REAL-TIME FACE ATTENDANCE SYSTEM USING COMPUTER VISION

Submitted in partial fulfillment of the requirement for the degree of

Master of Computer Application

By

AKASH NAUTIYAL

(Enrollment Number- PV-22010042)

MCA IV SEM

Under the Guidance of

Mr. Neeraj Panwar

Associate Professor

DEPARTMENT OF COMPUTER APPLICATION

GRAPHIC ERA HILL UNIVERSITY

(2022-2024)

CERTIFICATE

This is to certify that the thesis titled **"Real-Time Face Attendance System Using Computer Vision" submitted** by **Akash Nautiyal**, to Graphic Era Hill University for the award of the degree of **Master of Computer Application**, is a bona fide recordof the research work done by him/her under our supervision. The contents of this project in full or in parts have not been submitted to any other Institute or University for the award of any degree or diploma.

Mr. Neeraj Panwar
Project Guide
(Assistant Professor)
GEHU, Dehradun

Place: Dehradun
Date:

ACKNOWLEDGEMENT

This Project report was completed as a result of support from my mentor, I am greatly indebted to my good supervisor **Mr. Neeraj Panwar** for his useful and necessary observation, suggestions, contribution and corrections. I would not have been able to achieve anything in this project without your supervision.

Akash Nautiyal
University Roll No. - 2201042

Certificate Number : ICCICCT-432_1

ICCICCT

KSRCE - Department of Computer Science and Engineering

2nd International Conference on Challenges in Information, Communication and Computing Technology (ICCICCT - 2024)

KSR College of Engineering | Taylor & Francis Taylor & Francis Group | Indexed by Scopus

April 26th and 27th, 2024

CERTIFICATE OF PRESENTATION

This is to Certify that

AKASH NAUTIYAL

GRAPHIC ERA HILL UNIVERSITY

has successfully presented the paper entitled

FACIAL RECOGNITION ATTENDANCE SYSTEM: EMBRACING OPENCV FOR REAL-TIME MONITORING

in 2nd International Conference on Challenges in Information, Communication and Computing Technology (ICCICCT - 2024), Organized by Department of Computer Science and Engineering at K.S.R. College of Engineering, Tiruchengode, Namakkal, Tamil Nadu, India.

Dr. V. SHARMILA	Dr. A. RAJIV KANNAN	Dr. P. SENTHIL KUMAR
Convener	Conference Chair	Patron

INDEX

ABSTRACT

A face recognition attendance system is a biometric technology that uses artificial intelligence to identify and verify people based on their facial characteristics. It's a fast, high-accuracy system that's used in various sectors, including finance, retail, government, and industry.

Attendance systems have evolved significantly over the years, with the advent of technology providing innovative solutions to streamline the process. Facial recognition has emerged as a promising technology in this domain due to its accuracy, efficiency, and non-intrusiveness.

This paper presents an efficient real-time attendance system employing facial recognition technology. Traditional attendance systems often suffer from inaccuracies, buddy punching, and time-consuming manual processes. Leveraging advancements in computer vision and machine learning, the proposed system aims to address these challenges by automating attendance management through facial recognition.

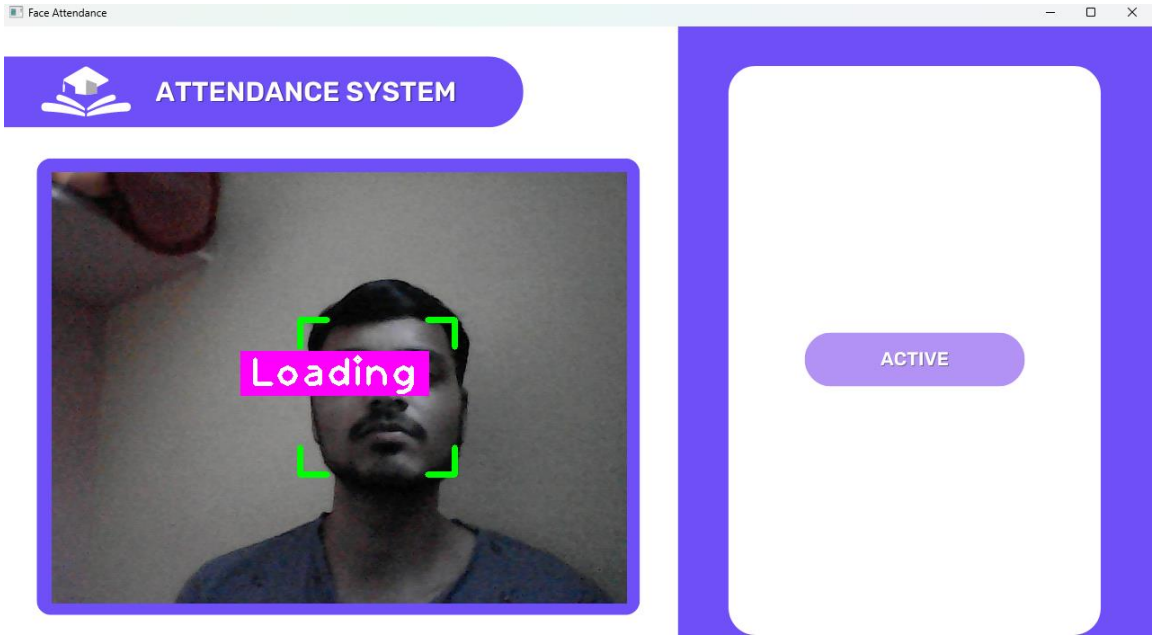
The system operates by capturing facial images using a camera and employing Machine learning algorithms to detect, extract, and analyze facial features. Utilizing Machine Learning Algorithm, we develop Face encoding of the captured images and then compare it with encoding present in database of the stored images. The system's architecture encompasses both hardware and software components, including camera modules, processing units, and a centralized database.

The proposed system has been evaluated through extensive testing and validation, demonstrating promising results in terms of accuracy, speed, and reliability. Furthermore, the system's adaptability and scalability make it suitable for various industries, including education, corporate environments, and public institutions.

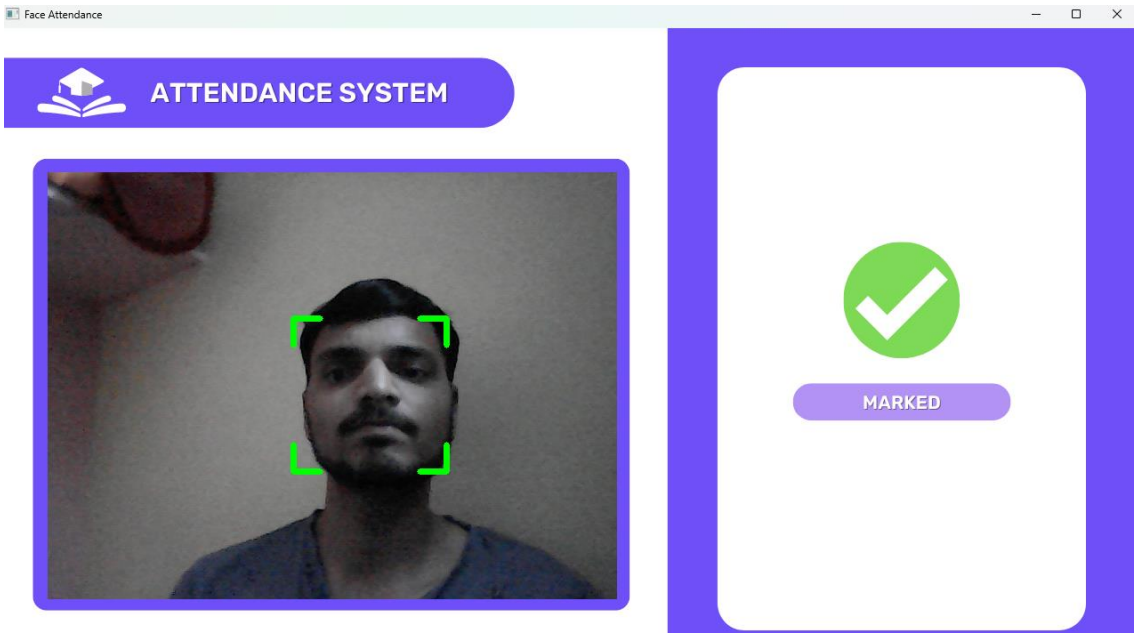
In conclusion, the real-time attendance system utilizing facial recognition technology presents a significant advancement in attendance management, offering a reliable, efficient, and secure solution for organizations seeking to streamline their attendance tracking processes.

LIST OF FIGURES

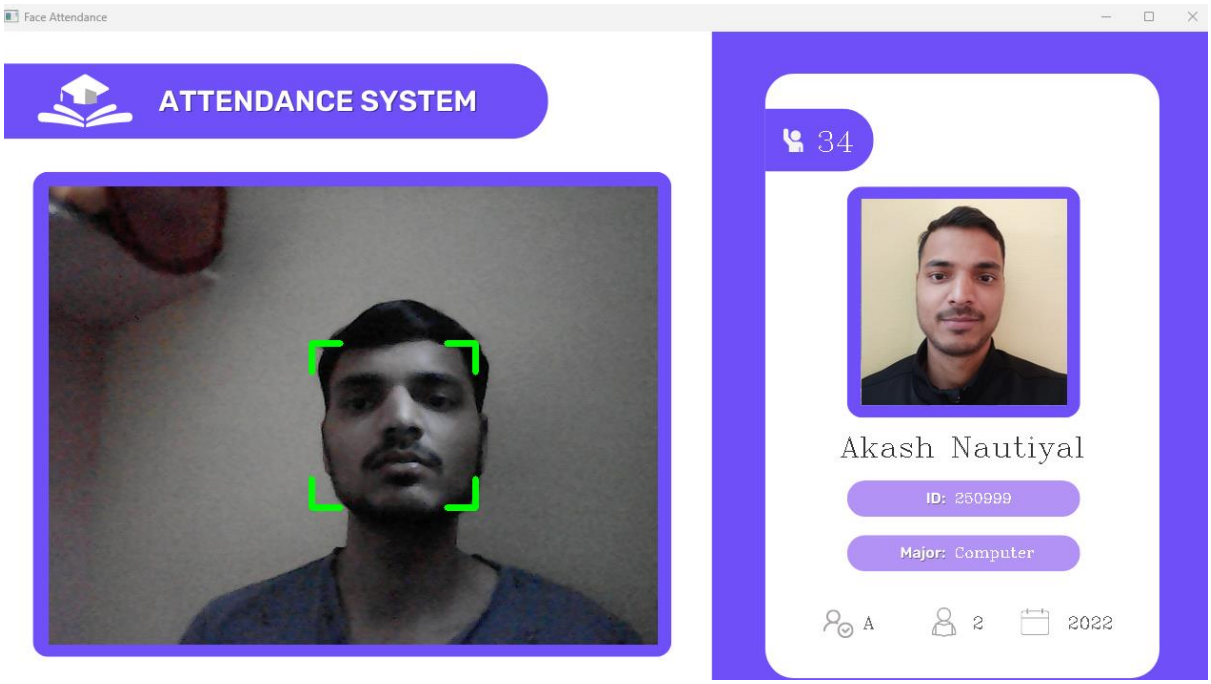
Active Display:



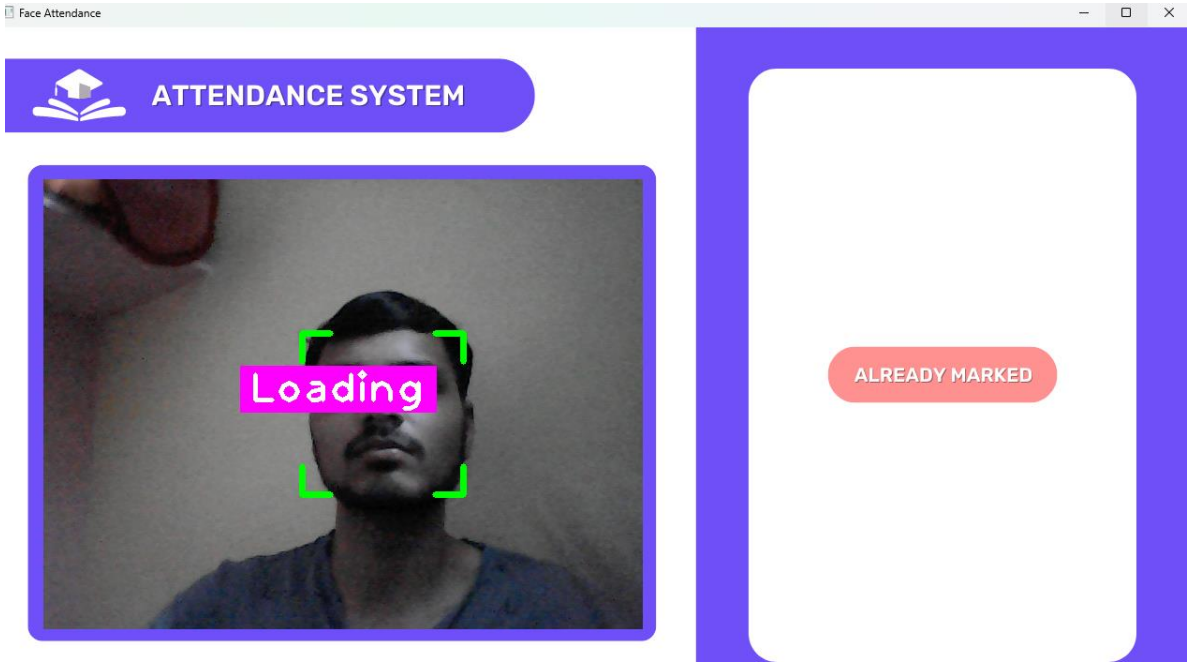
Attendance Marked Page:



User Information Page:



Already Marked Page:



INTRODUCTION

Facial recognition technology has become a common biometric feature in laptops, cell phones, attendance tracking systems, and other devices in recent years. Many organizations employ facial recognition systems for surveillance, home security automation, security access, and criminal identification procedures because of recent technological breakthroughs. In essence, face recognition software compares an individual's geometric traits with their facial features. The conventional approach stores user images in a database, which is then used to compare collected photographs with user images. A variety of traits were retrieved and applied as facial features, including corners, edges, and texture descriptors. For categorization, these features were used with machine learning methods. The system utilizes computer vision, a field of artificial intelligence, to automate the attendance process by recognizing and verifying individuals based on their facial features in real-time [1]. The primary components and functionalities of the system include:

Face Detection: Employing robust algorithms to identify and locate human faces within images or video streams. This initial step is crucial for subsequent processing.

Encoding Generation: Utilizing computer vision techniques to extract unique facial features, such as the distance between eyes, shape of the nose, and contours of the face. These features are translated into mathematical representations for recognition which is known as Encoding of an Image which is useful in comparing the images.

Real-Time Processing: The system operates in real-time, enabling immediate recognition and verification of individuals as they enter or interact with the system. This ensures swift attendance recording without causing delays or disruptions.

Database Management: Storing and managing a database of known faces or templates against which incoming faces are compared for identification and attendance tracking.

User Interface and Integration: Designing a user-friendly interface for administrators and users to interact with the system seamlessly. Integration with existing attendance management systems or APIs enhances its usability and compatibility.

Accuracy and Performance Evaluation: Conducting rigorous testing and evaluation to assess the system's accuracy, reliability, and performance under various conditions, including varying lighting, angles, and facial expressions.

OBJECTIVE

Automated facial recognition systems are commonly employed in corporations, educational institutions, and government agencies to monitor employee attendance. Compared to manual methods, it is quicker, more efficient, and completely removes the need for physical labor [2]. The objectives of a real-time face attendance system using computer vision typically revolve around improving efficiency, accuracy, and convenience in attendance management processes. Here are the primary objectives:

Automated Attendance Tracking: To automate the attendance recording process by using facial recognition technology. This eliminates the need for manual attendance marking, reducing errors and time spent on traditional methods.

Real-Time Monitoring: To provide instantaneous attendance updates. As individuals are recognized in real-time, the system can immediately log their attendance status without delays.

Accuracy and Reliability: To ensure a high level of accuracy in identifying individuals. The system aims to minimize false positives or negatives, accurately associating individuals with their attendance records.

Efficiency and Timesaving: To streamline attendance management for both administrators and attendees. By eliminating manual processes, the system saves time and effort, allowing staff to focus on other essential tasks.

Integration and Compatibility: To integrate seamlessly with existing attendance management systems or databases. Compatibility ensures smooth implementation without disrupting established workflows.

User-Friendly Interface: To offer an intuitive and user-friendly interface for both administrators and users interacting with the system. This promotes ease of use and acceptance among users.

Security and Access Control: To enhance security by monitoring entry points

and ensuring that only authorized individuals are granted access based on their recognition in the system.

Scalability and Adaptability: To design a system that can be easily scaled to accommodate varying numbers of users and locations. This adaptability allows the system to be implemented across different environments.

Compliance and Ethical Considerations: To adhere to ethical standards and legal requirements concerning data privacy, consent, and the responsible use of facial recognition technology.

DATA COLLECTION & PREPROCESSING

Data preprocessing is a crucial step in data analysis and machine learning that involves transforming raw data into a more usable format. It aims to clean, normalize, and prepare the data to enhance its quality and make it suitable for further analysis or model training. It also involves storing of pre-processed data in database so that it can be used further.

The data of an individual is collected in the form of Name (Name of the student), Major (in which subject user is doing Course), Standing (Garde of User), Starting Year (Starting year of the Course), Year (Course Duration). Last Attendance and Total Attendance is added by Code. All these variables are bound up in a folder named Roll Number of User and saved in the Firebase database.

The image which is taken from the webcam is of high pixel range and it can be a hectic task for our computer to generate encoding of that image and to store that high pixel image in our database can also be wastage of storage. The image captured from the webcam has extra information, which is not necessary for the face recognition algorithm, so to remove that extra information we first convert the high pixel image into 256 * 256 and pass the image using various filter for the optimization of the image so that the image used for encoding can have more information regarding facial features. When the image is read from open CV, its default format is BGR and our model uses RGB format, so with the help of Open CV we convert BGR into RGB. After that the image is named after the Roll Number of the user.

METHODOLOGY

The methods used in this research were trained and tested with PyCharm with some basic libraries installed. These algorithms are trained with a normal CPU, so there is no necessity for high-power GPUs. For storing data, Firebase Database is used which update information in Real-Time. This database can store user information in Real time and updating it on real time. Data is stored in JSON format in Firebase Realtime Database, a cloud-hosted NoSQL database. With its real-time synchronization feature, clients can instantly sync data changes across one another in milliseconds. The database is accessible from various platforms, including web, iOS, Android, and server environments. Changes made by one client are instantly reflected on all connected devices, making it ideal for collaborative or multi-user applications. Firebase also provides offline capabilities, enabling applications to remain functional even without an internet connection. Once reconnected, the database syncs any pending changes automatically update. Firebase offers robust security rules to control access to the database. Developers can define rules based on authentication, user roles, and data structures to ensure data security and privacy. It seamlessly integrates with other Firebase services such as Authentication, Cloud Functions, Analytics, and Storage, allowing developers to create comprehensive and scalable applications. Firebase Realtime Database is designed to handle scalability requirements, automatically scaling to accommodate increasing data and user loads [3]. The workflow of system is shown in Figure.

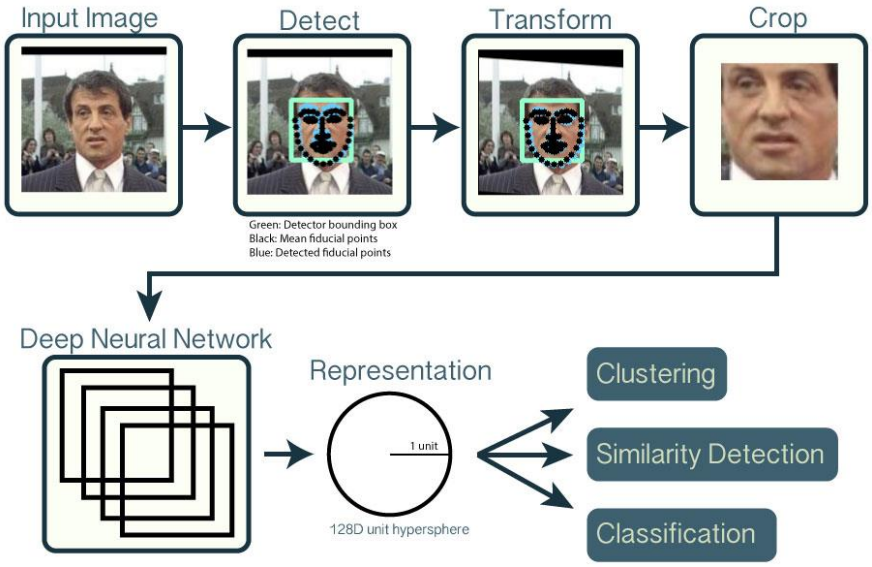


Fig. Workflow of System

The proposed workflow for Real-Time Face Attendance System encompasses: -

A. **Data Preprocessing:** Data preprocessing is a crucial step in data analysis and machine learning that involves transforming raw data into a more usable format. It aims to clean, normalize, and prepare the data to enhance its quality and make it suitable for further analysis or model training. It also involves storing of pre-processed data in database so that it can be used further. Individual data is gathered in the following formats:

   Name (student name), Major (subject area in which the user is enrolled in the course), Standing (user's grade), Beginning Year (course year beginning), and Year (length of the course).

   Code adds the Last Attendance and Total Attendance. All these variables are stored in the Firebase database and organized into a folder called Roll Number of User which is basically a 6-digit number. The image captured by the webcam has a high pixel range, which means that encoding it and storing it in our database could be a laborious process for our computer and result in storage waste. The webcam image contains extra information that the face recognition algorithm does not need. To get rid of this extra information, we first convert the high-pixel image to 256 by 256 pixel and then optimize the image using different filters. This way, the image used for encoding can contain more information about facial features. Since our model uses the RGB format and the default BGR format for images read from OpenCV, we use OpenCV to help us convert BGR to RGB. The image is then given a name based on the user's roll number.

B. **Face Detection:** Face detection is a computer vision technology used to identify and locate human faces within images or video frames. It's a foundational step in various applications, including facial recognition, security systems, photo editing software, and more. The technique of identifying a face in a picture is called detection. Facial recognition, made possible by computer vision, can recognize, and identify individual faces from a picture that contains the faces of one or more people [4]. A face detection example is shown in Figure.

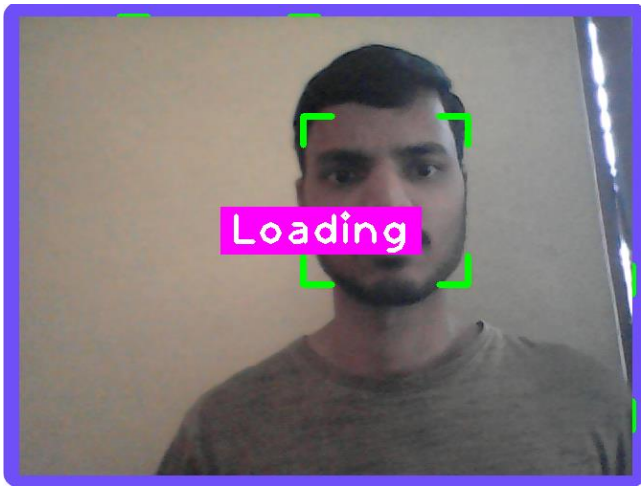


Fig. Face Detection using face recognition.

C. Generate Face Encodings: Generating face encodings involves converting facial images into numerical representations, enabling easy comparison and recognition of faces. One popular method for face encoding is using deep learning models, particularly face recognition models, to extract facial embeddings [5]. Dlib comes with pre-trained landmark detector that detects 68 different landmark features as shown in Figure.

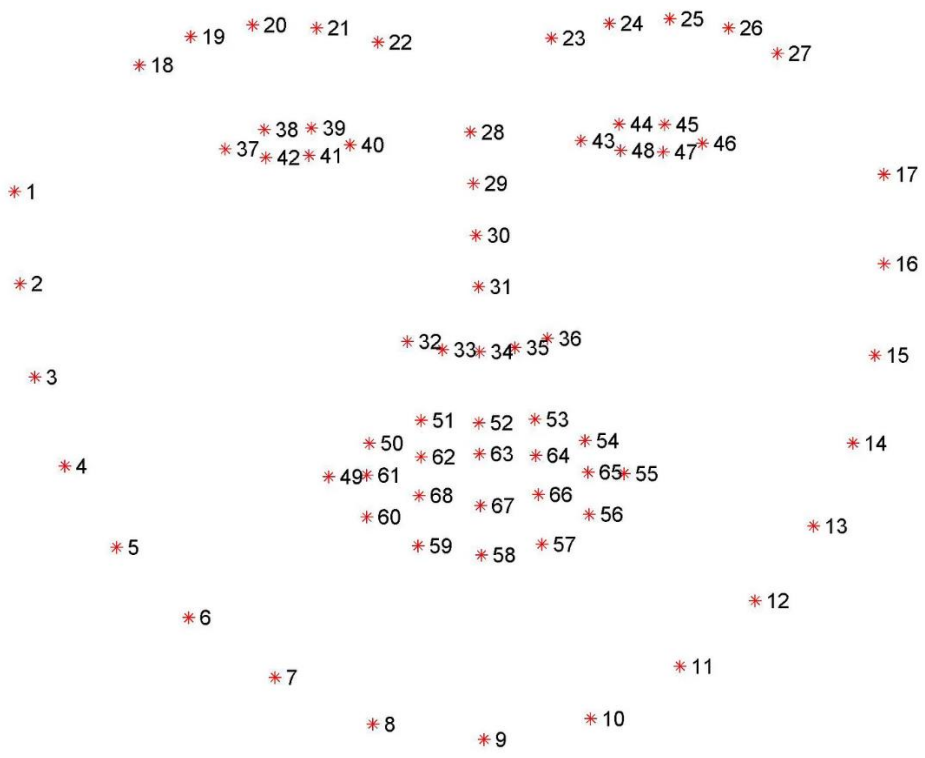


Figure. Dlib facial landmark detecto

A CNN that has been previously trained is used to generate the 128 encoding values. Due to its prior training on millions of training photos, the model is able to produce trustworthy encodings for faces it has never seen before. Images of the same individual ought to yield around the same 128-valued encoded vector. An extremely basic interface for extracting the encodings from an image and the set of rectangles that depict the faces in the image is offered by the facial_recognition library. Finding out which person the face belongs to is as easy as measuring the distance between the new facial image and those in the encodings and choosing the one with the lowest distance value. This is done once you have the 128 valued encoded vector and the person's name as the label [6]. Below you can see the face encoding following figures of two different input images taken from the webcam.

Figure. Input Image 1 from camera Model.

Encoding Started ...

array([-1.90755725e-01, 1.27633125e-01, 2.92013586e-02, -3.49817239e-02,
-1.23655237e-02, -9.81059074e-02, -2.48256847e-02, -1.01783261e-01,
1.08596422e-01, -3.11100110e-02, 2.03311846e-01, -1.00102231e-01,
-2.21952349e-01, -1.36154726e-01, -2.54909489e-02, 9.77982506e-02,
-1.09247833e-01, -1.99613065e-01, 1.42976791e-02, -1.01944096e-01,
1.17541626e-01, 5.11706993e-03, 1.54069178e-02, 1.08624332e-01,
-2.86817849e-01, -2.77195096e-01, -9.48177874e-02, -2.17730090e-01,
3.68094258e-03, -1.19397134e-01, -2.11995188e-02, 4.86608967e-02,
7.48404413e-02, 3.25713865e-02, 7.43889362e-02, -4.79312614e-02,
9.41099226e-02, -2.00933248e-01, 1.03238538e-01, 1.53080150e-01,
1.45939484e-01, 5.49700409e-02, 1.52193904e-01, -1.54745445e-01,
3.05712204e-02, 5.61157838e-02, -2.23978162e-01, 1.62887692e-01,
6.50427639e-02, 1.80913117e-02, -6.47907481e-02, -1.45804305e-02,
2.25748748e-01, 7.31183663e-02, -1.49839759e-01, -8.68962184e-02,
1.26961440e-01, -1.57963574e-01, -4.08169851e-02, 1.21845007e-01,
-8.39499384e-02, -1.63634926e-01, -2.68825203e-01, 7.65522718e-02,
-1.59626573e-01, 1.03154860e-01, -1.08275175e-01, -2.94140391e-02,
9.88917053e-02, -2.83227175e-01, 1.47268504e-01, 1.50552467e-01,
-8.42809603e-02, 8.90252069e-02, 7.27110654e-02, 1.72209013e-02,
2.99694594e-02, 1.84677541e-04, -9.40349102e-02, -9.36298594e-02,
6.18245490e-02, -7.97352046e-02, 5.16808107e-02, 5.50850760e-03)]

Encoding Complete
File Saved
Process finished with exit code 0

Figure. Encoding of input image 1

Figure. Input Image 2 from camera Model.

```
Encoding Started ...
array([-2.20260203e-01,  1.30789548e-01,  7.59498700e-02, -4.84002829e-02,
       -3.05678025e-02, -3.31506915e-02, -5.93391657e-02, -1.11219533e-01,
        1.63894996e-01, -7.84125775e-02,  2.31201515e-01,  2.20352970e-02,
       -9.91328359e-02, -7.81775936e-02, -6.82642609e-02,  1.59182370e-01,
       -1.73202962e-01, -1.30004942e-01, -5.82705997e-02, -1.26322702e-01,
       -9.83849168e-05,  2.29360834e-02, -1.96711738e-02,  5.66161275e-02,
       -8.76095295e-02, -3.29574496e-01, -9.54316184e-02, -1.23355776e-01,
        6.16217889e-02, -6.14502057e-02,  5.96307330e-02,  3.92164253e-02,
       -1.53687090e-01, -2.27859914e-02,  6.19656444e-02,  7.00810701e-02,
        3.96575592e-02,  4.29849550e-02,  1.62563711e-01,  3.10022812e-02,
        3.73470504e-03,  9.08966362e-02, -1.83375955e-01,  1.45096555e-01,
        1.32592246e-01,  1.29765749e-01, .....3731e-01,  1.41869143e-01,
       -6.02923557e-02, -6.05847128e-03, -3.53222303e-02,  3.23826298e-02,
        9.01911873e-03,  1.09084390e-01,  9.74849835e-02,  4.25669998e-02,
        1.69138983e-02, -2.45250817e-02, -9.32487175e-02, -1.09036803e-01,
        1.60095356e-02, -6.40238002e-02,  1.25130966e-01,  9.43907164e-03
Encoding Complete
File Saved
Process finished with exit code 0
```

Figure. Encoding of input image 2

D. Compare Encodings: In this process we generally compare both encoding of the database images and the captured images and the matched result is stored in list. The least value in the list is then identified by ID and is shown in the desired Graphics. We can see the least value in the array, and that value is then

converted into Boolean as minimum value gets true and rest gets false [7]. After we get the index value of matched index, we update attendance in Database with last attendance time as shown in Fig 8.

```
Loading Encode File ...
Encode File Loaded
matches [False, False, True, False, False, False]
faceDis [0.99285519 0.65350092 0.34120175 0.66463729 1.05193736 0.89501874]
Match Index 2
Known Face Detected
250999
{'last_attendance_time': '2024-04-11 12:19:41', 'major': 'Computer', 'name': 'Akash Nautiyal'
55.754506
matches [False, False, True, False, False, False]
faceDis [0.99040352 0.66188263 0.30666434 0.67987125 1.04448264 0.88833328]
Match Index 2
Known Face Detected
250999
```

Fig 8. Comparison of Encoding

E. Update Attendance in Database: After showing desired image in the graphics, the attendance of the identified ID is then updated in the database in real time (Fig. 9). If the attendance is already marked so it doesn't affect the database and the result "Already marked" is shown in the Output, otherwise the attendance is increased by 1 and last attendance time is also updated.

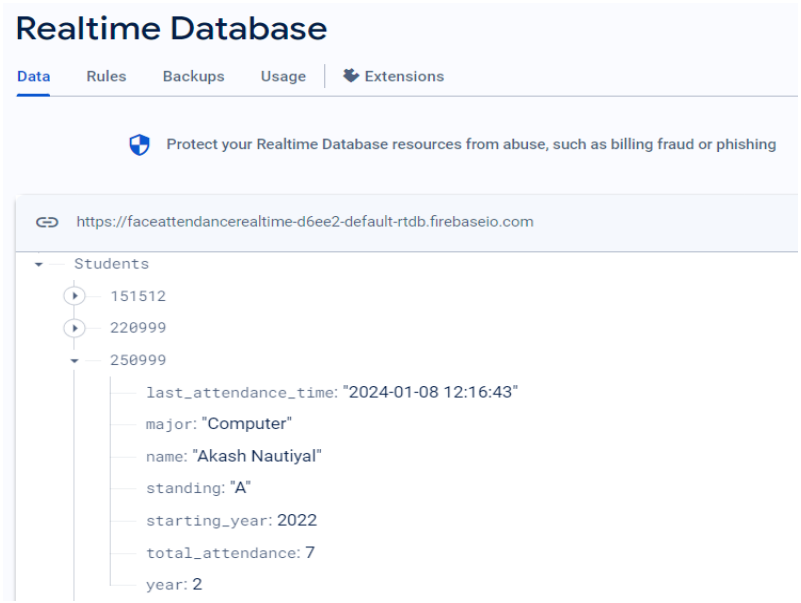


Fig 9. User Information in the database

FIREBASE: REALTIME DATABASE

The Firebase Realtime Database is a cloud-hosted database in which data is stored as JSON. The data is synchronized in real-time to every connected client. All of our clients share one Realtime Database instances and automatically receive updates with the newest data, when we build cross-platform applications with our iOS, and JavaScript SDKs.

The Firebase Realtime Database is a NoSQL database from which we can store and sync the data between our users in real-time. It is a big JSON object which the developers can manage in real-time. By using a single API, the Firebase database provides the application with the current value of the data and updates to that data. Real-time syncing makes it easy for our users to access their data from any device, be it web or mobile.

The Realtime database helps our users collaborate with one another. It ships with mobile and web SDKs, which allow us to build our app without the need for servers. When our users go offline, the Real-time Database SDKs use local cache on the device for serving and storing changes. The local data is automatically synchronized, when the device comes online.

Key Capabilities:

A Real-time database is capable of providing all offline and online services. These capabilities include accessibility from the client device, scaling across multiple databases, and many more.

Real-time: The Firebase Real-time database uses data synchronization instead of using HTTP requests. Any connected device receives the updates within

milliseconds. It doesn't think about network code and provides collaborative and immersive experiences.

Offline: The Firebase Database SDK persists our data to disk, and for this reason, Firebase apps remain responsive even when offline. The client device receives the missed changes, once connectivity is re-established.

Accessible from client devices: There is no need for an application server to access the Firebase Real-time database. We can access it directly from a mobile device or web browser. Data validation and security are available through the Firebase Real-time Database Security Rules, expression-based rules executed when data is read or written.

Scaling across multiple databases: With the Firebase Real-time Database on Blaze Pricing Plan, we can support the data needs of our app by splitting our data across multiple database instances in a single Firebase project. Streamline authentication with Firebase authentication on our project and authenticate users in our database instances. Controls access to data in each database with custom Firebase real-time database rules available for each database instance.

Other Alternatives:
Apart from Firebase's real-time database, there are several alternatives that are used.

Cloud Firestore: Cloud Firestore is a scalable and flexible database used for server development, mobile, and web from Firebase and Google Cloud Platform.

Firebase Remote Config: It stores developer specified key-value pairs to change the behaviour and appearance of our app without requiring users to download an update.

Firebase Hosting: It is used to hosts the HTML, CSS, and JavaScript of our website as well as other developer-provided assets like graphing, fonts, and icons.

Cloud Storage: It is used to store images, videos, and audio as well as other user-generated content.

TOOLS USED

OpenCV:

OpenCV, short for Open-Source Computer Vision Library, stands as a testament to the transformative power of open-source software in the field of computer vision. Developed by Intel in 1999 and later supported by Willow Garage and Itseez, OpenCV has evolved into one of the most popular and widely-used libraries for real-time computer vision tasks. In this exploration, we delve into the essence of OpenCV, its core features, and its profound impact on the realm of computer vision.

At its core, OpenCV is a comprehensive library that offers a plethora of tools and algorithms for image and video analysis, processing, and manipulation. Whether you're detecting objects in images, tracking motion in videos, or extracting meaningful information from visual data, OpenCV provides a rich set of functionalities to tackle a wide range of computer vision tasks with ease and efficiency.

One of the key strengths of OpenCV lies in its support for multiple programming languages, including C++, Python, Java, and MATLAB. This multi-language support ensures that developers from diverse backgrounds can leverage the power of OpenCV in their projects, regardless of their programming preferences or expertise levels.
Additionally, OpenCV's cross-platform compatibility allows it to run seamlessly on various operating systems, including Windows, Linux, macOS, iOS, and Android, further enhancing its accessibility and versatility.

OpenCV's extensive collection of algorithms spans a wide spectrum of computer vision tasks, ranging from basic image processing operations like filtering, thresholding, and morphological operations to advanced techniques such as feature detection, object recognition, and machine learning-based classification. Whether you're exploring fundamental concepts in computer vision or implementing cutting- edge research algorithms, OpenCV provides a rich toolbox to support your endeavors.

One of the hallmark features of OpenCV is its real-time processing capabilities, making it particularly well-suited for applications requiring rapid analysis and decision-making. Whether you're developing surveillance systems, autonomous vehicles, or augmented reality applications, OpenCV's efficient algorithms and optimized implementations ensure high-performance processing even in demanding real-world scenarios.

Furthermore, OpenCV integrates seamlessly with other popular libraries and frameworks, enabling developers to build end-to-end solutions for complex computer vision tasks. Whether you're leveraging deep learning frameworks like TensorFlow and PyTorch for advanced feature extraction or integrating OpenCV with robotics frameworks like ROS (Robot Operating System) for autonomous navigation, OpenCV serves as a versatile building block in the development of sophisticated vision-based systems.

In addition to its vast array of functionalities, OpenCV boasts a vibrant and supportive community of developers, researchers, and enthusiasts who actively contribute to its development and evolution. Through collaborative efforts, community-driven initiatives, and open communication channels, the OpenCV community continues to drive innovation and push the boundaries of what's possible in the realm of computer vision.

PyCharm:

PyCharm is a powerful integrated development environment (IDE) specifically designed for Python programming. Developed by JetBrains, PyCharm offers a comprehensive suite of tools that cater to both professional developers and beginners. The IDE supports a wide range of functionalities, including code completion, real-time error detection, and refactoring, which significantly enhances coding efficiency and accuracy.

One of PyCharm's standout features is its intelligent code editor, which provides context-aware suggestions and quick navigation through the codebase. This helps developers write clean, maintainable code with ease. Additionally, PyCharm integrates seamlessly with various version control systems like Git, facilitating efficient project management and collaboration.

PyCharm also supports web development frameworks such as Django and Flask, making it a versatile tool for full-stack development. Its integrated tools for testing, debugging, and profiling ensure that developers can maintain high code quality and performance. The IDE's support for scientific libraries like NumPy, SciPy, and matplotlib makes it an excellent choice for data science and machine learning projects.

Moreover, PyCharm's customizable interface and extensive plugin ecosystem allow developers to tailor the IDE to their specific needs, enhancing their productivity. With its robust feature set and user-friendly design, PyCharm remains a preferred choice for Python developers worldwide.

NumPy:

NumPy, short for Numerical Python, stands as a cornerstone in the Python ecosystem, revolutionizing the landscape of numerical computing. Born out of the need for efficient array operations and mathematical functions, NumPy has become an indispensable tool for scientists, engineers, and data analysts worldwide. In this short note, we'll explore the essence of NumPy, its core features, and its profound impact on the Python community.

At the heart of NumPy lies the ndarray (n-dimensional array) object, a powerful data structure that enables efficient storage and manipulation of multi-dimensional data. Unlike traditional Python lists, NumPy arrays are homogeneous, meaning that all elements within them must be of the same data type. This uniformity not only leads to optimized memory storage but also facilitates vectorized operations, allowing for lightning-fast computations without the need for explicit looping constructs.

One of the key advantages of NumPy is its extensive collection of mathematical functions and operations tailored for array processing. From simple arithmetic operations like addition and multiplication to more complex functions such as trigonometric, exponential, and logarithmic functions, NumPy provides a comprehensive suite of tools to handle a wide range of numerical tasks. Furthermore, NumPy seamlessly integrates with other Python libraries such as SciPy, Pandas, and Matplotlib, forming a robust ecosystem for scientific computing and data analysis.

NumPy's array indexing and slicing capabilities are another highlight, enabling users to extract, modify, and manipulate subsets of data with ease. Whether you're working with one-dimensional arrays, multi- dimensional matrices, or even structured arrays with named fields, NumPy offers intuitive and efficient mechanisms for accessing and manipulating data elements at various levels of granularity.

The concept of broadcasting further enhances NumPy's appeal, allowing for arithmetic operations between arrays of different shapes and sizes. Through broadcasting, NumPy automatically aligns the

dimensions of input arrays to perform element-wise operations, simplifying code implementation and improving performance. This feature is particularly useful in scenarios where arrays of different shapes need to be combined or manipulated efficiently.

Moreover, NumPy provides extensive support for linear algebra operations, making it an invaluable tool for tasks involving numerical simulations, machine learning, signal processing, and image processing. From matrix multiplication and eigenvalue decomposition to singular value decomposition and beyond, NumPy's linear algebra capabilities empower users to tackle complex mathematical problems with confidence and efficiency.

Dlib:

dlib is a powerful toolkit for making real-world machine learning and data analysis applications. Originally developed in C++, it has robust Python bindings that provide an easy-to-use interface for a variety of machine learning and computer vision tasks. Here's a detailed look at what makes dlib a popular choice among developers and researchers:

dlib includes a wide range of machine learning algorithms, such as support vector machines (SVM), k-nearest neighbors (KNN), and deep learning models. It supports both supervised and unsupervised learning methods.

dlib offers state-of-the-art face detection capabilities using Histogram of Oriented Gradients (HOG) and Convolutional Neural Network (CNN) based methods.

It includes facial landmark detection, which identifies key points on the face such as the eyes, nose, and mouth. The library can recognize and compare faces using deep metric learning.

Functions for image processing such as image transformation, filtering, and feature extraction are available. It supports reading, writing, and manipulating images in various formats.

dlib provides tools for training and deploying custom object detectors. It supports the use of Histogram of Oriented Gradients (HOG) features.

Face Recognition:

Face recognition in Python involves using libraries and frameworks that facilitate detecting, recognizing, and analysing human faces in images or videos. This technology has a wide range of applications, from security systems and identity verification to social media and augmented reality. Here's a detailed overview of face recognition in Python:

Key Libraries for Face Recognition
1. face_recognition
face_recognition is one of the simplest and most powerful libraries for face recognition in Python, built on top of dlib.

Features:

- Detects faces in images.
- Recognizes faces by comparing them with known faces.
- Identifies facial features like eyes, nose, and mouth.
- Encodes faces into 128-dimensional vectors for comparison.

Pickle:

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What Pickle does is it "serializes" the object first before writing it to a file. Pickling is a way to convert a Python object (list, dictionary, etc.) into a character

stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another Python script. It provides a facility to convert any Python object to a byte stream. This Byte stream contains all essential information about the object so that it can be reconstructed, or "unpickled" and get back into its original form in any Python.

Recursive objects (objects containing references to themselves): Pickle keeps track of the objects it has already serialized, so later references to the same object won't be serialized again.

Object sharing (references to the same object in different places): This is similar to self-referencing objects. Pickle stores the object once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.

User-defined classes and their instances: Marshal does not support these at all, but Pickle can save and restore class instances transparently. The class definition must be importable and live in the same module as when the object was stored.

While Python Pickle offers capabilities for object serialization, developers that maintain limitations, especially while working across various Python versions or dealing with the large datasets. It's important to remember always consider the specific needs of your application to determine if ickle or an alternative like JSON, XML is suited for serialization.

CVzone:

CVzone is a high-level computer vision library in Python that simplifies the use of the OpenCV and MediaPipe libraries. It is designed to make common computer vision tasks easier to implement and more accessible for developers, especially those who may not have extensive experience with computer vision.

Simplifies the complex functionalities of OpenCV and MediaPipe into easy-to-use functions. Reduces the amount of boilerplate code required for common tasks Provides high-level functions for detecting and tracking objects in images and videos. Integrates seamlessly with MediaPipe for hand and face detection.

Supports the creation of AR effects and applications. Facilitates overlaying digital content onto the real world through video streams.

Optimized for real-time performance, making it suitable for applications such as interactive installations and live video processing.

Ideal for learning computer vision concepts and quickly prototyping computer vision projects.

Firebase:

Firebase is a platform developed by Google for creating mobile and web applications. While Firebase is primarily known for its robust integration with JavaScript and mobile frameworks (like Android and iOS), there are Python libraries available that allow developers to interact with Firebase services.

One of the most popular Python libraries for Firebase is firebase-admin. This library allows Python applications to interact with various Firebase services, including authentication, real-time database, cloud storage, and Firestore.

Authentication:

- Manage users and authentication flows.
- Verify ID tokens and retrieve user information.

Realtime Database:

- Read and write operations on the Firebase Realtime Database.
- Perform complex queries and transactions.

Cloud Firestore:

- Real-time updates and offline support.
- Structured query support.

Cloud Storage:

- Upload and download files.
- Manage storage buckets.

Cloud Messaging:

- Send notifications and messages to client apps.
- Manage device groups and topics.

Machine Learning:

- Integrate with Firebase ML for on-device and cloud-based machine learning models.

APPENDIX A
CODE

#main.py

```
import os
import pickle
import numpy as np
import cv2
import face_recognition
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
from datetime import datetime


cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
  'databaseURL': "https://faceattendancerealtime-d6ee2-default-
rtdb.firebaseio.com/",
  'storageBucket': "faceattendancerealtime-d6ee2.appspot.com"
})

bucket = storage.bucket()

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

imgBackground = cv2.imread('Resources/background.png')

# Importing the mode images into a list
folderModePath = 'Resources/Modes'
modePathList = os.listdir(folderModePath)
```

```
imgModeList = []
for path in modePathList:
  imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))
# print(len(imgModeList))

# Load the encoding file
print("Loading Encode File ...")
file = open('EncodeFile.p', 'rb')
encodeListKnownWithIds = pickle.load(file)
file.close()
encodeListKnown, studentIds = encodeListKnownWithIds
# print(studentIds)
print("Encode File Loaded")

modeType = 0
counter = 0
id = -1
imgStudent = []

while True:
  success, img = cap.read()


  imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
  imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)


  faceCurFrame = face_recognition.face_locations(imgS)
  encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)


  imgBackground[162:162 + 480, 55:55 + 640] = img
  imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
```

```
 if faceCurFrame:
      for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
         matches = face_recognition.compare_faces(encodeListKnown,
encodeFace)
         faceDis = face_recognition.face_distance(encodeListKnown,
encodeFace)
         print("matches", matches)
         print("faceDis", faceDis)


         matchIndex = np.argmin(faceDis)
         print("Match Index", matchIndex)


         if matches[matchIndex]:
            print("Known Face Detected")
            print(studentIds[matchIndex])
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
            imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)
            id = studentIds[matchIndex]
            if counter == 0:
               cvzone.putTextRect(imgBackground, "Loading", (275, 400))
               cv2.imshow("Face Attendance", imgBackground)
               cv2.waitKey(1)
               counter = 1
               modeType = 1

      if counter != 0:

         if counter == 1:
            # Get the Data
            studentInfo = db.reference(f'Students/{id}').get()
            print(studentInfo)
            # Get the Image from the storage
```

```
            blob = bucket.get_blob(f'Images/{id}.png')
            array = np.frombuffer(blob.download_as_string(), np.uint8)
            imgStudent = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
            # Update data of attendance
            datetimeObject =
datetime.strptime(studentInfo['last_attendance_time'],
                                      "%Y-%m-%d %H:%M:%S")
            secondsElapsed = (datetime.now() - datetimeObject).total_seconds()
            print(secondsElapsed)
            if secondsElapsed > 30:
                ref = db.reference(f'Students/{id}')
                studentInfo['total_attendance'] += 1
                ref.child('total_attendance').set(studentInfo['total_attendance'])
                ref.child('last_attendance_time').set(datetime.now().strftime("%Y-
%m-%d %H:%M:%S"))
            else:
                modeType = 3
                counter = 0
                imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]

        if modeType != 3:

            if 10 < counter < 20:
                modeType = 2

            imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]

            if counter <= 10:
                cv2.putText(imgBackground, str(studentInfo['total_attendance']),
(861, 125),
                            cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 1)
                cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                            cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
                cv2.putText(imgBackground, str(id), (1006, 493),
```

```
                    cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
            cv2.putText(imgBackground, str(studentInfo['standing']), (910,
625),
                    cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)
            cv2.putText(imgBackground, str(studentInfo['year']), (1025, 625),
                    cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)
            cv2.putText(imgBackground, str(studentInfo['starting_year']),
(1125, 625),
                    cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)

            (w, h), _ = cv2.getTextSize(studentInfo['name'],
cv2.FONT_HERSHEY_COMPLEX, 1, 1)
            offset = (414 - w) // 2
            cv2.putText(imgBackground, str(studentInfo['name']), (808 + offset,
445),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)

            imgBackground[175:175 + 216, 909:909 + 216] = imgStudent

        counter += 1

        if counter >= 20:
            counter = 0
            modeType = 0
            studentInfo = []
            imgStudent = []
            imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]
  else:
      modeType = 0
      counter = 0

  # cv2.imshow("Webcam", img)
  cv2.imshow("Face Attendance", imgBackground)
  cv2.waitKey(1)
```

#AddDataToDatabase.py

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db



cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
  'databaseURL': "https://faceattendancerealtime-d6ee2-default-
rtdb.firebaseio.com/"
})


ref = db.reference('Students')



data = {
  "151512":
     {
        "name": "Emma Watson",
        "major": "Arts",
        "starting_year": 2021,
        "total_attendance": 4,
        "standing": "A",
        "year": 3,
        "last_attendance_time": "2022-12-11 00:54:34"
     },
  "220999":
     {
        "name": "Akash Negi",
        "major": "Electronics",
        "starting_year": 2022,
        "total_attendance": 4,
```

```
        "standing": "A",
        "year": 2,
        "last_attendance_time": "2023-12-11 00:54:34"
    },
"250999":
    {
        "name": "Akash Nautiyal",
        "major": "Computer",
        "starting_year": 2022,
        "total_attendance": 5,
        "standing": "A",
        "year": 2,
        "last_attendance_time": "2023-12-11 00:54:34"
    },
"321654":
    {
        "name": "Murtaza Hassan",
        "major": "Robotics",
        "starting_year": 2017,
        "total_attendance": 7,
        "standing": "G",
        "year": 4,
        "last_attendance_time": "2022-12-11 00:54:34"
    },
"852741":
    {
        "name": "Emly Blunt",
        "major": "Economics",
        "starting_year": 2021,
        "total_attendance": 12,
        "standing": "B",
        "year": 1,
        "last_attendance_time": "2022-12-11 00:54:34"
    },
"963852":
    {
```

```
        "name": "Elon Musk",
        "major": "Physics",
        "starting_year": 2020,
        "total_attendance": 7,
        "standing": "G",
        "year": 2,
        "last_attendance_time": "2022-12-11 00:54:34"
    }
}


for key, value in data.items():
  ref.child(key).set(value)
```

#EncodeGenerator.py

```
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import  storage

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
  'databaseURL': "https://faceattendancerealtime-d6ee2-default-
rtdb.firebaseio.com/",
  'storageBucket': "faceattendancerealtime-d6ee2.appspot.com"
})


# Importing student images
folderPath = 'Images'
pathList = os.listdir(folderPath)
print(pathList)
imgList = []
studentIds = []
for path in pathList:
  imgList.append(cv2.imread(os.path.join(folderPath, path)))
  studentIds.append(os.path.splitext(path)[0])

  fileName = f'{folderPath}/{path}'
  bucket = storage.bucket()
  blob = bucket.blob(fileName)
  blob.upload_from_filename(fileName)


  # print(path)
```

```
   # print(os.path.splitext(path)[0])
print(studentIds)


def findEncodings(imagesList):
   encodeList = []
   for img in imagesList:
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      encode = face_recognition.face_encodings(img)[0]
      encodeList.append(encode)

   return encodeList

print("Encoding Started ...")
encodeListKnown = findEncodings(imgList)
print(encodeListKnown)
encodeListKnownWithIds = [encodeListKnown, studentIds]
print("Encoding Complete")

file = open("EncodeFile.p", 'wb')
pickle.dump(encodeListKnownWithIds, file)
file.close()
print("File Saved")
```

RESULT

The world's most straightforward face recognition library allows you to recognize and work with faces from Python or the command line developed with dlib's cutting-edge deep learning-based face recognition technology. On the labelled Faces in the Wild benchmark, the model's accuracy is 99.38%.

A database of face photos called labelled Faces in the Wild was created with the purpose of researching the issue of unrestrained face recognition. More than 13,000 facial photos that were gathered from the internet are included in the data collection. The name of each person in the photo is written on each face.

The image provided to face_encoder has its (128,1) dimension encoding obtained using face_encoder. It is a 29-convolution-layer pretrained ResNet network model. A dataset containing roughly 3 million faces is used to train the model. This system performs well in each and every aspect and perform with same accuracy for every user whenever it is used for face attendance. As it uses Face Recognition library whose accuracy is 99.38% so because of that this system is able to recognize face with the same accuracy and gives positive result every time. As we are using Google Firebase as a database and it is known for its smooth operation or updating data in real time so as face is recognized by model its attendance is marked in no time. We have tried this system with the information of 10 people and it has marked `attendance of them every time.

CONCLUSION

Applications for facial image processing include face recognition systems, and in recent years, their importance as a research topic has grown. The system's implementations include person verification, video monitoring, crime prevention, and other related security measures. University implementations of face recognition systems are possible. The goal of the Face Recognition Based Attendance System is to decrease the errors that happen with the manual, traditional method of recording attendance. Automating and creating a system that benefits a company, like an institute, is the goal. This approach is workable, dependable, and sufficiently safe. The suggested method can identify several faces, and the system's performance yields respectable, decent outcomes.

REFERENCES

- Modern Face Recognition with Deep Learning [Online] Avaliable: Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning | by Adam Geitgey | Medium

- Empowering Students and Teachers: The Role of Face Recognition Attendance System in Educational Institutions[Online] Available: https://www.rydotinfotech.com/blog/benefits-of-face-recognition-attendance-system-for-schools-and-colleges/

- Rosebrock, Adrian. "Opencv face recognition." PyImageSearch. Dostopno na: https://www. pyimagesearch. com/2018/09/24/opencv-face-recognition/[14. 4. 2019] (2018).

- Facial Recognition — a visual step by step [Online] Avaliable: https://medium.com/swlh/facial-recognition-a-visual-step-by-step-d679289bab11

- [Hangaragi, Shivalila, Tripty Singh, and N. Neelima. "Face detection and Recognition using Face Mesh and deep neural network." Procedia Computer Science 218 (2023): 741-749.]

- Step by Step Face Recognition Code Implementation From Scratch In Python [Online]Available:https://towardsdatascience.com/step-by-step-face-recognition-code-implementation-from-scratch-in-python-cc95fa041120

- How to compare two images and get an accuracy level with Python [Online]Available:https://medium.com/@gowtham180502/how-to-compare-two-images-and-get-an-accuracy-level-with-python-61d8d953a942

- ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line

- https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-facerecognition-with-deep-learning-c3cffc121d78

- https://firebase.google.com/

- https://www.computervision.zone/