SMART ATTENDANCE SYSTEM USING FACE RECOGNITION

***Submitted by* ADIVIGNESHWARAN K (RA2311026020071) ARUN J (RA2311026020077)**

## OM AKASH M.S (RA2311026020122)

Under the guidance of

**Dr S. MANOHAR**

**Associate Professor**

**Department of Computer Science and Engineering *In partial fulfillment for the award of the degree* of**

## BACHELOR OF TECHNOLOGY

**in**

## COMPUTER SCIENCE AND ENGINEERING

**with specialization in**

## ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

**of**

**FACULTY OF ENGINEERING AND TECHNOLOGY**



**RAMAPURAM, CHENNAI-600089 APRIL 2025**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini Project titled “**SMART ATTENDANCE SYSTEM USING FACE RECOGNITION**” is the bonafide certificate of **ADIVIGNESHWARAN K (RA2311026020071), ARUN J (RA2311026020077), OM AKASH M.S (RA2311026020122)**

of II Year CSE AIML submitted for the course **21CSE251T – DIGITAL IMAGE PROCESSING** for the Academic Year 2024 – 25 Even Semester.

#### SIGNATURE

**Dr.S.MANOHAR**

**Associate Professor**

Computer Science and Engineering

SRM Institute of Science and Technology Ramapuram, Chennai-89

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **TITLE NO.** | **TITLE NAME** | **PAGE NO** |
|  | **ABSTRACT** |  |
| **1** | **INTRODUCTION** | **1** |
| **2** | **SCOPE AND MOTIVATION** | **4** |
| **3** | **DESCRIPTION** | **5** |
| **4** | **ALGORITHM OR PSEUDOCODE** | **9** |
| **5** | **IMPLEMENTATION** | **12** |
| **6** | **OUTPUT** | **17** |
| **7** | **APPLICATIONS** | **19** |
| **8** | **CONCLUSION** | **20** |

## ABSTRACT

The Smart Attendance System Using Face Recognition and Python is an innovative approach to automate attendance tracking through real-time facial recognition technology. This system leverages Python programming along with the face recognition and OpenCV libraries to streamline the attendance process in schools, colleges, and workplaces. The primary objective of this project is to eliminate the manual task of attendance marking, which is often prone to errors and inefficiency. The system utilizes a webcam to capture live video frames, from which it detects and recognizes faces using face recognition. A preloaded dataset of known faces allows the system to match the captured face with an individual in the database. When a match is detected, the system automatically logs the name and timestamp of the recognized person into a CSV file, ensuring accurate attendance records without duplication. The Tkinter library is used to create a user-friendly graphical user interface (GUI) that enables easy interaction with the system. The GUI displays real-time webcam feed, status updates, and offers options to start and stop the webcam. The system works efficiently, providing real-time feedback on whether the webcam is running and attendance is marked successfully. Although the project functions effectively, there are areas for improvement. The system recalculates face encodings every time it runs, which can increase startup time. Additionally, the application lacks the functionality for adding new users through the interface, which would make the system more user-friendly. Furthermore, there is no feature for viewing or exporting attendance data in more convenient formats, such as Excel. This project has practical applications in various settings that require automated attendance tracking, showcasing the potential of AI in simplifying routine tasks. The system can be expanded by adding features such as face registration via camera, admin login, and database integration, making it a deployable solution for organizations.

.

## CHAPTER 1 INTRODUCTION

### Introduction to Smart Attendance System

Attendance management has been a critical and time-consuming task in educational institutions, workplaces, and other organizations. Traditional methods of tracking attendance, such as manual marking or card-based systems, are prone to errors, require significant administrative effort, and are vulnerable to manipulation. With the advancements in technology, particularly in computer vision and facial recognition, there is a growing opportunity to automate and streamline the attendance process, making it more efficient, accurate, and secure. The Smart Attendance System Using Face Recognition and Python seeks to automate this process by leveraging face recognition and OpenCV libraries. The system captures real-time video feed from a webcam, detects faces, and matches them with a preloaded dataset of known individuals. Upon recognition, it automatically logs the name and timestamp into a CSV file, eliminating manual attendance marking. A user-friendly interface built with Tkinter allows users to start and stop the webcam and view live updates. This system offers an efficient, accurate, and secure solution for attendance tracking, with potential for future improvements like user registration and data export.

### System Overview

The Smart Attendance System Using Face Recognition and Python is an advanced, automated solution designed to revolutionize attendance tracking. By leveraging cutting-edge facial recognition technology, the system eliminates the manual effort traditionally associated with attendance marking, ensuring accuracy and efficiency. The system utilizes OpenCV to capture live video from a webcam and face recognition to process the captured frames. It detects faces within the video feed, compares them to a preloaded dataset of known individuals, and accurately identifies each person. Upon successful identification, the system records the individual’s name and timestamp in a CSV file, ensuring no duplicate attendance entries are made.

The system features a Tkinter-based graphical user interface (GUI) that provides users with a simple and intuitive platform to interact with the system. Through this interface, users can start and stop the webcam, view the live video stream, and receive real-time updates about the system's status, such as whether the webcam is running or if attendance has been successfully recorded.

#### Key features of the system include:

* + - **Real-Time Face Recognition**: The system recognizes faces in real time, making attendance tracking quick and efficient.
    - **Automatic Attendance Logging**: Once a face is identified, the system automatically logs the person’s

name and timestamp in a CSV file, ensuring accurate and reliable attendance data.

* + - **User-Friendly Interface**: The clean and simple interface allows users to easily start/stop the webcam and monitor the system’s status.
    - **Enhanced Security**: The system can be further enhanced with user authentication features, limiting access to authorized personnel only.

### Technological Foundation

The system's effectiveness stems from its technological underpinnings:

1. **OpenCV**: Used for capturing and processing live video from the webcam. It handles image resizing, conversion to RGB format, and real-time video display
2. **Tkinter**: The GUI framework used for building a user-friendly interface. It provides buttons for controlling the webcam and displays real-time status updates and video feed.
3. **NumPy**: Utilized for efficient array operations, aiding in image processing and face recognition tasks.
4. **Datetime**: Helps in generating accurate timestamps for attendance logs.

### Impact of Smart Attendance Process

The Smart Attendance System Using Face Recognition and Python has the potential to significantly improve attendance tracking in educational institutions, workplaces, and other organizations. By automating the process of attendance marking, it reduces human error and eliminates the need for manual intervention, saving time and effort.

#### Key impacts include:

1. **Increased Efficiency**: The system automates attendance marking, ensuring quicker, more accurate records, and reducing administrative workload.
2. **Accuracy and Reliability**: Facial recognition minimizes errors, ensuring only authorized individuals are marked present.
3. **Convenience**: With real-time tracking, users no longer need to manually mark attendance, making the process seamless and convenient.
4. **Scalability**: The system can be easily expanded to support larger datasets and integrated into various environments, such as large offices or schools.
5. **Enhanced Security**: By utilizing face recognition, the system provides a higher level of security compared to traditional methods like paper logs or PIN-based systems.

### Broader Implications for Workforce Management

The Smart Attendance System Using Face Recognition and Python has significant implications for modern workforce management, particularly in industries and sectors where accuracy, security, and efficiency are critical.

1. **Streamlined Time and Attendance Tracking**: By automating attendance with facial recognition, the system reduces administrative overhead associated with traditional timekeeping methods. Employees no longer need to manually clock in or out, eliminating the possibility of time fraud, and ensuring accurate records are maintained for payroll and performance tracking.
2. **Enhanced Security and Access Control:** The face recognition technology provides an additional layer of security for sensitive work environments. It helps ensure that only authorized personnel are allowed to access restricted areas or systems, contributing to overall workplace safety.
3. **Improved Employee Accountability:** The automatic and real-time attendance logging ensures that every employee is held accountable for their attendance, which can improve punctuality and productivity. Employers can easily track patterns of absenteeism, tardiness, or other behaviors, allowing for more data-driven decision-making.
4. **Data-Driven Insights for HR**: The system provides a wealth of data that can be used for performance reviews, identifying trends in employee attendance, and managing human resources more effectively. HR teams can analyze attendance records to gain insights into employee behavior and optimize staffing, scheduling, and resource allocation.
5. **Reduction in Administrative Costs:** By automating attendance, organizations can reduce costs associated with manual time tracking, including paper-based systems, administrative salaries, and data entry errors. This contributes to more efficient operations and allows human resources teams to focus on higher-value tasks.

## CHAPTER 2

**SCOPE AND MOTIVATION**

### Project Scope

The Smart Attendance System Using Face Recognition and Python aims to automate the process of tracking attendance through facial recognition technology. The scope of the project includes the development of a real- time face detection system using a webcam, powered by OpenCV and the face\_recognition library. The system will capture live video, process the images to detect faces, and match them with a preloaded dataset of known individuals. Upon successful recognition, the system will automatically record the individual’s name and timestamp in a CSV file, eliminating the need for manual attendance marking. The project will also include the creation of a user-friendly graphical user interface (GUI) using Tkinter, allowing users to start/stop the webcam and view live video feed. Additionally, the system will ensure that attendance entries are not duplicated by checking for previously recorded names in the dataset. Future enhancements could include the ability to add new users directly through the GUI, export attendance records to Excel, and incorporate user authentication for secure access to the attendance data. The system is designed to be scalable, capable of handling large datasets and adaptable to different environments, such as schools, offices, and other institutions.

### Project Motivation

The motivation behind developing the Smart Attendance System Using Face Recognition and Python stems from the need for a more efficient, accurate, and secure method of tracking attendance. Traditional manual attendance systems are prone to errors, time-consuming, and vulnerable to issues like proxy attendance, which can compromise the integrity of records. With the growing reliance on technology, there is a clear demand for automation in routine administrative tasks like attendance marking. Face recognition technology offers a promising solution by providing a highly accurate and secure means of identifying individuals. By incorporating facial recognition into attendance tracking, this project aims to eliminate human errors, reduce the risk of attendance fraud, and save valuable time for both students and employees. Furthermore, with the increasing shift toward digitization in education and the workplace, this system supports the transition to paperless, streamlined operations, making it a valuable tool for modern institutions. The project also addresses the need for scalability and adaptability, as it can be implemented in a variety of environments, from classrooms to offices. By integrating computer vision and machine learning technologies, the system not only solves a real-world problem but also demonstrates the practical applications of these advanced technologies in everyday scenarios

## CHAPTER 3 DESCRIPTION

### Overview of the System

The system is composed of several key modules: the face detection and recognition module, the real-time webcam feed, the attendance logging module, and the graphical user interface (GUI). These modules work together to provide a seamless and efficient experience for the user. The overall architecture of the system can be broken down into the following components:

* **Webcam Interface**: Captures live video stream for facial recognition.
* Face Recognition Engine: Detects and recognizes faces using preloaded datasets.
* **Attendance Logger:** Records the identified individual’s attendance in a CSV file with timestamp.
* **User Interface:** Allows users to control the webcam and monitor the system’s status through a Tkinter-based interface.

### System Requirements

Before diving into the design, it is essential to outline the system requirements:

### Software Requirements

* + - * **Python**: The core programming language for building the system.
      * **OpenCV**: A library for computer vision tasks, used for handling the webcam input, video streaming, and image processing.
      * **Face recognition**: A Python library built on top of dlib that provides simple methods to identify faces from an image or video feed.
      * **Tkinter**: The GUI framework used to build the interface, allowing users to interact with the system easily.
      * **Pillow (PIL)**: Used to convert OpenCV images into a format compatible with Tkinter for display.
      * **CSV Module**: For managing attendance data storage in CSV format.
      * **NumPy**: For array and numerical operations, which helps in managing face encodings and image arrays.

### Hardware Requirements

* + - * **Webcam**: Used to capture the video feed for real-time facial recognition.
      * **Computer with Python environment**: Any basic computer system with Python installed, with enough computational power to handle image processing tasks in real time.

### System Architecture

The architecture of the system can be represented in a modular form, where each component performs a distinct function. The following diagram provides a high-level view of how these modules interact:

### Module 1: Webcam Interface

The webcam interface captures real-time video from the user’s camera. OpenCV is used to handle video frames, allowing for continuous streaming. The camera feed is resized to reduce processing time and enhance performance. It is important to note that the webcam stream must be continuously updated to detect faces in each frame.

### Module 2: Face Recognition

Once the video feed is captured, the face\_recognition library is used to identify faces in the frames. The find encodings function is employed to extract facial encodings for comparison. These encodings are unique representations of a person’s face, stored and matched against a pre-loaded dataset. The system searches for matches between the captured faces and known faces from the dataset.

This module works in real time, constantly comparing newly detected faces with the known dataset. When a match is found, the system proceeds to mark attendance for that individual.

### Module 3: Attendance Logging

After a face is successfully recognized, the Attendance Logger module records the individual’s name along with the timestamp in a CSV file. The system checks for duplicates by looking at the existing entries in the CSV file. If an entry with the same name exists, no new entry is made, preventing the marking of duplicate attendance.

This is a critical part of the system, as it ensures that only one attendance record is created per individual for each session.

### Module 4: Graphical User Interface (GUI)

The user interacts with the system via the GUI built using Tkinter. The interface allows users to start or stop the webcam, view the live video feed, and receive real-time status updates about the face recognition process. The GUI is designed to be simple and intuitive, making it easy for users to control the system.

The status label provides updates such as “Webcam Active” or “Attendance Marked”, allowing users to track the system’s progress. The buttons to start and stop the webcam interface are also included in

the GUI.

### Data Flow

The data flow in the system begins with the webcam capturing the video feed. Each frame from the webcam is passed through the face recognition engine to detect faces. If a face is detected and matches an entry in the preloaded dataset, the system logs the attendance in a CSV file.

The overall process can be summarized in the following steps:

1. The webcam captures a frame.
2. The frame is processed by the face recognition system to detect and encode faces.
3. The system compares the detected face with the preloaded dataset.
4. If a match is found, the system marks attendance by recording the individual’s name and timestamp.
5. The real-time status is updated on the GUI for the user.

### System Workflow

A step-by-step breakdown of the workflow:

1. **Start Webcam:** The user presses the "Start Webcam" button on the Tkinter interface.
2. **Capture Video Feed**: The webcam starts capturing frames continuously.
3. **Face Detection and Recognition:** Each frame is processed to detect faces. If a face matches any of the known faces, the system identifies the person and proceeds to mark attendance.
4. **Mark Attendance:** The system logs the name and timestamp of the recognized individual in the CSV file.
5. **Update GUI:** The status label updates to indicate that attendance has been marked successfully.

### Database and Data Storage

For simplicity, the attendance data is stored in a CSV file. Each entry in the CSV consists of the individual’s name and the timestamp of when their attendance was marked. This structure is easy to maintain and can be quickly expanded in the future to store additional information.

Name, Timestamp

John Doe, 2025-04-19, 10:15:30

Jane Smith, 2025-04-19, 10:16:45

### Error Handling and Edge Cases

The system is designed to handle various errors gracefully:

* No Face Detected: If no face is detected in the webcam feed, the system will continue to run and wait for a face to appear.
* Duplicate Attendance: The system checks for duplicate attendance entries before writing to the CSV file.
* Failed Webcam Initialization: If the webcam fails to initialize, the system provides a clear error message through the GUI.

### Security Considerations

While the system offers an efficient method for tracking attendance, it lacks advanced security features such as encryption for the attendance data or user authentication. Future iterations could include:

* Encryption: To ensure data privacy, attendance records can be encrypted before storage.
* Admin Authentication: The system can include an authentication system to restrict access to attendance data or settings.

## CHAPTER 4 ALGORITHM OR PSEUDOCODE

**4.1 PSEUDOCODE**

Pseudocode is a simplified, informal way of writing the logic or steps of a program using plain language and basic programming structure. It isn’t actual code that a computer can run, but it's used to plan and visualize algorithms before writing real code in any programming language.

### Step 1: System Initialization

**Objective**: Initialize the system, set up libraries, and load necessary resources.

#### Initialize Libraries:

Import required libraries such as cv2, face\_recognition, os, time, and tkinter

#### Load Pre-recorded Face Data:

Load a set of known face encodings from a directory of pre-stored images (one image per known individual).

#### Initialize the GUI:

Set up the user interface for interacting with the system, using Tkinter for webcam control and status updates.

#### Start Webcam Feed:

Open the webcam using OpenCV and start capturing frames.

### Step 2: Webcam Frame Capture

**Objective**: Capture video frames and continuously feed them into the system.

#### Start Frame Capture Loop:

Begin a loop to continuously capture frames from the webcam.

#### Convert the Frame to RGB:

OpenCV captures frames in BGR format, but the face recognition library expects RGB. Convert the frame accordingly.

#### Step 3: Face Detection and Recognition

**Objective**: Detect faces in the captured frame and recognize them by comparing to the known face dataset.

#### Detect Faces in the Frame:

Use the face\_recognition library to detect faces in the current frame.

#### Extract Face Encodings:

Extract encodings for the detected faces in the frame.

#### Compare with Known Faces:

For each detected face encoding, compare it with the stored encodings of known individuals.

#### Record Attendance for Recognized Faces:

If a match is found, mark the person’s attendance and avoid duplication.

### Step 4: Display Live Video Feed and Status Updates

**Objective**: Show live video feed with detected faces and real-time status updates.

#### Display Video Feed with Face Locations:

Draw rectangles around the detected faces on the live video feed to show where the faces were detected.

#### Display Name for Recognized Faces:

Display the recognized name above the detected face.

#### Update GUI with Status:

Continuously update the GUI with status, e.g., "Attendance marked for [name]" or "No face detected".

#### Display Frame in GUI:

Convert the OpenCV frame to a format that Tkinter can display.

### Step 5: Save and Export Attendance Data

**Objective**: Store the attendance data in a CSV file and export it when required.

#### Attendance Log Format:

Each attendance record is saved with the name and timestamp in a CSV file.

#### Export Attendance:

Provide a button in the GUI to export the logged attendance data.

**Step 6: Handle User Interface Actions Objective**: Handle user input via the Tkinter interface. **Start/Stop Webcam**:

Use a button in the Tkinter interface to start or stop the webcam feed.

#### Handle Attendance Export:

Implement a button to allow the user to export the attendance log.

### Step 7: Error Handling and System Shutdown

**Objective**: Handle errors, clean up resources, and shut down the system properly.

#### Handle Webcam Errors:

If the webcam cannot be opened, display an error message.

#### Close Resources:

Properly release the webcam and close any open resources upon exiting the program.

### Step 8: Final Testing and Deployment

**Objective**: Test the system with multiple users and deploy the application.

#### Test with Multiple Faces:

Ensure the system correctly identifies and logs multiple users.

#### System Deployment:

Package the system for deployment to the target environment. This can include creating an executable or preparing the system for cloud deployment.

## CHAPTER 5 IMPLEMENTATION

import face\_recognition import cv2

import numpy as np import os

from datetime import datetime import tkinter as tk

from tkinter import messagebox from PIL import Image, ImageTk

# Load images from dataset path = 'dataset'

images = [] names = []

for filename in os.listdir(path):

img = cv2.imread(os.path.join(path, filename)) if img is not None:

images.append(img) names.append(os.path.splitext(filename)[0])

def find\_encodings(images): encoding\_list = []

for img in images:

rgb\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) encodes = face\_recognition.face\_encodings(rgb\_img)

if encodes: encoding\_list.append(encodes[0])

return encoding\_list

known\_encodings = find\_encodings(images) print("Encoding Complete!")

# Attendance marker

def mark\_attendance(name):

with open('Attendance.csv', 'a+') as f: f.seek(0)

lines = f.readlines()

name\_list = [line.split(',')[0] for line in lines] if name not in name\_list:

now = datetime.now()

dt\_string = now.strftime('%Y-%m-%d,%H:%M:%S') f.write(f'{name},{dt\_string}\n')

# Global variable for webcam cap = None

# Create Tkinter Window root = tk.Tk()

root.title("Smart Attendance System") root.geometry("800x600") root.configure(bg="#f0f0f0")

# Title label

title = tk.Label(root, text="Smart Attendance System", font=("Helvetica", 24, "bold"), bg="#f0f0f0", fg="#333")

title.pack(pady=10)

# Status Label

status\_label = tk.Label(root, text="Status: Ready", font=("Arial", 16), bg="#f0f0f0", fg="green") status\_label.pack(pady=10)

# Label to show video stream

video\_label = tk.Label(root) video\_label.pack(pady=10)

# Video frame update function def update\_frame():

global cap

if cap is None or not cap.isOpened(): status\_label.config(text="Status: Webcam not active", fg="red") return

success, frame = cap.read() if not success:

status\_label.config(text="Status: Failed to capture frame", fg="red") return

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25) rgb\_small\_frame = cv2.cvtColor(small\_frame, cv2.COLOR\_BGR2RGB)

faces\_cur\_frame = face\_recognition.face\_locations(rgb\_small\_frame)

encodes\_cur\_frame = face\_recognition.face\_encodings(rgb\_small\_frame, faces\_cur\_frame)

for encode\_face, face\_loc in zip(encodes\_cur\_frame, faces\_cur\_frame): matches = face\_recognition.compare\_faces(known\_encodings, encode\_face)

face\_distances = face\_recognition.face\_distance(known\_encodings, encode\_face) best\_match\_index = np.argmin(face\_distances)

if matches[best\_match\_index]:

name = names[best\_match\_index].capitalize() y1, x2, y2, x1 = [v \* 4 for v in face\_loc]

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2) mark\_attendance(name)

status\_label.config(text=f"Status: Attendance marked for {name}", fg="green")

# Convert OpenCV image to ImageTk format and display it in Tkinter cv2image = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

img = Image.fromarray(cv2image)

imgtk = ImageTk.PhotoImage(image=img) video\_label.imgtk = imgtk video\_label.configure(image=imgtk) video\_label.after(10, update\_frame)

# Start Webcam Button def start\_webcam():

global cap

if cap is None or not cap.isOpened(): cap = cv2.VideoCapture(0)

if cap.isOpened():

status\_label.config(text="Status: Webcam Started", fg="blue") update\_frame()

else:

status\_label.config(text="Status: Failed to start webcam", fg="red")

# Stop Webcam Button def stop\_webcam():

global cap

if cap and cap.isOpened(): cap.release()

video\_label.config(image='') # Clear video display status\_label.config(text="Status: Webcam Stopped", fg="red")

# Buttons

button\_frame = tk.Frame(root, bg="#f0f0f0") button\_frame.pack(pady=20)

start\_button = tk.Button(button\_frame, text="Start Webcam", font=("Arial", 14), bg="#4CAF50", fg="white",

width=15, command=start\_webcam) start\_button.grid(row=0, column=0, padx=20)

stop\_button = tk.Button(button\_frame, text="Stop Webcam", font=("Arial", 14), bg="#f44336", fg="white", width=15, command=stop\_webcam)

stop\_button.grid(row=0, column=1, padx=20)

# Run the GUI loop root.mainloop()

## CHAPTER 6 OUTPUT

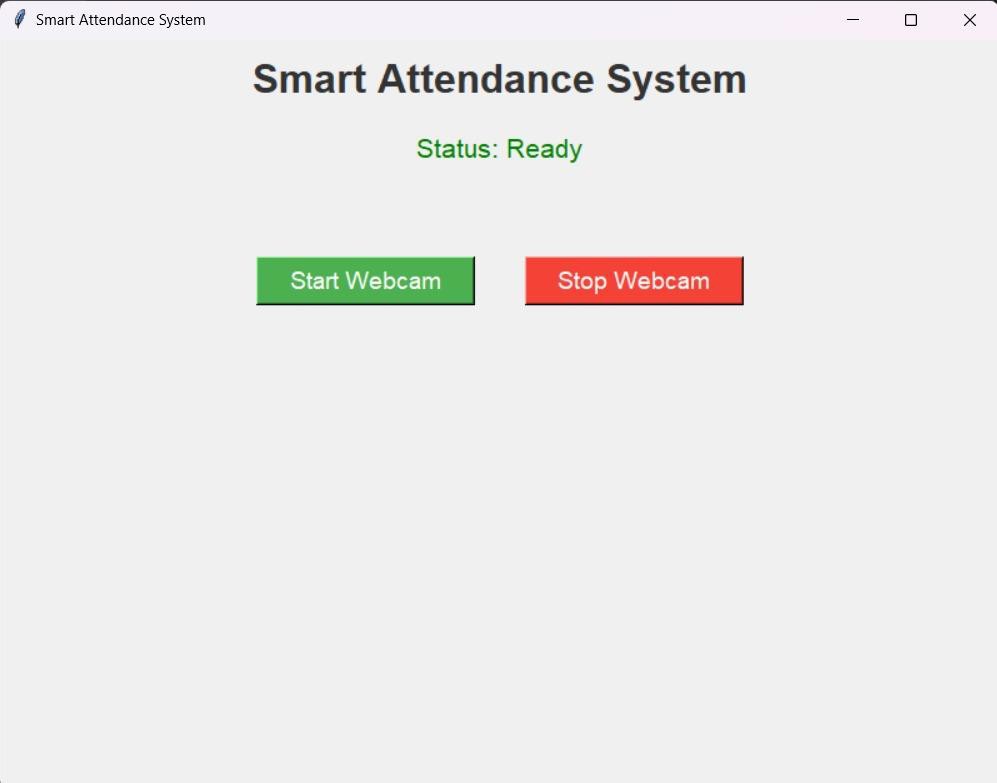


Fig 6.1

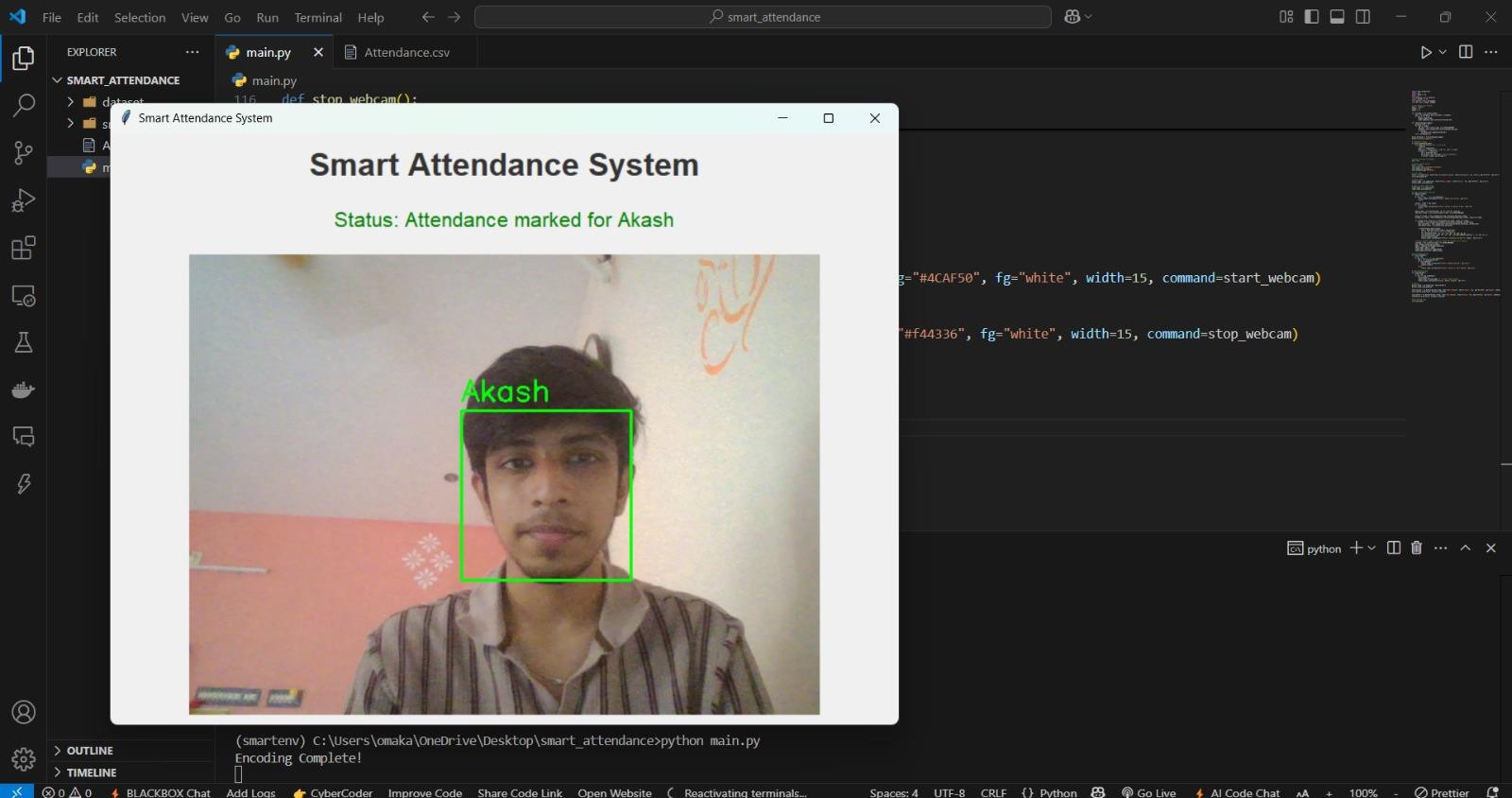


Fig 6.2

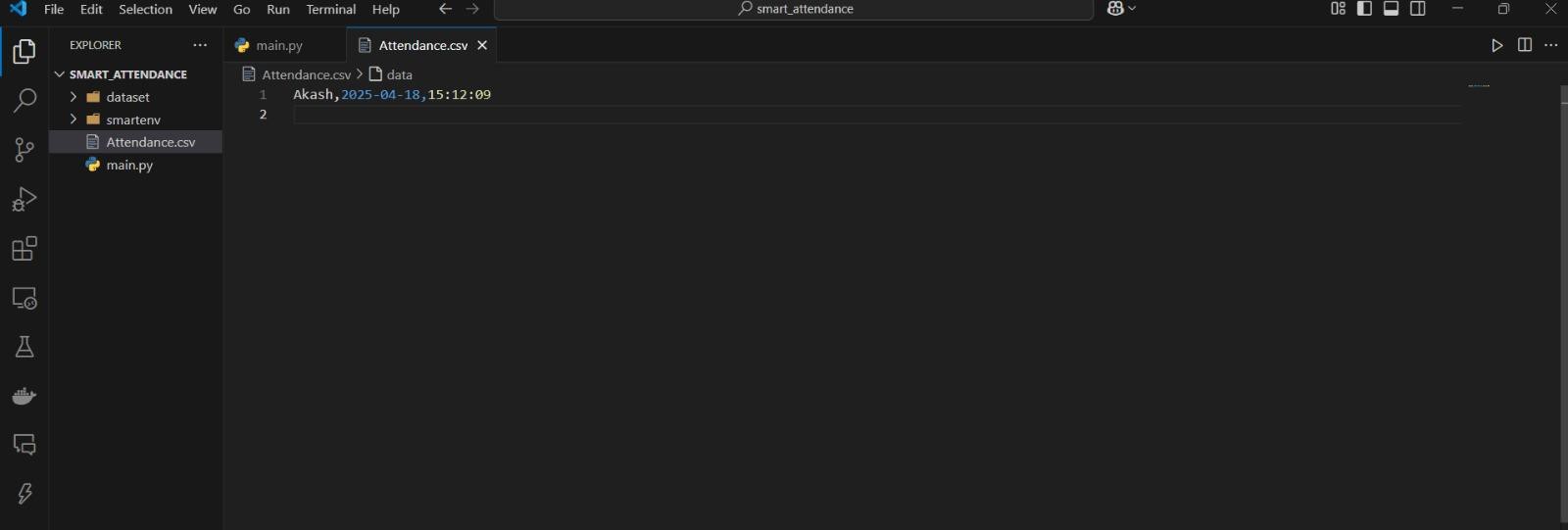


Fig 6.3

## CHAPTER 7 APPLICATIONS

#### Educational Institutions:

Schools, colleges, and universities can use this system to automate student attendance, saving time and reducing manual errors during roll calls.

#### Corporate Offices:

Companies can deploy this for employee attendance tracking, eliminating the need for biometric or manual sign-in systems and improving workplace security.

#### Examination Halls:

It can be used to verify student identity during exams to prevent impersonation and ensure fair practices.

#### Seminars & Conferences:

For events with large numbers of attendees, the system can quickly record participants' presence without human intervention.

#### Co-working Spaces:

Shared office environments can use this system to track members' usage patterns and enhance billing accuracy.

#### Secure Areas:

In high-security zones, this system can serve as a double layer of verification along with ID cards or QR codes.

#### Healthcare Facilities:

Hospitals and clinics can use it to manage staff shifts and verify authorized personnel entry in restricted zones.

#### Factories and Industrial Sites:

Attendance for laborers and staff can be recorded even with minimal interaction, useful in environments requiring hygiene or safety gear.

#### Government and Public Sector:

Offices and departments can reduce paper-based attendance and improve accountability among employees.

#### Libraries or Labs:

Used to track individual access times, helping manage resources and time limits efficiently.

## CHAPTER 8 CONCLUSION

The Smart Attendance System using Face Recognition provides an efficient, secure, and contactless method for marking attendance. By integrating technologies like OpenCV, face recognition, and Tkinter, it automates the entire process with real-time face detection and recognition. This system reduces manual effort, prevents proxy attendance, and ensures accurate record-keeping, making it a valuable solution for schools, offices, and other institutions aiming to modernize their attendance tracking. It enhances user convenience by requiring no physical interaction. The system is scalable and can be improved with features like cloud integration and analytics.

Overall, it represents a step forward in applying AI for daily administrative tasks. With further development, it can support multiple cameras and real-time remote monitoring. Such innovations pave the way for smarter, AI- driven workplace and academic management systems.