>

            <!-- Preprocess Embeddings -->

            <div class="col-md-4 mb-3">

                <div class="card shadow">

                    <div class="card-body
text-center">

                        <i class="fas fa-
cogs fa-3x mb-3 text-success"></i>

                        <h5 class="card-
title">Preprocess Embeddings</h5>

                        <form method="POST"
action="/preprocess_embeddings">

                            <button
type="submit" class="btn btn-success btn-
block">Start</button>

                        </form>

                    </div>
```

```
                </div>

            </div>

            <!-- Train Model -->

            <div class="col-md-4 mb-3">

                <div class="card shadow">

                    <div class="card-body text-center">

                        <i class="fas fa-brain fa-3x mb-3 text-warning"></i>

                        <h5 class="card-title">Train Model</h5>

                        <form method="POST" action="/train_model">

                            <button type="submit" class="btn btn-warning btn-block">Start</button>

                        </form>

                    </div>

                </div>

            </div>

            <!-- Recognize Faces -->

            <div class="col-md-6 mb-3">

                <div class="card shadow">

                    <div class="card-body text-center">

                        <i class="fas fa-play-circle fa-3x mb-3 text-danger"></i>

                        <h5 class="card-title">Start Recognition</h5>

                        <a href="/recognize" class="btn btn-danger btn-block">Start</a>

                    </div>

                </div>

            </div>

            <!-- View Attendance -->

            <div class="col-md-6 mb-3">

                <div class="card shadow">

                    <div class="card-body text-center">

                        <i class="fas fa-clipboard-list fa-3x mb-3 text-info"></i>

                        <h5 class="card-title">View Attendance</h5>

                        <a href="/view_attendance" class="btn btn-info btn-block">View</a>

                    </div>

                </div>

            </div>

            <!-- View Registered Students -->

            <div class="col-md-6 mb-3">

                <div class="card shadow">

                    <div class="card-body text-center">

                        <i class="fas fa-user-graduate fa-3x mb-3 text-primary"></i>

                        <h5 class="card-title">View Registered Students</h5>

                        <a href="/view_students" class="btn btn-primary btn-block">View</a>

                    </div>

                </div>

            </div>

            <!-- Export Attendance -->

            <div class="col-md-6 mb-3">

                <div class="card shadow">

                    <div class="card-body text-center">

                        <i class="fas fa-file-csv fa-3x mb-3 text-dark"></i>

                        <h5 class="card-title">Export Attendance</h5>

                        <a href="/export_attendance" class="btn btn-dark btn-block">Download CSV</a>

                    </div>

                </div>

            </div>

        </div>
```

```
        <!-- Flash Messages -->

        {% with messages =
get_flashed_messages(with_categories=true)
%}

        {% if messages %}

        <div class="mt-4">

            {% for category, message in
messages %}

            <div class="alert alert-{{
category }} alert-dismissible fade show"
role="alert">

                {{ message }}

                <button type="button"
class="close" data-dismiss="alert" aria-
label="Close">

                    <span aria-
hidden="true">&times;</span>

                </button>

            </div>

            {% endfor %}

        </div>

        {% endif %}

        {% endwith %}

    </div>

    <!-- Footer -->

    <footer class="footer mt-5 bg-dark text-
white text-center py-3">

        <p>&copy; 2024 Face Recognition
System. All rights reserved.</p>

    </footer>

    <!-- Scripts -->

    <script
src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>

    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/
core@2.5.3/dist/umd/popper.min.js"></script>

    <script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>
```

create_dataset.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Create Dataset with
TensorFlow.js</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.5.2/css/bootstrap.min.css">

    <script
src="https://cdn.jsdelivr.net/npm/@tensorflo
w/tfjs"></script>

    <script
src="https://cdn.jsdelivr.net/npm/@tensorflo
w-models/blazeface"></script>

    <style>

        #camera {

            position: relative;

        }

        #videoElement, #canvasOverlay {

            position: absolute;

            top: 100px;

            left: 0;  }

    </style>

</head>

<body>

    <div class="container mt-5">

        <div class="row justify-content-
center">

            <div class="col-md-6">

                <div class="card shadow">

                    <div class="card-header
bg-primary text-white text-center">

                        <h2>Create
Dataset</h2>

                    </div>
```

```
<div class="card-body">

<form method="POST" action="/upload_dataset" id="datasetForm">

<div class="form-group">

<label for="name">Name</label>

                                <input type="text" class="form-control" id="name" name="name" required placeholder="Enter your name">

                            </div>

                            <div class="form-group">

                                <label for="roll_number">Roll Number</label>

                                <input type="text" class="form-control" id="roll_number" name="roll_number" required placeholder="Enter your roll number">

                            </div>

                            <button type="button" class="btn btn-primary btn-block" onclick="startCapturing()">Start Capturing</button>

                        </form>

                        <div id="camera">

                            <video id="videoElement" width="100%" height="auto" autoplay></video>

                            <canvas id="canvasOverlay"></canvas>

                        </div>

                        <div id="status"></div>

                    </div>

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

                        <a href="{{ url_for('index') }}" class="btn btn-secondary">Back to Home</a>

                    </div></div></div></div></div>

    <script>

        let video = document.getElementById("videoElement");

        let canvasOverlay = document.getElementById("canvasOverlay");

        let contextOverlay = canvasOverlay.getContext("2d");

        let capturedImages = [];

        let totalCaptures = 0;

        let blazefaceModel;

        // Load the BlazeFace model

        async function loadBlazeFaceModel() {

            blazefaceModel = await blazeface.load();

            console.log("BlazeFace model loaded");

        }

        // Start the camera

        async function startCamera() {

            try {

                const stream = await navigator.mediaDevices.getUserMedia({ video: true });

                video.srcObject = stream;

                video.onloadedmetadata = () => {

                    video.play();

                    canvasOverlay.width = video.videoWidth;

                    canvasOverlay.height = video.videoHeight;

                };

            } catch (err) {

                console.error('Error accessing camera:', err);

                alert("Camera access failed: " + err.message);

            }

        }

        // Initialize the camera and load the model

        loadBlazeFaceModel();

        startCamera();

        // Start capturing face images

        function startCapturing() {
```

```
            let name =
document.getElementById('name').value;

            let roll_number =
document.getElementById('roll_number').value
;

            if (!name || !roll_number) {

                alert("Please enter your
name and roll number.");

                return;

            }

            capturedImages = [];

            totalCaptures = 0;

            captureImages(name,
roll_number);

        }

        // Capture images

        async function captureImages(name,
roll_number) {

            if (totalCaptures < 50) {

                contextOverlay.clearRect(0,
0, canvasOverlay.width,
canvasOverlay.height);

                contextOverlay.drawImage(vid
eo, 0, 0, canvasOverlay.width,
canvasOverlay.height);

                // Detect faces using
BlazeFace

                const predictions = await
blazefaceModel.estimateFaces(video, false);

                for (let prediction of
predictions) {

                    const start =
prediction.topLeft;

                    const end =
prediction.bottomRight;

                    const size = [end[0] -
start[0], end[1] - start[1]];



                    contextOverlay.strokeSty
le = "green";

                    contextOverlay.lineWidth
= 2;

                    contextOverlay.strokeRec
t(start[0], start[1], size[0], size[1]);

                    const faceCanvas =
document.createElement("canvas");

                    faceCanvas.width =
size[0];

                    faceCanvas.height =
size[1];

                    const faceContext =
faceCanvas.getContext("2d");

                    faceContext.drawImage(vi
deo, start[0], start[1], size[0], size[1],
0, 0, faceCanvas.width, faceCanvas.height);

                    const dataURL =
faceCanvas.toDataURL("image/png");

                    capturedImages.push(data
URL);

                    totalCaptures++;

                    document.getElementById(
"status").innerHTML = `Captured:
${totalCaptures}/50`;

                }

                setTimeout(() =>
captureImages(name, roll_number), 200);

            } else {

                uploadImages(name,
roll_number);

            }

        }

        // Upload captured images

        async function uploadImages(name,
roll_number) {

            const formData = new FormData();

            formData.append("name", name);

            formData.append("roll_number",
roll_number);

            capturedImages.forEach((image,
index) => {

                formData.append(`image_${ind
ex}`, image);

            });

            try {

                const response = await
fetch('/upload_dataset', { method: 'POST',
body: formData });

                if (response.ok) {

                    alert("Dataset upload
successful!");
```

```
                    document.getElementById(
"status").innerHTML = "Dataset uploaded
successfully!";

                } else {

                    alert("Error uploading
dataset!");

                }

            } catch (err) {

                console.error("Error
uploading images:", err);

                alert("Error uploading
dataset!");

            }

        }

    </script>

</body>

</html>
```

recognize.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</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.5.2/css/bootstrap.min.css">

    <style>

        .card {

            max-width: 700px;

            margin: auto;

            margin-top: 50px;

        }

        .webcam-stream {

            width: 100%;

            border-radius: 8px;

        }

        .results {

            text-align: center;

            margin-top: 20px;

        }

        #loading {

            display: none;

            text-align: center;

            margin-top: 10px;

        }

        #video, #canvas {

            display: none;

        }

        .result-item {

            margin-bottom: 15px;

            font-size: 1.2em;

        }

        .face-box {

            position: absolute;

            border: 2px solid red;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="card shadow">

            <div class="card-header bg-
primary text-white text-center">

                <h2>Face Recognition</h2>

            </div>

            <div class="card-body text-
center">

                <div class="mt-3 mb-4">

                    <!-- Webcam Stream -->
```

```
                    <video id="video"
autoplay></video>

                    <canvas
id="canvas"></canvas>

                    <img id="output"
class="webcam-stream" alt="Live Webcam
Stream">

                </div>

                <!-- Flash Messages -->

                {% with messages =
get_flashed_messages(with_categories=true)
%}

                    {% if messages %}

                        <div class="mt-2">

                            {% for category,
message in messages %}

                                <div
class="alert alert-{{ category }} alert-
dismissible fade show" role="alert">

                                    {{
message }}

                                    <button
type="button" class="close" data-
dismiss="alert" aria-label="Close">

                                        <spa
n aria-hidden="true">&times;</span>

                                    </button
>

                                </div>

                            {% endfor %}

                        </div>

                    {% endif %}

                {% endwith %}

                <!-- Recognition Results -->

                <div id="results"
class="results"></div>

                <!-- Loading Spinner -->

                <div id="loading">

                    <div class="spinner-
border text-primary" role="status">

                        <span class="sr-
only">Processing...</span>

                    </div>

                </div>

                <a href="{{ url_for('index')
}}" class="btn btn-secondary mt-3">Back to
Home</a>

            </div>

        </div>

    </div>

    <script>

        const video =
document.getElementById("video");

        const canvas =
document.getElementById("canvas");

        const output =
document.getElementById("output");

        const resultsDiv =
document.getElementById("results");

        const loadingDiv =
document.getElementById("loading");

        let processing = false; // To
prevent multiple requests while processing

        // Start webcam stream

        navigator.mediaDevices.getUserMedia(
{ video: true })

            .then(stream => {

                video.srcObject = stream;

                video.style.display =
"block";

            })

            .catch(err => {

                console.error("Error
accessing webcam:", err);

                alert("Unable to access the
webcam. Please make sure your device has a
working camera.");

            });

        // Capture frame, send to server,
and display results

        async function
fetchRecognitionResults() {

            if (processing) return; // Skip
if already processing

            try {

                processing = true;

                loadingDiv.style.display =
"block"; // Show loading spinner
```

```
                // Draw the video frame onto a canvas
                const context = canvas.getContext("2d");
                canvas.width = video.videoWidth;
                canvas.height = video.videoHeight;
                context.drawImage(video, 0, 0, canvas.width, canvas.height);
                // Convert canvas content to a Base64 image
                const frame = canvas.toDataURL("image/jpeg");
                output.src = frame; // Optional: Display captured frame
                // Send the frame to the server
                const response = await fetch("/recognize", {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ frame: frame })
                });
                const data = await response.json();
                displayResults(data.results);
            } catch (error) {
                console.error("Error fetching recognition results:", error);
                resultsDiv.innerHTML = `<p class="text-danger">Error fetching recognition results. Please try again.</p>`;
            } finally {
                processing = false;
                loadingDiv.style.display = "none"; // Hide loading spinner
            }
        }
        // Display recognition results
        function displayResults(results) {
            resultsDiv.innerHTML = ""; // Clear previous results
            if (results.length === 0) {
                resultsDiv.innerHTML = `<p>No faces detected.</p>`;
                return;
            }
            results.forEach(result => {
                const { name, confidence, status, box } = result;
                const resultText = `
                    <div class="result-item">
                        <strong>${name}</strong> (${confidence}%) - ${status}
                        <div class="face-box" style="top: ${box[1]}px; left: ${box[0]}px; width: ${box[2] - box[0]}px; height: ${box[3] - box[1]}px;"></div>
                    </div>`;
                resultsDiv.innerHTML += resultText;
            });
        }
        // Poll for recognition results every 2 seconds
        setInterval(fetchRecognitionResults, 2000);
    </script>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body></html>
```

view_attendance.html:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Attendance Records</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .card {
            margin: auto;
            max-width: 800px;
        }
        .table-container {
            max-height: 400px;
            overflow-y: auto;
        }
    </style>
</head>
<body>
    <div class="container mt-5">
        <div class="card shadow">
            <div class="card-header bg-primary text-white text-center">
                <h3>Attendance Records</h3>
            </div>
            <div class="card-body">
                <div class="table-container">
                    <table class="table table-striped table-hover">
                        <thead class="thead-dark">
                            <tr>
                                <th>Name</th>
                                <th>Timestamp</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for record in records %}
                                <tr>
                                    <td>{{ record.name }}</td>
                                    <td>{{ record.timestamp }}</td>
                                </tr>
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
                <div class="text-center mt-3">
                    <a href="{{ url_for('index') }}" class="btn btn-secondary">Back to Home</a>
                </div>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

view_students.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

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

    <title>Registered Students</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

    <div class="container mt-5">

        <div class="card shadow">

            <div class="card-header bg-
primary text-white">

                <h3>Registered Students</h3>

            </div>

            <div class="card-body">

                {% if students %}

                    <table class="table
table-striped table-hover">

                        <thead class="thead-
dark">

                            <tr>

                                <th
scope="col">ID</th>

                                <th
scope="col">Name</th>

                                <th
scope="col">Roll Number</th>

                            </tr>

                        </thead>

                        <tbody>

                            {% for student
in students %}

                            <tr>

                                <td>{{
student[0] }}</td>

                                <td>{{
student[1] }}</td>

                                <td>{{
student[2] }}</td>

                            </tr>

                        {% endfor %}

                    </tbody>

                </table>

            {% else %}

                <p class="text-
center">No students registered yet.</p>

            {% endif %}

        </div>

        <div class="card-footer text-
center">

            <a href="{{ url_for('index')
}}" class="btn btn-secondary">Back to
Home</a>

        </div>

    </div>

</div>

<script
src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/
core@2.5.3/dist/umd/popper.min.js"></script>

<script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>
```

4.3.3 BACKEND:APP.PY

4.3.3.1 IMPORTING LIBRARIES AND DB CONFIGURATION

```
from flask import Flask, request, render_template, redirect, url_for, flash, Response, jsonify, send_file

import os

import cv2

import numpy as np

import base64

from datetime import datetime

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

import pickle

from imutils import paths

import dlib

import logging

import mysql.connector

import csv

# Configuration

app = Flask(__name__)

# MySQL Configuration

db_config = {

    'host': 'localhost',

    'user': 'root',

    'password': '1234',

    'database': 'attendance_system_new'

}

# Database connection

def get_db_connection():

    return mysql.connector.connect(**db_config)

@app.route('/')

def index():

    return render_template('index.html')
```

4.3.3.2 LOAD THE MODELS

```
DATASET_DIR = 'dataset'

MODEL_PATH = 'model'

OUTPUT_PATH = 'output'

CASCADE_PATH = 'haarcascade_frontalface_default.xml'

EMBEDDING_FILE = os.path.join(OUTPUT_PATH, "embeddings.pickle")

RECOGNIZER_FILE = os.path.join(OUTPUT_PATH, "recognizer.pickle")

LABEL_ENCODER_FILE = os.path.join(OUTPUT_PATH, "le.pickle")

SHAPE_PREDICTOR = os.path.join(MODEL_PATH, "shape_predictor_68_face_landmarks.dat")

FACE_RECOGNITION_MODEL = os.path.join(MODEL_PATH, "dlib_face_recognition_resnet_model_v1.dat")

# Load Dlib models

detector = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor(SHAPE_PREDICTOR)

embedder = dlib.face_recognition_model_v1(FACE_RECOGNITION_MODEL)
```

4.3.3.3 CREATE DATASET MODULE

```
@app.route('/create_dataset')

def create_dataset():

    return
render_template('create_dataset.html')

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

def upload_dataset():

    try:

        # Validate form data

        name = request.form.get('name')

        roll_number =
request.form.get('roll_number')

        if not name or not roll_number:

            return jsonify({"error": "Name
and Roll Number are required!"}), 400

        # Connect to the database

        connection = get_db_connection()

        cursor = connection.cursor()

        # Check if the student already
exists in the database

        cursor.execute("SELECT * FROM
students WHERE roll_number = %s",
(roll_number,))

        existing_student = cursor.fetchone()

        if existing_student:

            return jsonify({"error":
"Student with this roll number already
exists!"}), 400

        # Insert student details into the
database

        cursor.execute("INSERT INTO students
(name, roll_number) VALUES (%s, %s)", (name,
roll_number))

        connection.commit()

        # Create user-specific directory

        user_dir = os.path.join(DATASET_DIR,
name)

        os.makedirs(user_dir, exist_ok=True)

        # Process each image from the
request

        for key in request.form:

            if key.startswith('image_'):

                img_data =
request.form[key].split(",")[1]

                img_bytes =
base64.b64decode(img_data)

                img_path =
os.path.join(user_dir, f"{key}.png")

                with open(img_path, "wb") as
img_file:

                    img_file.write(img_bytes
)

        return jsonify({"message": "Dataset
uploaded and student registered
successfully!"}), 200

    except Exception as e:

        logging.exception("Error during
dataset upload.")

        return jsonify({"error": str(e)}),
500

    finally:

        if connection:

            cursor.close()

            connection.close()
```

4.3.3.4 PREPROCESS EMBEDDINGS MODULE

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

def preprocess_embeddings():

    try:

        image_paths = list(paths.list_images(DATASET_DIR))

        known_embeddings = []

        known_names = []

        for (i, image_path) in enumerate(image_paths):

            logging.info(f"Processing image {i + 1}/{len(image_paths)}: {image_path}")

            name = os.path.basename(os.path.dirname(image_path))

            image = cv2.imread(image_path)

            rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

            boxes = detector(rgb_image, 1)

            for box in boxes:

                shape = predictor(rgb_image, box)

                face_descriptor = embedder.compute_face_descriptor(rgb_image, shape)

                known_names.append(name)

                known_embeddings.append(np.array(face_descriptor))

        data = {"embeddings": known_embeddings, "names": known_names}

        with open(EMBEDDING_FILE, "wb") as f:

            pickle.dump(data, f)

        flash("Embeddings preprocessed successfully!", "success")

        return redirect(url_for('index'))

    except Exception as e:

        logging.exception("Error during preprocessing embeddings.")

        flash(f"Error: {str(e)}", "danger")

        return redirect(url_for('index'))
```

4.3.3.5 TRAIN MODEL MODULE

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

def train_model():

    try:

        if not os.path.exists(EMBEDDING_FILE):

            flash("Embeddings file not found. Please preprocess embeddings first.", "danger")

            return redirect(url_for('index'))

        with open(EMBEDDING_FILE, "rb") as f:

            data = pickle.load(f)

        known_names = data["names"]

        if len(set(known_names)) < 2:

            flash("Error: At least two unique individuals required for training.", "danger")

            return redirect(url_for('index'))

        label_encoder = LabelEncoder()

        labels = label_encoder.fit_transform(known_names)

        recognizer = SVC(C=1.0, kernel="linear", probability=True)
```

```
        recognizer.fit(data["embeddings"], labels)

        with open(RECOGNIZER_FILE, "wb") as f:

            pickle.dump(recognizer, f)

        with open(LABEL_ENCODER_FILE, "wb") as f:

            pickle.dump(label_encoder, f)

        flash("Model trained successfully!", "success")

        return redirect(url_for('index'))

    except Exception as e:

        logging.exception("Error during model training.")

        flash(f"Error: {str(e)}", "danger")

        return redirect(url_for('index'))
```

4.3.3.6 FACE RECOGNITION MODULE

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

def recognize():

    if request.method == 'GET':

        return render_template('recognize.html')  # For GET requests, render the page.

    elif request.method == 'POST':

        try:

            # Decode the incoming base64 frame

            frame_data = request.json.get('frame')

            if not frame_data:

                return jsonify({"error": "No frame data received"}), 400

            # Process the frame (base64 to image)

            frame_data = frame_data.split(',')[1]  # Strip base64 prefix

            frame_bytes = base64.b64decode(frame_data)

            frame = cv2.imdecode(np.frombuffer(frame_bytes, np.uint8), cv2.IMREAD_COLOR)

            rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # Load recognizer and label encoder (ensure they exist)

            recognizer = None

            le = None

            if os.path.exists(RECOGNIZER_FILE) and os.path.exists(LABEL_ENCODER_FILE):

                with open(RECOGNIZER_FILE, "rb") as f:

                    recognizer = pickle.load(f)

                with open(LABEL_ENCODER_FILE, "rb") as f:

                    le = pickle.load(f)

            if recognizer is None or le is None:

                return jsonify({"error": "Recognizer or label encoder not found. Please train the model first."}), 400

            # Detect faces in the frame

            faces = detector(rgb_frame)

            results = []

            for face in faces:
```

```
                shape = predictor(rgb_frame,
face)

                face_embedding =
embedder.compute_face_descriptor(rgb_frame,
shape)

                # Perform recognition

                predictions =
recognizer.predict_proba([np.array(face_embe
dding)])[0]

                max_index =
np.argmax(predictions)

                confidence =
predictions[max_index]

                name =
le.classes_[max_index] if confidence >
CONFIDENCE_THRESHOLD else "Unknown"

                if name != "Unknown":

                    status =
log_attendance(name)

                    if status == "marked":

                        display_text =
f"{name} ({confidence * 100:.2f}%):
Attendance Marked"

                    else:

                        display_text =
f"{name} ({confidence * 100:.2f}%): Already
Marked"

                else:

                    display_text = "Unknown"

                results.append({

                    "name": name,

                    "confidence":
round(confidence * 100, 2),

                    "status": display_text,

                    "box": [face.left(),
face.top(), face.right(), face.bottom()]

                })

            return jsonify({"results":
results})

        except Exception as e:

            return jsonify({"error": f"Error
processing frame: {str(e)}"}), 500

def log_attendance(name):

    # Log attendance in database

    conn = get_db_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT * FROM attendance
WHERE name=%s AND DATE(timestamp) =
CURDATE()", (name,))

    existing_record = cursor.fetchone()

    if existing_record:

        status = "already_marked"

    else:

        timestamp = datetime.now()

        cursor.execute("INSERT INTO
attendance (name, timestamp) VALUES (%s,
%s)", (name, timestamp))

        conn.commit()

        status = "marked"

    cursor.close()

    conn.close()

    return status
```

4.3.3.7 ATTENDANCE RECORDS MODULE

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

def view_attendance():

    conn = get_db_connection()

    cursor = conn.cursor(dictionary=True)
```

```
    cursor.execute("SELECT name, timestamp FROM attendance ORDER BY timestamp DESC")

    records = cursor.fetchall()

    cursor.close()

    conn.close()

    return render_template('view_attendance.html', records=records)
```

4.3.3.8 REGISTERED STUDENTS MODULE

```
# Route to display the registered students

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

def view_students():

    try:

        connection = get_db_connection()

        cursor = connection.cursor()

        # Fetch all students from the database

        cursor.execute("SELECT id, name, roll_number FROM students")

        students = cursor.fetchall()

        # Return a list of students

        return render_template('view_students.html', students=students)

    except Exception as e:

        logging.exception("Error fetching students.")

        return jsonify({"error": str(e)}), 500

    finally:

        if connection:

            cursor.close()

            connection.close()
```

4.3.3.9 EXPORT ATTENDANCE MODULE

```
@app.route('/export_attendance')

def export_attendance():

    conn = get_db_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT name, timestamp FROM attendance")

    rows = cursor.fetchall()

    csv_file = 'attendance.csv'

    with open(csv_file, mode='w', newline='') as file:

        writer = csv.writer(file)

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

        writer.writerows(rows)
```

4.4 SINGLE IMAGE EMBEDDING CONVERSION:

Take the single image and covert the image to embeddings by using Dlib Models.

For face landmark detection : shape_predictor_68_face_landmarks.dat

The shape_predictor_68_face_landmarks.dat is a file containing a pre-trained machine learning model used to detect 68 specific points (or landmarks) on the human face. These landmarks outline the face's structure and include features like the jawline, eyebrows, nose, eyes, and mouth.

Details of the 68 Landmarks

The model maps the face into specific regions:

1. **Jawline (points 1–17):** Defines the contour of the lower face.
2. **Eyebrows (points 18–27):** Captures the shape of the eyebrows.
3. **Nose (points 28–36):** Covers the bridge and tip of the nose.
4. **Eyes (points 37–48):** Includes the corners and contours of both eyes.
5. **Mouth (points 49–68):** Outlines the lips and inner mouth area.

How It Works

The model works in combination with the dlib library:

1. **Face Detection:** A face is first detected in an image using a face detector (e.g., dlib’s get_frontal_face_detector).
2. **Landmark Prediction:** Once a face is detected, the shape_predictor_68_face_landmarks.dat model is applied to identify the 68 points on the face.

For face embeddings : dlib_face_recognition_resnet_model_v1.dat

The dlib_face_recognition_resnet_model_v1.dat is another pre-trained model provided by the dlib library. It is specifically used for **face recognition**. This model employs a deep learning-based ResNet (Residual Neural Network) architecture to compute **face embeddings**, which are numerical representations of faces. These embeddings are essential for comparing and identifying faces.

How It Works

1. **Face Detection**: Before using this model, you first need to detect faces in an image (e.g., using dlib's face detector or another method).
2. **Face Landmarking**: Once a face is detected, landmarks (e.g., from shape_predictor_68_face_landmarks.dat) are used to align the face.
3. **Face Embedding Generation**: The aligned face is passed through the dlib_face_recognition_resnet_model_v1 model, which outputs a 128-dimensional vector (embedding) that uniquely represents the face.
4. **Face Comparison**: The embeddings can be compared using distance metrics (like Euclidean distance) to determine how similar two faces are. If the distance is below a certain threshold, the faces are considered a match.

Details

- **Architecture**: The model is based on a ResNet-34 architecture, which is a popular deep learning network designed for image classification and recognition tasks.
- **Output**: A 128-dimensional embedding vector.
- **Purpose**: It is specifically trained to differentiate between faces, making it highly effective for face recognition tasks.

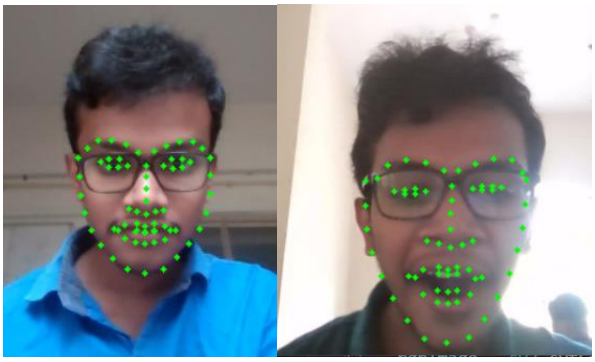
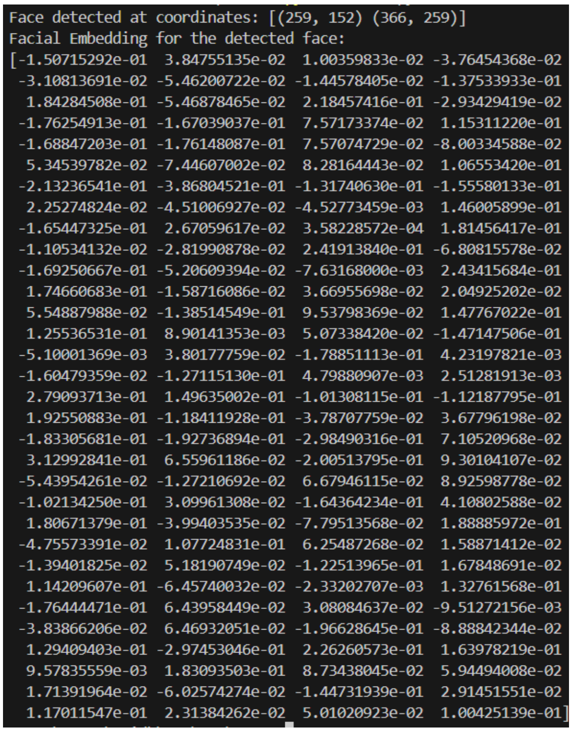
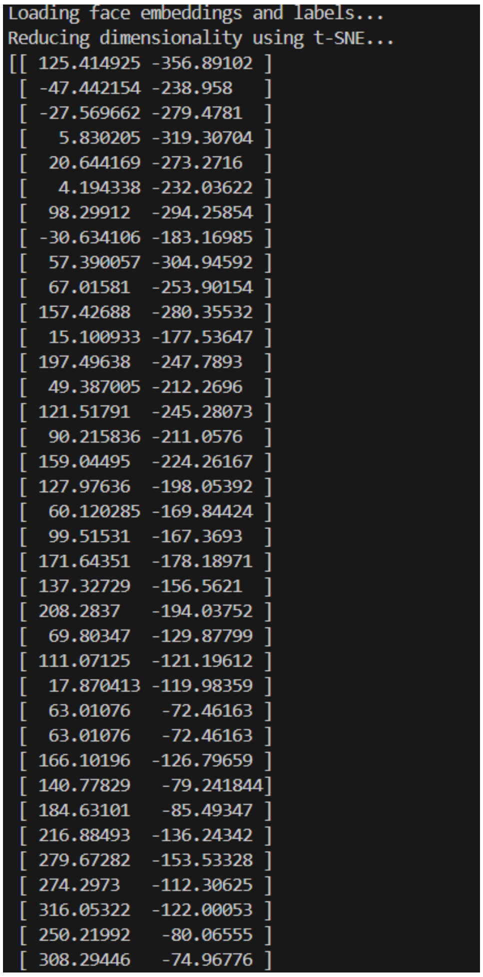


Figure 4.1 Face Landmark detection



```
Face detected at coordinates: [(259, 152) (366, 259)]
Facial Embedding for the detected face:
[-1.50715292e-01  3.84755135e-02  1.00359833e-02 -3.76454368e-02
 -3.10813691e-02 -5.46200722e-02 -1.44578405e-02 -1.37533933e-01
  1.84284508e-01 -5.46878465e-02  2.18457416e-01 -2.93429419e-02
 -1.76254913e-01 -1.67039037e-01  7.57173374e-02  1.15311220e-01
 -1.68847203e-01 -1.76148087e-01  7.57074729e-02 -8.00334588e-02
  5.34539782e-02 -7.44607002e-02  8.28164443e-02  1.06553420e-01
 -2.13236541e-01 -3.86804521e-01 -1.31740630e-01 -1.55580133e-01
  2.25274824e-02 -4.51006927e-02 -4.52773459e-03  1.46005899e-01
 -1.65447325e-01  2.67059617e-02  3.58228572e-04  1.81456417e-01
 -1.10534132e-02 -2.81990878e-02  2.41913840e-01 -6.80815578e-02
 -1.69250667e-01 -5.20609394e-02 -7.63168000e-03  2.43415684e-01
  1.74660683e-01 -1.58716086e-02  3.66955698e-02  2.04925202e-02
  5.54887988e-02 -1.38514549e-01  9.53798369e-02  1.47767022e-01
  1.25536531e-01  8.90141353e-03  5.07338420e-02 -1.47147506e-01
 -5.10001369e-03  3.80177759e-02 -1.78851113e-01  4.23197821e-03
 -1.60479359e-02 -1.27115130e-01  4.79880907e-03  2.51281913e-03
  2.79093713e-01  1.49635002e-01 -1.01308115e-01 -1.12187795e-01
  1.92550883e-01 -1.18411928e-01 -3.78707759e-02  3.67796198e-02
 -1.83305681e-01 -1.92736894e-01 -2.98490316e-01  7.10520968e-02
  3.12992841e-01  6.55961186e-02 -2.00513795e-01  9.30104107e-02
 -5.43954261e-02 -1.27210692e-02  6.67946115e-02  8.92598778e-02
 -1.02134250e-01  3.09961308e-02 -1.64364234e-01  4.10802588e-02
  1.80671379e-01 -3.99403535e-02 -7.79513568e-02  1.88885972e-01
 -4.75573391e-02  1.07724831e-01  6.25487268e-02  1.58871412e-02
 -1.39401825e-02  5.18190749e-02 -1.22513965e-01  1.67848691e-02
  1.14209607e-01 -6.45740032e-02 -2.33202707e-03  1.32761568e-01
 -1.76444471e-01  6.43958449e-02  3.08084637e-02 -9.51272156e-03
 -3.83866206e-02  6.46932051e-02 -1.96628645e-01 -8.88842344e-02
  1.29409403e-01 -2.97453046e-01  2.26260573e-01  1.63978219e-01
  9.57835559e-03  1.83093503e-01  8.73438045e-02  5.94494008e-02
  1.71391964e-02 -6.02574274e-02 -1.44731939e-01  2.91451551e-02
  1.17011547e-01  2.31384262e-02  5.01020923e-02  1.00425139e-01]
```

Figure 4.2 Embeddings for single image



```
Loading face embeddings and labels...
Reducing dimensionality using t-SNE...
[[ 125.414925 -356.89102 ]
 [ -47.442154 -238.958   ]
 [ -27.569662 -279.4781  ]
 [   5.830205 -319.30704 ]
 [  20.644169 -273.2716  ]
 [   4.194338 -232.03622 ]
 [  98.29912  -294.25854 ]
 [ -30.634106 -183.16985 ]
 [  57.390057 -304.94592 ]
 [  67.01581  -253.90154 ]
 [ 157.42688  -280.35532 ]
 [  15.100933 -177.53647 ]
 [ 197.49638  -247.7893  ]
 [  49.387005 -212.2696  ]
 [ 121.51791  -245.28073 ]
 [  90.215836 -211.0576  ]
 [ 159.04495  -224.26167 ]
 [ 127.97636  -198.05392 ]
 [  60.120285 -169.84424 ]
 [  99.51531  -167.3693  ]
 [ 171.64351  -178.18971 ]
 [ 137.32729  -156.5621  ]
 [ 208.2837   -194.03752 ]
 [  69.80347  -129.87799 ]
 [ 111.07125  -121.19612 ]
 [  17.870413 -119.98359 ]
 [  63.01076   -72.46163 ]
 [  63.01076   -72.46163 ]
 [ 166.10196  -126.79659 ]
 [ 140.77829   -79.241844]
 [ 184.63101   -85.49347 ]
 [ 216.88493  -136.24342 ]
 [ 279.67282  -153.53328 ]
 [ 274.2973   -112.30625 ]
 [ 316.05322  -122.00053 ]
 [ 250.21992   -80.06555 ]
 [ 308.29446   -74.96776 ]
```

Figure 4.3 Reduced Dimension of Embeddings

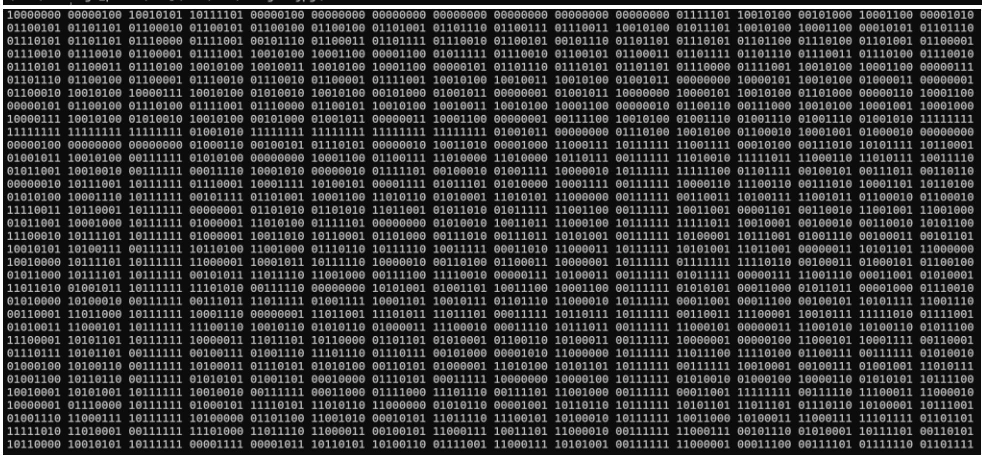
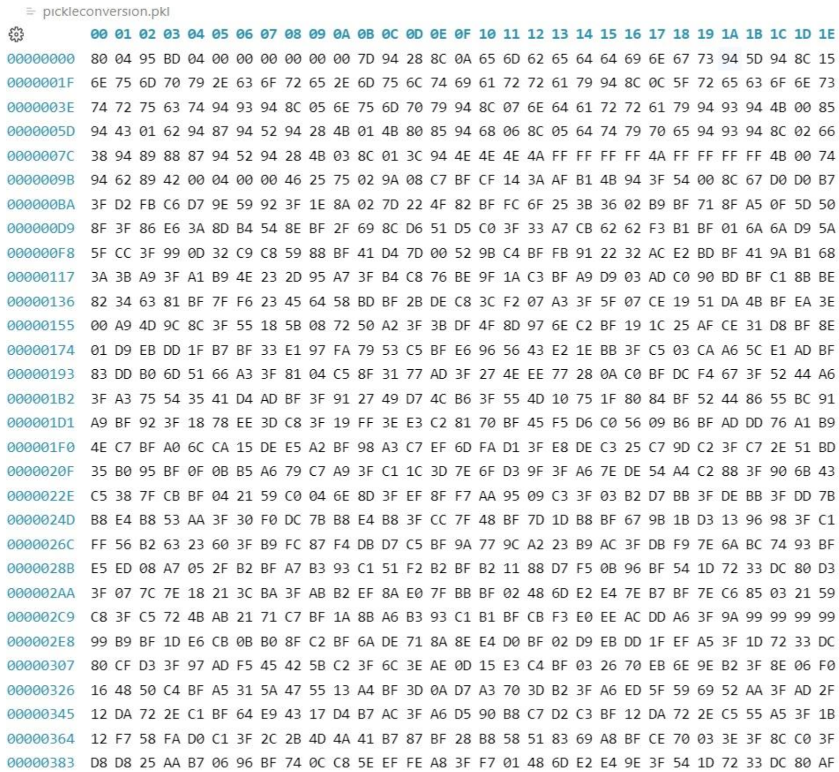


Figure 4.4 Storing of Embeddings as byte stream(pickle)




```
pickleconversion.pkl
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E
00000000  80 04 95 BD 04 00 00 00 00 00 00 7D 94 28 8C 0A 65 6D 62 65 64 64 69 6E 67 73 94 5D 94 8C 15
0000001F  6E 75 6D 70 79 2E 63 6F 72 65 2E 6D 75 6C 74 69 61 72 72 61 79 94 8C 0C 5F 72 65 63 6F 6E 73
0000003E  74 72 75 63 74 94 93 94 8C 05 6E 75 6D 70 79 94 8C 07 6E 64 61 72 72 61 79 94 93 94 4B 00 85
0000005D  94 43 01 62 94 87 94 52 94 28 4B 01 4B 80 85 94 68 06 8C 05 64 74 79 70 65 94 93 94 8C 02 66
0000007C  38 94 89 88 87 94 52 94 28 4B 03 8C 01 3C 94 4E 4E 4E 4A FF FF FF FF 4A FF FF FF FF 4B 00 74
0000009B  94 62 89 42 00 04 00 00 46 25 75 02 9A 08 C7 BF CF 14 3A AF B1 4B 94 3F 54 00 8C 67 D0 D0 B7
000000BA  3F D2 FB C6 D7 9E 59 92 3F 1E 8A 02 7D 22 4F 82 BF FC 6F 25 3B 36 02 B9 BF 71 8F A5 0F 5D 50
000000D9  8F 3F 86 E6 3A 8D B4 54 8E BF 2F 69 8C D6 51 D5 C0 3F 33 A7 CB 62 62 F3 B1 BF 01 6A 6A D9 5A
000000F8  5F CC 3F 99 0D 32 C9 C8 59 88 BF 41 D4 7D 00 52 9B C4 BF FB 91 22 32 AC E2 BD BF 41 9A B1 68
00000117  3A 3B A9 3F A1 B9 4E 23 2D 95 A7 3F B4 C8 76 BE 9F 1A C3 BF A9 D9 03 AD C0 90 BD BF C1 8B BE
00000136  82 34 63 81 BF 7F F6 23 45 64 58 BD BF 2B DE C8 3C F2 07 A3 3F 5F 07 CE 19 51 DA 4B BF EA 3E
00000155  00 A9 4D 9C 8C 3F 55 18 5B 08 72 50 A2 3F 3B DF 4F 8D 97 6E C2 BF 19 1C 25 AF CE 31 D8 BF 8E
00000174  01 D9 EB DD 1F B7 BF 33 E1 97 FA 79 53 C5 BF E6 96 56 43 E2 1E BB 3F C5 03 CA A6 5C E1 AD BF
00000193  83 DD B0 6D 51 66 A3 3F 81 04 C5 8F 31 77 AD 3F 27 4E EE 77 28 0A C0 BF DC F4 67 3F 52 44 A6
000001B2  3F A3 75 54 35 41 D4 AD BF 3F 91 27 49 D7 4C B6 3F 55 4D 10 75 1F 80 84 BF 52 44 86 55 BC 91
000001D1  A9 BF 92 3F 18 78 EE 3D C8 3F 19 FF 3E E3 C2 81 70 BF 45 F5 D6 C0 56 09 B6 BF AD DD 76 A1 B9
000001F0  4E C7 BF A0 6C CA 15 DE E5 A2 BF 98 A3 C7 EF 6D FA D1 3F E8 DE C3 25 C7 9D C2 3F C7 2E 51 BD
0000020F  35 B0 95 BF 0F 0B B5 A6 79 C7 A9 3F C1 1C 3D 7E 6F D3 9F 3F A6 7E DE 54 A4 C2 88 3F 90 6B 43
0000022E  C5 38 7F CB BF 04 21 59 C0 04 6E 8D 3F EF 8F F7 AA 95 09 C3 3F 03 B2 D7 BB 3F DE BB 3F DD 7B
0000024D  B8 E4 B8 53 AA 3F 30 F0 DC 7B B8 E4 B8 3F CC 7F 48 BF 7D 1D B8 BF 67 9B 1B D3 13 96 98 3F C1
0000026C  FF 56 B2 63 23 60 3F B9 FC 87 F4 DB D7 C5 BF 9A 77 9C A2 23 B9 AC 3F DB F9 7E 6A BC 74 93 BF
0000028B  E5 ED 08 A7 05 2F B2 BF A7 B3 93 C1 51 F2 B2 BF B2 11 88 D7 F5 0B 96 BF 54 1D 72 33 DC 80 D3
000002AA  3F 07 7C 7E 18 21 3C BA 3F AB B2 EF 8A E0 7F BB BF 02 48 6D E2 E4 7E B7 BF 7E C6 85 03 21 59
000002C9  C8 3F C5 72 4B AB 21 71 C7 BF 1A 8B A6 B3 93 C1 B1 BF CB F3 E0 EE AC DD A6 3F 9A 99 99 99 99
000002E8  99 B9 BF 1D E6 CB 0B B0 8F C2 BF 6A DE 71 8A 8E E4 D0 BF 02 D9 EB DD 1F EF A5 3F 1D 72 33 DC
00000307  80 CF D3 3F 97 AD F5 45 42 5B C2 3F 6C 3E AE 0D 15 E3 C4 BF 03 26 70 EB 6E 9E B2 3F 8E 06 F0
00000326  16 48 50 C4 BF A5 31 5A 47 55 13 A4 BF 3D 0A D7 A3 70 3D B2 3F A6 ED 5F 59 69 52 AA 3F AD 2F
00000345  12 DA 72 2E C1 BF 64 E9 43 17 D4 B7 AC 3F A6 D5 90 B8 C7 D2 C3 BF 12 DA 72 2E C5 55 A5 3F 1B
00000364  12 F7 58 FA D0 C1 3F 2C 2B 4D 4A 41 B7 87 BF 28 B8 58 51 83 69 A8 BF CE 70 03 3E 3F 8C C0 3F
00000383  D8 D8 25 AA B7 06 96 BF 74 0C C8 5E EF FE A8 3F F7 01 48 6D E2 E4 9E 3F 54 1D 72 33 DC 80 AF
```


Figure 4.5 Hexadecimal representation of pickle file

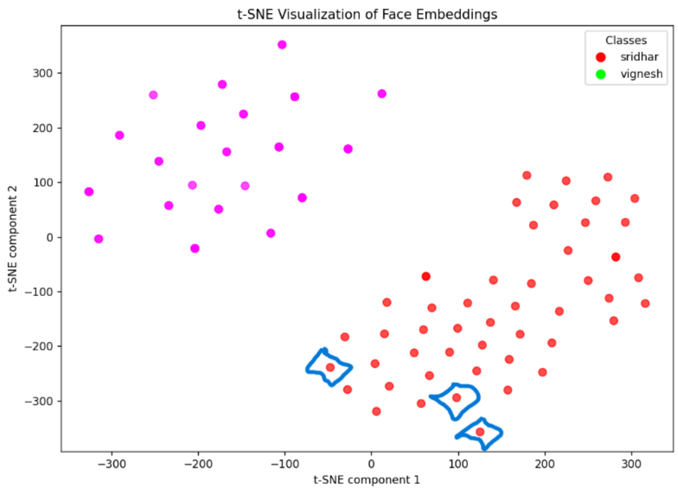


Figure 4.6 Visualization of Embeddings

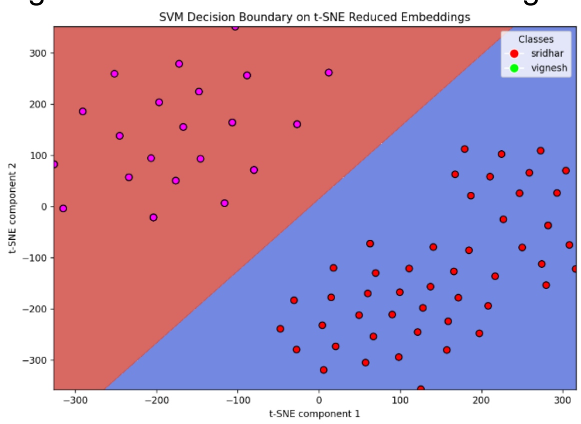


Figure 4.7 SVM Decision Boundary on Face Embedding

4.5 EVALUATION

4.5.1 Accuracy

Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of correctly classified samples to the total number of samples. Accuracy provides a basic assessment of how well the model performs overall.

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

4.5.2 Precision

Precision focuses on the accuracy of positive predictions, representing the ratio of correctly predicted positive cases to the total predicted positive cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

4.5.3 Recall (Sensitivity)

Recall measures the model's ability to correctly identify all actual positive cases. It is the ratio of correctly predicted positive cases to all actual positive cases.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4.5.4 F1-Score

The F1-score is the harmonic mean of precision and recall, providing a balanced metric when the dataset is imbalanced. A higher F1-score reflects a good balance between precision and recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.5.5 Confusion Matrix

A confusion matrix is a structured table used to evaluate the performance of a classification model. It provides a detailed comparison between the predicted outcomes and the actual (true) outcomes of the dataset, making it an essential tool for analyzing and understanding model accuracy. By summarizing the model's predictions, the confusion matrix offers insights into not only the number of correct predictions but also the types of errors the model makes.

Structure of the Confusion Matrix

- **True Positives (TP):** These are cases where the model correctly predicted the positive class.
- **True Negatives (TN):** These are cases where the model correctly predicted the negative class.
- **False Positives (FP):** (Type I Error) These are cases where the model incorrectly predicted the positive class.
- **False Negatives (FN):** (Type II Error) These are cases where the model incorrectly predicted the negative class.

4.6 RESULTS

4.6.1 USER INTERFACE MODULE

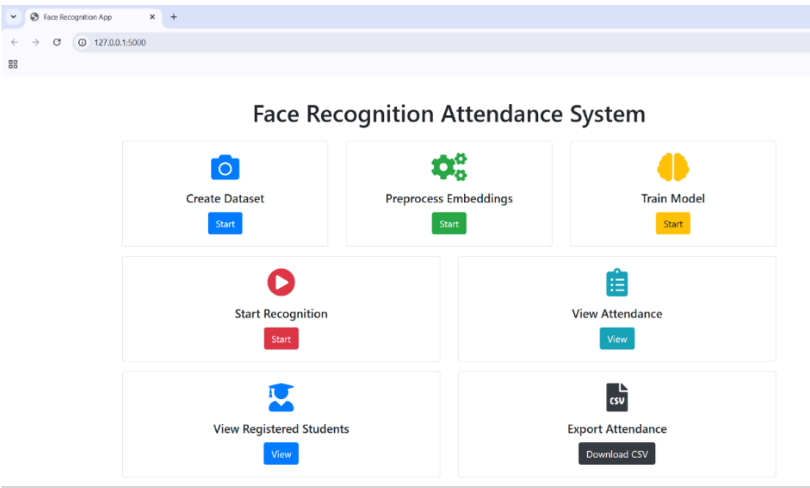


Figure 4.8 User Interface Module

4.6.2 CREATE DATASET MODULE

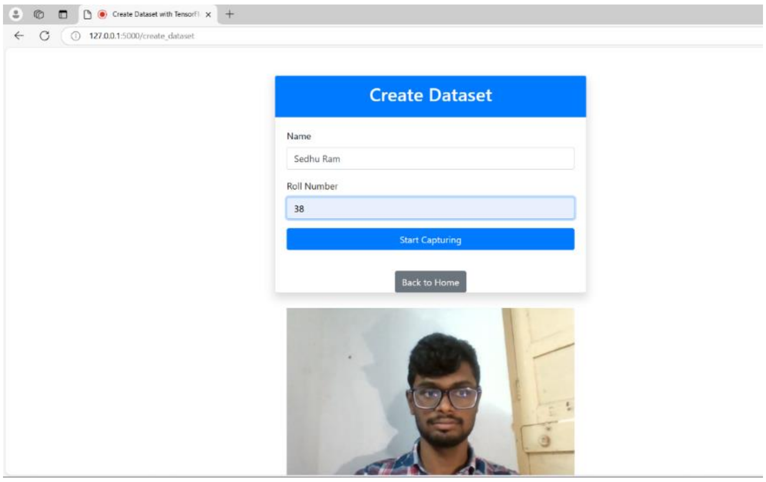


Figure 4.9 Create dataset Module

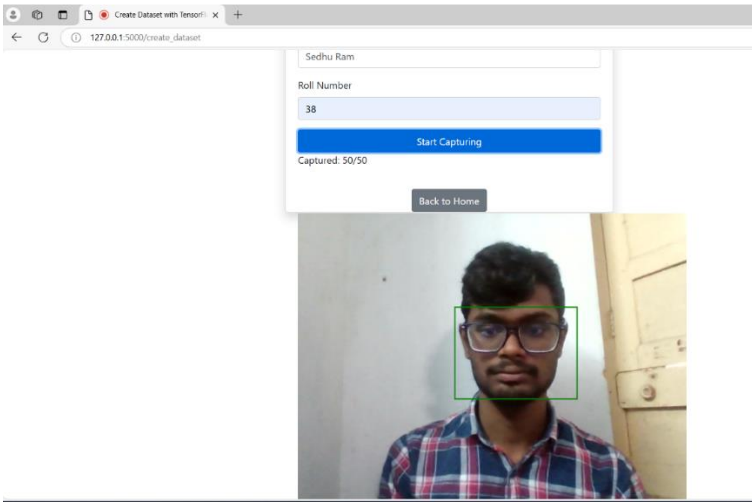


Figure 4.10 Image Capturing for dataset

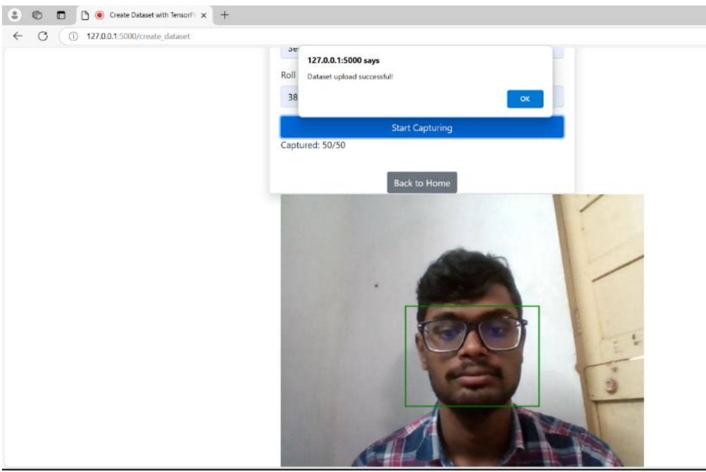


Figure 4.11 Dataset creation successful

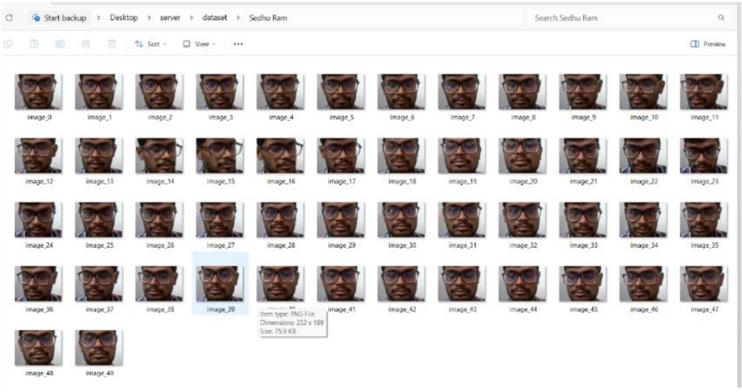


Figure 4.12 Displaying Dataset

4.6.3 PREPROCESS EMBEDDING MODULE

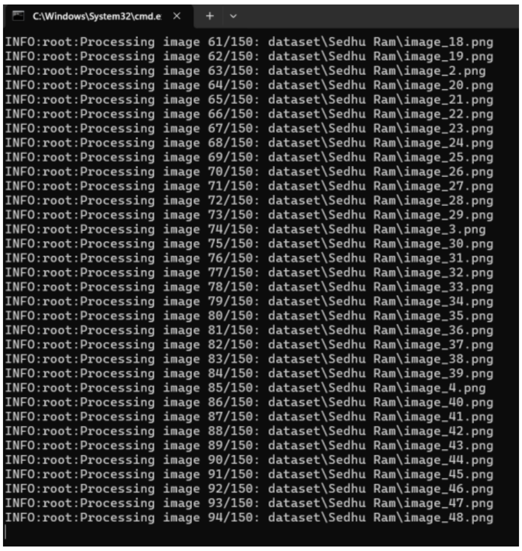


Figure 4.13 Preprocess Embedding Module

4.6.4 TRAIN MODEL MODULE

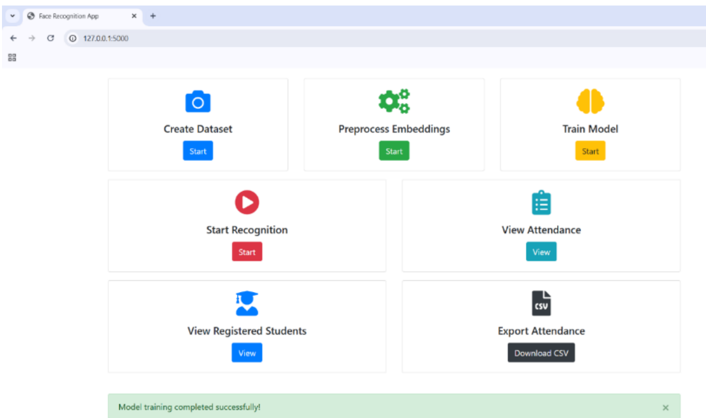


Figure 4.14 Model Training Module

4.6.5 FACE RECOGNITION MODULE

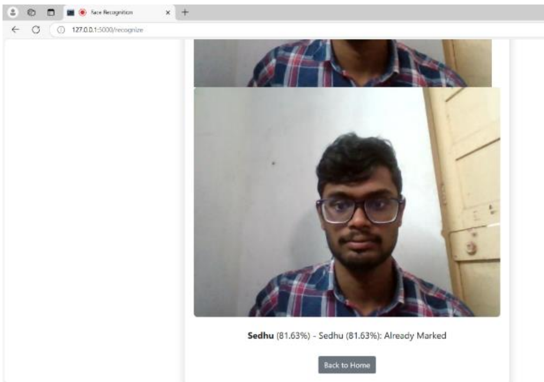
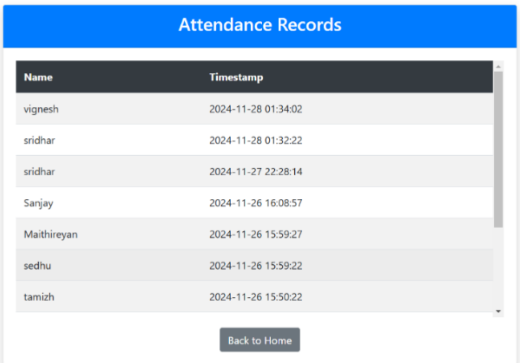


Figure 4.15 Face recognition Module

4.6.6 ATTENDANCE RECORDS MODULE



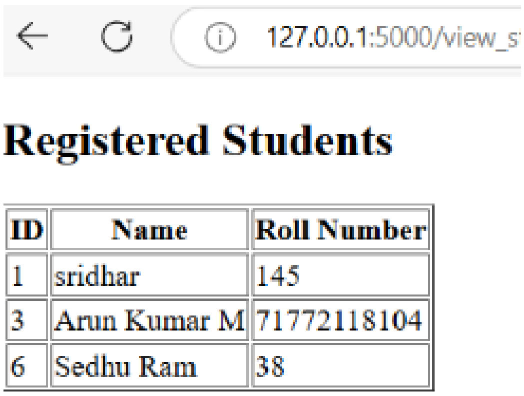
Attendance Records

Name	Timestamp
vignesh	2024-11-28 01:34:02
sridhar	2024-11-28 01:32:22
sridhar	2024-11-27 22:28:14
Sanjay	2024-11-26 16:08:57
Maithireyan	2024-11-26 15:59:27
sedhu	2024-11-26 15:59:22
tamizh	2024-11-26 15:50:22

Back to Home

Figure 4.16 Attendance Records Module

4.6.7 REGISTERED STUDENTS MODULE



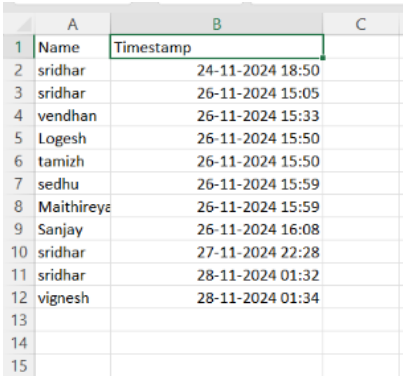
127.0.0.1:5000/view_s

Registered Students

ID	Name	Roll Number
1	sridhar	145
3	Arun Kumar M	71772118104
6	Sedhu Ram	38

Figure 4.17 Registered Students Module

4.6.8 EXPORT ATTENDANCE MODULE



	A	B	C
1	Name	Timestamp	
2	sridhar	24-11-2024 18:50	
3	sridhar	26-11-2024 15:05	
4	vendhan	26-11-2024 15:33	
5	Logesh	26-11-2024 15:50	
6	tamizh	26-11-2024 15:50	
7	sedhu	26-11-2024 15:59	
8	Maithireya	26-11-2024 15:59	
9	Sanjay	26-11-2024 16:08	
10	sridhar	27-11-2024 22:28	
11	sridhar	28-11-2024 01:32	
12	vignesh	28-11-2024 01:34	
13			
14			
15			

Figure 4.18 Export attendance Module

4.6.9 NGROK IMPLEMENTATION

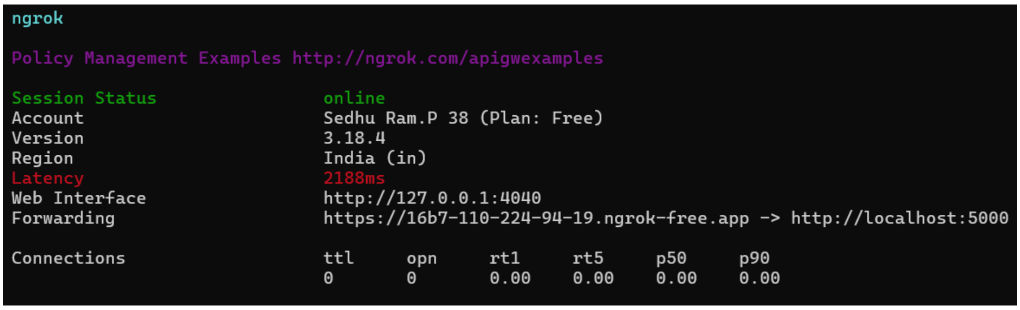


Figure 4.19 Connecting to ngrok

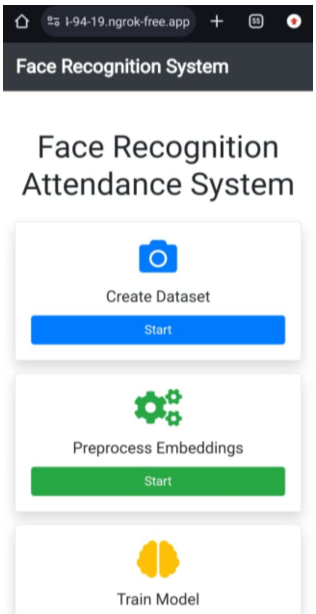


Figure 4.20 Accessing Application Mobile

Figure 4.21 Face Recognition for Attendance

4.6.10 EVALUATION METRICS OUTPUT

```
Training the SVM model...

Accuracy: 0.98

Classification Report:
                 precision    recall  f1-score   support

       Abilash V      1.00      1.00      1.00        10
     Arunkumar M      0.91      1.00      0.95        10
     Arunkumar S      0.91      1.00      0.95        10
          Guru M      1.00      1.00      1.00        10
Jebin kamalesh I      1.00      1.00      1.00        10
     Logeshwaran      1.00      1.00      1.00        10
        Rakesh R      1.00      1.00      1.00        10
    Saravanakumar     1.00      1.00      1.00        10
     Sedhu Ram P      1.00      0.90      0.95        10
         Sridhar      1.00      1.00      1.00         9
   Tamizharasu M      1.00      1.00      1.00        10
       VIGNESH s      1.00      0.89      0.94         9
   Vijaya gopal A     1.00      1.00      1.00        10

        accuracy                          0.98       128
       macro avg      0.99      0.98      0.98       128
    weighted avg      0.99      0.98      0.98       128
```

```
Class 8:
  TP: 9
  TN: 118
  FP: 0
  FN: 1
  Accuracy: 0.99
  Precision: 1.00
  Recall: 0.90
  F1-Score: 0.95

Class 9:
  TP: 9
  TN: 119
  FP: 0
  FN: 0
  Accuracy: 1.00
  Precision: 1.00
  Recall: 1.00
  F1-Score: 1.00

Class 10:
  TP: 10
  TN: 118
  FP: 0
  FN: 0
  Accuracy: 1.00
  Precision: 1.00
  Recall: 1.00
  F1-Score: 1.00

Class 11:
  TP: 8
  TN: 119
  FP: 0
  FN: 1
  Accuracy: 0.99
  Precision: 1.00
  Recall: 0.89
  F1-Score: 0.94
```

Figure 4.22 Evaluation Metrics

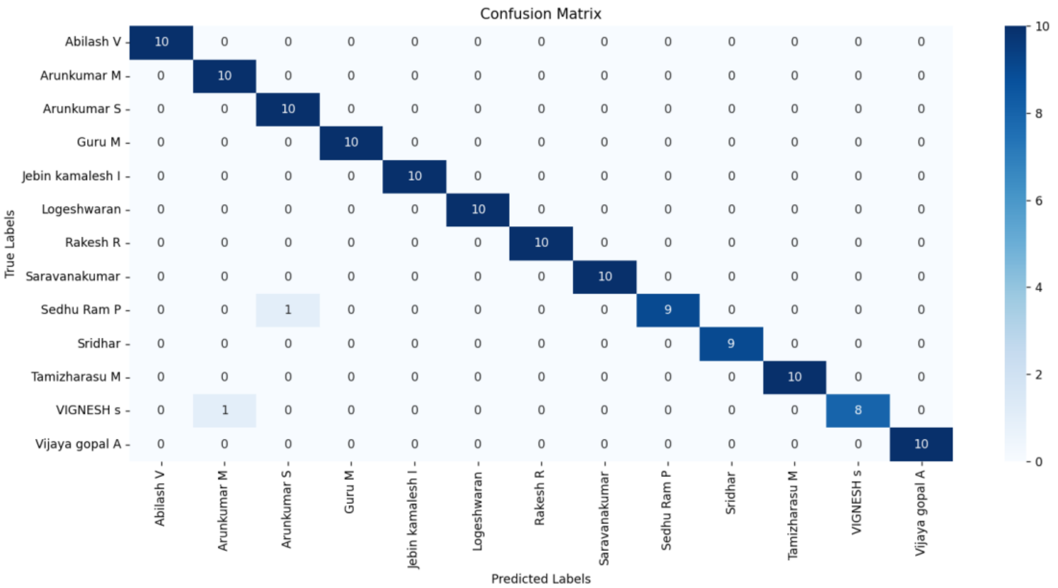


Figure 4.23 Confusion Matrix

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

The **Face Recognition Attendance System** demonstrates the seamless integration of modern technologies like computer vision and machine learning to address real-world challenges. By automating the attendance process, this system offers significant improvements in efficiency, accuracy, and security over traditional methods. The use of face recognition ensures a non-intrusive and user-friendly experience while minimizing the risk of errors or fraudulent practices. Through the integration of a robust backend, a dynamic frontend, and advanced machine learning models, the system efficiently manages datasets, processes embeddings, and performs real-time recognition. Additionally, features such as attendance tracking, student registration, and export functionalities make it a comprehensive solution for various organizations. This project not only highlights the potential of AI-driven applications but also lays the foundation for future enhancements, such as scalability, multi-camera integration, and advanced facial analytics, making it a valuable tool for the evolving needs of digital infrastructure.

5.2 FUTURE SCOPE

The **Face Recognition Attendance System** offers a promising foundation for further advancements and enhancements. Below are potential areas for future development:

1. **Scalability and Multi-Camera Integration**
   The system can be extended to support large-scale environments, such as universities or corporate offices, by incorporating multiple cameras and centralized data management. This would enable simultaneous attendance tracking across different locations.
2. **Enhanced Facial Recognition Accuracy**
   By integrating more advanced machine learning models like deep neural networks or transformers, the system can improve recognition accuracy,

particularly in challenging conditions such as poor lighting, occlusions, or variations in facial expressions.

3. **Mobile and IoT Integration**

   A mobile application can be developed to allow users to check attendance records, manage registration, or receive real-time notifications. IoT-enabled smart devices can also enhance real-time data collection and processing.

4. **Emotion and Behavior Analysis**

   Incorporating emotion detection or behavioral analysis can add additional layers of insights, such as monitoring attentiveness or identifying unusual behavior during sessions.

5. **Cloud Storage and Processing**

   Migrating to cloud-based storage and processing will allow for more efficient data management, enhanced scalability, and better accessibility for administrators and users.

6. **Biometric Fusion**

   Integrating face recognition with other biometric techniques like fingerprint or iris scanning can provide multi-modal authentication, increasing reliability and security.

7. **Real-Time Analytics and Reporting**

   Advanced analytics tools can be integrated to provide detailed reports, trends, and visualizations on attendance patterns, helping organizations make data-driven decisions.

8. **Privacy and Security Enhancements**

   The system can incorporate more robust data encryption methods and privacy-compliant frameworks to ensure the secure handling of sensitive user data.

9. **Cross-Domain Applications**

   Beyond attendance, the system can be adapted for use in other domains, such as secure access control, visitor management, or customer identification in retail and banking sectors.

These future enhancements not only promise improved system efficiency and user experience but also pave the way for expanding its applications across diverse fields, ensuring its relevance in an increasingly digital world.

CHAPTER 6

REFERENCE

[1] Muhammad Haikal Mohd Kamil, Norliza Zaini, Lucyantie Mazalan, Afiq Harith Ahamad – 2023 “Online attendance system based on facial recognition with face mask detection”, Springer, vol 82, pp. 34437–34457.

[2] Dr. Sarita Sanap,Ms. Sakshi Narwade,Mr. Sahil Goge,Mr. Krishna Pandit - 2023 “Face Recognition Based Attendance System Using Histogram of Oriented Gradients and Linear Support Vector Machine”, EJTAS, vol 1, No.1,pp. 87.

[3] Andre Budiman, Fabiana, Ricky Aryatama Yaputera, Said Achmad, Aditya Kurniawan – 2023 “Student attendance with face recognition (LBPH or CNN)”, ELSEVIER, vol 216, pp. 31-38.

[4] Aditya UmalkarShivang Singh ManhasImazChandiwalaNarendra Bhagat -2023 “Face Recognition Based Attendance System Using Real Time Data”, IJCRT, vol 11, pp. 2320-2882.

[5] Ashish Kumar Shukla, Archana Shukla, Raghvendra Singh-2023 “Automatic attendance system based on CNN–LSTM and face recognition”, Springer, vol 16, pp. 1293–1301.

[6] Olufemi S. Ojo , Mayowa O. Oyediran , Babatunde J. Bamgbade ID , Abidemi Emmanuel Adeniyi ID , Godwin Nse Ebong ID, and Sunday Adeola Ajagbe – 2023 “Development of an Improved Convolutional Neural Network for an Automated Face-Based University Attendance System”, PARADIGM Plus, vol 4, No.1, pp. 18-28.

[7] Ahmad S. Lateef , Mohammed Y. Kamil -2023 “Face Recognition-Based Automatic Attendance System in a Smart Classroom”, IJEEE, vol 20, pp. 37-47.

[8] Priyanka Manke, Mohammed Hamza Siddiqui, Himanshu Pednekar, Pawan Sakat, Qureshi Abdul Qadir-2024 “Facial Recognition-Based Attendance System”, IJISRT,vol 9, pp. 673-679.

[9] Ms. K. Vyshnavi,Suhel Shaik, Mylavarapu Santosh Kumar,M.S.V Praveen, Swetha Potti - 2024 “Facial Recognition-Based Attendance System Using Opencv”, JCSE , vol 2, pp. 1-10.

[10] Dhanush Gowda H.L, K Vishal, Keertiraj B. R, Neha Kumari Dubey , Pooja M. R. – 2020 "Face Recognition based Attendance System",IJERT, vol 9, pp. 761-767.

[11] Dr. T. Chandrasekhar Rao, N. Sathvik Sharma, M. Mahammad Shadik, P. Usha Sree, P. Sri Harsha, K. Suryavardhan – 2024 "Automated Attendance System Using Facial Recognition",IJSRST, vol 11, No.2, pp. 67-74.

[12] D. Prasad , S. Girish Chandra , Ch. Sirisha, G. Lakshmi Tanuja, M. Durga Harinadh, M. Ravi Kumar – 2023 "Face Recognition Attendance System Using HOG Algorithm",DRSR,vol 13,pp. 255-261.

[13] Mahmoud Ali, Anjali Diwan, Dinesh Kumar – 2024 "Attendance System Optimization through Deep Learning Face Recognition",IJCDS,vol 15,No.1,pp. 1527-1540.

[14] Phul Babu Jha, Arjun Basnet, Biraj Pokhrel, Bishnu Pokhrel, Gopal Kumar Thakur, Surya Chhetri – 2023 "An Automated Attendance System Using Facial Detection and Recognition Technology",Apex Journal,vol 1,pp. 103-120.

[15] Avoy Sain, Samrat Dutta, Rimpi Saha, Unmesh Mandal – 2023 "Automated Facial Recognition based Attendance System using OpenCV in Python",IJSRCSEIT,vol 9,pp. 105-111.