**ATTENDANCE MANAGEMENT SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**Varshini Terli (AP22110011242)**

Under the Guidance of

**Dr. Hemantha Kumar Kalluri**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[Nov,2024]**

# Certificate

Date: 29-11-2024

This is to certify that the work present in this Project entitled “**FACE ATTENDANCE MANAGEMENT SYSTEM**” has been carried out by **Varshini Terli** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Designation, Affiliation.

**Acknowledgements**

This project, Attendance Management System Using Face Recognition, would not have been possible without the support and contributions of various individuals and organizations, whom we would like to extend our heartfelt gratitude.

I would like to express my sincere gratitude to **Dr. Hemantha Kumar Kalluri** for his invaluable guidance, support, and expertise throughout the development of this project. Their insightful feedback and encouragement have been instrumental in shaping the direction and success of this endeavor.

I am also grateful to our **project advisors and mentors** for their invaluable guidance, expertise, and encouragement throughout the duration of this project. Their insights and feedback have been instrumental in shaping our approach and achieving our goals.

We would like to acknowledge **SRM University AP** for providing the necessary resources, facilities, and funding to conduct this research. Their commitment to advancing innovation and accessibility in technology has been fundamental to the success of our endeavor.

**Table of Contents**

Certificate..............................................................................................................................2

Acknowledgements...............................................................................................................3

Table of Contents..................................................................................................................4-5

Abstract..................................................................................................................................6

List of Figures………………………………………………………………....……………...7

Introduction............................................................................................................................8

1.1.Introduction to Attendance Management Using Facial Recognition

### **1.2. The Role of Face Recognition Technology**

### **1.3. Enhancing Attendance Processes Through Automation**

### **1.4. Objectives and Scope of the Project**

2. Methodology...................................................................................................................9-14

* 1. Technologies and Libraries…………........................................................................9-10

2.1.1. Open-CV for Computer Vision

2.1.2. NumPy for Numerical Computation

2.1.3. Pandas for Data Management

2.1.4. Random Forest for Classification

2.1.5. Pickle for Model Serialization

2.1.6. Scikit-learn for Machine Learning

2.1.7. Tkinter for User Interface

* 1. Procedure……...............................................................................................................10-11

2.2.1. Data Collection and Pre-processing

2.2.2. Model Training

2.2.3. Attendance Marking

2.2.4. Performance Optimization

2.2.5. Accuracy Calculation

2.2.6. Real-Time Alerts and GUI

* 1. Explanation According to Python Files.....................................................................11-14

2.3.1. Data Collection and Pre-processing

2.3.2. Model Training

2.3.3. Attendance Marking Workflow

2.3.4. Performance Optimization

2.3.5. Accuracy Calculation

2.3.6. Real-Time Alerts and GUI

2.3.7. Continuous Improvement

3.Concluding Remarks..................................................................................................……18

4.Future Work........................................................................................................................19

5.References..........................................................................................................................20

**Abstract**

Understanding and accurately recognizing individuals' identities is crucial in various settings, particularly in educational and professional environments. This paper presents an approach to developing an Attendance Management System Using Face Recognition, aimed at automating attendance tracking and improving accuracy. The proposed system employs Python-based computer vision algorithms along with the OpenCV, Tkinter, and Scikit-Learn packages.

The system is designed to identify and authenticate individuals based on facial recognition, providing a contactless and efficient attendance solution. This paper details the steps involved in image acquisition, pre-processing, feature extraction, model training, and evaluation. The dataset comprises images of individuals captured under diverse conditions, including varying angles, expressions, and lighting scenarios.

For facial feature extraction, Open-CV is utilized to detect key facial landmarks, enabling accurate recognition. Image processing techniques such as scaling, normalization, and filtering are applied to enhance the system's robustness. Scikit-Learn is employed to train machine learning models for facial classification, while Tkinter provides an intuitive user interface for seamless interaction with the system.

The experimental results demonstrate the effectiveness of the face recognition system in accurately identifying individuals and marking attendance. The system’s real-time capabilities ensure quick and reliable attendance tracking, significantly reducing manual errors and saving time in educational and professional contexts.

This research contributes to the development of advanced automated systems that enhance efficiency and convenience in routine tasks. Future work could focus on extending the system’s features, improving its scalability, and integrating it with cloud-based solutions for widespread deployment.

**Keywords**: Face Recognition, Machine Learning, OpenCV, Tkinter, Scikit-Learn, Attendance Management, Automation

**List of figures:**

Figure-1:Interface of Attendance Management System…………………..………….....15

Figure-2 : Subject Details for Automatic Attendance…………………………………15

Figure-3 : Sheet of Automatic Attendance………………..…………………………….15

Figure-4 : Face Recognition for Automatic Attendance………………………………...16

Figure-5 : Subject details for Manual Attendance……………………………………….16

Figure-6 : Student Details for Manual Attendance……………………… ……………...17

Figure-7 : Sheet of Manual Attendance………………………………………………….17

**1.Introduction**

### ****1.1. Introduction to Attendance Management Using Face Recognition****

In today’s fast-paced academic and professional environments, automating routine processes has become essential for efficiency and accuracy. Attendance tracking, a fundamental yet time-consuming task, can benefit immensely from advanced technological solutions. A face recognition-based attendance system provides a seamless and contactless method to ensure accurate attendance records while minimizing manual errors.

### ****1.2. The Role of Face Recognition Technology****

Facial recognition technology offers an innovative approach to attendance management by enabling individuals to be identified and authenticated based on their unique facial features. Leveraging computer vision and machine learning, these systems provide a reliable, non-intrusive solution for automating attendance tracking. This paper focuses on developing a face recognition system tailored for educational and professional settings, where accuracy and real-time processing are paramount.

### ****1.3. Enhancing Attendance Processes Through Automation****

While the project’s primary goal is to automate attendance tracking, its implications extend beyond efficiency. By integrating facial recognition technology, institutions can ensure secure, accurate, and scalable solutions for managing attendance. This eliminates the need for physical registers or manual entry systems, fostering a modernized, streamlined process. The following sections outline the methodologies employed in developing this system using Python, along with key libraries like Open-CV, Tkinter, and Scikit-Learn.

### ****1.4. Objectives and Scope of the Project****

The objective of this project is to design and implement a reliable and efficient system capable of recognizing and authenticating faces to record attendance. This involves a combination of data acquisition, pre-processing, feature extraction, and machine learning techniques. The scope of the project includes:

* Developing a user-friendly interface for data entry and system operation.
* Ensuring high accuracy in facial recognition under diverse conditions.
* Providing real-time attendance tracking and data storage capabilities.
* Offering an option for manual attendance in case of system unavailability.

The proposed system aims to revolutionize traditional attendance management while ensuring accuracy, security, and ease of use.

**2.Methodology**

### ****2.1.Technologies and Libraries:****

#### ****2.1.1. Open-CV for Computer Vision****

Open-CV, an open-source library, forms the backbone of this project’s image processing and computer vision tasks. It facilitates:

* Face detection using Haar Cascade classifiers.
* Scaling, normalization, and feature extraction for training datasets.

#### ****2.1.2. NumPy for Numerical Computation****

NumPy simplifies the handling of multidimensional data arrays. It is crucial for:

* Converting image data into numerical arrays.
* Supporting mathematical computations for model pre-processing.

#### ****2.1.3. Pandas for Data Management****

Pandas handles structured data operations such as:

* Reading and writing attendance logs into .csv files.
* Efficiently organizing and updating datasets.

#### ****2.1.4. Random Forest for Classification****

Implemented using Scikit-learn, Random Forest enhances face classification accuracy by:

* Building multiple decision trees for robust prediction.
* Handling high-dimensional data with minimal parameter tuning.

#### ****2.1.5. Pickle for Model Serialization****

Pickle is utilized for:

* Saving trained machine learning models as binary files.
* Allowing the system to deploy models without retraining.

#### ****2.1.6. Scikit-learn for Machine Learning****

This library streamlines machine learning operations, enabling:

* Pre-processing and feature extraction.
* Training with algorithms like Random Forest.

#### ****2.1.7. Tkinter for User Interface****

Tkinter delivers an interactive GUI, providing access to functionalities such as:

* Capturing images for dataset creation.
* Initiating real-time face recognition.
* Manual attendance updates.

### ****2.2.Procedure:****

#### ****2.2.1. Data Collection and Pre-processing****

* **Image Capture:**
  + Facial images are collected via webcam and stored in a structured hierarchy.
  + Images are converted to gray scale to ensure uniformity in lighting and texture.
* **Face Detection:**
  + Haar Cascade Classifiers detect and crop faces, which are resized for training datasets.

#### ****2.2.2. Model Training****

* **Feature Extraction:**
  + Gray scale images are processed to extract key features using Local Binary Patterns Histogram (LBPH).
  + Unique IDs are assigned to labeled images.
* **Training and Saving:**
  + Models are trained using recognizer.train() and stored as trainner.yml for deployment.

#### ****2.2.3. Attendance Marking****

* **Real-Time Recognition:**
  + A live feed detects faces, matches them against trained models, and logs attendance with timestamps in .csv format.
* **Manual Entry:**
  + GUI options allow for manual adjustments to attendance logs.

#### ****2.2.4. Performance Optimization****

* **Caching Bottleneck Features:**
  + Pre-trained models like MobileNet or Inception v3 enhance feature extraction efficiency.
* **Retraining Mechanism:**
  + Additional training data is incorporated to improve recognition accuracy.

#### ****2.2.5. Accuracy Calculation****

* **Validation:**
  + Test datasets evaluate the model’s predictive accuracy.
* **Metrics:**
  + Metrics such as the ratio of correct to incorrect predictions help refine the system.

#### ****2.2.6. Real-Time Alerts and GUI****

* **Notifications:**
  + Alerts for detected and misclassified faces are provided.
* **User Interaction:**
  + The GUI simplifies interactions like initiating face recognition or updating attendance.

### ****Project Workflow****

1. **Data Collection:** Facial images are captured and organized.
2. **Pre-Processing:** Faces are detected, cropped, and pre-processed.
3. **Model Training:** Features are extracted, and the model is trained and saved.
4. **Attendance Logging:** Real-time recognition logs attendance into .csv files.
5. **System Deployment:** Serialized models enable efficient deployment.
6. **Continuous Improvement:** New data improves model accuracy through retraining.

### ****2.3.Explanation According to Python Files:****

#### ****2.3.1. Data Collection and Preprocessing****

**File: AMS\_Run.py**

* **Image Collection:**
  + The script uses a webcam to capture images of individuals. These images are saved in a folder structure where each folder corresponds to a specific individual, named using their unique ID.
  + Images are converted to grayscale for uniformity and to simplify the data used for feature extraction.
* **Face Detection:**
  + The Haar Cascade Classifier (haarcascade\_frontalface\_default.xml) is employed to detect faces in the captured images.
  + Detected faces are cropped and resized to standard dimensions (e.g., 100x100 pixels), ensuring consistent input data for model training.

#### ****2.3.2. Model Training****

**File: training.py**

* **Feature Extraction:**
  + The script processes the grayscale images to extract important facial features using the **FisherFace Recognizer**, which identifies the variations in the dataset that differentiate one face from another.
  + Each image is labeled with a unique ID based on the filename, linking it to the corresponding individual.
* **Training and Saving:**
  + The recognizer.train() method is used to train the model with extracted features.
  + The trained model is saved as trainner.yml. This file stores the learned patterns and IDs, enabling face recognition during real-time operations.

#### ****2.3.3. Attendance Marking Workflow****

**Files: AMS\_Run.py, meteric.py, testing.py**

* **Real-Time Recognition:**
  + A live video feed from the webcam is processed to detect faces using the Haar Cascade Classifier.
  + Detected faces are compared against the model saved in trainner.yml. Recognition is handled by either the **Local Binary Patterns Histogram (LBPH)** or the **FisherFace Recognizer**.
  + Recognized faces are assigned their corresponding IDs, and attendance is logged with the ID and a timestamp in a CSV file (attendance\_log.csv).
* **Manual Entry:**
  + AMS\_Run.py includes a GUI for manual attendance entry, allowing operators to log attendance for unregistered or unrecognized faces.

#### ****2.3.4. Performance Optimization****

**File: retrain.py**

* **Caching Bottleneck Features:**
  + Transfer learning is implemented using pre-trained models like **Inception v3** or **MobileNet**. These models extract bottleneck features, which are high-level representations of facial data.
  + These features are cached for reuse, significantly reducing computation time during retraining.
* **Model Update:**
  + When new training data is added, bottleneck features are reused, and the top layer of the pre-trained model is fine-tuned to adapt to the updated dataset.

#### ****2.3.5. Accuracy Calculation****

**File: meteric.py**

* **Validation Dataset:**
  + A separate folder containing test images is used for accuracy evaluation. Each test image is labelled with the correct ID.
  + The script processes these images and compares the predictions against the true labels.
* **Accuracy Metrics:**
  + The script calculates accuracy using the formula: Accuracy=Correct Predictions/Total Predictions
  + Logs of correct and incorrect classifications are maintained for debugging and analysis.

#### ****2.3.6. Real-Time Alerts and GUI****

**File: AMS\_Run.py**

* **GUI Implementation:**
  + A Tkinter-based GUI provides an interface for:
    - Capturing images and creating new user datasets.
    - Running the face recognition system in real time.
    - Logging attendance manually.
    - Viewing notifications and system alerts.
* **Alerts and Notifications:**
  + When a face is detected, a notification with the ID and name of the individual is displayed.
  + Errors, such as unregistered faces or system malfunctions, trigger alerts for manual intervention.

#### ****2.3.7. Continuous Improvement****

**Files: training.py, retrain.py**

* **Model Retraining:**
  + New facial data can be added, and the training.py script retrains the model to include the updated dataset.
  + The retrain.py script optimizes this process using pre-trained models and cached features.
* **Error Handling:**
  + The system includes robust error-handling mechanisms for:
    - Missing files (e.g., XML classifiers or model files).
    - Invalid inputs during manual attendance logging.
    - Failures in webcam connectivity or image saving.
  + Error logs are saved for troubleshooting and continuous system improvement.

**List Of Figures:**

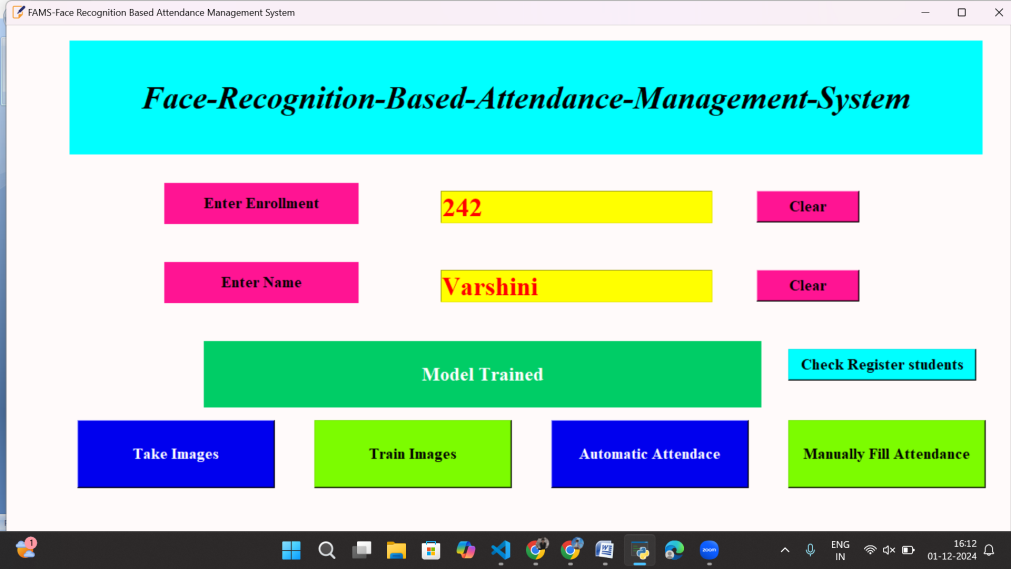


Figure-1: Interface of Attendance Management System

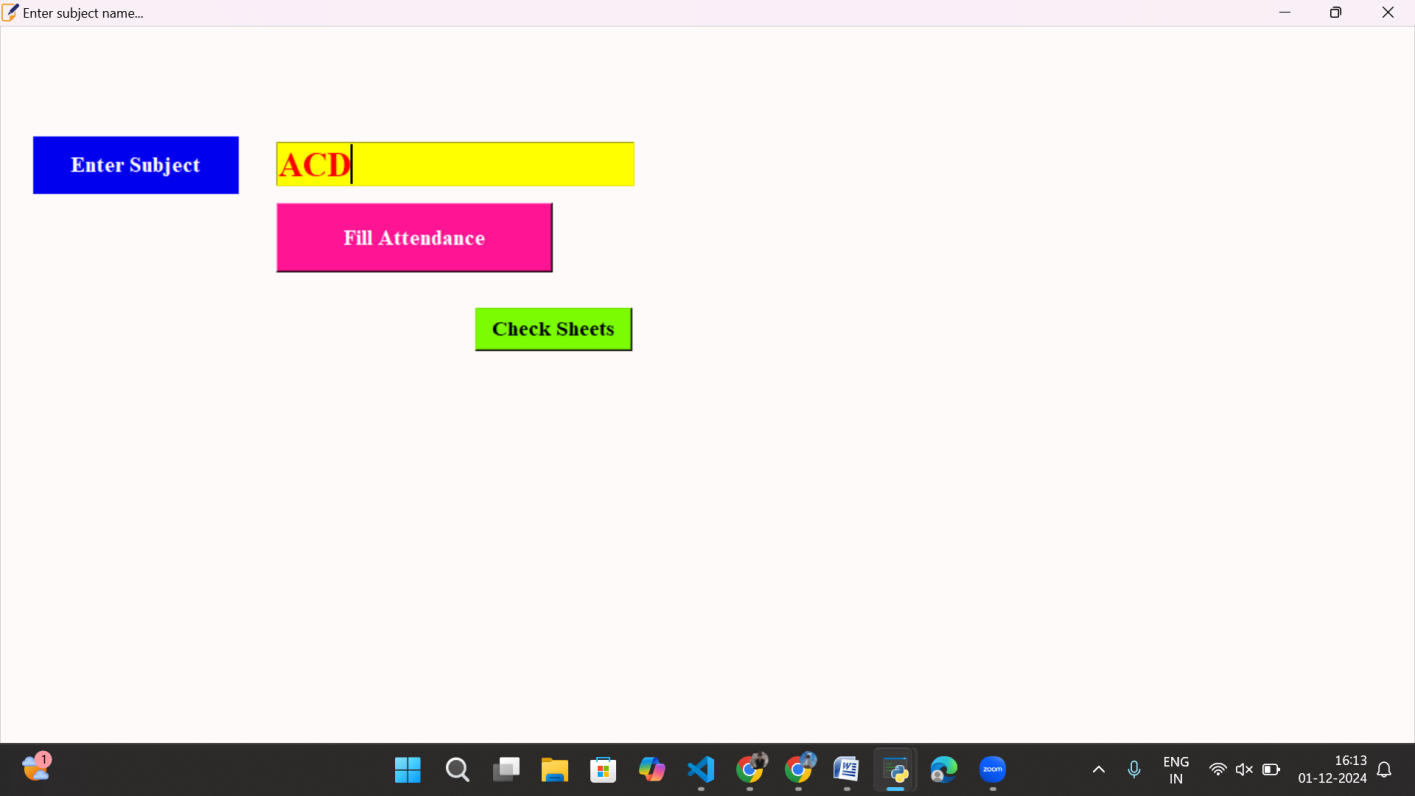


Figure-2 : Subject details for Automatic Attendance

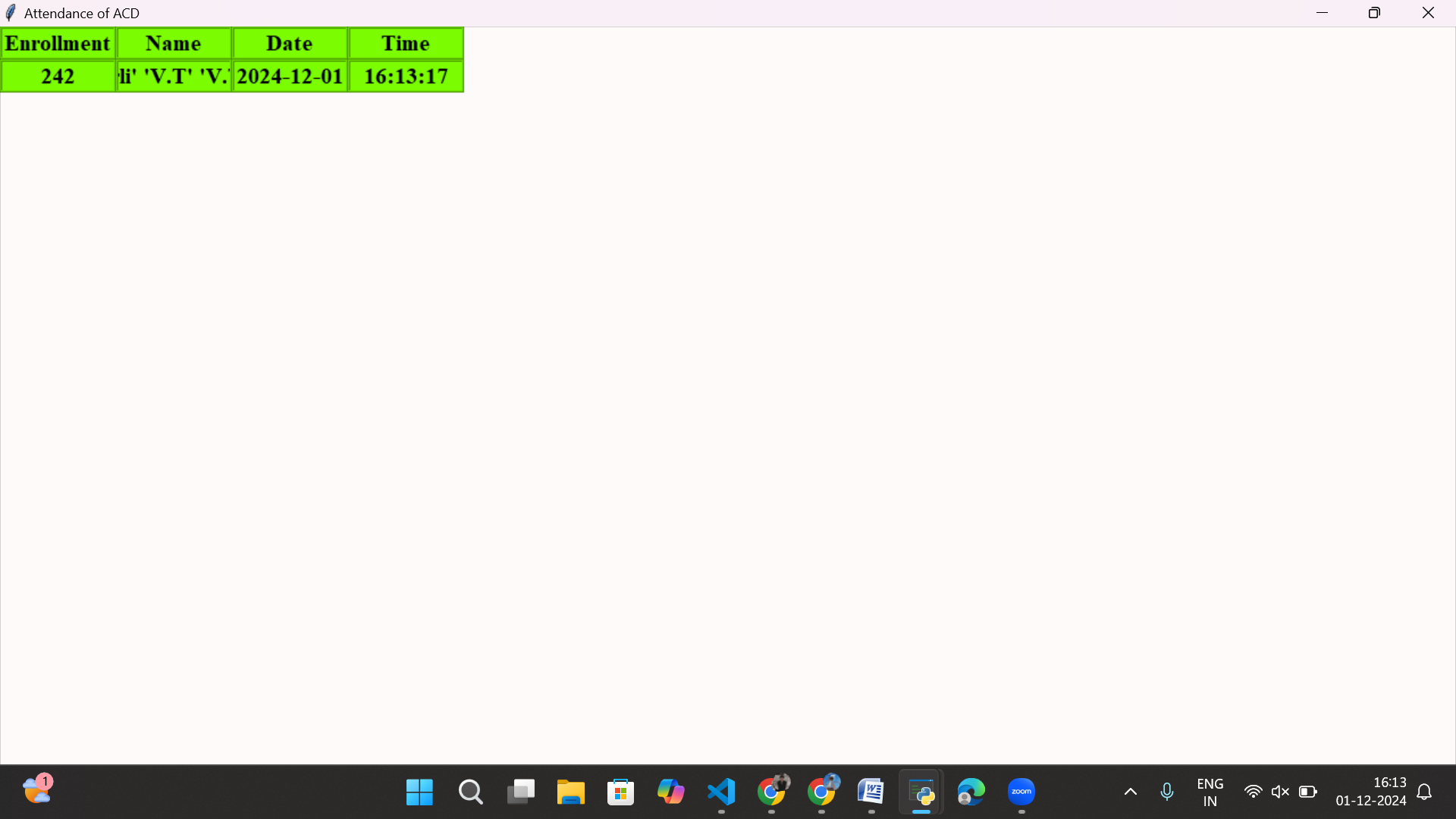


Figure-3 : Sheet of Automatic Attendance

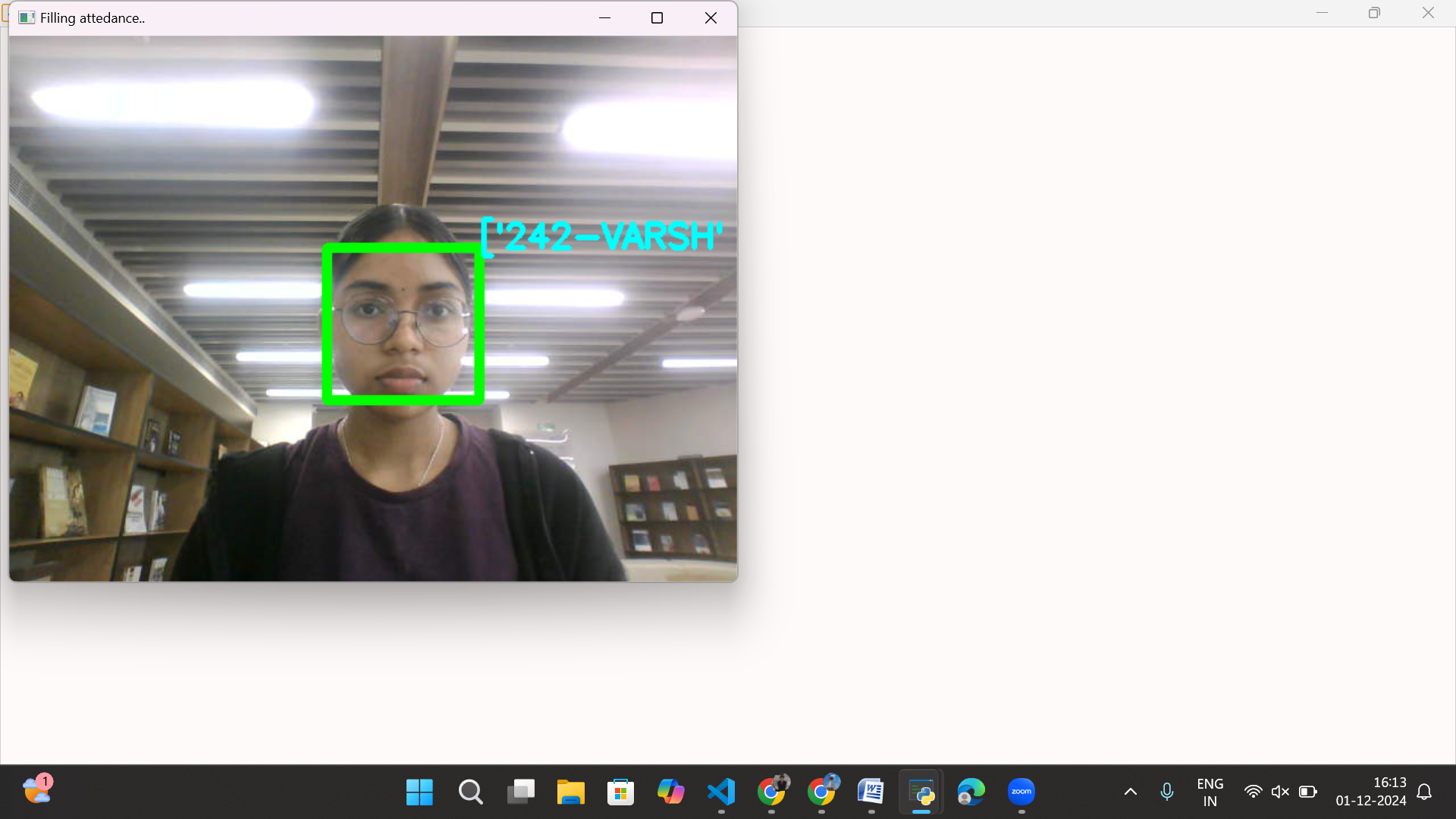


Figure-4 : Face Recognition for Automatic Attendance

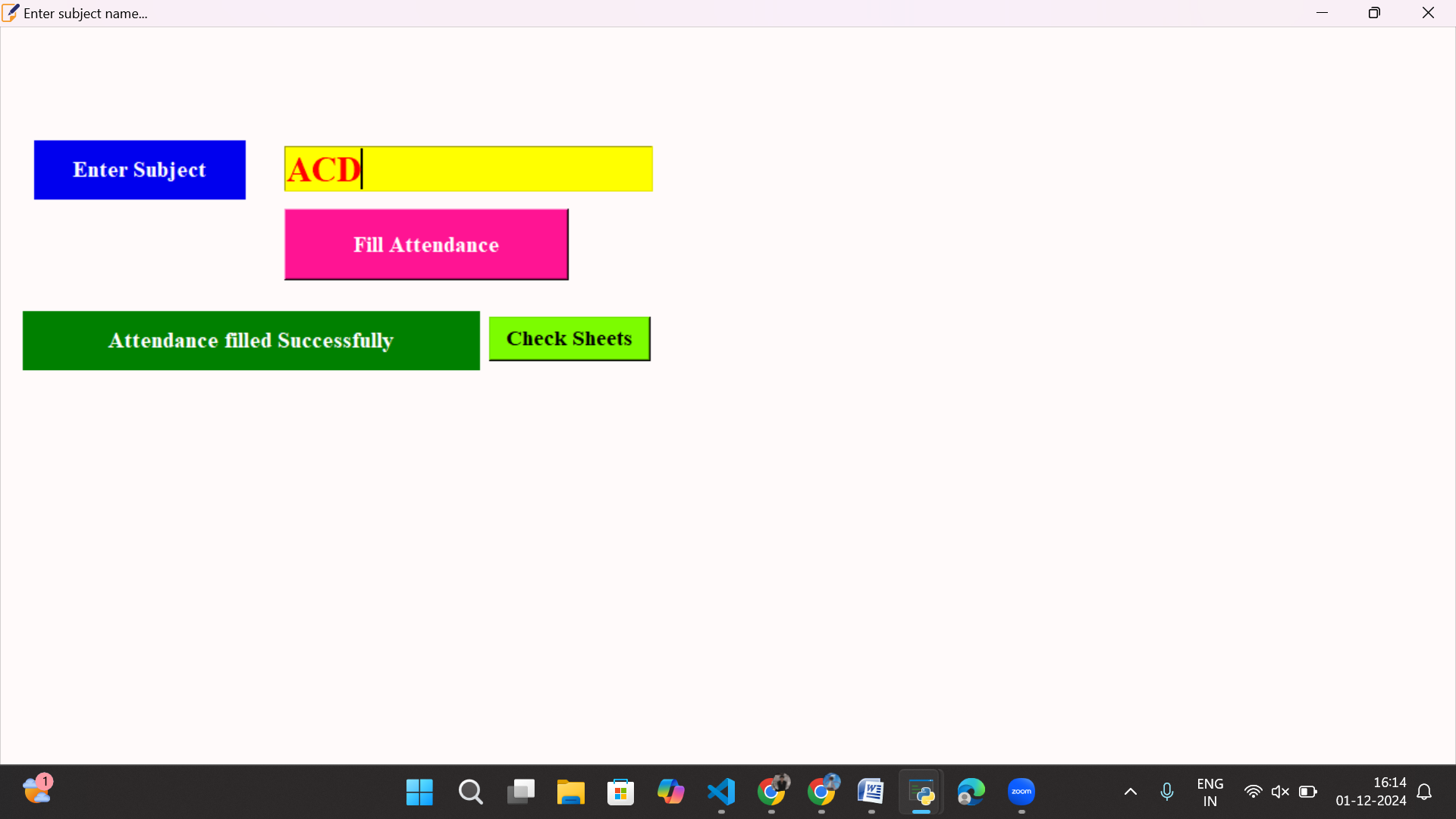


Figure-5 : Subject details for Manual Attendance

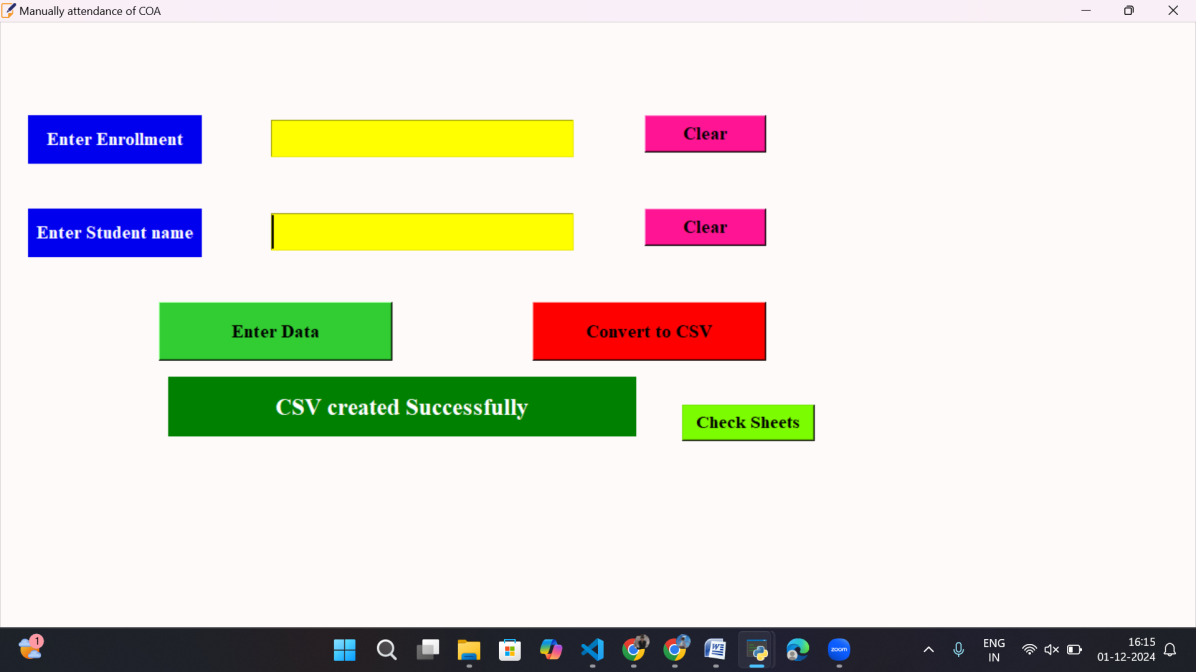


Figure-6 : Student Details for Manual Attendance

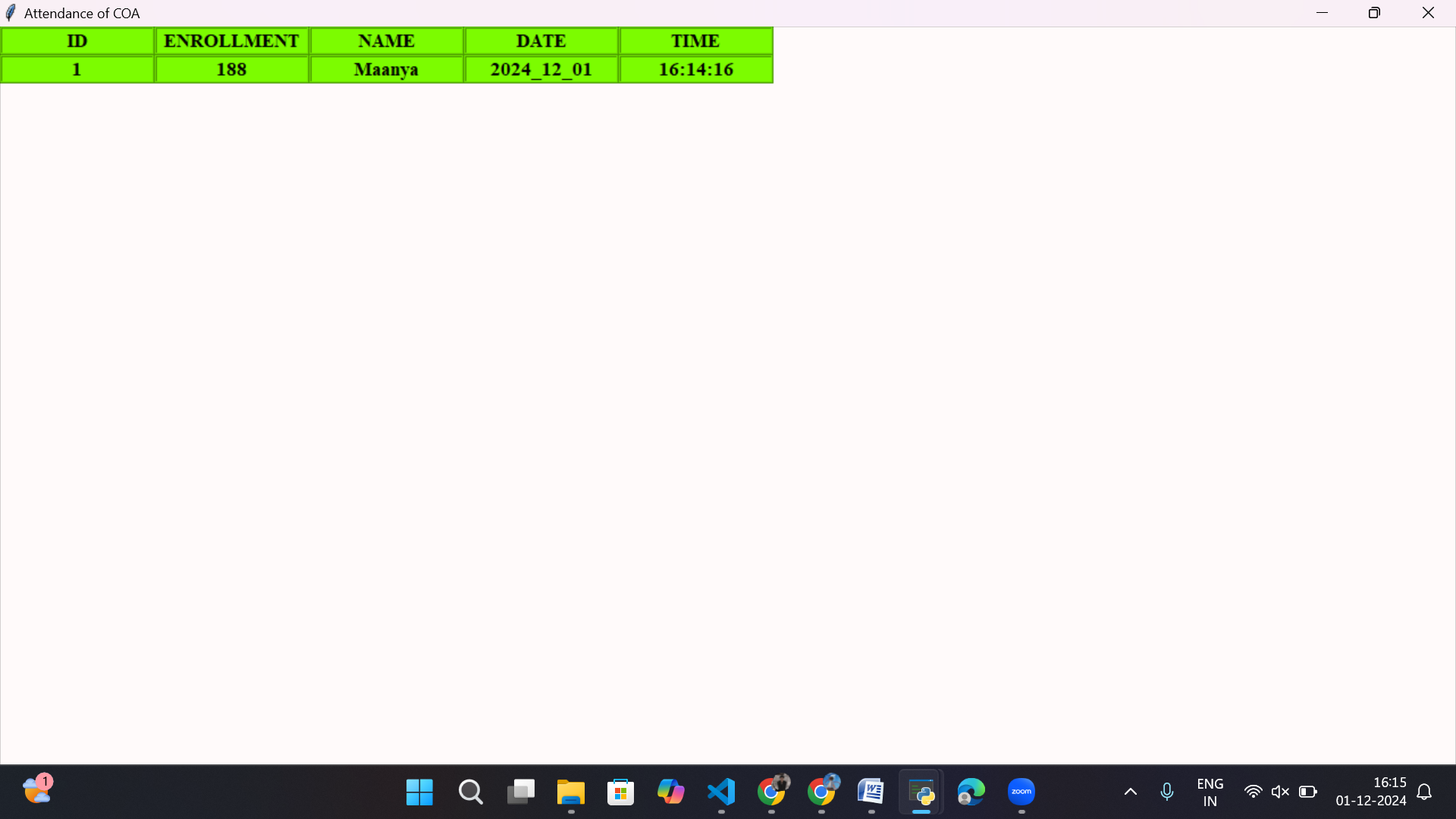


Figure-7 : Sheet of Manual Attendance

**3. Concluding Remarks**

This research explored the development of a face recognition-based attendance management system using Python and machine learning libraries such as Scikit-learn, OpenCV, and others. The primary objective was to create an efficient and automated solution for marking attendance, leveraging the potential of computer vision to replace traditional, manual methods.

The project successfully demonstrated the steps involved in data collection, preprocessing, feature extraction, and model training using the Random Forest algorithm. The chosen libraries and tools provided robust capabilities for capturing and processing facial images, training machine learning models, and integrating these functionalities into an interactive user interface. The importance of a diverse and representative dataset was emphasized as a cornerstone for ensuring accurate and reliable recognition.

Future enhancements could incorporate advanced face recognition techniques, such as deep learning-based models and multimodal data processing, to further improve accuracy and robustness. Integrating the system with cloud-based storage, real-time notification mechanisms, and mobile applications could make it more versatile and user-friendly.

This project highlights the potential of face recognition technology to streamline administrative tasks and improve efficiency in academic and professional environments. By automating attendance marking, the system can save time, reduce errors, and provide a secure, tech-driven solution that aligns with modern institutional needs.

**4.Future Work**

The project lays a solid foundation for hand sign detection systems, but there are several avenues for future development to enhance its capabilities and expand its applications:

1. **Enhanced Context Awareness**
   * **Background Activity Analysis**: Extend the system to analyze common gestures and activities within the camera's field of view, enabling recognition of deviations that might indicate potential issues.
   * **Facial Expression Recognition**: Integrate facial expression analysis to assess individuals' emotional states. When paired with specific hand gestures, this can enhance the detection of distress or incidents like bullying and harassment.
2. **Policy Integration and Support Systems**
   * Align the sign detection system with institutional policies and support frameworks. This could involve creating detailed response protocols, fostering communication among stakeholders, and offering counseling and support services to address bullying, harassment, and safety concerns effectively.
3. **Multi-Modal Surveillance**
   * Expand beyond visual detection by incorporating audio, text, and sensor data to create a comprehensive surveillance system. This approach could provide preventive measures to mitigate risks and offer deeper insights into potential threats before escalation.
4. **Real-Time Alerts and Interventions**
   * **Instant Notifications**: Implement real-time alert systems to immediately notify security staff, guardians, or relevant personnel when pre-defined gestures are detected.
   * **Discreet Interventions**: Enable subtle interventions, such as increasing lighting or playing pre-recorded messages, to discourage inappropriate behaviour or signal support without escalating situations.

These enhancements aim to create a more robust, context-aware system capable of addressing safety and communication challenges in diverse environments. By integrating advanced technologies and aligning with institutional frameworks, the system can significantly contribute to safer and more inclusive spaces.

**5.References**

1. Liu, Y., &Szeliski, R. (2009). Single view metrology in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (<https://doi.org/10.1109/CVPR.2009.5206796>)

2. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. (<https://doi.org/10.1023/A:1010933404324>)

3. Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools. <https://www.drdobbs.com/open-source/the-opencv-library/184404319>)

4. van der Walt, S., Colbert, S. C., &Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2), 22-30. (<https://doi.org/10.1109/MCSE.2011.37>)

5. Python Software Foundation. (2020). Pickle - Python object serialization. Python Software Foundation. (<https://docs.python.org/3/library/pickle.html>)

6. Kulkarni, R. A., &Harati, A. (2020). Hand Gesture Recognition System Using OpenCV. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 9(5), 280-283.