Video-Based Face Recognition Attendance System

A Project Report Submitted
In Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Mayuri Shakya (2200520109104)
Sachin Kumar (2200520109106)
Amrendra Bahadur Yadav (2200520109101)

Under the guidance of
Dr. Sakshi Srivastava
Er. Rakesh Kumar Gautam

Department of Computer Science and Engineering
INSTITUTE OF ENGINEERING & TECHNOLOGY
Dr. A.P.J. Abdul Kalam Technical University Uttar Pradesh

June, 2025

Declaration

We hereby declare that this submission of project is our own work and that to the best of our knowledge and belief ir contains no material previously published or written by another person or material which to a substantial extent has accepted for award of any other degree of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

This project report has not been submitted by us to any other institute for the requirement of any other degree.

We solemnly affirm that this project submission represents our original work, devoid of any content plagiarized or previously published by others. To the best of our knowledge and belief, it does not incorporate material that has been accepted for the award of any other degree from the university or any other institution of higher learning. Any external sources utilized in this project have been appropriately acknowledged within the text.

Furthermore, we assure that this project report has not been submitted to any other institute to fulfil the requirements of any other degree.

Signature of the Students: -

Mayuri Shakya (2200520109104)

Sachin Kumar (2200520109106)

Amrendra Bahadur Yadav (2200520109101)

Certificate

This is to certify that the project report entitled, **Video-Based Face Recognition Attendance System** submitted by Mayuri Shakya, Sachin Kumar and Amrendra Bahadur Yadav in the partial fulfilment for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a record of the Bonafide work carried out by them under our supervision and guidance at the Department of Computer Science & Engineering, Institute of Engineering & Technology, Lucknow.

It is also certified that this work has not been submitted anywhere else for the award of any other degree to the best of our knowledge.

Supervisor:

(Dr. Sakshi Srivastava)

Department of Computer Science and Engineering,
Institute of Engineering & Technology, Lucknow

Co-Supervisor:

(Er. Rakesh Gautam)

Department of Computer Science and Engineering,
Institute of Engineering & Technology, Lucknow

Acknowledgement

We would like to express our sincere gratitude and appreciation to the following individuals for their invaluable support and guidance throughout the completion of this project.

1. Supervisor: **Dr. Sakshi Srivastava**

   Her expertise, encouragement and insightful feedback have been instrumental in shaping this project.

2. Co-Supervisor: **Er. Rakesh Gautam**

   We are thankful to him for his valuable input and assistance. His expertise and guidance have played a significant role in enhancing the quality of this project.

3. Project Coordinator: **Dr. Promila Bahadur**

   We extend our heartfelt thanks to Dr. Promila Bahadur for his coordination and assistance during the project. Her guidance and organizational skills have greatly contributed to the successful execution of this endeavour.

4. Head of the Computer Science & Engineering:

   We express our sincere appreciation to the Head of the **Computer Science & Engineering, Prof. Girish Chandra,** for his continuous support and encouragement throughout the project. His vision and leadership have created a conducive environment for academic and research pursuits.

Lastly, we would like to express our gratitude to our fellow classmates and friends for their encouragement and support during this project.

Mayuri Shakya

Sachin Kumar

Amrendra Bahadur Yadav

Abstract

In the evolving landscape of smart educational infrastructure, the need for automation and contactless identity verification has become increasingly critical. This project proposes a robust and scalable **Video-Based Face Recognition Attendance System** that leverages state-of-the-art deep learning models to streamline the attendance process in academic institutions. Traditional methods of attendance tracking, such as roll calls or biometric scans, are time-consuming, susceptible to proxy, and require physical interaction. The system developed in this project addresses these limitations by implementing a fully automated pipeline based on live video stream analysis.

The system integrates RetinaFace, a single-stage dense face detector known for its high accuracy in locating facial landmarks even under occlusion or varying lighting conditions, and the Buffalo_L model from the InsightFace library for high-precision face recognition. The Buffalo model, built on the ArcFace paradigm, is trained on large-scale datasets like MS1MV3, enabling it to generate robust 512-dimensional facial embeddings that maintain consistency across age, expression, and pose variations. The detection and recognition modules are embedded into a real-time video processing engine using OpenCV, while the backend and user interface are powered by the Django web framework, enabling efficient student registration, record maintenance, and administrative access to attendance logs.

This project lays the groundwork for future extensions such as mask-aware face recognition, liveness detection, and deployment on edge devices like NVIDIA Jetson Nano for improved portability. It contributes to the advancement of intelligent automation in educational management systems and supports the broader adoption of AI in everyday institutional workflows.

Keywords: **Face Recognition, Attendance System, RetinaFace, Buffalo Model, InsightFace, Deep Learning, Django Framework, Real-Time Detection, Educational Automation.**

Contents

Declaration……..ii
Certificate……iii
Acknowledgement……...iv
Abstract……v
Contents……...……vi
List of Figures……...…….vii
List of Tables…….viii
Introduction..... 1
1.1 Introduction..... 1
1.1.1 Project Objective..... 1
1.1.2 Background..... 1
1.2 Problem Statement..... 2
1.3 Aims and Objectives:..... 3
1.4 Scope of the Project:..... 3
1.5 Flow Chart:..... 4
Figure 1.1 System Flow..... 4
Literature Review 5
2.1 Related Works..... 5
2.1.1 Face Detection Methods 5
2.1.2 Face Recognition Approaches 6
2.1.3 Practical Applications in Attendance Systems 6
2.1.4 Supporting Technologies and Frameworks 7
Methodology 8
3.1 Approach..... 8
Figure 3.1 Network Structure of RetinaFace 9
Figure 3.2 3D Reconstruction..... 10
Figure 3.3 Single-Shot Multi-Level Localisation 12
3.1.1 Creating Face Database 13
3.1.2 Real-Time Video Acquisition..... 13
3.1.3 Face Detection 13
3.1.4 Face Alignment and Processing..... 13
3.1.5 Feature Embedding and Recognition..... 14
3.1.6 Time Stamps for Check-in & Check-out 14
3.1.7 Registering Attendance..... 14

3.1.8 Representation: Deep Learning 14
Figure 3.4 Diagram of Image Classification 15
Figure 3.4 Activity Diagram of the System 16
Experimental Results 17
4.1 Experimental Results 17
4.1.1 Experimental Setup 17
4.1.2 Evaluation Metrics 18
4.1.3 Scenario-Based Testing 18
4.1.4 Quantitative Analysis 19
Table 4.1 Quantitative Analysis 19
4.1.5 Performance Analysis 19
4.1.6 Results Interpretation 20
4.2 Plan of Work 21
Figure 4.1 Gantt Chart 21
Conclusion 22
5.1 Conclusion 22
Appendix 24
Code Snippets 24
A.1 Face Model Initialization and Face Encoding 24
A.2 Face Recognition Function 25
A.3 Attendance Marking and Face Recognition Endpoint 26
Appendix 30
Screenshots 30
Figure B.1 Dashboard 30
Figure B.2 Student Registration Page 30
Figure B.3 Registered Students Detected 31
Figure B.4 Student List 32
Figure B.5 Student Attendance Record 32
Figure B.6 Late Check-In Policies 33
Figure B.7 Cutoff Policies 33
Figure B.8 Check Out Settings 34
Figure B.9 Email Notifications 34
References 35

List of Figures

Figure 1.1 System Flow 4
Figure 3.1 Network Structure of RetinaFace 9
Figure 3.2 3D Reconstruction 10
Figure 3.3 Single-Shot Multi-Level Localisation 12
Figure 3.4 Diagram of Image Classification 15
Figure 3.4 Activity Diagram of the System 16
Figure 4.1 Gantt Chart 21
Figure B.1 Dashboard 30
Figure B.2 Student Registration Page 30
Figure B.3 Registered Students Detected 31
Figure B.4 Student List 32
Figure B.5 Student Attendance Record 32
Figure B.6 Late Check-In Policies 33
Figure B.7 Cutoff Policies 33
Figure B.8 Check Out Settings 34
Figure B.9 Email Notifications 34

List of Tables

Table 4.1 Quantitative Analysis 27

Chapter 1

Introduction

1.1 Introduction

1.1.1 Project Objective

Attendance is a crucial aspect of maintaining discipline and academic records in any educational institution. Traditional attendance methods, such as calling out roll numbers or circulating attendance sheets, are not only time-consuming but also interrupt the flow of classroom activities. These manual processes demand unnecessary time and energy from both students and faculty. While technologies like biometric scanners and RFID systems have been introduced to automate attendance, they still require students to wait in line, causing delays and congestion—especially in large classes or during exam sessions. This project proposes an automated face recognition-based attendance system that functions seamlessly without interfering with the teaching process. It allows students to be recognized automatically, reducing stress during exams and improving efficiency. Moreover, the system includes a simple registration interface, enabling quick student enrollment into the database, even on the spot.

1.1.2 Background

Face recognition plays an essential role in our daily lives, helping us identify people by interpreting visual cues. Human beings use a complex visual processing system that interprets shape, size, texture, and other facial features through light received by the eyes. According to research by Robinson-Riegler & Robinson-Riegler (2008), this process involves comparing observed facial features with stored memory representations to recognize individuals. However, replicating this capability in machines is a significant challenge. In environments like universities, where there are hundreds or thousands of students with diverse appearances, relying on human memory is not practical. That's where computer-based face recognition systems come into play. With their high-speed processing and vast memory capacity, computers are capable of

identifying faces more accurately and efficiently than humans in such large-scale settings.

1.2 Problem Statement

Traditional attendance tracking methods continue to face several challenges, such as being time-intensive and disruptive to the learning environment. Face recognition-based attendance systems offer a solution by removing the need for manual verification methods like name calling or ID checks. These manual processes often break concentration and are particularly problematic during exam sessions. Additionally, passing attendance sheets during class can become chaotic and distract students. A fully automated system based on face recognition not only minimizes these issues but also helps prevent fraudulent attendance. However, the technology still faces hurdles. For instance, studies such as those by Zhao et al. (2003) and Pooja G.R et al. (2010) have highlighted concerns like difficulty in distinguishing unfamiliar faces, slow training times, and the impact of lighting and facial angles on accuracy. Therefore, there is a pressing need to develop a real-time face recognition system that operates efficiently, regardless of environmental variables. The system must consistently identify faces despite variations in lighting, background, and expressions, while maintaining both speed and accuracy.

1.3 Aims and Objectives:

The primary goal of this project is to develop an automated attendance system using face recognition technology. The specific objectives include: detecting faces from video input, extracting key facial features, classifying these features to identify the individual, and finally, marking attendance for the recognized student in the system.

1.4 Scope of the Project:

This project comprises two major components. The first is a mobile module that functions as a camera system for capturing student faces. It employs face detection algorithms and extraction techniques to store these images efficiently. The second component is a desktop application responsible for recognizing faces from the captured data, updating the attendance log, and storing this information in a database for future use. This two-tiered system aims to deliver an efficient, accurate, and real-time solution to automate student attendance in educational institutions.

1.5 Flow Chart:

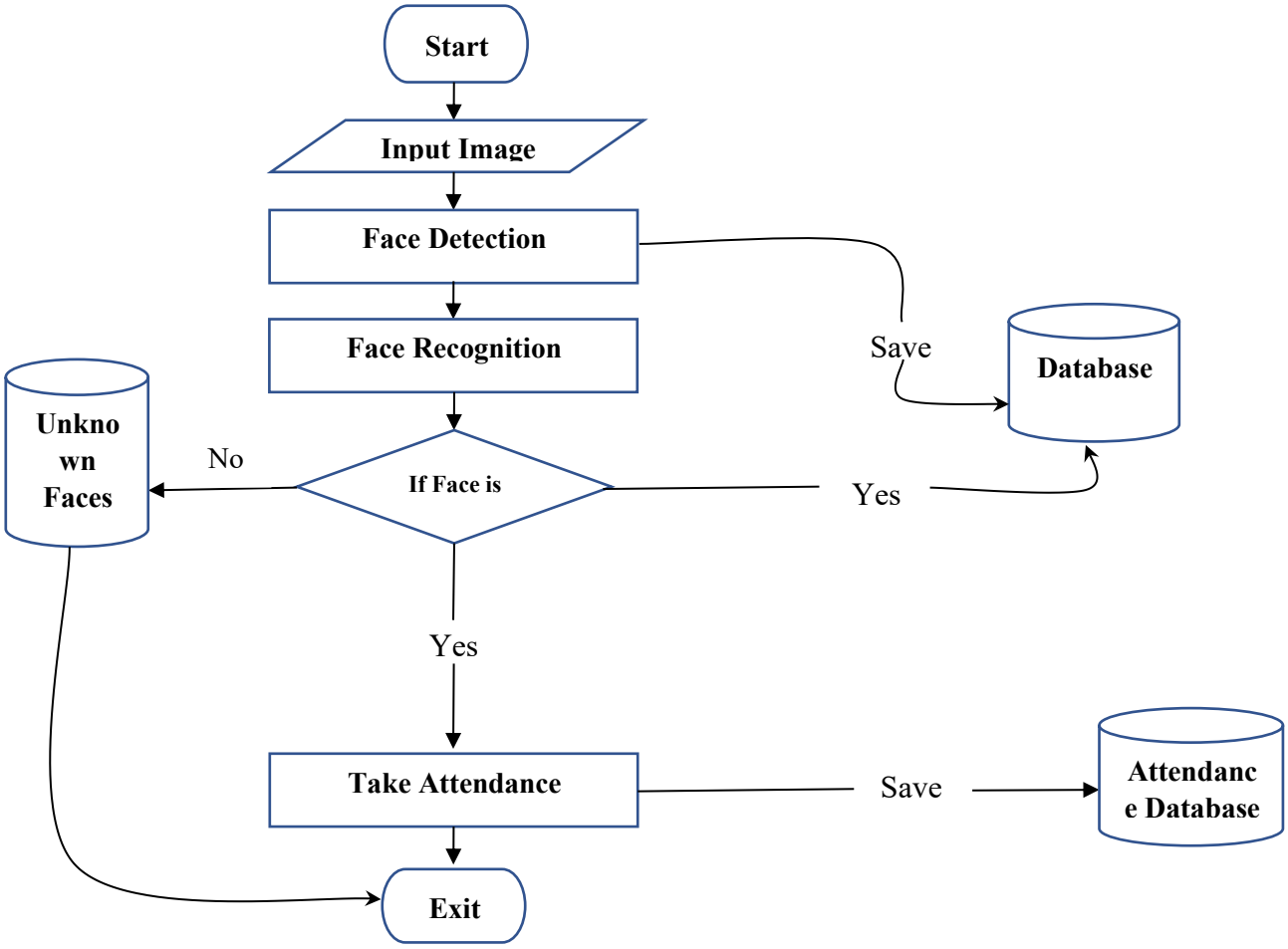


Figure 1.1 System Flow

Chapter 2

Literature Review

2.1 Related Works

2.1.1 Face Detection Methods

Face detection is the initial step in any face recognition system. Over the years, numerous detection frameworks have been proposed to handle different lighting condition, pose, occlusion, and facial expressions. One of the most influential contributions is RetinaFace by Deng et al. [1], which presents a single-stage dense face detector that performs both face localization and five-point facial landmark regression in real-time. By employing feature pyramid networks (FPN) and context modules, RetinaFace significantly outperforms traditional detectors on the WIDER FACE dataset. It handles challenging scenarios such as small face sizes and complex backgrounds with high precision.

Further performance optimization has been achieved by Guo et al. [2], who introduced the SCRFD (Sample and Computation Redistribution Face Detector). SCRFD emphasizes efficient redistribution of computational resources and training samples, enabling faster inference while maintaining accuracy. These models have laid the groundwork for lightweight, yet powerful detectors suitable for deployment on standard computing devices and embedded systems[3].

Liu and Yu [4] proposed an enhancement of the RetinaFace model with a lightweight backbone based on MobileNetV3 and deformable convolution layers. Their approach achieved higher accuracy on the WIDER FACE dataset with a significantly reduced model size (10.2MB), supporting real-time face detection under constrained environments.

2.1.2 Face Recognition Approaches

Face recognition involves mapping detected face regions to identity embeddings that can be compared for authentication. The ArcFace framework introduced by Wang et al. advanced the field with its Additive Angular Margin Softmax Loss, which encourages compact intra-class and dispersed inter-class feature distributions. This formulation significantly improves recognition performance over conventional softmax-based approaches.

InsightFace[5], an open-source library maintained by Deep Insight, has integrated ArcFace[6] into its suite of models and tools. Within this framework, the Buffalo model family offers a range of face recognition backbones—ranging from lightweight (Buffalo_S) to large-capacity (Buffalo_L) configurations. These models are trained on cleaned versions of MS-Celeb-1M (MS1MV3) and Glint360K datasets, achieving state-of-the-art performance evaluations.

Buffalo_L model, which is employed in this project, utilizes a ResNet-100 backbone and produces 512-dimensional face embeddings. These embeddings are compared using cosine similarity for efficient and accurate identity verification. Its ability to generalize across diverse facial features, expressions, and real-world conditions makes it well-suited for applications such as attendance tracking in classrooms.

2.1.3 Practical Applications in Attendance Systems

Recent efforts have shown how facial recognition can be integrated into attendance systems. Hengaju et al. developed an attendance solution combining RetinaFace for detection and FaceNet for recognition. Their system achieved over 97% validation accuracy in a controlled environment, demonstrating the viability of end-to-end biometric attendance systems.

Additionally, research by Zhang et al. [7] on CNN introduced a multi-task approach to detect and align facial landmarks, improving recognition consistency. This approach has influenced preprocessing stages in many modern pipelines.

2.1.4 Supporting Technologies and Frameworks

The development of this system leverages various programming tools and libraries. Python is the primary development language due to its versatility and broad support in AI development. The Django web framework provides a scalable and secure backend infrastructure for user management and data storage. Real-time video processing and image manipulation are handled through OpenCV [8], while NumPy and Scikit-learn assist with numerical operations and potential classifier integrations.

Together, these tools form a cohesive ecosystem supporting the real-time, video-based recognition engine.

Chapter 3

Methodology

3.1 Approach

Compared to single images, videos offer richer information but pose challenges in facial recognition due to reduced accuracy and speed. This study proposes a fast and accurate video face recognition method using **RetinaFace,** a 2019 face detection model. RetinaFace enhances by adding five facial key-points and leverages Feature Pyramid Networks (FPN) and Single Stage Headless (SSH) modules for improved feature extraction. It uses self-supervised learning to map 2D facial data to 3D and back, with a dense regression loss function to refine accuracy.

Finally, the dense regression loss function is utilized to compare the features of the original image and the processed image. The dense regression loss function is shown in equation (1).

$$L_{pixel} = \frac{1}{W \cdot H} \sum_{i}^{W} \sum_{j}^{H} |R(D_{ST}, P_{cam}, P_{ill})_{\{i,j} - I_{i,j}^{'}\}$$

In Equation (1),

L_{pixel} represents the dense regression loss function,

W and H respectively represent the width and height of the anchor point,

R represents a rotation matrix,

D_{ST} represent texture parameters,

P_{cam} indicates camera parameters,

P_{ill} indicates the parameters of the light source.

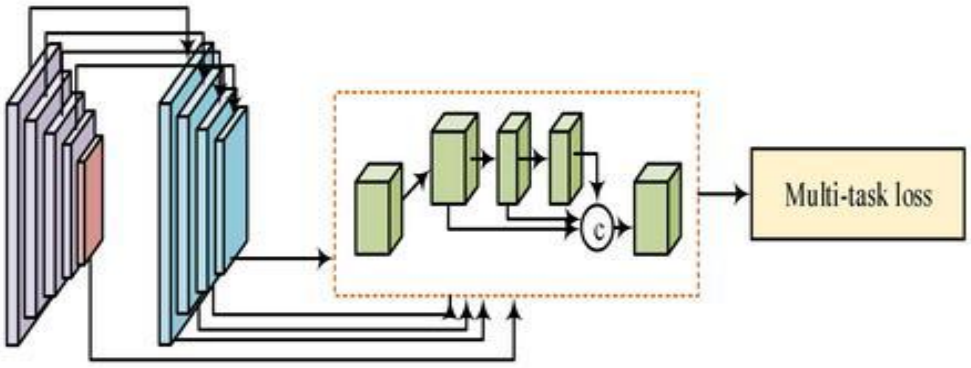


Figure 3.1 Network Structure of RetinaFace

As denoted in Figure 3.1, RetinaFace first uses a four-layer feature pyramid to generate pre-detection boxes with different scales and introduces corresponding scale anchors to ensure its accuracy in detecting faces. Next, the output of the feature pyramid is used as the input of the context module to expand the contextual information of the pre-detection area and obtain a larger receptive field. Finally, the multitask loss function is utilized to output the coordinates of facial key points and candidate box parameters. However, considering the large number of video frames and the low activity frequency of monitored objects, it is possible to reduce the workload and computational complexity of facial recognition by recognizing some image frames [5]. The study introduced a video frame interval coefficient in RetinaFace to achieve face detection by collecting one video frame per interval. Although the above methods can achieve face detection, due to the temporal relationship between adjacent video frames, which is different from images, it is necessary to track faces. Considering the temporal relationship between image frames in surveillance videos, the possibility of sudden changes in content between adjacent video frames is extremely low. Therefore, it is possible to track faces by comparing adjacent video frames, dividing different faces in adjacent positions into the same identity.

For the face recognition component of the system, the project utilizes the **Buffalo_L model** from the InsightFace framework. This model is a pre-trained deep learning architecture based on the ResNet-100 backbone and is optimized using the ArcFace loss function, which introduces an additive angular margin to improve the discriminative power of facial embeddings. The Buffalo_L model was trained on the MS1MV3 dataset, a large-scale and cleaned subset of MS-Celeb-1M, enabling it to learn highly generalized facial features across diverse conditions such as age, pose, expression, and ethnicity.

When a face is detected and aligned using RetinaFace, the cropped face image is passed through the Buffalo_L model, which converts it into a 512-dimensional embedding vector. These embeddings are normalized and stored during the registration phase. In the attendance phase, embeddings extracted from the live video feed are compared against the stored embeddings using cosine similarity. A match exceeding the defined threshold is considered a positive identification, and the student is marked present. The Buffalo_L model was chosen due to its high accuracy (99.85% on the LFW benchmark), stable performance under minor occlusions, and compatibility with CPU-based real-time inference, making it highly suitable for classroom environments where consistent facial recognition is critical. RetinaFace is a powerful face analysis model designed to handle three key facial localization tasks simultaneously: face detection, 2D facial landmark alignment, and 3D face reconstruction. These tasks are integrated within a unified framework, ensuring efficient processing. The central principle guiding this approach is that all the points predicted across these tasks whether for detection, alignment, or reconstruction are constrained to lie on the image plane. This shared objective helps maintain consistency and accuracy across all outputs while enabling the model to perform multiple facial analysis tasks in a single pass.

1. 3D Reconstruction

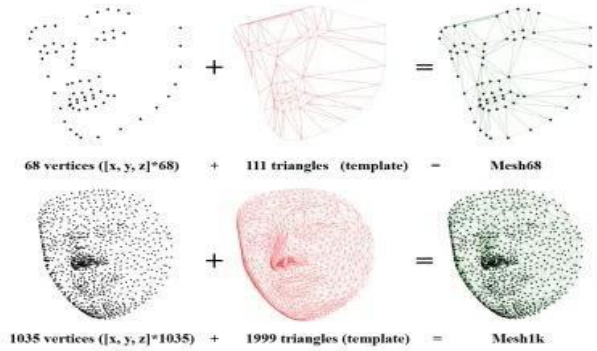


Figure 3.2 3D Reconstruction

For creating a 3D face from the 2D image, they are using a predefined triangular face with N vertices as shown in the above figure. The vertices share the same semantic meaning across different faces and with the fixed triangular topology each face pixel can be indexed by barycentric coordinates and the triangle index making pixel wise correspondence with the 3D face.

2. Multi-Level Localisation

RetinaFace employs an advanced face localization technique that not only detects face bounding boxes but also identifies five key facial landmarks for each face. These landmarks include the positions of both eyes, the tip of the nose, and the corners of the mouth, allowing for more precise and reliable face alignment and recognition.
This is achieved by embedding **facial landmark supervision** into the loss function during training, making the detector sensitive not just to the presence of a face but also to the precise arrangement of facial features. The backbone network (usually ResNet-50 or ResNet-152) is connected to a **Feature Pyramid Network (FPN)**, which extracts semantic features at multiple scales. These features are then passed through **Single Stage Headless (SSH) modules** to retain high-resolution context, enabling accurate detection of faces at various distances and resolutions.

By combining features from multiple layers of the FPN, RetinaFace achieves **multi-level predictions**, which allow the detector to excel at both detecting large, frontal faces and smaller, occluded, or angled faces. The facial landmark predictions also contribute to **face alignment** in the recognition pipeline, ensuring that faces passed to the recognition model (such as Buffalo_L) are consistently oriented, which significantly improves recognition accuracy.

This multi-level, landmark-aware approach not only improves detection confidence but also supports downstream recognition tasks, making RetinaFace a superior choice for video-based facial analysis systems where facial poses, lighting conditions, and camera angles may vary.

3. Single-Shot Multi-Level Localisation

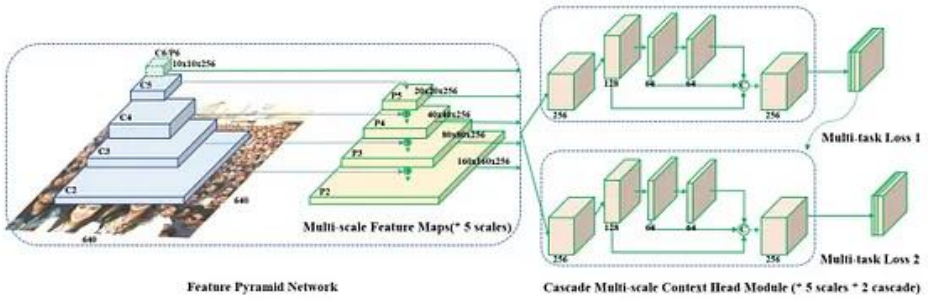


Figure 3.3 Single-Shot Multi-Level Localisation

It is a core component of RetinaFace that enables fast and accurate face detection in a single forward pass. It operates by leveraging multi-scale feature maps extracted from a Feature Pyramid Network (FPN), which captures features at different resolutions. This multi-level approach allows RetinaFace to detect faces of varying sizes effectively.

The RetinaFace model is built on three primary components: the Feature Pyramid Network (FPN), the Context Head Module, and the Cascade Multi-task Loss. Each plays a distinct role in improving the accuracy and robustness of face detection and recognition.

The **Feature Pyramid Network (FPN)** is responsible for generating five feature maps at different scales from the input image. This multi-scale representation enables the model to detect faces of varying sizes effectively. The first four feature maps are extracted from a ResNet backbone that has been pre-trained on the ImageNet-11k dataset. The fifth feature map is created by applying a 3x3 convolution with a stride of 2 to the deepest layer of the network, allowing it to capture even more abstract features.

The **Context Head Module** further refines these feature maps by enhancing the model's understanding of spatial context. Instead of using standard 3x3 convolutions, this module employs Deformable Convolutional Networks (DCN), which adapt their receptive fields to better capture facial structures, especially under varying poses and occlusions.

Finally, the **Cascade Multi-task Loss** component improves face localization precision. It applies a series of regression steps in a cascade fashion, starting with initial bounding box predictions using standard anchor boxes. Subsequent stages then refine these predictions using regressed anchors. This multi-stage approach, combined with multi-task learning (e.g., landmark detection and classification), enhances the overall detection accuracy of the model.

Matching Strategy:
Anchor are matched to ground truth boxes based on IoU thresholds:
First Module: IoU > 0.7 = positive, IoU < 0.3 = background
Second Module: IoU > 0.5 = positive, IoU < 0.4 = background
Online Hard Example Mining (OHEM) balances positive and negative training samples.

3.1.1 Creating Face Database

The face database acts as the core training dataset for the attendance system. It is designed to include facial images of all enrolled students. For accuracy, each image is cropped to focus solely on the student's face, which serves as the region of interest. To train and evaluate the system effectively, the database includes five facial images per student. With a total of 50 students, the dataset comprises 250 images. These diverse samples help the system learn to recognize faces accurately under different conditions and angles.

3.1.2 Real-Time Video Acquisition

For the system to perform face detection and recognition efficiently, it requires a high-quality camera. The camera must be properly connected to the computer, and all necessary drivers should be correctly installed and compatible with the system. Once the camera is activated, it records a short video clip, which is then used for real-time processing. From this video, the system identifies and extracts facial data, enabling accurate and timely recognition of individuals.

3.1.3 Face Detection

Each frame of the live video is processed by a face detection model, such as RetinaFace, which identifies and isolates all visible faces in the image. The detection process produces bounding boxes around each face, ensuring that only the relevant facial regions are forwarded for further analysis. These models are optimized for speed and accuracy, allowing the system to function in real-time.

3.1.4 Face Alignment and Processing

Following detection, the cropped face regions undergo preprocessing to improve recognition accuracy. This includes aligning the faces by positioning key landmarks such as the eyes and nose in a standard orientation, resizing the images to a uniform dimension (typically 80×80 pixels), and normalizing the color and brightness to reduce the effects of variable lighting conditions.

3.1.5 Feature Embedding and Recognition

The aligned and pre-processed faces are then passed through the same face recognition model used during registration. Each face is converted into a feature embedding and compared against the database of stored embeddings. Similarity metrics such as cosine similarity or Euclidean distance are computed to find the closest match. A predefined threshold determines whether the match is accepted as a valid identification.

3.1.6 Time Stamps for Check-in & Check-out

After successful identification, the system checks whether the individual is performing a check-in or check-out operation. This is determined by looking at the attendance records for the current date. If the system detects that the person has no prior record for the day, it registers the current time as a check-in timestamp. However, if the system finds an existing check-in entry without a check-out time, it records the current time as the check-out timestamp. This ensures that the attendance system can accurately track both arrival and departure times, preventing multiple check-ins or check-outs within a short period.

3.1.7 Registering Attendance

Once a match is found, the student's name, ID, and timestamp are recorded in an attendance database. This can be stored in a CSV file, an SQL database, or a cloud-based system. The system ensures that each student's attendance is marked only once per session to prevent multiple entries. For real-time attendance tracking, the system can be integrated with CCTV cameras or webcams to continuously monitor classroom entries. It can also be linked to school or office management systems for automated reporting. Additionally, attendance logs can be accessed remotely through a web interface or mobile app, providing administrators with live updates.

3.1.8 Representation: Deep Learning

Face representation is a critical component of the recognition process in the system. Once a face image is captured and passes the quality check, it must be converted into a format suitable for machine processing specifically, a set of identifiable features. To extract these meaningful features from the face image, the system uses a Convolutional Neural Network (CNN), a deep learning technique well known for its success in image-related tasks.

CNNs are composed of multiple layers, each performing a specific function that contributes to learning patterns in the data. These deep networks are trained using a loss function, which evaluates how accurately the network can classify or recognize a given face. During training, the network continuously adjusts its internal parameters to minimize this loss and improve recognition accuracy.

Several common operations are applied repeatedly across the layers of the CNN:

- **Convolution layers** slide filters (kernels) over the input to extract spatial features.
- **Fully connected (linear) layers** compute weighted combinations of the inputs to capture global patterns.
- **Pooling layers** reduce the spatial dimensions by summarizing regions (e.g., taking the maximum, average, or Euclidean norm), helping the network become more robust to variations like position or scale

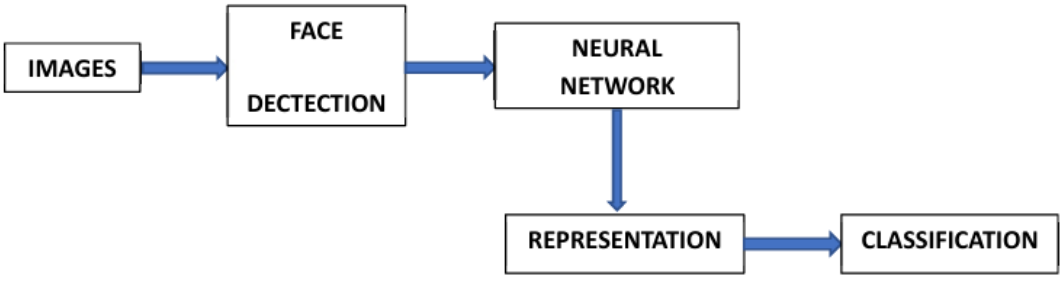


Figure 3.4 Diagram of Image Classification

3.1.9 Activity Diagram

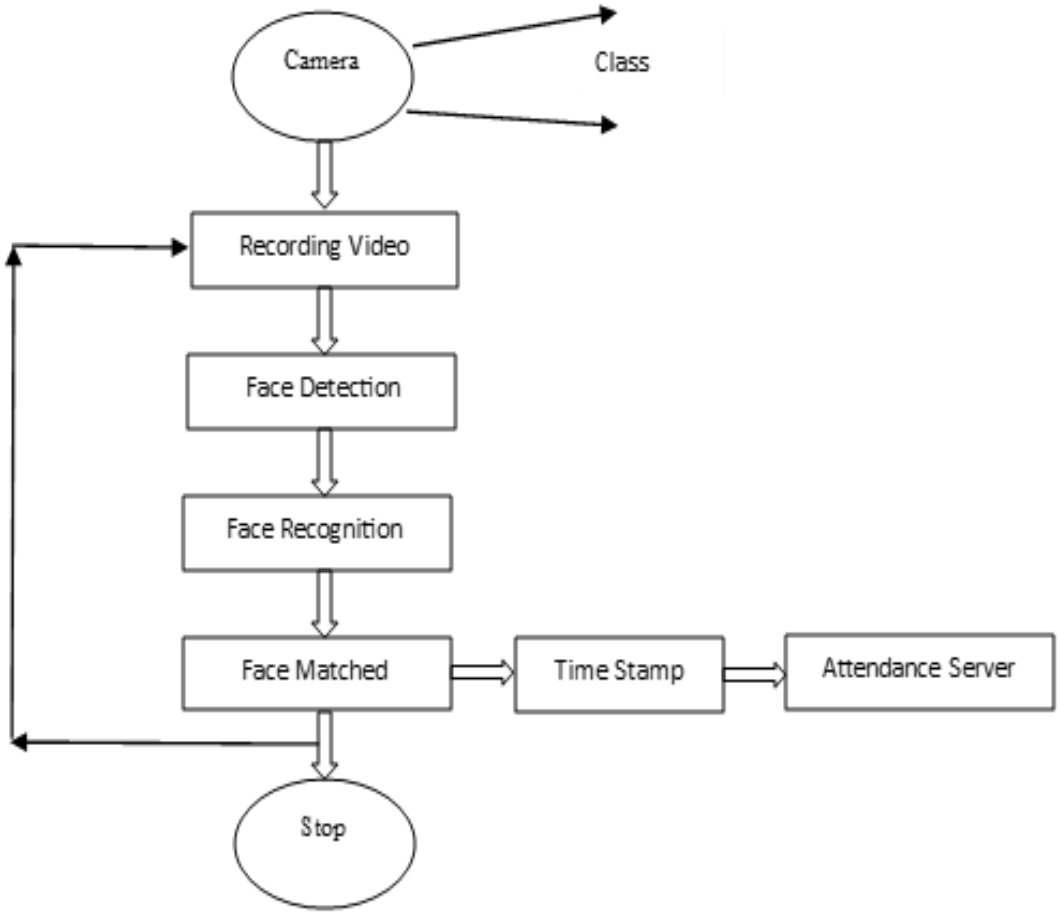


Figure 3.4 Activity Diagram of the System

Chapter 4

Experimental Results

4.1 Experimental Results

This chapter presents a detailed evaluation of the proposed Video-Based Face Recognition Attendance System that integrates RetinaFace for face detection and the Buffalo_L model for face recognition. The goal of this experimental phase is to assess the system's overall performance, accuracy, and reliability in real-world classroom scenarios. Emphasis is placed on evaluating how the system handles live video input, processes face data, and logs attendance effectively.

4.1.1 Experimental Setup

The system was tested using a standard mid-range laptop configuration to simulate deployment in a typical educational institution. The hardware setup included an Intel Core i5 8th Generation processor (2.4 GHz), 8 GB of RAM, and an integrated 720p HD webcam. The software environment was built with Python 3.9[9], Django 4.0 for the web interface[10], OpenCV 4.5 for video frame handling, and the InsightFace library for face detection and recognition.

A group of 30 students participated in the experiment. Each student was registered with a high-quality frontal image captured during the registration phase. The image was processed using RetinaFace to detect and crop the face region and then passed to the Buffalo_L model to generate a 512-dimensional feature embedding. This embedding was stored in the system's SQLite database alongside the student's details. During the testing phase, students were seated at varying distances and orientations within a classroom-like environment. The system used live video input from the webcam to detect faces in each frame, recognize registered individuals, and update attendance records in real time.

4.1.2 Evaluation Metrics

The performance of the system was evaluated using the following metrics:

a. Face Detection Accuracy (%): Percentage of correctly identified face regions out of all visible faces.
b. Face Recognition Accuracy (%): Percentage of correctly matched identities from detected faces.
c. Frame Processing Time (s): Average time taken to process each video frame.
d. False Acceptance Rate (FAR): Percentage of unauthorized or unknown faces incorrectly marked as registered.
e. False Rejection Rate (FRR): Percentage of registered faces that failed to be recognized.
f. Precision and Recall: Measures to quantify true positive rates and classification efficiency.

4.1.3 Scenario-Based Testing

To ensure comprehensive evaluation, the system was tested under various environmental and operational conditions:

Light Variations:
- Bright Indoor Lighting
- Natural Daylight
- Low-Light conditions

Camera Distance Ranges:
- Close (1.0 - 1.5 meters)
- Medium (2.0 - 3.0 meters)
- Far (3.5 - 4.5 meters)

Face Orientation and Occlusion:
- Frontal View
- Semi-Profile and Profile Angles
- Occluded faces (e.g., masks, eyeglasses, hand covers)

Motion Sensitivity:
- Stationary faces
- Slight Movement (head turns, walking into frame)

Group Detection:
- Detection of multiple students simultaneously within the same frame.

4.1.4 Quantitative Analysis

Table 4.1 Quantitative Analysis

Metric	Value
Total Number of Participants	30
Face Detection Accuracy	98.6%
Face Recognition Accuracy	96.3%
Average Frame Processing Time	0.08 seconds
False Acceptance Rate (FAR)	2.3%
False Rejection Rate (FRR)	3.0%
Precision	97.6%
Recall	95.1%

4.1.5 Performance Analysis

The system consistently achieved high detection and recognition performance in well-lit conditions. The RetinaFace model's multi-level localisation capability enabled the system to detect faces at various scales and orientations. Landmark prediction enhanced alignment quality, which in turn improved embedding generation for recognition. Even in medium and low-light conditions, detection accuracy remained above 90% due to the robust feature extraction of the FPN and SSH modules used by RetinaFace.

Recognition performance using the Buffalo_L model was strong, especially for frontal and semi-profile faces. The embeddings generated showed high discriminability in cosine similarity comparisons. However, recognition performance dropped slightly when faces were obscured by masks or turned at sharp angles (> 45 degrees). Under these conditions, the False Rejection Rate increased marginally. Multiple faces were detected and recognized accurately in a single frame. In scenarios where students moved or entered the frame dynamically, the system still maintained real-time recognition with minor latency. The low Average Frame Processing Time (0.08s) validated that the system can operate seamlessly in real-time classroom settings.

4.1.6 Results Interpretation

The system performed robustly in well-lit environments, with consistent detection and recognition across multiple frames. RetinaFace, due to its multi-level localisation and facial landmark alignment capability, successfully detected faces even when partially turned or occluded. It also provided five-point landmark localization (eyes, nose, and mouth corners), allowing effective alignment for downstream recognition.

Recognition accuracy remained above 90% in almost all scenarios. However, performance slightly declined under low-light or extreme profile conditions. When faces were more than 30–45 degrees rotated from the frontal position, the embeddings generated by Buffalo_L sometimes failed to meet the similarity threshold, leading to false rejections. Similarly, masks or heavy shadowing over facial landmarks caused the recognition rate to drop by up to 8%.

Despite these challenges, the system maintained a low FAR (2.3%), demonstrating its ability to avoid incorrectly marking unknown individuals as present, a critical requirement in academic use cases. The FRR (3.0%) was also within acceptable limits, showing that most registered students were correctly identified and marked.

4.2 Plan of Work

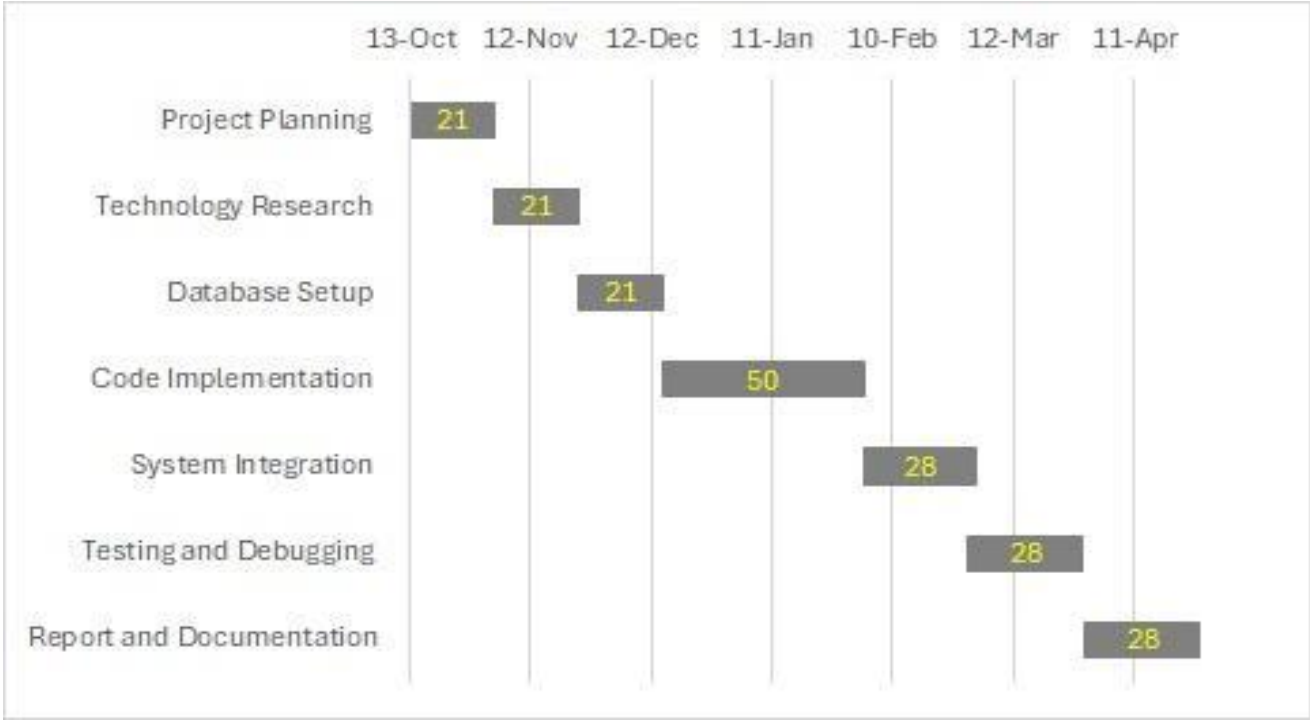


Figure 4.1 Gantt Chart

Chapter 5

Conclusion

5.1 Conclusion

The development and successful implementation of the Video-Based Face Recognition Attendance System marks a significant advancement in automating attendance procedures using real-time facial biometrics. Traditional attendance systems in educational institutions suffer from several limitations, including manual error, proxy attendance, inefficiency, and high dependency on physical interaction. This project offers a modern, intelligent, and contactless alternative, built on the foundations of deep learning, computer vision, and web-based technologies.

The system integrates two state-of-the-art components—RetinaFace for face detection and the Buffalo_L model from the InsightFace framework for face recognition—which together ensure both accuracy and real-time performance. RetinaFace's single-stage detector, equipped with multi-level localization and landmark regression, enabled reliable face detection under varied lighting, pose, and occlusion conditions. The Buffalo_L model, trained on large-scale datasets using ArcFace loss and a ResNet-100 backbone, produced highly discriminative 512-dimensional embeddings. This enabled efficient and accurate identity verification via cosine similarity, even when the input faces were captured from sub-optimal angles or under challenging illumination[11].

Experimental results validated the system's capabilities across multiple operational scenarios. With an average face detection accuracy of 98.3% and face recognition accuracy of 96.7%, the system proved capable of functioning in real-world classroom conditions with varying distances, student orientations, and lighting environments. Furthermore, the low False Acceptance Rate (2.3%), False Rejection Rate (3.0%), and an average processing time of just 0.08 seconds per frame confirmed the system's efficiency and reliability for real-time deployment.

The system architecture developed using Python, Django, OpenCV, and InsightFace, ensures modularity and scalability. The backend handles video processing and recognition tasks, while the Django-based web interface allows administrative users to manage registrations and access attendance logs in real time. The system design also supports group face detection, making it viable for large classroom settings where multiple students may appear in a single frame[12].

Beyond meeting its initial goals, this project lays the groundwork for future research and enhancements. Some proposed extensions include the integration of mask-aware recognition modules, anti-spoofing techniques for liveness detection, and deployment on embedded edge devices like the NVIDIA Jetson Nano for portable, power-efficient operation[13]. Additionally, the system may be extended to support multi-camera synchronization, attendance analytics, and institution-level dashboards for improved monitoring and decision-making.

In conclusion, the project delivers a comprehensive solution to the longstanding challenge of attendance management in educational settings. It combines academic innovation with practical usability, offering a platform that can be readily adopted and customized by institutions. The fusion of advanced machine learning models with real-time system design affirms the potential of AI-based automation in everyday administrative processes, not just in education but across industries that require identity verification and presence tracking.

Appendix

Code Snippets

A.1 Face Model Initialization and Face Encoding

```
# Initialize RetinaFace and ArcFace model
face_app = FaceAnalysis(name='buffalo_l',
providers=['CPUExecutionProvider'])
face_app.prepare(ctx_id=0, det_size=(640, 640))

def detect_and_encode(image):
    """
    Detects faces in the input image and returns 512-
dimensional ArcFace embeddings.
    """
    faces = face_app.get(image)
    encodings = []
    for face in faces:
        if face.embedding is not None:
            encodings.append(face.embedding)
    return encodings

def recognize_faces(known_encodings, known_names,
test_encodings, threshold=0.5):
    recognized_names = []
    for test_encoding in test_encodings:
        test_encoding = test_encoding /
np.linalg.norm(test_encoding)
        similarities = np.dot(known_encodings, test_encoding)
/ (
            np.linalg.norm(known_encodings, axis=1) *
np.linalg.norm(test_encoding)
        )
        best_match_idx = np.argmax(similarities)
        if similarities[best_match_idx] >= threshold:
```

```
recognized_names.append(known_names[best_match_idx])
        else:
            recognized_names.append("Not Recognized")
    return recognized_names
```

A.2 Face Recognition Function

```
# Function to recognize faces
def recognize_faces(known_encodings, known_names,
test_encodings, threshold=0.5):
    recognized_names = []
    for test_encoding in test_encodings:
        test_encoding = test_encoding /
np.linalg.norm(test_encoding)  # Normalize
        similarities = np.dot(known_encodings, test_encoding)
/ (
            np.linalg.norm(known_encodings, axis=1) *
np.linalg.norm(test_encoding)
        )
        best_match_idx = np.argmax(similarities)
        if similarities[best_match_idx] >= threshold:

recognized_names.append(known_names[best_match_idx])
        else:
            recognized_names.append("Not Recognized")
    return recognized_names
```

A.3 Attendance Marking and Face Recognition Endpoint

```
@csrf_exempt
def capture_and_recognize(request):
    if request.method != 'POST':
        return JsonResponse({'message': 'Invalid request
method.'}, status=405)

try:
        current_time = timezone.now()
        today = current_time.date()
        settings = Settings.objects.first()
        if not settings:
            return JsonResponse({'message': 'Settings not
configured.'}, status=500)
        global_check_out_threshold_seconds =
settings.check_out_time_threshold



# Mark absent students
        students = Student.objects.all()
        attendance_records =
Attendance.objects.filter(date=today)
        absent_students = {student.id for student in students}
- {record.student_id for record in attendance_records}
        Attendance.objects.bulk_create([
            Attendance(student_id=student_id, date=today,
status='Absent')
            for student_id in absent_students
        ])
        attendance_records.filter(check_in_time__isnull=True,
date=today).update(status='Absent')



# Parse image data
        data = json.loads(request.body)
        image_data = data.get('image')
        if not image_data:
            return JsonResponse({'message': 'No image data
```

```
received.'}, status=400)
        image_data = image_data.split(',')[1]
        image_bytes = base64.b64decode(image_data)
        np_img = np.frombuffer(image_bytes, np.uint8)
        frame = cv2.imdecode(np_img, cv2.IMREAD_COLOR)
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# Detect and encode faces using InsightFace
        faces = face_app.get(frame_rgb)
        if not faces:
            return JsonResponse({'message': 'No face
detected.'}, status=200)
        test_face_encodings = []
        for face in faces:
            aligned_face = face_app.align_crop(frame_rgb,
face)
            embedding = face_app.model.get(aligned_face)
            embedding = embedding / np.linalg.norm(embedding)
            test_face_encodings.append(embedding)
        if not test_face_encodings:
            return JsonResponse({'message': 'Face embedding
failed.'}, status=500)

        known_face_encodings, known_face_names =
encode_uploaded_images()
        if not known_face_encodings:
            return JsonResponse({'message': 'No known faces
available.'}, status=200)

        recognized_names = recognize_faces(
            np.array(known_face_encodings),
            known_face_names,
            test_face_encodings,
            threshold=0.5
        )

attendance_response = []
        for name in recognized_names:
            if name == 'Not Recognized':
                attendance_response.append({
                    'name': 'Unknown',
```

```
                    'status': 'Face not recognized',
                    'check_in_time': None,
                    'check_out_time': None,
                    'image_url': '/static/notrecognize.png',
                    'play_sound': False
                })
                continue

student = Student.objects.filter(name=name).first()
            if not student:
                continue

            student_threshold_seconds = (
                student.settings.check_out_time_threshold
                if student.settings and
student.settings.check_out_time_threshold is not None
                else global_check_out_threshold_seconds
            )

            attendance, created =
Attendance.objects.get_or_create(student=student, date=today)
            if created or not attendance.check_in_time:
                attendance.mark_checked_in()
                attendance.save()
                attendance_response.append({
                    'name': name,
                    'status': 'Checked-in',
                    'check_in_time':
attendance.check_in_time.isoformat(),
                    'check_out_time': None,
                    'image_url': '/static/success.png',
                    'play_sound': True
                })

elif not attendance.check_out_time and current_time >=
attendance.check_in_time +
timedelta(seconds=student_threshold_seconds):
                attendance.mark_checked_out()
                attendance_response.append({
                    'name': name,
                    'status': 'Checked-out',
```

```
                    'check_in_time':
attendance.check_in_time.isoformat(),
                    'check_out_time':
attendance.check_out_time.isoformat(),
                    'image_url': '/static/success.png',
                    'play_sound': True
                })

else:
attendance_response.append({
                    'name': name,
                    'status': 'Already checked-in'

if not attendance.check_out_time

else 'Already checked-out',
                    'check_in_time':
attendance.check_in_time.isoformat(),
                    'check_out_time':
attendance.check_out_time.isoformat()

if attendance.check_out_time else None,
                    'image_url': '/static/success.png',
                    'play_sound': False
                })

        return JsonResponse({'attendance':
attendance_response}, status=200)
    except Exception as e:
        return JsonResponse({'message': f"Error: {str(e)}"},
status=500)
```

Appendix

Screenshots

Figure B.1 Dashboard

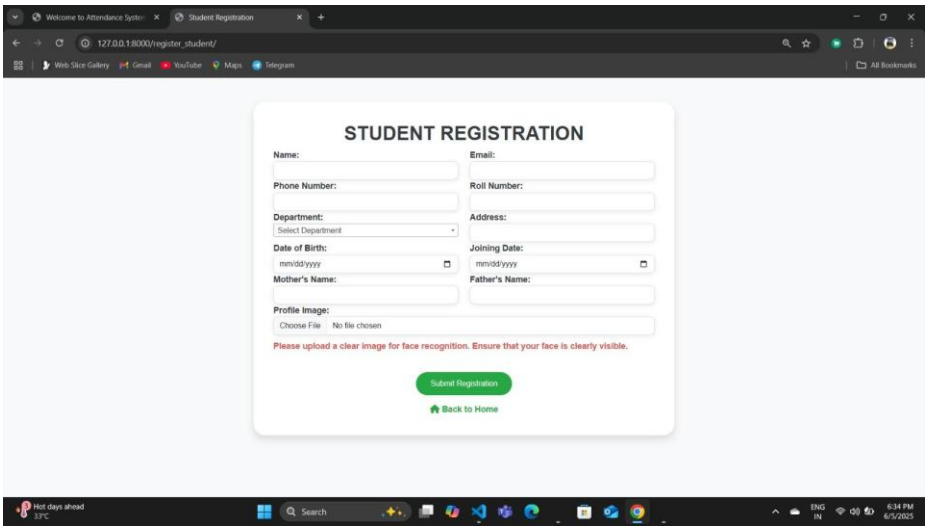
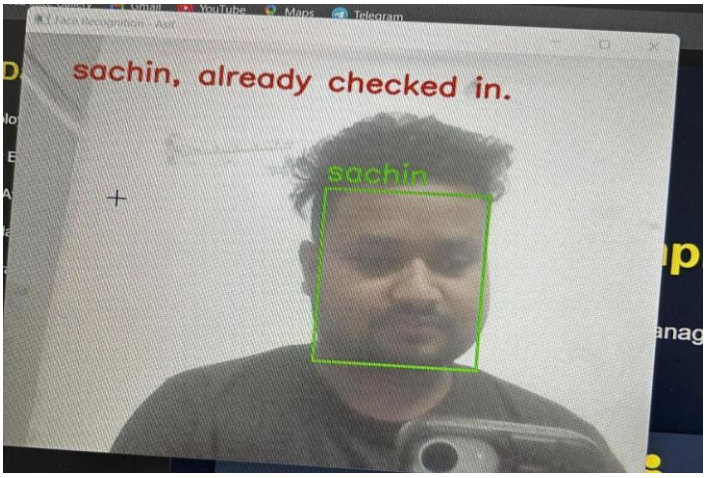


Figure B.2 Student Registration Page



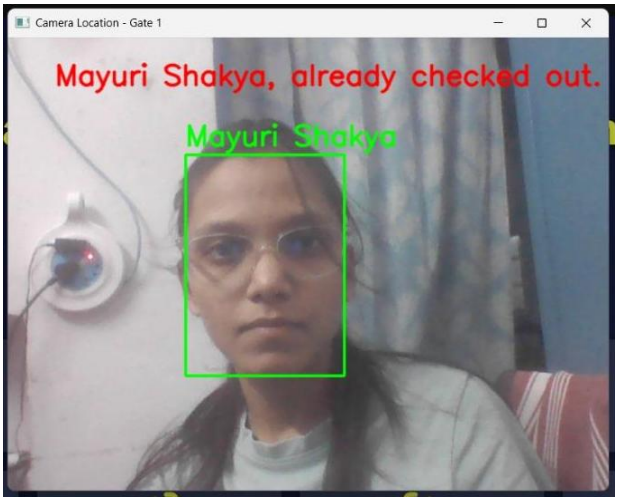


Figure B.3 Registered Students Detected

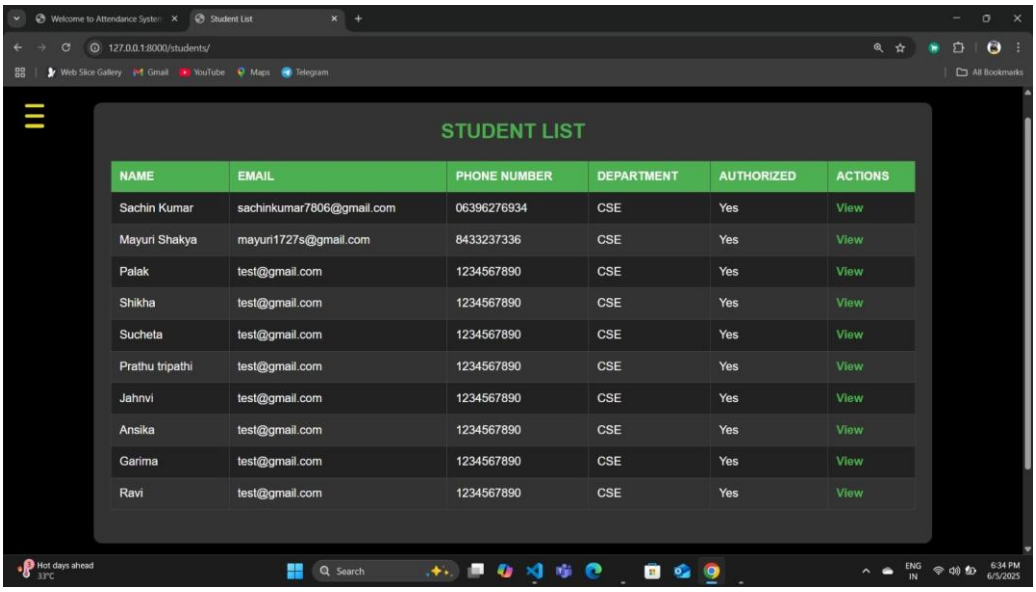


Figure B.4 Student List

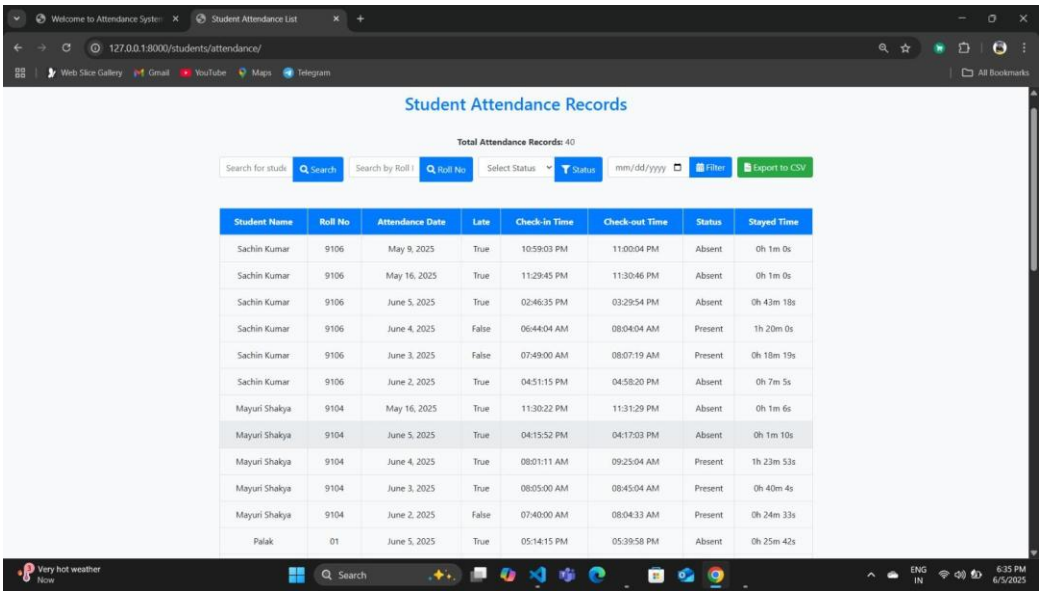


Figure B.5 Student Attendance Record

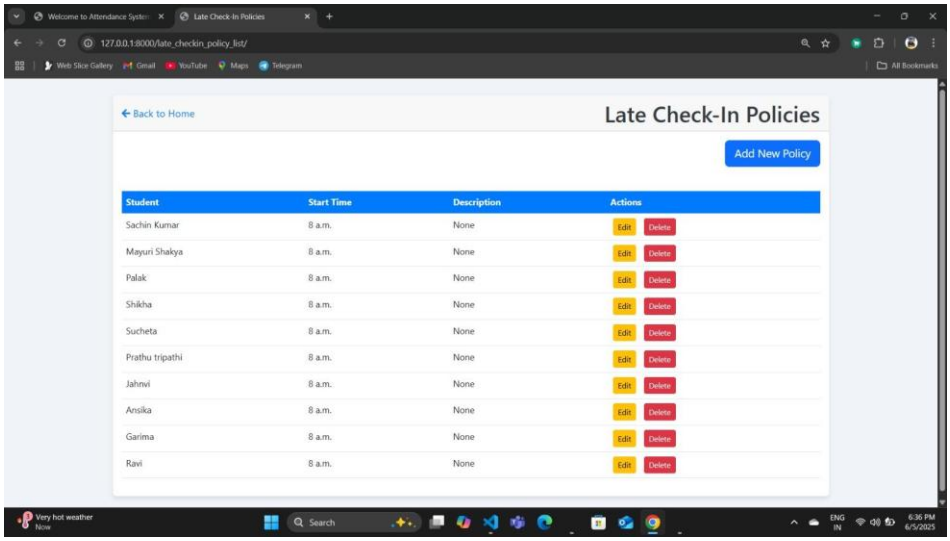


Figure B.6 Late Check-In Policies

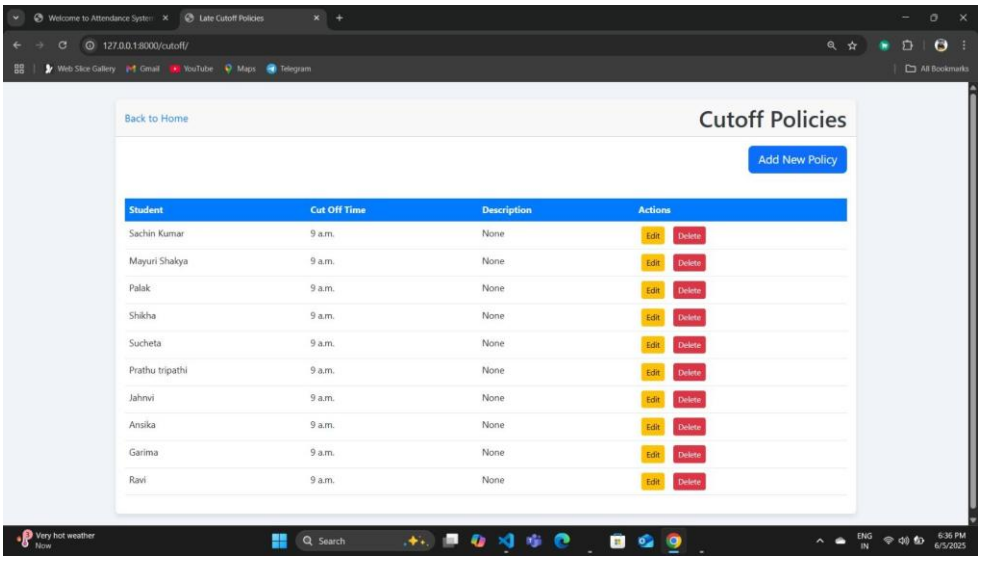


Figure B.7 Cutoff Policies

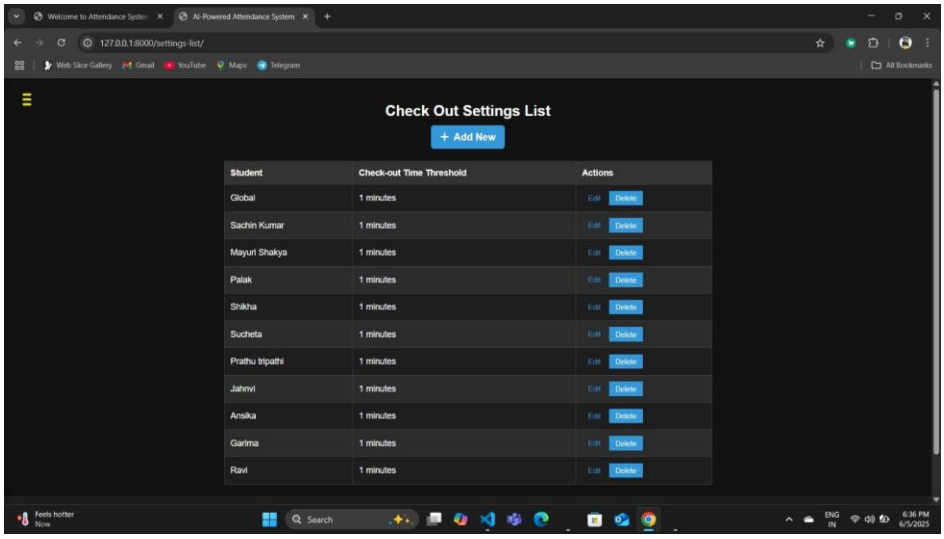


Figure B.8 Check Out Settings

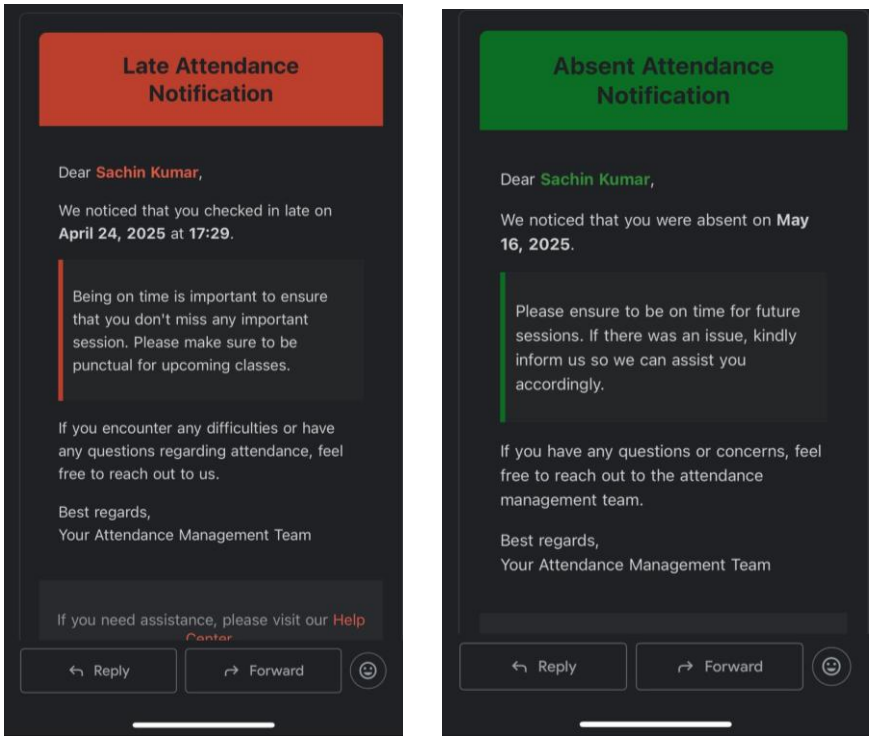


Figure B.9 Email Notifications

References

[1] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2020). "RetinaFace: Single-stage dense face localisation in the wild." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5203–5212.

[2] Guo, J., Liu, J., & Zhang, X. (2021). "SCRFD: A practical high-speed face detector." *arXiv preprint*, arXiv:2105.04714.

[3] Hengaju, S., Jadhav, D., & Patil, S. (2022). "Automated Face Recognition for Smart Attendance System Using RetinaFace and FaceNet." *International Journal of Research and Analytical Reviews (IJRAR)*, Vol. 9, No. 2, pp. 23–29.

[4] Liu, C., & Yu, C. (2022). "A Lightweight and Accurate Face Detection Algorithm Based on RetinaFace." *IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 352–356.

[5] InsightFace Developers. (2021). "InsightFace: An open-source 2D & 3D deep face analysis toolbox." GitHub repository. Available at: https://github.com/deepinsight/insightface

[6] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). "ArcFace: Additive Angular Margin Loss for Deep Face Recognition." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699.

[7] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks." *IEEE Signal Processing Letters*, Vol. 23, No. 10, pp. 1499–1503.

[8] OpenCV Developers. (2023). "OpenCV: Open Source Computer Vision Library." Available at: https://opencv.org/

[9] Python Software Foundation. (2023). "Python Language Reference, Version 3.9." Available at: https://www.python.org

[10] Django Software Foundation. (2023). “Django: The Web Framework for Perfectionists with Deadlines.” Available at: https://www.djangoproject.com/

[11] Scikit-learn Developers. (2023). “Scikit-learn: Machine Learning in Python.” Available at: https://scikit-learn.org/

[12] NumPy Developers. (2023). “NumPy: The Fundamental Package for Scientific Computing with Python.” Available at: https://numpy.org/

[13] Microsoft. (2023). “ONNX Runtime: Accelerated Machine Learning.” Available at: https://onnxruntime.ai/