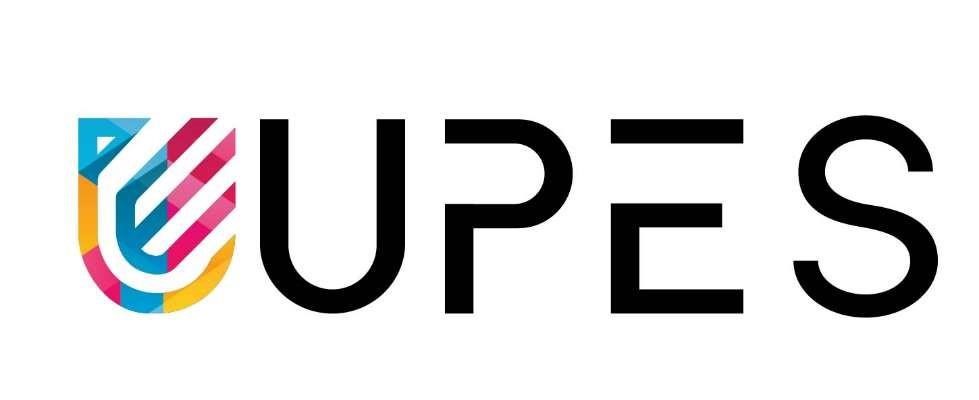
Software Requirements Specification

**Attendance Recording System using Facial Recognition**

Prepared By

|  |  |  |
| --- | --- | --- |
| Specialization | Sap id | Name |
| Data Science | 500084615 | Dhanraj Bhosale |
| Data Science | 500086349 | Khushi Bansal |
| Data Science | 500085562 | Ritika Kumari |



Data Science Cluster

School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

DEHRADUN-248007, Uttarakhand

Dr. Shaurya Gupta Mr. Shamik Tiwari

Project Mentor Cluster Head

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | **Topic** | **Page no.** |
| **1.** | **INTRODUCTION** | 4 |
| 1.1 | Purpose of the project | 4 |
| 1.2 | Target Beneficiary | 4-5 |
| 1.3 | Project Scope | 5 |
| 1.4 | References | 5-6 |
| **2.** | **PROJECT DESCRIPTION** | 6 |
| 2.1 | Reference algorithm | 6 |
| 2.2 | Data/Data Structure | 7 |
| 2.3 | Design and implementation constraints | 7 |
| 2.4 | Design Diagram | 8-9 |
| **3.** | **SYSTEM REQUIREMENTS** | 10 |
| 3.1 | System Interface | 10 |
| 3.2 | Database Interface | 10 |
| **4.** | **NON-FUNCTIONAL REQUIREMENTS** | 10 |
| 4.1 | Performance Requirements | 10 |
| 4.2 | Software Quality Attributes | 10 |

1. **INTRODUCTION**

Imagine a world where attendance no longer requires tedious signing papers or battling noisy roll calls. Imagine a system that effortlessly verifies your presence with a simple glance. This vision becomes a reality with Automated Attendance Recording Systems powered by Facial Recognition (FRS).

FRS eliminates the inefficiencies and frustrations of traditional methods, offering a plethora of benefits:

* Effortless Efficiency: Forget tedious manual processes. FRS streamlines attendance, freeing up valuable time for instructors and students.
* Unwavering Accuracy: Say goodbye to human error and proxy attendance. FRS ensures meticulous and reliable records, boosting accountability.
* Enhanced Security: Unfamiliar faces are flagged, fostering a secure environment where only authorized individuals gain access.
* Seamless Convenience: No more scrambling for pens or fumbling with papers. FRS offers a contactless and intuitive experience for everyone.

# Purpose of the project

The purpose of attendance prediction using machine learning models is to forecast attendance rates accurately in various contexts such as events, classes, or workplaces. By analyzing historical attendance data alongside relevant factors like time, date, weather, and promotional activities, these models can provide valuable insights to organizers, educators, and managers. With precise predictions, stakeholders can optimize resource allocation, staff scheduling, and marketing efforts, ultimately enhancing efficiency, maximizing participation, and improving overall decision-making processes. This predictive capability empowers organizations to anticipate attendance trends, mitigate potential risks, and ensure better preparedness for future events or activities.

# Target Beneficiary

The target beneficiary for attendance prediction using an ML model includes educational institutions, HR departments in companies, event organizers, and any organization relying on managing attendance efficiently. By accurately forecasting attendance patterns, these entities can optimize resource allocation, streamline scheduling, and enhance overall operational efficiency. Students benefit from optimized class schedules, ensuring they attend necessary sessions. Employees experience improved productivity with optimized work schedules. Event organizers can better plan logistics and resources.

# Project Scope

# The project aims to develop a machine learning model for attendance prediction. The scope includes data collection from attendance records, feature engineering to extract relevant information, model training and validation, and implementation for real-time predictions. The model will utilize algorithms such as regression or classification to forecast attendance patterns based on factors like historical data, time of year, and external variables. The final deliverables will include a trained ML model integrated into a user-friendly interface for efficient attendance forecasting, aiding organizations in resource allocation and planning.ject Scope

**1.4 References:**

[1] Smitha, Pavithra S Hegde, Afshin: Face Recognition based Attendance Management System, International Journal of Engineering and

Technical Research V9(05), June 2020

[2] Enjie Jiang, “A review of the comparative studies on traditional and intelligent face recognition methods”, (CVIDL), 2020.

[3] Dilpreet Singh Brar, Amit Kumar, Pallavi, Usha Mittal, Pooja Rana, “Face Detection for Real World Application”, IEEE,2021.

[4] Bharath Tej Chinimilli, Anjali T, Akhil Kotturi, h Vihas Reddy Kaipu, Jathin Varma Mandapati “Face Recognition based

Attendance System using Haar Cascade and Local Binary Pattern Histogram Algorithm", IEEE, 2020.Aini, N., Noor, R. M., & Othman, M. S. (2019). A Review of Automated Attendance System Using Biometric: Fingerprints, Irises, and Faces. Journal of Telecommunication, Electronic Computer Engineering, 11(2), 81-86.

[5] Abhishek, K., Agarwal, S., & Gupta, K. (2018). A Deep Learning Framework for Face Recognition with Pose and Illumination Variations. Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), 852-857.

[6] Garvie, C. (2019). The Ethics of Facial Recognition. Journal of Applied Philosophy, 36(3), 306-323.

[7] Gadekar, S. P., Patil, S. M., & Yeole, S. S. (2019). Face Recognition Attendance System Using OpenCV. International Journal of Engineering and Advanced Research Technology, 3(4), 501-504.

[8] Hava, A., Kocaman, I., & Keskin, C. (2017). A Mobile Real-time Attendance System Using Face Recognition on Android Platform. International Journal of Engineering and Technology, 9(8), 2963-2967.

[9] Molla, M. I., Hossain, M. B., & Amin, M. B. (2017). A Comparative Study of Biometric Attendance Systems. International Journal of Computer Applications, 176(8), 29-34.

[10] Noor, A. S. M., Zakaria, N. A., & Abdullah, A. H. (2015). An Efficient Attendance Management System Using Facial Recognition Technique. Applied Soft Computing, 37, 840-848.

# 2. PROJECT DESCRIPTION

**2.1 Reference Algorithm**

1. The algorithm used in the code is a combination of face detection and face recognition techniques. The code uses the face\_recognition library, which is a Python library for performing face recognition. The library is based on the **Local Binary Patterns Histogram (LBPH) algorithm**, which is a popular face recognition algorithm.

2. The code first loads images of known faces and extracts their face encodings using the face\_recognition library. These encodings are then stored in the `known\_face\_encoding` list.

3. Next, the code performs face detection on the video frames captured from the webcam. It uses the `face\_locations` and `face\_encodings` functions from the face\_recognition library to detect faces and extract their encodings.

4. The extracted face encodings are then compared with the known face encodings using the `compare\_faces` function from the face\_recognition library[6]. If a match is found, the corresponding face name is added to the `face\_names` list, and the attendee's name is displayed on the video frame along with the message "Present".

5. The code also keeps track of the students who have attended the class and writes their attendance to a CSV file.

6. The algorithm used in the code is based on the Local Binary Patterns Histogram (LBPH) algorithm for face recognition and the face\_recognition library for face detection and encoding.

## 2.2 Data/Data Structure

The data structures used in the face recognition code are not explicitly defined in the code itself, but they can be inferred from the operations performed on the data. Here are some potential data structures used in the code:

1. Face embeddings: Face embeddings are vector representations of faces that capture their unique features. They are typically generated using a pre-trained deep learning model that takes an image of a face as input and outputs a vector of fixed length. In the code, the face\_recognition library is used to generate face embeddings for known faces and detected faces.

2. Face encodings: Face encodings are similar to face embeddings, but they are typically normalized to have unit length. In the code, the face\_recognition library returns face encodings for known faces and detected faces.

3. Database Management: The system needs to store and manage the known face encodings and corresponding names. In the provided code, this is done using the known\_face\_encoding and known\_faces\_names lists

4. Face locations: Face locations are bounding boxes that specify the location of a face in an image. In the code, the face\_recognition library returns face locations for detected faces.

5. Person objects: Person objects are data structures that hold information about a person, including their face embeddings or encodings. person objects are used to group faces belonging to the same person.

6. CSV files: CSV files are used to store attendance data in the code. Each row in the CSV file represents an instance of attendance for a person.

datetime.now().strftime("%Y-%m-%d"). It then opens the file for writing using csv.writer().

7. Initializing Video Capture: The script then initializes a video capture object by calling cv2.VideoCapture(0), which captures video from the default camera (camera index 0).

## 2.3 Design and Implementation constraints

The program is a Python script that uses the face\_recognition library to perform face recognition on live video from a webcam. Here is a step-by-step explanation of how the code works:

**1. Importing Libraries:** The script starts by importing the necessary libraries, including `cv2` for OpenCV, `numpy` for numerical operations, `csv` for working with CSV files, `os` for interacting with the operating system, and `datetime` for handling date and time.

**2. Initializing Video Capture:** The script then initializes a video capture object by calling `cv2.VideoCapture(0)`, which captures video from the default camera (camera index 0).

**3. Loading Known Faces:** The script loads images of known faces from the `photos` directory using the `face\_recognition.load\_image\_file()` function. For each image, it extracts the face encoding using `face\_recognition.face\_encodings()`. These encodings and corresponding face names are stored in `known\_face\_encoding` and `known\_faces\_names` lists, respectively.

**4. Initializing Variables:** The script initializes empty lists for storing face locations, encodings, and names. It also sets a flag `s` to True, indicating the first frame of the video has been processed.

**5. Creating a CSV File:** The script creates a CSV file with the current date as the filename using `datetime.now().strftime("%Y-%m-%d")`. It then opens the file for writing using `csv.writer()`.

**6. Processing Video Frames:** The script enters a loop that continuously captures video frames using `video\_capture.read()`. For each frame, it resizes the frame to a quarter of its original size using `cv2.resize()`, converts it to RGB format using `cv2.cvtColor()`, and loads it into the `rgb\_small\_frame` variable.

**7. Face Detection:** If the `s` flag is True, the script performs face detection on the resized frame using `face\_recognition.face\_locations()` and `face\_recognition.face\_encodings()`. These functions detect faces and extract their encodings.

**8. Face Matching:** For each detected face, the script compares its encoding to the known face encodings using `face\_recognition.compare\_faces()`. If a match is found, the corresponding face name is added to the `face\_names` list

**9. Displaying Results:** The script then displays the detected faces and their names on the video frame using `cv2.putText()`. If a known face is detected, it prints the name of the face to the console.

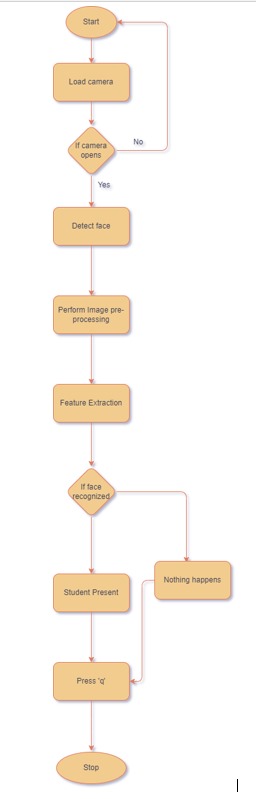
**10. Writing to CSV:** The script writes the detected faces and their corresponding times to the CSV file using `csv.writer().writerow()`.

**11. User Interaction:** The script displays the video frame using `cv2.imshow()` and waits for a 'q' key press using `cv2.waitKey()`.

**12. Cleaning Up:** Finally, the script releases the video capture object, closes all windows, and closes the CSV file.

In summary, the code uses the face\_recognition library to detect faces in real-time video from a webcam, compares them to known faces, and writes the results to a CSV file

**2.4Design Diagram**

****

# 3. SYSTEM REQUIREMENTS

**3.1 System Interface**

The Language we will use is Python.

## 3.2 Database Interface

The Database interface will detect the face and data will be stored in excel.

## 4. NON-FUNCTIONAL REQUIREMENTS

### 4.1 Performance Requirements

### The ML model for attendance prediction must achieve a minimum accuracy of 90%, with a precision and recall of at least 85% each. The model should handle a dataset of up to 10,000 records with a processing time of under 5 seconds per prediction. Additionally, it must be scalable to accommodate future data growth and capable of providing real-time predictions. The system should maintain data privacy and security standards, ensuring compliance with relevant regulations. Finally, the model should undergo regular updates and monitoring to maintain its predictive accuracy and relevance over time.

* 1. **Software Quality Attributes**

1. Accuracy: Ensuring the ML model predicts attendance with high precision and minimal error.

2.Reliability:Consistent performance across various datasets and conditions, maintaining dependable results.

3. Scalability: Ability to handle increasing data volume without compromising prediction accuracy or speed.