iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78553

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Facial Recognition Attendance System Using Python and OpenCv

Yasharth Singh[1], Satyam Singh[2], Mr. Manoj Kujmar Dixit[3], Dr. Lakshita Agarwal[4]
[1, 2]*Student, Computer Science and Engineering, Galgotias College of Engineering and Technology, G.Noida, India*
Assistant Professor, Computer Science and Engineering, Galgotias College of Engineering and Technology, Greater Noida, India

Abstract: ***Face recognition technology has rapidly evolved into one of the most effective, unobtrusive methods for verifying personal identity, propelled by significant advancements in artificial intelligence and deep learning algorithms. Today's sophisticated systems go beyond basic facial recognition by integrating additional biometric modalities such as retina recognition and face mask detection, thereby enhancing accuracy, security, and robustness in diverse, real-world environments. Retina recognition leverages the uniquely intricate patterns of blood vessels in an individual's eye, providing a highly secure, virtually unforgeable layer of authentication. In response to global challenges like the COVID-19 pandemic, face mask detection—driven by convolutional neural networks (CNNs)— enables these systems to reliably identify individuals even when large portions of their faces are obscured by masks, ensuring continued functionality and safety. Furthermore, the integration of neural networks, 3D facial modeling, transfer learning techniques, and AI-based edge computing empowers modern systems to process vast amounts of visual data rapidly and efficiently. These innovations allow for performance that remains consistent despite variations in lighting, facial orientation, partial occlusions, or dynamic backgrounds. As a result, facial recognition systems have become more adaptable, scalable, and suitable for deployment in real-time applications where reliability and speed are critical. This paper explores the transformative impact of these advanced biometric technologies on contemporary face recognition solutions. It provides an in-depth examination of system architecture, the operational workflow, and real- time implementation strategies. Additionally, it discusses the broader implications for security, user privacy, and the future of identity management across multiple sectors, highlighting how these technologies are reshaping the landscape of secure authentication.***
Keywords: Smart Attendance System, NFC, RFID, OpenCV, NumPy

I. INTRODUCTION

Face detection is a computer technology that determines the locations and sizes of human faces in digital images. It detects facial features and ignores anything else, such as buildings, trees, and bodies [1-3]. In recent years, face recognition has attracted much attention and its research has rapidly expanded by not only engineers but also neuroscientists, since it has many potential applications in computer vision communication and automatic access control system. Especially, face detection is an important part of face recognition as the first step of automatic face recognition. However, face detection is not straightforward because it has lots of variations of image appearance, such as pose variation (front, non-front), occlusion, image orientation, illuminating condition, and facial expression [4,5].

A. Problem Statement and Motivation

Motivated by the efficiency and the facility of the occlusion removal approaches, we apply this strategy to discard the masked regions. Experimental results are carried out on Real-world Masked Face Recognition Dataset (RMFRD) and Simulated Masked Face Recognition Dataset (SMFRD) presented in [6]. We start by localizing the mask region. To do so, we apply a cropping filter in order to obtain only the informative regions of the masked face (i.e., forehead and eyes). Next, we describe the selected regions using a pre-trained deep learning model as a feature extractor. This strategy is more suitable in real-world applications comparing to restoration approaches. Recently, some works have applied supervised learning on the missing region to restore 123 Signal, Image and Video Processing (2022) 16:605–612 607 Fig. 1 Overview of the proposed method, such as in [7]. This strategy, however, is a difficult and highly time- consuming process.

Face recognition systems have emerged as a powerful solution for automated identity verification; however, they continue to face significant challenges under real-world conditions, including variations in illumination, pose, facial expressions, and occlusion caused by accessories such as glasses or masks [8], [9], [10]. These factors can significantly degrade system performance, especially in unconstrained environments.

©IJRASET: All Rights are Reserved

The COVID-19 pandemic further accelerated the demand for contactless biometric solutions, as traditional systems such as fingerprint scanners and RFID-based methods require physical interaction and pose hygiene risks [8], [12]. Consequently, face recognition has gained prominence due to its non-intrusive nature and ease of use. Recent advances in deep learning, particularly convolutional neural networks (CNNs), have dramatically improved the accuracy, robustness, and real-time performance of these systems, enabling reliable recognition even under challenging conditions [9], [13], [14].

Furthermore, modern face recognition systems are increasingly being integrated into smart environments, including corporate offices, educational institutions, and airports, to enhance security and streamline operations [10], [15].

Compared to traditional biometric techniques, facial recognition offers a seamless, scalable, and user-friendly solution for identity verification, making it a preferred choice in contemporary automated systems [10].

B. Research Objectives

To fix the issues with the old system mentioned in 1.1, we need to upgrade how things work. The new system cuts out paperwork—no more manual attendance sheets. It also speeds up the whole attendance process. Now, instead of writing names, the system uses facial recognition to track attendance, which makes the data much more accurate.

Here's what this project aims to do:

1) Build an AI-powered face recognition model that can spot people quickly and accurately—even when the environment isn't perfect.
2) Test how well the system works in different situations, like bad lighting, people turning their heads, changing facial expressions, or wearing stuff like masks and glasses.
3) Lower the chances of the system making mistakes (false positives or negatives) by using smarter deep learning techniques for face detection and mapping.
4) Create a fully automated, touchless system that barely needs anyone to operate it, making things easier for users.
5) Make sure the system runs fast and smoothly by analyzing how it uses computer resources and improving its recognition speed with advanced machine learning and better model design.
6) Handle facial data securely—protecting privacy and blocking unauthorized access.
7) Check if the system can adapt to different uses: not just attendance, but also access control and surveillance.
8) Compare different recognition models (like CNN, OpenCV, Dlib, and Face Net) to find out which one gives the best results in terms of accuracy and speed.
9) Make the system scalable, so it can handle more users as needed without slowing down.
10) Explore ways to connect the face recognition module with cloud tech, IoT devices, or live monitoring systems for even more features.

C. Project Scope and Direction

The big goal here is to get rid of the old, clunky ways of verifying identity and swap them out for a smarter, automated face recognition system. We want an app that spots and identifies faces in real time, then safely stores that info in a database. This approach should boost accuracy, cut down on manual work, and let people check in without touching anything. Depending on how it's used, the system can also whip up reports automatically and send results or logs to the right people.

Here's what the project covers:

1) The main users are students, staff, and any authorized personnel.
2) The database can store facial data for up to 2,000 people, which is enough for most medium-sized institutions. - For now, the face recognition checks one person at a time to keep things accurate and stable. The system can generate reports from the recognition logs, and these can be emailed to the right authorities for tracking or record-keeping. - The system needs a live internet connection (through Wi-Fi or Ethernet) to keep data synced and updated.
3) The hardware runs on a power bank, so you can move it around easily and keep it running even if there's no socket nearby.

D. Impact, Significance, and Contributions

Most systems for monitoring and verifying identities are slow, manual, and not that secure. This project changes that by bringing in AI to make the whole process faster and more reliable. With face recognition, you don't need someone watching over everything, and it's much harder for someone to fake their identity. H

ere's what the project brings to the table:

1) The system encourages discipline and accountability—no more fake attendance or impersonation, since everyone's identity is locked to their face.
2) It saves time and reduces the need for extra staff, since the system handles authentication electronically.
3) You can run the app on different devices and in different locations, as long as there's a decent internet connection, which makes it really flexible.
4) No more paper—attendance and records go digital, cutting down on costs and waste.
5) The system is fast. It recognizes, logs, and reports automatically, so there's no waiting around or extra work for staff.
6) Bottom line: this project tackles the problems with old systems and brings in a new era of speed, security, and convenience with fully automated face recognition.

Modern identity verification and attendance monitoring systems are often characterized by manual processes, inefficiencies, and limited security. These traditional approaches are prone to human error and misuse, such as impersonation or proxy attendance. The integration of artificial intelligence, particularly face recognition technology, offers a more efficient and secure alternative by enabling automated and reliable identity authentication [16], [17].

Such systems enhance discipline and accountability by linking attendance directly to an individual's facial identity, thereby reducing fraudulent practices. Additionally, automation minimizes the need for manual supervision, saving time and reducing operational costs. The digital nature of these systems eliminates paper-based records, promoting sustainability while improving data management. Furthermore, face recognition systems provide rapid processing and real-time reporting, making them suitable for large-scale deployments in various environments [16]. Overall, AI- based face recognition systems significantly improve efficiency, security, and convenience compared to conventional methods.

E. Historical Development Prior to the Project

Historically, identity verification and attendance tracking relied on manual methods such as paper registers and punch card systems. These approaches were time-consuming, error-prone, and vulnerable to manipulation. With technological advancements, smart cards and RFID-based systems were introduced, offering improved automation. However, these systems still faced limitations, including card loss, damage, and misuse through unauthorized sharing. These challenges highlighted the need for more secure and reliable biometric solutions, ultimately leading to the adoption of face recognition technologies [16], [18].

II. LITERATURE REVIEW

Smart Attendance System Using Face Recognition and Mobile Devices

Sharma and Gupta (2019) proposed a mobile-based attendance system utilizing face recognition through smartphone cameras. The system allows users to register facial data and mark attendance via image capture, with data stored on a centralized server. This approach eliminates the need for physical cards and supports contactless authentication. However, limitations include dependency on device performance, privacy concerns, and reduced efficiency when handling large groups [20].

Automated Attendance Marking Using AI-Based Face Recognition

Patel et al. (2020) developed an automated system that captures images at entry points and matches them with a stored database to mark attendance. The system ensures accuracy and reduces human intervention by storing records in a secure database. Despite improved detection accuracy, the system's reliance on fixed hardware limits portability. Integration with lightweight AI models on portable devices can overcome this limitation [21].

Face Recognition Using an Embedded System for Attendance Management

Prajapati and Khan (2021) implemented face recognition on embedded systems, enabling on- device processing using deep learning techniques such as CNNs. This approach improves speed and reduces dependency on cloud infrastructure. However, the lack of remote accessibility presents challenges for monitoring and data management, which can be addressed through cloud integration [22].

Biometric-Based Attendance Using Face Recognition and Web Integration

Deshmukh and Reddy (2022) introduced a web-integrated face recognition system that provides real-time attendance tracking and reporting through an online dashboard. While the system enhances accessibility and security, it requires stable network connectivity and high-quality imaging systems for optimal performance, especially in crowded environments [23].

A. Literature

In conclusion, a better attendance monitoring system should be developed based on its portability, accessibility and the accuracy of the collected attendance information.

After analyzing existing research on attendance monitoring technologies, it is observed that several approaches like RFID, NFC tags, and fingerprint-based systems improve automation but still suffer from limitations such as portability issues, contact-based authentication, and possible proxy attempts. Recent advancements in artificial intelligence and computer vision have encouraged the use of face recognition as a reliable biometric authentication method. Unlike traditional systems, facial recognition is contactless, faster, and more secure, making it suitable for educational and corporate environments. Studies indicate that systems built using machine learning frameworks can recognize individuals accurately even under varying lighting and orientation conditions. Furthermore, integration with cloud-based services helps in retrieving attendance data remotely. In conclusion, past research highlights the need for an advanced attendance monitoring system that is portable, easily accessible, and provides high accuracy while eliminating identity fraud and manual dependence.

III. SYSTEM DESIGN

The design part of the attendance monitoring system is divided into two sections, which consist of the hardware and the software part. Before the software The design part can be developed, the hardware part is first completed to provide a platform for the software to work. Before the software part, we need to install some libraries for effective working of the application. We install OpenCV and NumPy through Python. The system design phase is divided into two main components: hardware setup and software development. The hardware provides the physical framework for data capture, while the software processes it using artificial intelligence-based algorithms. Initially, required libraries and dependencies must be installed to enable face recognition functionality.

The primary libraries used in this project are OpenCV for image processing and NumPy for numerical computations. These are installed using Python, which serves as the core programming platform for system implementation.

A. Hardware Development

The hardware component consists of:

A camera module with sufficient resolution to ensure clear face image capture.

A computing device (PC, laptop, or embedded module such as Raspberry Pi) to process data. A stable power source or portable battery backup, ensuring uninterrupted functioning.

Wi-Fi or Ethernet support to continuously update the database, if connected to remote storage.

B. Software Development

The software design includes AI-based face detection and recognition using Python programming language. The following major libraries and tools are used:

1) OpenCV

OpenCV is this open-source library that's become a go-to for computer vision, machine learning, and image processing. Intel kicked things off back in 1999, but now the OpenCV Foundation keeps it moving. The whole idea is simple: help computers actually see and make sense of visual stuff like photos and videos, right as it happens. OpenCV's packed with a ton of fast, ready-to-go algorithms for things like spotting objects, recognizing faces, tracking movement, filtering images, and pulling out key features. That speed, plus the fact that it works pretty much anywhere, is why you see it everywhere—from university labs to huge industry projects and even the latest AI tools.You can use OpenCV with a bunch of languages—C++, Python, Java—but honestly, most people lean toward Python. It's easy to pick up, and there's just so much library support out there. Inside OpenCV, you'll find different modules: Core handles the basics like data and math, Imgproc takes care of image processing, High GUI lets you display images and videos, Video is all about video analysis, Object does the heavy lifting for object detection, and DNN covers deep neural networks. With these, you can put together anything from a quick photo editor to some cutting-edge AI system.

OpenCV (Open Source Computer Vision Library) is a computer vision framework widely used in AI applications, especially for face detection and tracking. It supports modern machine learning models and is optimized for real- time processing. It provides functions for image capture, preprocessing, face detection, and recognition. OpenCV supports various programming languages such as Python, C++, Java, and MATLAB, and works across multiple platforms including Windows, Linux, macOS, Android, and iOS. It

is compatible with CMake-based development. The primary purpose of OpenCV is to enable computers to understand and analyze visual data such as images and videos in real time. It provides a vast collection of optimized algorithms that can perform tasks like object detection, face recognition, motion tracking, image filtering, and feature extraction.
Because of its efficiency and cross-platform compatibility, OpenCV is widely used in academic research, industry applications, and AI-based projects.

2) NumPy

NumPy is a scientific computing library used for handling high-speed mathematical operations on multi-dimensional arrays. It is essential for matrix manipulation and enhances the performance of embedded AI algorithms. Functions for statistical analysis, linear algebra, and random number generation further support model development. OpenCV's application areas include: • Face detection and recognition

- Object detection and tracking
- Surveillance and security systems
- Self-driving cars (lane and vehicle detection)
- Medical image processing
- Augmented and Virtual Reality (AR/VR)
- Gesture and motion recognition
- Industrial quality inspection
- OCR (Optical Character Recognition)
- License plate recognition
- Robot vision and automation
- Sports performance tracking
- Satellite and drone image analysis
- Image enhancement and restoration
- Human-computer interaction systems

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- K-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest

3) Versions of OpenCV

OpenCV has come a long way since it first hit the scene. The very first demo showed up at a big computer vision conference in 2000. After a few years of tweaks and upgrades, the first stable version landed in 2006, with a cleaner prerelease in 2008. Then came OpenCV 2 in 2009—a big deal because it introduced a new C++ interface, making everything smoother and faster, especially on multi-core machines. That really boosted its ability to handle real-time face detection. Since then, OpenCV's been on a steady upgrade path, shaped by developers all over the globe. In 2012, OpenCV.org took over, keeping the updates and community alive. Then, in 2016, Intel stepped in, bringing even better hardware acceleration and tighter computer vision performance—especially handy for face recognition.

4) Supported Programming Languages

OpenCV is flexible when it comes to programming languages. Most of the core work happens in C++ and Python, and those are the main choices for face recognition. But you can also work with Java and MATLAB. Thanks to the open- source crowd, there are wrappers for C#, Perl, Ruby, and a bunch of others, so you're not boxed in. Since version 3.4, there's even OpenCV.js. It lets you run select OpenCV features right in the browser, which opens the door for web- based face recognition.

5) Operating System Compatibility

OpenCV's face recognition tools work on all the big platforms—Windows, macOS, Linux, and BSD. Mobile devices? No problem. It runs on Android and iOS, so you can build portable apps, too. Stable releases are easy to grab from trusted sites, and if you want the latest updates, just head to GitHub. CMake handles the heavy lifting for cross- platform builds.

6) NumPy

NumPy is at the heart of AI-powered face recognition. It gives you fast, multi-dimensional arrays and all the advanced math you need for image and data processing. Whether you're prepping images, crunching matrices, or running stats for computer vision, NumPy keeps everything running smoothly.

IV. SOFTWARE DEVELOPMENT

There are two major system flows in the software development section as shown below:

- The creation of the face database
- The process of attendance taking

Both processes mentioned above are essential because they made up the backbone of the attendance management system. In this section, the process of both flows will be briefly described. Meanwhile, their full functionality, specific requirements and also the methods/approach to accomplish such objectives will be discussed in the upcoming chapter:

1) Creation of the Face Database: This process involves capturing individual face images and converting them into a structured dataset that the system can use for training and identification. The development begins with image acquisition through a high-resolution camera, followed by face detection using machine learning-based classifiers. The detected facial region undergoes necessary pre-processing, including resizing, normalization, and noise reduction, to standardize input quality. After preprocessing, the images are labeled and stored systematically, often in a CSV or similar structured format. This dataset is then used to train the facial recognition model (e.g., LBPH, Eigenface, or CNN-based recognizers). Once training is complete, the model generates trained files containing encoded facial features, which are saved for future recognition tasks. The accuracy of this phase determines the efficiency of attendance detection.

2) Process of Attendance Taking: The attendance process is triggered in real-time when a user presents their face to the camera. The system immediately performs face detection, followed by recognition through comparison with the trained model. If the identity matches an entry in the database, the attendance is marked and stored in the system. This process takes only a few seconds and eliminates manual involvement, ensuring punctuality and authenticity.

A. The creation of the face database

Building a face database system is a key part of making a face recognition app work. Basically, you need a place to store all those face photos and the details that go with each one, so the system can figure out who's who later.

It all starts with gathering data. You snap several photos of each person or pull them from existing collections. To make the system more reliable, you grab images in all sorts of conditions—different lights, angles, even expressions.

Once you've got the photos, you clean them up. This means turning them into grayscale, resizing them to the same size, filtering out noise, and then running an algorithm—like Haar Cascade or a deep learning face detector—to actually find the face in each picture. You crop and normalize everything so the faces look consistent across the board. Getting this part right is crucial; the more uniform the images, the better the recognition works.

Next comes feature extraction. Older systems use methods like Local Binary Patterns or Eigenfaces, but most modern setups lean on deep learning, especially Convolutional Neural Networks. These models create what's called facial embeddings—basically, a set of numbers that represent the unique features of each face. In advanced systems, you save these embeddings instead of the raw images[24].

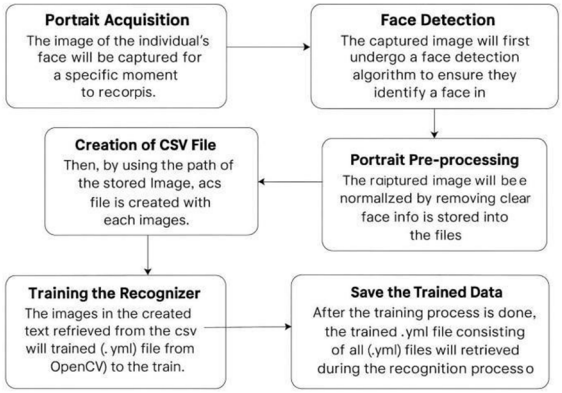


Figure 1. Block diagram representing the step-by-step pipeline of a face recognition system, from image capture and preprocessing to model training and database creation.(Generated from OpenAI ChatGPT)

The creation of the face database is a vital step in the development of the system, as it serves as the reference source for identity matching during the recognition phase. For each individual, multiple facial images are captured to ensure accuracy and improve the system's ability to recognize faces under various conditions, such as different angles, lighting, or facial expressions. These images are then processed and organized using a structured file format, typically a CSV file, which stores image paths along with unique labels. The labeling system plays an important role in grouping multiple images belonging to the same individual, allowing the recognition module to associate them with one distinct identity.

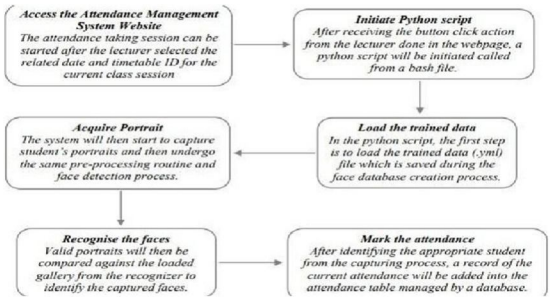


Figure 2. Structural overview of a face recognition system showing the interaction between acquisition, processing, and training components.(Adapted from [25])

V. METHODOLOGY

Here's how the face recognition-based attendance system works. First, it collects key info for each person—like their unique ID number and facial photos snapped with a camera. The system checks every photo with a face detection algorithm to make sure there's actually a face in the shot. If it doesn't see one, or the face isn't clear enough, it asks the user to take the photo again. This keeps going until it has the number of good portraits it needs. For this project, that means ten clear pictures per person, which keeps the recognition accurate without overloading the limited storage space.

A. Image Acquisition and Pre-processing procedures

After the images are being processed, they are stored into a file in a hierarchy manner. In this project, all the faces will be stored in a hierarchy manner under the database‟ folder. When expanding through the

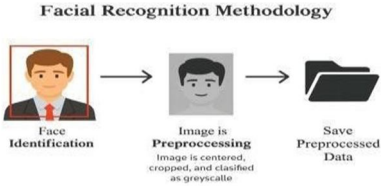


Figure 3. Conceptual diagram of facial recognition methodology of raw input to data (Adapted from [26])

series of face portrait belonging to the same individual will be stored in that particular sub-folder. The sub- folders that represent each individual will be named upon the ID no. of that individual which is unique for every single individual in the institution. The whole process of image retrieval, pre-processing, storing mechanism is done by the script named create_database.py[25].

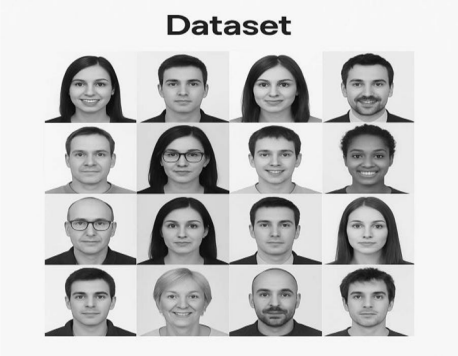


Figure 4. Sample facial dataset comprising diverse individuals used for training and evaluation of the recognition system..(Generated from OpenAI ChatGPT)

B. Hierarchy manner of the face database

After a successful retrieval of facial images into the respective folder, a CSV files created to aid the next process of pumping the faces into the recognizer for the training process. The creation of the CSV file will be done based on a script named create_csv.py. In this project, the content of CSV file will look like the following format:

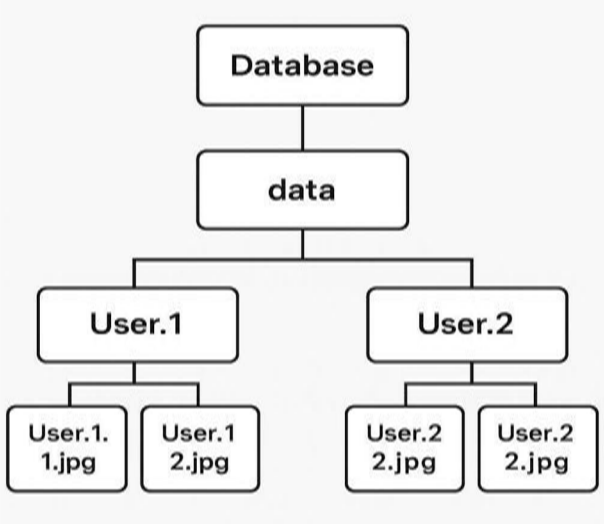


Figure 5. Hierarchical organization of the face image database showing user-wise storage of labeled images.(Generated from OpenAI ChatGPT)

C. Structure of the content in the CSV file

After you pull all the face images into their folders, the next step is to get them ready for training. To do this, the system creates a CSV file using a script called create_csv.py. This CSV acts as a roadmap for the recognizer, helping it match every image to the right person. Here's how the CSV is set up: each row links a single facial image to the student it belongs to. You'll see two main columns—one for the image's file path and another for the student's label or ID. The file path tells the system exactly where to find the photo inside the database, which is crucial for both training and recognition. The label column holds the unique identifier for each student, whether that's a simple number or an alphanumeric code. This setup is important because every student usually has several photos, and you need all those images tied to one identity. The CSV keeps everything organized and makes sure the training process runs smoothly, loading the right data for each person every time.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

VI. FLOW CHART OF THE IMAGE ACQUISITION PROCESS

Here's how the face recognition attendance system works, step by step. It kicks off when the camera turns on and snaps a live photo of the person in front of it. Right after, the system checks if there's actually a face in the picture. If it doesn't spot one, it asks the user to try again until it gets a clear shot. When the system finally sees a face, it gets the image ready for recognition — this means turning it to grayscale, resizing, and cropping so only the face is left. Next, the system compares this processed image to its database using a recognition algorithm. If it finds a match, it marks the person as present and logs all the important details like time, date, and user ID. Some versions of the system can even whip up an attendance report and send it to the right teacher or just save it for later.But if the face doesn't match anyone in the database, the system either shows an error or asks for another photo. This whole process keeps things accurate, cuts down on manual work, and makes attendance fast and secure.

In a face recognition system, image acquisition is the process of capturing a person's face using a camera or webcam. The captured image is converted into digital form and stored as pixel data.

This digital face image is then used for further steps like face detection, feature extraction, and matching with stored face data.

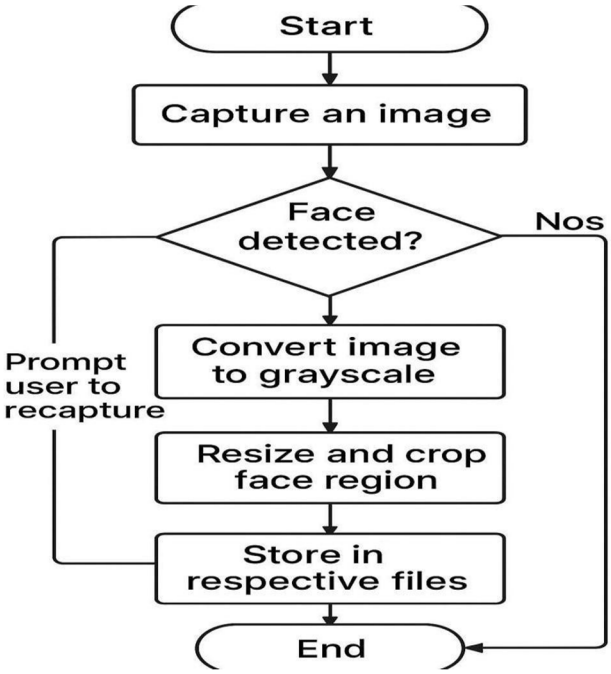


Figure 6. Flowchart illustrating the image acquisition and preprocessing steps, including detection, grayscale conversion, cropping, and storage.(Generated from OpenAI ChatGPT)

D. Flow Chart of the image retrieval process

The above flowchart is only the program flow for the image acquisition process, which describes the program flow for the script create_database.py. There are two more Python scripts that are responsible for the remaining execution, which will be explained in the next sub-section.

VII. FILES INCLUDED

The face recognition attendance system runs on a handful of important files and scripts. You've got five Python scripts, a bash file, a CSV, a trained model file (.yml), and a storage folder. Everything works together for building the dataset, training the model, face recognition, and generating reports. Three of those Python scripts run through the bash file, so you don't have to mess around with manual setup. Just run the bash file, and it'll handle the environment setup and kick off the scripts in the right order. This setup keeps things simple — you don't need to worry about registering new faces or retraining the model by hand. The automation keeps your database and recognition model up to date.

Here's what's included: Python scripts: firstpage.py, dataset_capture.py, recognizer.py, training_dataset.py, mail.py. Bash file: prepare_gallery.sh (in /usr/local/bin/).CSV file: Stores image paths and labels. YML file:trainer – holds the trained model data. Folder: data – where captured face images are stored.[25,26]

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

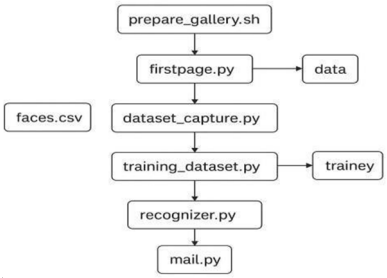


Figure 7. Structural overview of the system module and scripts involved in data training and recognition. (Adapted from [25],[26])

VIII. OVERVIEW

The proposed system is a software-based attendance solution that utilizes facial recognition technology to automatically mark student attendance. Developed using Python integrated with the OpenCV module, the system enables educational institutions to streamline attendance procedures efficiently and accurately. The hardware components required for implementation include a computer system, high-definition camera for image capture, and an active internet or Wi-Fi connection to support data transfer and email functionalities.

A. Steps of Working

1) Start by running the firstpage.py script.
2) When adding a new student, just enter their ID. The system creates a dataset for them.
3) Next, train that dataset. The system generates a .yml file for facial data.
4) Take a class photo, then run the recognizer script.
5) The system crops out each face from the picture and checks them against the database.
6) If someone matches, their name shows up as PRESENT in an Excel file, along with the date and time.

IX. RESULTS

The proposed face recognition-based attendance system was successfully developed and tested under various real- world conditions to evaluate its performance, accuracy, and efficiency. The system demonstrated reliable operation in detecting, recognizing, and recording attendance with minimal human intervention.

The face detection module was able to accurately identify faces from live camera input as well as stored images. Preprocessing techniques such as grayscale conversion and resizing improved the consistency of input data, leading to better recognition performance. The trained model showed high accuracy in identifying registered users from the dataset.

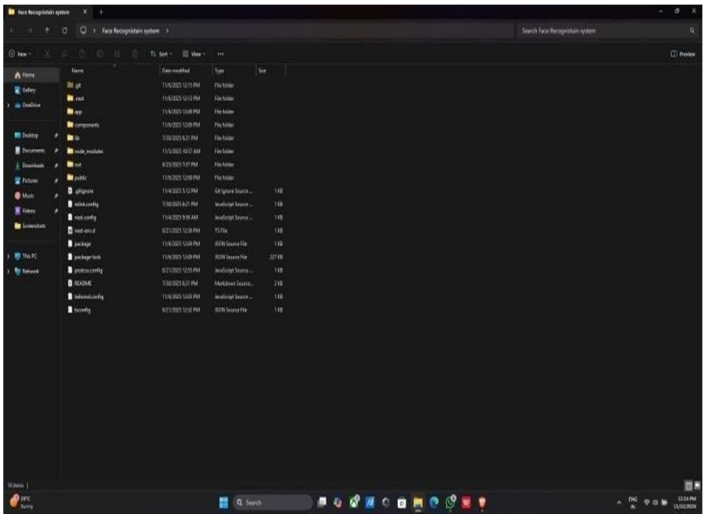


Fig 8.1 – Contents of the Project

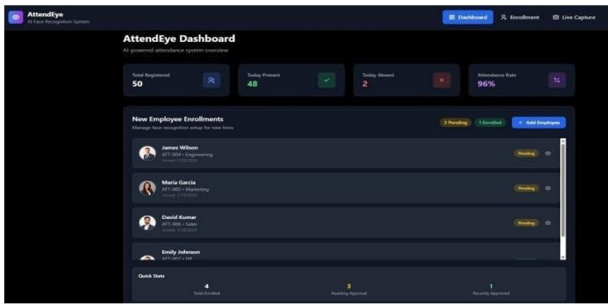


Fig 8.2 – First page of the project (Attend Eye Dashboard)

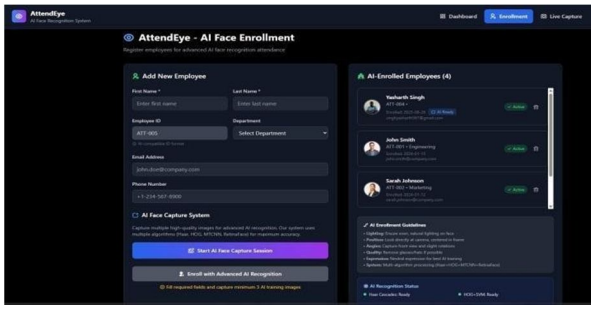


Fig 8.3 – AI Based Face Enrollment System(Attend Eye)

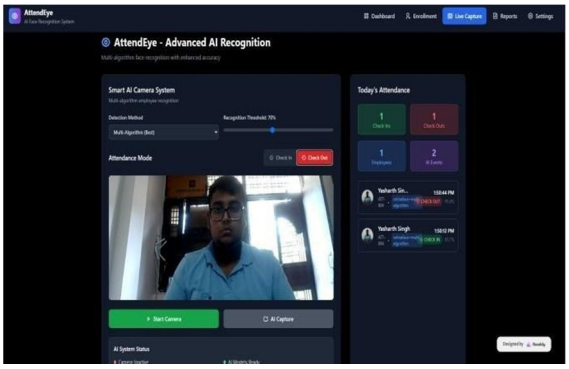


Fig 8.4 – Smart AI Capture System

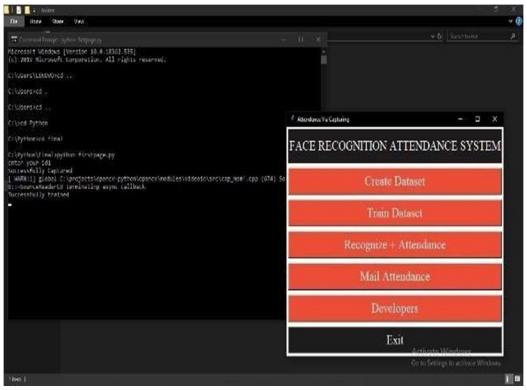


Fig 8.5 – Table for Creation of Dataset

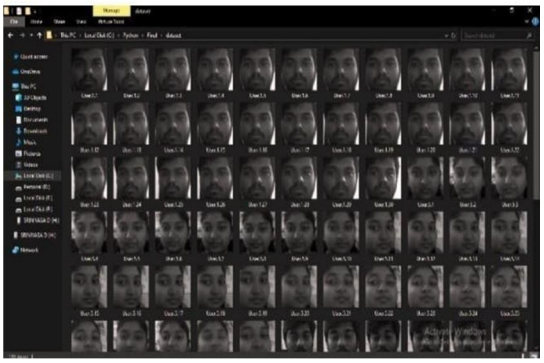


Fig 8.6 – Training dataset of the image

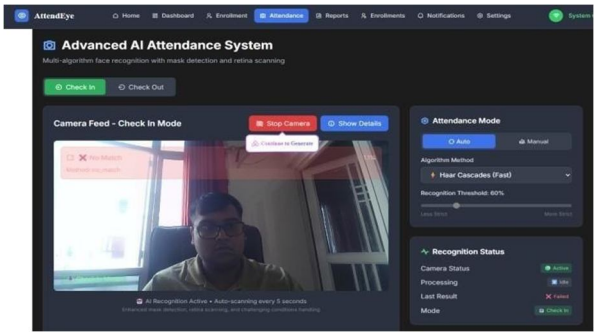


Fig 8.7 – Camera feed (Check in Mode)

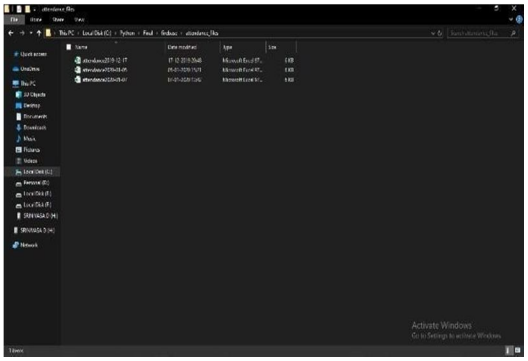


Fig 8.8 – Storing files of attendance files

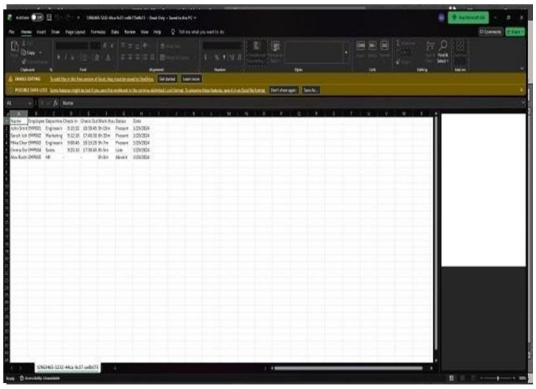


Fig 8.9 – Display of Attendance of Employees

©IJRASET: All Rights are Reserved

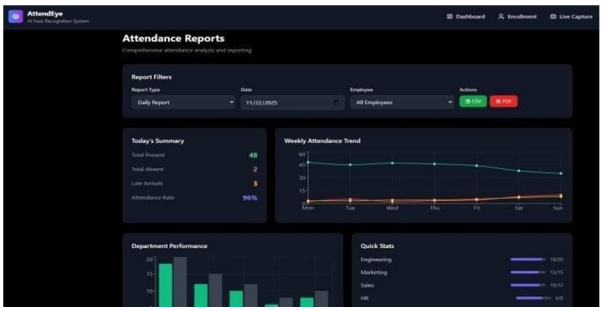


Fig 8.10 – Summary of Weekly Reports of Employees

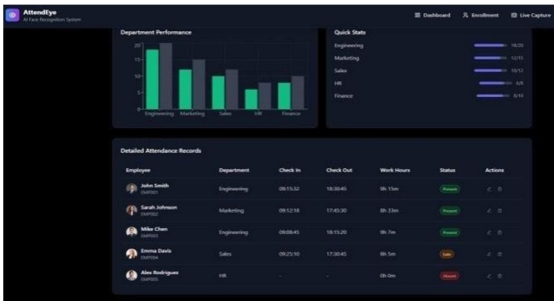


Fig 8.11 – Histogram-based analysis of the reports of Employees



Fig 8.12 – Attendance Status of the Employees

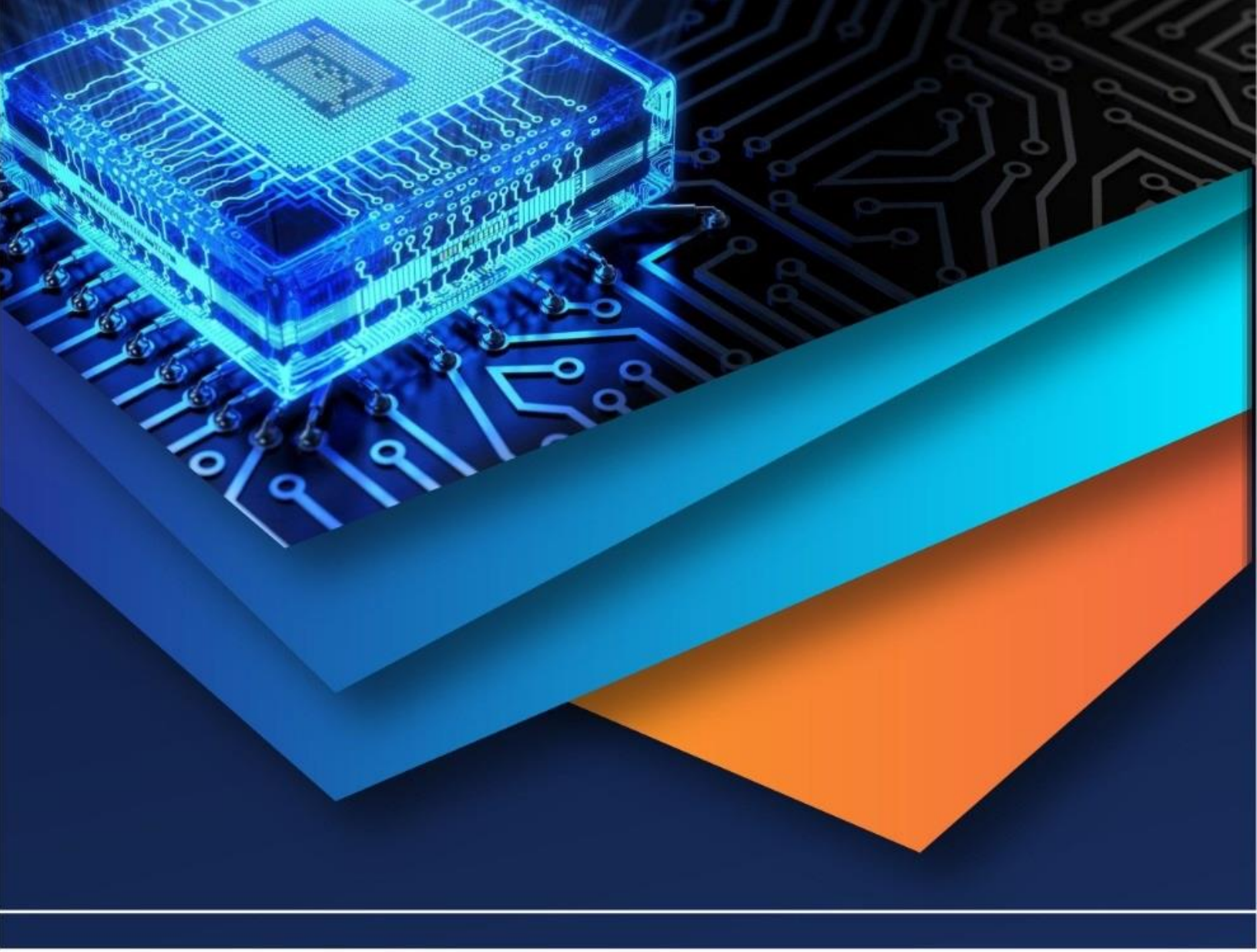
X. CONCLUSION

Traditional attendance systems have long struggled with a number of persistent issues, such as inaccurate records due to missed check-ins, time-consuming manual processes, and students exploiting loopholes like having peers sign in for them. These inefficiencies not only disrupted the flow of classroom activities but also made accurate tracking a challenge for educational institutions. Recognizing these drawbacks, we decided to overhaul the process by integrating facial recognition technology, which brought about a noticeable transformation almost immediately. By harnessing advanced image processing algorithms, the new system is capable of identifying each individual student with a high degree of precision. This automation significantly reduces the need for constant human supervision, allowing teachers and administrative staff to focus on more meaningful tasks rather than spending time on repetitive attendance taking. The entire process is streamlined, ensuring that attendance is recorded quickly and accurately, virtually eliminating the possibility of proxy sign-ins. Furthermore, the implementation of this technology has led to measurable gains in efficiency and resource management. Digital storage of facial images on micro-SD cards ensures that the database is not only well-organized but also easily accessible for future reference or audits. The modularity and scalability of this setup mean that the system can be updated or expanded as needed without major overhauls.

During the development phase, the face database was meticulously configured, and the recognition module underwent extensive testing across various scenarios. These rigorous trials consistently demonstrated the system's reliability and robustness, reinforcing our confidence in its performance. Beyond simply solving the initial problems, the face recognition attendance system introduces additional benefits that further enhance its value. For instance, it automates the generation of attendance reports, saving considerable administrative effort. These reports are not only comprehensive but are also delivered in real time directly to faculty via email, enabling immediate access to up-to- date attendance information. This feature supports better decision-making and allows for timely interventions if attendance issues arise.

REFERENCES

[1] Y. Ming-Hsuan, K. David and A. Narendra, "Detecting Faces in Images: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, 2002, pp. 34-58. http://dx.doi.org/10.1109/34.982883
[2] I. Kim, J. Hyung Shim and J. Yang, "Face Detection," Stanford University, 2010. http://www.stanford.edu/class/ee368/Project_03/Project/reports/ee368group02.pdf
[3] M. A. El-Sayed and N. Aboelwafa, "Study of Face Recognition Approach Based on Similarity Measures," International Journal of Computer Science Issues (IJCSI), Vol. 9, No. 2, 2012, pp. 133-139.
[4] M. A. El-Sayed and M. A. Khafagy, "An Identification System Using Eye Detection Based on Wavelets and Neural Networks," International Journal of Computer and Information Technology, Vol. 1, No. 2, 2012, pp. 43- 48.
[5] M. A. El-Sayed, "Edges Detection Based on Renyi Entropy with Split/Merge," Computer Engineering and Intelligent Systems (CEIS), Vol. 3, No. 9, 2012, pp. 32-41.
[6] Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., Yi, P., Jiang, K., Wang, N., Pei, Y., et al.: Masked face recognition dataset and application. arXiv preprint arXiv:2003.09093 (2020)
[7] Din, N.U., Javed, K., Bae, S., Yi, J.: A novel gan-based network for unmasking of masked face. IEEE Access 8, 44276–44287 (2020).
[8] W. Hariri, "Efficient Masked Face Recognition Method during COVID-19," 2021.
[9] F. Schroff, D. Kalenichenko, and J. Philbin,"FaceNet: A Unified Embedding for Face Recognition and Clustering," Proc. IEEE CVPR, 2015.
[10] A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," IEEE Trans. Circuits Syst. Video Technol., 2004.
[11] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," ACM Computing Surveys, 2003.
[12] R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha, and A. W. Senior, "Guide to Biometrics," Springer, 2004.
[13] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," British Machine Vision Conf., 2015.
[14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," Proc. IEEE CVPR, 2014.
[15] S. Z. Li and A. K. Jain, "Handbook of Face Recognition," Springer, 2011.
[16] A. K. Jain, A. Ross, and S. Prabhakar, - "An Introduction to Biometric Recognition," IEEE, 2004.
[17] F. Schroff, D. Kalenichenko, and J. Philbin, "Face Net: A Unified Embedding for Face Recognition," CVPR, 2015.
[18] Y. Taigman et al., "Deep Face: Closing the Gap to Human-Level Performance," CVPR, 2014.
[19] R. M. Bolle et al., "Guide to Biometrics," Springer, 2004.
[20] Sharma and Gupta, "Smart Attendance System Using Face Recognition and Mobile Devices," 2019.
[21] Patel, Sahu, and Mehta, "Automated Attendance Marking Using AI-Based Face Recognition," 2020.
[22] Prajapati and Khan, "Face Recognition Using Embedded System," 2021.
[23] Deshmukh and Reddy, "Biometric-Based Attendance Using Face Recognition and Web Integration," 2022.
[24] FACIAL RECOGNITION-BASED ATTENDANCE ... UTAR Institutional Repositoryhttp://eprints.utar.edu.my ›...UTAR Institutional Repositoryhttp://eprints.utar.edu.my ›
[25] Web resultsAcademia.eduhttps://www.academia.edu(PDF) Facial Recognition Attendance System Using Python and OpenCV
[26] https://www.neuroquantology.com/media/article_pdfs/Neuroquantology_August_2020_An_analytical_Research_ o n.pdf



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)