INTRODUCTION TO PYTHON

"ATTENDANCE SYSTEM WITH FACE RECOGNITION"

A THESIS

SUBMITTED BY

VENKATA SATYA **(CB.EN.U4AIE22005)**

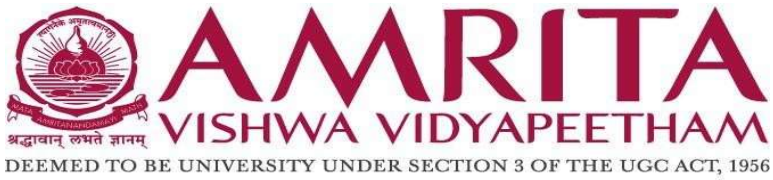
KOLLA LOKESH **(CB.EN.U4AIE22027)**

GOLI SURYA TEJA **(CB.EN.U4AIE22069)**

RAMA KRISHNA PRASAD **(CB.EN.U4AIE22070)**

IN PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY IN CSE (AI)

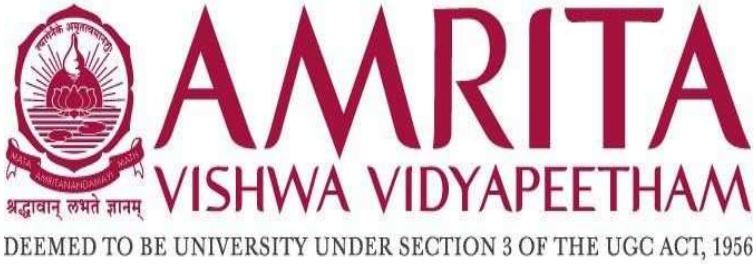


CENTRE FOR COMPUTATIONAL ENGINEERING AND NETWORKING

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITAVISHWAVIDYAPEETHAM
COIMBATORE-641112(INDIA)

DECEMBER-2023

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE-641112



BONAFIDE CERTIFICATE

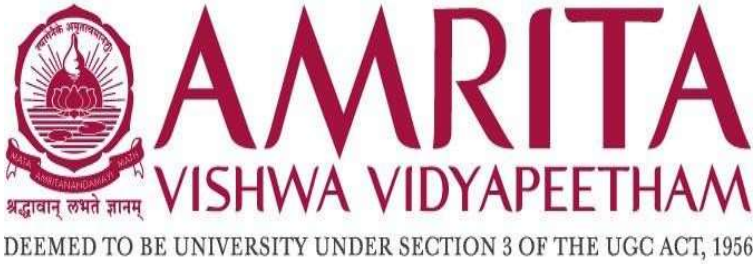
This is to certify that the thesis entitled "**ATTENDANCE SYSTEM WITH FACE RECOGNITION**" submitted by group-12 of batch(A), for the award of the Degree of Bachelor of Technology in the "CSE(AI) " is a bonafide record of the work carried out by her under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Ms.Sreelakshmi K
Project Guide

Dr. K.P.Soman
Professor and Head of CEN Department

Submitted for the university examination held on 20-12-2023

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE-641112



DECLARATION

We,group-12 of batch(A),hereby declare that this thesis entitled "**ATTENDANCE SYSTEM WITH FACE RECOGNITION**" is the record of the original workdone by us under the guidance of Ms.Sreelakshmi K, Assistant Professor, Centre for Computational Engineering and Networking, Amrita School of Artificial Intelligence, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree/diploma/ associate ship/fellowship/or a similar award to any candidate in any University.

Place:Coimbatore

Date:20-12-2023 **Signature of the Student**

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher Ms. SREELAKSHMI K ma'am, who gave us the golden opportunity to do this wonderfull project on the topic "ATTENDANCE SYSTEM WITH FACE RECOGNITION",which also helped usin doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.We would also like to thank our group members,as with out their cooperation,we would not have been able to complete the project within the prescribed time.

ABSTRACT

This project presents a complete Python-based solution that combines facial recognition technology with an intuitive graphical user interface (GUI) in response to the rising demand for effective attendance monitoring solutions. For simplicity and scalability, the system makes use of a folder database structure, which improves management ease.

The graphical user interface (GUI) makes it simple for users to enroll, and administrators may easily add, edit, or remove user profiles. The system can manage real-time scenarios in a variety of contexts, such as classrooms, corporate offices, and events, thanks to its folder-based database structure, which improves flexibility to dynamic environments.

Face detection is the first step in the procedure, and then features are extracted and compared. The GUI presents pertinent data upon successful identification, and attendance records are centrally maintained. The addition of a graphical user interface (GUI) improves the system's usability and accessibility for people with different technical backgrounds.

To sum up, the attendance system that uses Python is incorporated with state-of-the-art facial recognition technology and has an intuitive graphical user interface. These characteristics work together to provide a robust, flexible, and user-friendly system that can be used in enterprises, educational institutions, and event management.

1. ATTENDANCE WITH FACE RECOGNITION

INTRODUCTION

In the field of attendance systems, face recognition technology has become a game-changer by providing an alternative to manual techniques. This biometric technique provides accurate identification and verification of individuals by utilizing distinctive face traits. Unlike fingerprint or iris scans, face recognition is a convenient and user-friendly option since it is non-intrusive and relies only on facial photos taken by a camera. The solution solves long-standing issues with manual methods by processing face data in real-time, enabling quick and precise attendance tracking.

System administration is made easier and the user experience is improved when facial recognition technology is used with a graphical user interface (GUI). By using a simplified interface to capture facial traits, the GUI makes it easy for people to enroll. With this strategy, a wide range of users—including those with little technical experience—can utilize the technology. The GUI also makes maintenance easier, making it simple for administrators to add or delete users from the attendance system. Using a folder-based database structure improves scalability even further, allowing for dynamic organizational changes and maintaining system flexibility.

Face recognition systems provide data encryption and protection first priority in order to address security and privacy issues. Usually, distinctive face traits or templates are used instead of storing facial photographs, protecting people's biometric information. Access control systems and the technology can work together smoothly to improve security in regions that are off-limits. In addition, the system keeps track of attendance data centrally and generates detailed reports for administrators. Face recognition in attendance systems is a comprehensive answer for contemporary organizational demands as these reports are useful for payroll processing, performance analysis, and compliance monitoring. Face recognition in attendance systems is being widely used, and as technology develops, this will redefine security and efficiency in a variety of businesses.

OBJECTIVE

This project's main goal is to design, develop, and implement a sophisticated Attendance System that uses a folder-based database structure, a user-friendly Graphical User Interface (GUI), and Face Recognition technology. By utilizing face recognition's accuracy and efficiency, the initiative seeks to overcome the drawbacks of conventional attendance monitoring techniques.
The goal of the project is to provide an improved attendance system that meets the changing demands of contemporary businesses by emphasizing usability, security, and adaptability while simultaneously improving accuracy and efficiency.

METHODOLOGY

Using computer vision techniques and algorithms, faces are found and identified inside picture or video frames as part of the face detection process. Here is a step-by-step explanation of the typical methodology for face detection:

1. IMAGE ACQUISITION:

Obtain the face-containing input picture frame. This taken from a camera. In computer vision applications, picture capture is an essential first step. In this case, it's critical for tasks like face identification and attendance monitoring. OpenCV is a well-liked option for these kinds of applications as it makes camera access and picture processing simpler.

2. GRAYSCALE CONVERSION:

Image input should be converted to grayscale. Color information speeds up processing and simplifies computations, however it is not always required for face identification.

Figure 1: image to grayscale

3. FACE DETECTION ALGORITHM:

Utilize a face detection technique on the image in grayscale. The Haar Cascade Classifier is a widely used and effective algorithm. This technique locates areas of a picture that most likely contain faces using a collection of atrained classifiers.

4. DRAW RECTANGLES AROUND DETECTED FACES:

If faces are detected, draw rectangles around them to highlight their positions in the image.

5. DISPLAY OR SAVE RESULTS

Display the image with detected faces or save the image with marked faces, depending on the application.

The general approach may be modified to accommodate different application needs, face detection algorithm selections, and extra features (real-time processing, multi-facial detection, face tracking in video streams, etc.). Face detection is an essential component of many computer vision applications, ranging from systems for human-computer interaction to security and surveillance.

2. IMPLEMENTATION IN PYTHON

CODE

```
import  tkinter as tk
from tkinter import messagebox
import cv2
import os
import datetime
import face_recognition
from PIL import Image, ImageTk

blocked_images_db = "blocked_images.txt"
def register_student():
    username = reg_username_entry.get()
    password = reg_password_entry.get()

    if username == "" or password == "":
        messagebox.showerror("Error", "Please enter both username and
password!")
    else:
        with open("student_database.txt", "a") as file:
            file.write(f"{username},{password}\n")
        messagebox.showinfo("Success", "Student registered successfully!")
        reg_window.destroy()  # Close registration window after successful
registration

def open_register_window():
    global reg_window
    reg_window = tk.Toplevel(root)
    reg_window.title("Register Student")

    reg_frame = tk.LabelFrame(reg_window, text="Register Student", padx=20,
pady=20)
    reg_frame.pack(padx=20, pady=20)

    tk.Label(reg_frame, text="Username:").grid(row=0, column=0, pady=5,
sticky="w")
    global reg_username_entry
    reg_username_entry = tk.Entry(reg_frame)
    reg_username_entry.grid(row=1, column=0, pady=5)

    tk.Label(reg_frame, text="Password:").grid(row=2, column=0, pady=5,
sticky="w")
    global reg_password_entry
    reg_password_entry = tk.Entry(reg_frame, show="*")
    reg_password_entry.grid(row=3, column=0, pady=5)

    register_button = tk.Button(reg_frame, text="Register",
command=register_student)
    register_button.grid(row=4, column=0, pady=10)
def capture_image_student():
    username = login_username_entry.get()

    with open(blocked_images_db, "r") as file:
```

```
        blocked_users = file.readlines()
    if username + "\n" in blocked_users:
        messagebox.showerror("Error", "Camera access is blocked for this
user.")
        return

    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()

    image_folder = "student_images"
    if not os.path.exists(image_folder):
        os.makedirs(image_folder)

    image_path = os.path.join(image_folder, f"{username}.png")
    cv2.imwrite(image_path, frame)
    messagebox.showinfo("Image Captured", "Image captured successfully!")
def capture_attendance_dialog():
    capture_attendance_window = tk.Toplevel(root)
    capture_attendance_window.title("Capture Attendance")

    courses = ["Math", "Science", "History"]  # Example courses
    faculties = ["Dr. Smith", "Prof. Johnson", "Dr. Brown"]  # Example
faculties

    course_label = tk.Label(capture_attendance_window, text="Select Course:")
    course_label.pack()
    course_var = tk.StringVar()
    course_dropdown = tk.OptionMenu(capture_attendance_window, course_var,
*courses)
    course_dropdown.pack()

    faculty_label = tk.Label(capture_attendance_window, text="Select
Faculty:")
    faculty_label.pack()
    faculty_var = tk.StringVar()
    faculty_dropdown = tk.OptionMenu(capture_attendance_window, faculty_var,
*faculties)
    faculty_dropdown.pack()

    capture_button = tk.Button(capture_attendance_window, text="Capture
Image", command=lambda: capture_student_attendance(course_var.get(),
faculty_var.get()))
    capture_button.pack()
# ... (Previous code remains the same)
def capture_student_attendance(selected_course, selected_faculty):
    username = login_username_entry.get()
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()

    # Assuming you have a folder named 'attendance_images' to store
attendance images
    attendance_folder = "attendance_images"
    if not os.path.exists(attendance_folder):
        os.makedirs(attendance_folder)
```

```
    attendance_image_path = os.path.join(attendance_folder,
f"{selected_course}_{selected_faculty}_{username}.png")
    cv2.imwrite(attendance_image_path, frame)

    # Load the captured image for face recognition
    captured_image = face_recognition.load_image_file(attendance_image_path)
    captured_face_encodings = face_recognition.face_encodings(captured_image)

    # Load the registered student's image for comparison
    student_image_folder = "student_images"
    student_image_path = os.path.join(student_image_folder,
f"{username}.png")
    if not os.path.exists(student_image_path):
        messagebox.showerror("Error", "No registered image found for the
student.")
        return

    registered_image = face_recognition.load_image_file(student_image_path)
    registered_face_encodings =
face_recognition.face_encodings(registered_image)

    if not captured_face_encodings or not registered_face_encodings:
        messagebox.showerror("Error", "Face encodings not found. Attendance
not captured.")
        return

    # Compare the face encodings
    for captured_face_encoding in captured_face_encodings:
        for registered_face_encoding in registered_face_encodings:
            results =
face_recognition.compare_faces([captured_face_encoding],
registered_face_encoding)

            # If faces match, record attendance details
            if results[0]:  # Assuming results indicate successful
recognition
                # Get the current date
                current_date = datetime.datetime.now().strftime("%Y-%m-%d")

                # Append attendance record to the file
                with open("attendance_records.txt", "a") as file:

file.write(f"{current_date},{selected_course},{selected_faculty},{username},P
resent\n")

                # Show attendance captured message
                messagebox.showinfo("Attendance Captured", "Attendance
captured successfully!")
                return

    # Show error if face recognition fails
    messagebox.showerror("Error", "Attendance not captured. Face recognition
failed.")

def login():
    username = login_username_entry.get()
    password = login_password_entry.get()
```

```
    with open("student_database.txt", "r") as file:
        for line in file:
            stored_username, stored_password = line.strip().split(",")
            if username == stored_username and password == stored_password:
                messagebox.showinfo("Success", f"Welcome {username}
(Student)")
                open_student_panel(username)
                return
    if username == "teacher" and password == "teacher@123":
        messagebox.showinfo("Success", "Welcome Teacher")
        open_teacher_panel()
        return

    messagebox.showerror("Error", "Invalid credentials!")
def read_student_data():
    try:
        with open("student_database.txt", "r") as file:
            lines = file.readlines()

        return [line.strip().split(",") for line in lines]
    except FileNotFoundError:
        messagebox.showerror("Error", "Student database not found.")
        return []
def view_student_details():
    student_data = read_student_data()
    if student_data:
        details = "\n".join([f"Username: {line[0]}, Password: {line[1]}" for
line in student_data])
        messagebox.showinfo("Student Details", f"Student
Details:\n{details}")
    else:
        messagebox.showinfo("Student Details", "No students found.")

def view_student_images():
    image_folder = "student_images"
    if not os.path.exists(image_folder):
        messagebox.showinfo("Info", "No student images found.")
        return

    image_files = os.listdir(image_folder)
    if not image_files:
        messagebox.showinfo("Info", "No student images found.")
    else:
        # Create a new window to display the images and names
        image_display_window = tk.Toplevel(root)
        image_display_window.title("Student Images")

        # Fetch student data (assuming the images' names correspond to the
student usernames)
        student_data = read_student_data()

        # Loop through each image file and display it along with the name
        for image_file in image_files:
            username, _ = os.path.splitext(image_file)  # Get the username
from the image file name
```

```
            # Find the corresponding name from student data
            student_name = ""
            for student in student_data:
                if student[0] == username:
                    student_name = student[0]  # Assuming username is the
first element in student_data

            if student_name:
                image_path = os.path.join(image_folder, image_file)

                # Open and resize the image using PIL
                image = Image.open(image_path)
                image.thumbnail((200, 200))  # Adjust the size as needed

                # Convert the image to a format compatible with Tkinter
                tk_image = ImageTk.PhotoImage(image)

                # Display the name and image in a label
                name_label = tk.Label(image_display_window,
text=student_name)
                name_label.pack()

                image_label = tk.Label(image_display_window, image=tk_image)
                image_label.image = tk_image  # Keep a reference to avoid
garbage collection
                image_label.pack()

def block_camera_access():
    username_to_block = block_camera_entry.get()
    with open(blocked_images_db, "a") as file:
        file.write(username_to_block + "\n")
    messagebox.showinfo("Success", f"Camera access blocked for
{username_to_block}")
def unblock_camera_access():
    username_to_unblock = block_camera_entry.get()
    with open(blocked_images_db, "r") as file:
        lines = file.readlines()
    with open(blocked_images_db, "w") as file:
        for line in lines:
            if line.strip() != username_to_unblock:
                file.write(line)
    messagebox.showinfo("Success", f"Camera access unblocked for
{username_to_unblock}")

def delete_student():
    student_data = read_student_data()
    if student_data:
        username_to_delete = delete_username_entry.get()
        remaining_students = [student for student in student_data if
student[0] != username_to_delete]

        with open("student_database.txt", "w") as file:
            for student in remaining_students:
                file.write(",".join(student) + "\n")
        messagebox.showinfo("Success", f"Student {username_to_delete}
deleted.")
    else:
```

```
        messagebox.showinfo("Student Details", "No students found.")

# Function to open the student panel
def open_student_panel(username):
    student_panel = tk.Toplevel(root)
    student_panel.title(f"Welcome {username} (Student)")

    welcome_label = tk.Label(student_panel, text=f"Welcome {username}!",
font=("Arial", 20))
    welcome_label.pack(padx=20, pady=20)

    capture_attendance_button = tk.Button(student_panel, text="Capture
Attendance", command=capture_attendance_dialog)
    capture_attendance_button.pack(padx=0, pady=10)

    capture_image_button = tk.Button(student_panel, text="Capture Image",
command=capture_image_student)
    capture_image_button.pack(padx=0, pady=10)

    logout_button = tk.Button(student_panel, text="Logout",
command=student_panel.destroy)
    logout_button.pack(padx=0, pady=10)
def recognize_face(username):
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()

    # Assuming you have a folder named 'student_images' to store student
images
    student_image_folder = "student_images"
    if not os.path.exists(student_image_folder):
        messagebox.showerror("Error", "No student images found.")
        return

    # Load the student's image
    student_image_path = os.path.join(student_image_folder,
f"{username}.png")
    student_image = face_recognition.load_image_file(student_image_path)
    student_face_encodings = face_recognition.face_encodings(student_image)

    # Encode the captured face
    captured_face_encoding = face_recognition.face_encodings(frame)

    # Compare the face encodings
    if captured_face_encoding:
        for captured_encoding in captured_face_encoding:
            for student_encoding in student_face_encodings:
                results = face_recognition.compare_faces([captured_encoding],
student_encoding)

                # If faces match, show a success message
                if results[0]:
                    messagebox.showinfo("Face Recognition", "Face recognized
successfully!")
                    return

        # If faces do not match, show an error message
```

```
        messagebox.showerror("Face Recognition", "Face recognition failed.
Face does not match the student's image.")

def read_attendance_data():
    try:
        with open("attendance_records.txt", "r") as file:
            attendance_data = file.readlines()
        return attendance_data
    except FileNotFoundError:
        return []
def view_attendance_dialog():
    view_attendance_window = tk.Toplevel(root)
    view_attendance_window.title("View Attendance")

    attendance_data = read_attendance_data()  # Function to read attendance
data

    if attendance_data:
        for record in attendance_data:
            record_details = record.split(',')
            if len(record_details) == 5:
                current_date, selected_course, selected_faculty, username,
status = record_details

                # Assuming the images are stored in the 'attendance_images'
folder
                attendance_image_path =
f"attendance_images/{selected_course}_{selected_faculty}_{username}.png"

                attendance_frame = tk.Frame(view_attendance_window, padx=10,
pady=10)
                attendance_frame.pack(padx=20, pady=10, fill=tk.BOTH,
expand=True)

                # Display the attendance details
                details_label = tk.Label(attendance_frame, text=f"Date:
{current_date}, Course: {selected_course}, Faculty: {selected_faculty},
Username: {username}, Status: {status}")
                details_label.pack()

                # Open the captured image using PIL
                try:
                    image = Image.open(attendance_image_path)
                    image.thumbnail((200, 200))  # Adjust the size as needed

                    # Convert the image to a format compatible with Tkinter
                    tk_image = ImageTk.PhotoImage(image)

                    # Display the image in a label
                    image_label = tk.Label(attendance_frame, image=tk_image)
                    image_label.image = tk_image  # Keep a reference to avoid
garbage collection
                    image_label.pack()
                except FileNotFoundError:
                    error_label = tk.Label(attendance_frame, text="Image not
found.")
                    error_label.pack()
```

```
            else:
                error_label = tk.Label(view_attendance_window, text=f"Invalid
attendance record: {record}")
                error_label.pack(padx=20, pady=20)
    else:
        no_data_label = tk.Label(view_attendance_window, text="No attendance
records found.")
        no_data_label.pack(padx=20, pady=20)

def open_teacher_panel():
    teacher_panel = tk.Toplevel(root)
    teacher_panel.title("Welcome Teacher")

    welcome_label = tk.Label(teacher_panel, text="Welcome Teacher!",
font=("Arial", 20))
    welcome_label.pack(padx=20, pady=20)

    view_attendance_button = tk.Button(teacher_panel, text="View Attendance",
command=view_attendance_dialog)
    view_attendance_button.pack(pady=(0, 10))  # Add vertical space between
this button and the next one

    view_details_button = tk.Button(teacher_panel, text="View Student
Details", command=view_student_details)
    view_details_button.pack(pady=(0, 10))  # Add vertical space between this
button and the next one

    delete_username_label = tk.Label(teacher_panel, text="Enter Student
Username to Delete:")
    delete_username_label.pack()

    global delete_username_entry
    delete_username_entry = tk.Entry(teacher_panel)
    delete_username_entry.pack()

    delete_student_button = tk.Button(teacher_panel, text="Delete Student",
command=delete_student)
    delete_student_button.pack(pady=(10, 0))  # Add vertical space between
this button and the previous one

    view_images_button = tk.Button(teacher_panel, text="View Student Images",
command=view_student_images)
    view_images_button.pack(pady=(10, 0))  # Add vertical space between this
button and the previous one

    block_camera_label = tk.Label(teacher_panel, text="Enter Username to
Block Camera Access:")
    block_camera_label.pack()

    global block_camera_entry
    block_camera_entry = tk.Entry(teacher_panel)
    block_camera_entry.pack()

    block_camera_button = tk.Button(teacher_panel, text="Block Camera",
command=block_camera_access)
    block_camera_button.pack(pady=(10, 0))  # Add vertical space between this
button and the previous one
```

```
    unblock_camera_button = tk.Button(teacher_panel, text="Unblock Camera",
command=unblock_camera_access)
    unblock_camera_button.pack(pady=(10, 0))  # Add vertical space between
this button and the previous one

    logout_button = tk.Button(teacher_panel, text="Logout",
command=teacher_panel.destroy)
    logout_button.pack(padx=0, pady=10)

root = tk.Tk()
root.title("Student Attendance System")
root.configure(bg="black")
def set_styles():
    root.tk_setPalette(background="#000000", foreground="#ffffff")
    root.option_add("*Font", "Arial 10")
    root.option_add("*Label.Font", "Arial 12 bold")
    root.option_add("*Button.Background", "#4caf50")
    root.option_add("*Button.Foreground", "white")
    root.option_add("*Button.Relief", "flat")
    root.option_add("*Button.Width", 15)
    root.option_add("*Entry.Width", 20)

set_styles()

title_label = tk.Label(root, text="Attendance Recognition System",
bg="black", fg="white")
title_label.pack(padx=20, pady=20)

login_frame = tk.LabelFrame(root, text="Login", padx=20, pady=20)
login_frame.pack(padx=20, pady=20)

tk.Label(login_frame, text="Username:").grid(row=0, column=0, pady=5,
sticky="w")
login_username_entry = tk.Entry(login_frame)
login_username_entry.grid(row=1, column=0, pady=5)

tk.Label(login_frame, text="Password:").grid(row=2, column=0, pady=5,
sticky="w")
login_password_entry = tk.Entry(login_frame, show="*")
login_password_entry.grid(row=3, column=0, pady=5)

login_button = tk.Button(login_frame, text="Login", command=login)
login_button.grid(row=4, column=0, pady=10)

register_window_button = tk.Button(login_frame, text="Register",
command=open_register_window)
register_window_button.grid(row=5, column=0, pady=10)

root.mainloop()
```

CODE OVERVIEW

A python application that uses facial recognition to create a basic student attendance system. The Tkinter library was used to create the system's graphical user interface (GUI). Students may use the software to see their attendance records, log in, and take pictures with their cameras to record their attendance. Instructors may check attendance reports, update student information, and restrict which students can use the camera.

1. GUI INITIALIZATION AND STYLING:

The application establishes styles for the GUI elements, including the background color, font, button look, and width of the entry field, as well as initializes the main Tkinter window . The set_styles() function defines the styling preferences using the option_add method.

2. LOGIN INTERFACE:

Users (teachers and students) can input their passwords and usernames on the login window. The entered credentials are compared to the student data saved in the student_database.txt file and a teacher password that is hardcoded when the "Login" button is clicked.

3. STUDENT REGISTRATION:

There is a separate registration window (reg_window) where students can register. Students can set a password and username in the registration box, and those details are saved in the student_database.txt file.

4. ATTENDANCE CAPTURE:

By selecting the "Capture Attendance" option, students may record their attendance. Face recognition software uses the taken image, which is stored in the attendance_images folder.The functions for taking attendance, storing photos, and doing facial recognition are all handled by the capture_student_attendance function.

5. FACE RECOGNITION:

The face_recognition library is used to implement face recognition. The software makes a comparison between the registered student's face encoding and the captured face encoding.The student's attendance is noted in the attendance_records.txt file if a match is discovered.

6. TEACHER PANEL:

In addition, teachers may examine attendance records, manage student information, remove students, view student photos, and manage camera access, among other features.The teacher can use comparable functions to implement actions like eliminating pupils or preventing access to the webcam.

7. VIEWING STUDENT DETAILS AND IMAGES:

Teachers can view student details and images through dedicated buttons in the teacher panel. The view_student_details and view_student_images functions handle these functionalities.

8. BLOCKING AND UNBLOCKING IMAGES:

By inputting the usernames of those pupils, teachers can restrict or allow access to the camera for certain kids. The blocked_images.txt file contains the data.

9. LOGGING OUT:

The "Logout" option allows both teachers and students to log out of the system. The corresponding windows are closed using the destroy() function.

The code creates a straightforward interface that allows teachers and students to interact with the system, as well as implementing a rudimentary attendance system with facial recognition features. Keep in mind that the particular needs and use cases should determine how the security and privacy components of the system are improved.

OUTPUT

After running the code in command prompt we will get the login portal where students and teacher can login with their credentials.

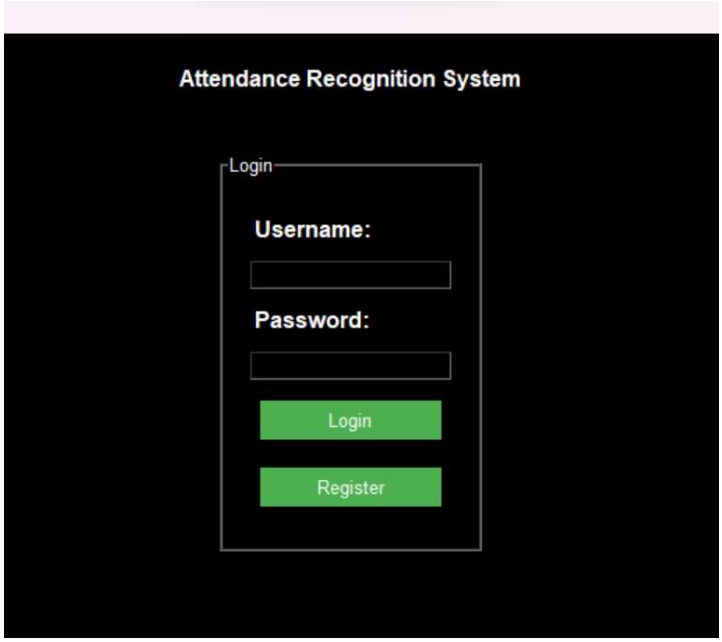


Figure 2:login portal

If student not register he can get registered in register portal.

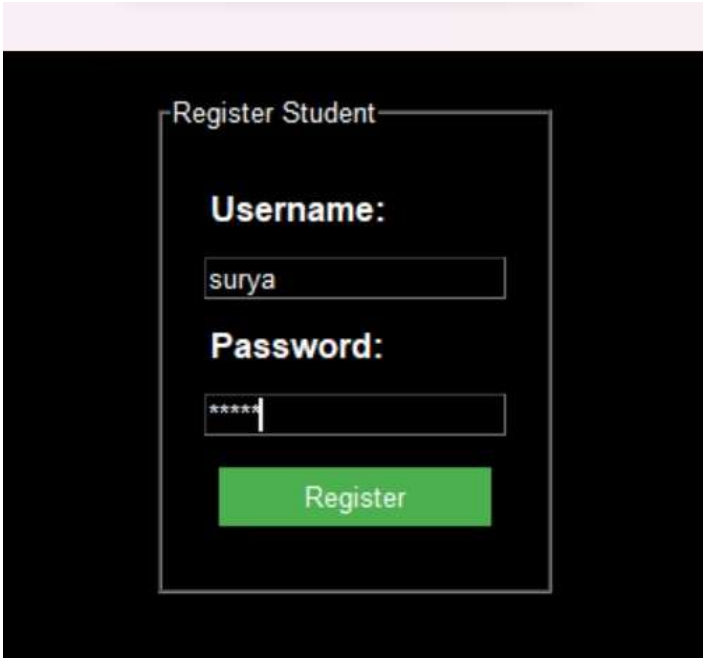


Figure 3:register portal

After get registered student can login with their respective credentials.

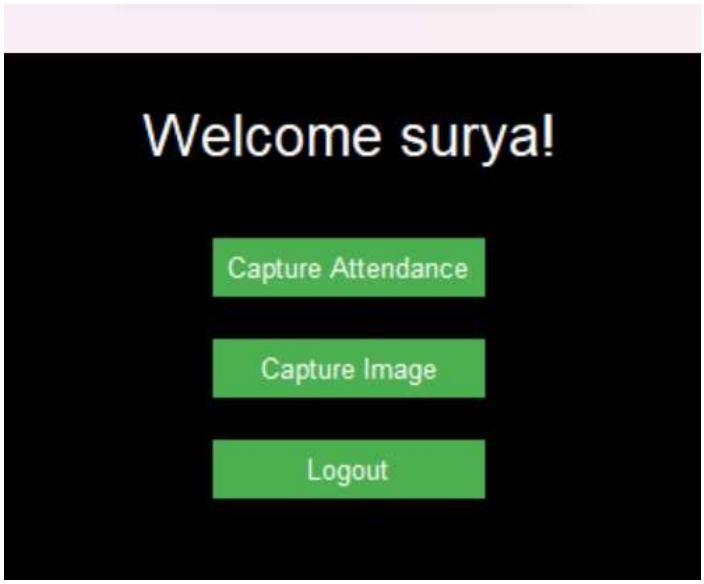


Figure 4:student desk

Teacher has default login credential i.e user name:teacher and password:teacher@123 with these teacher can login.

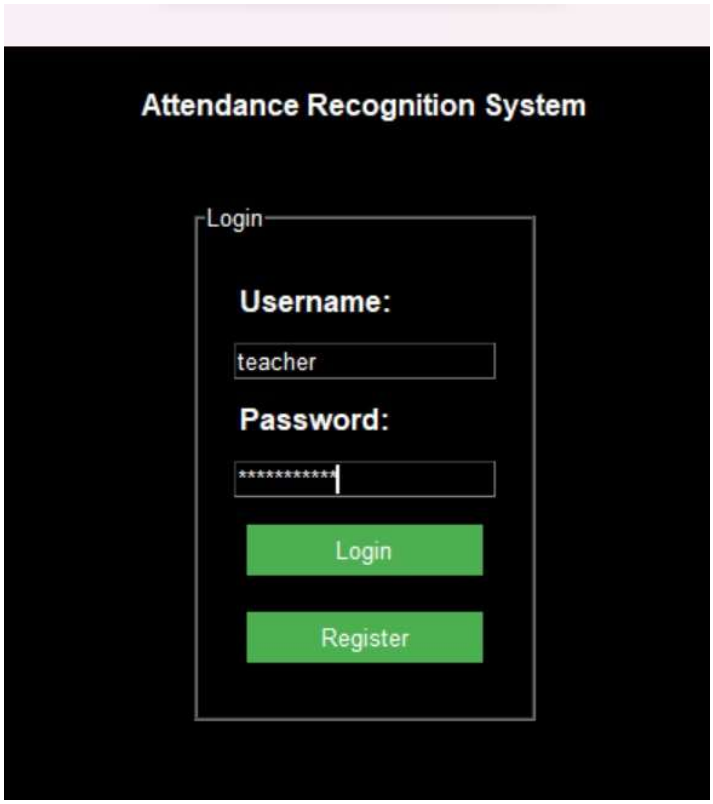


Figure 5:teacher login

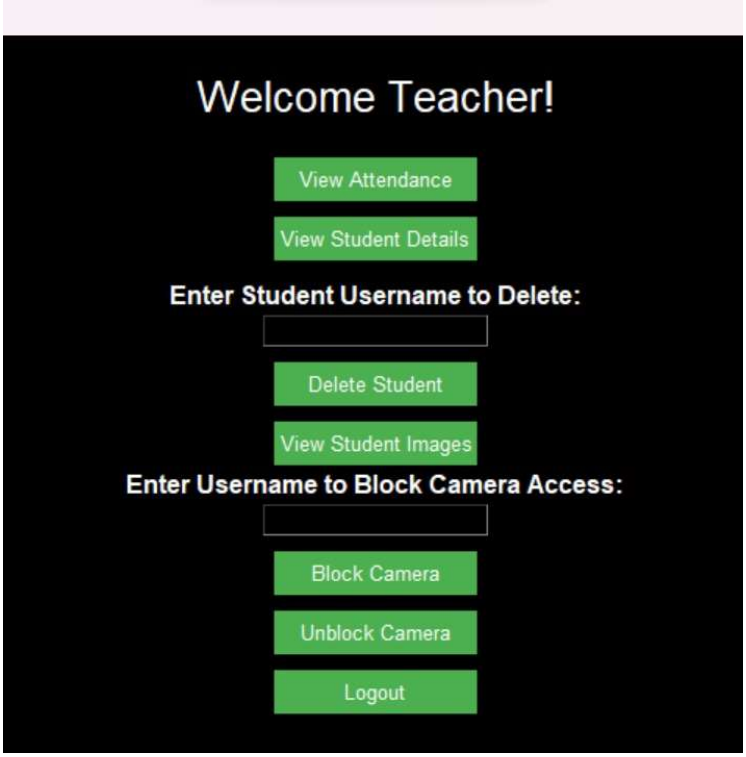


Figure 6:teacher desk

Once student login to his/her desk they have to capture their image first so that image can used while giving attendance with face detection.

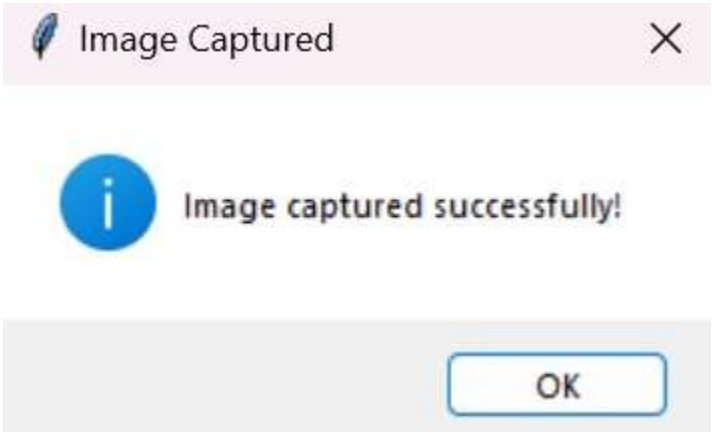


Figure 7:capturing of student image

After capturing student image teacher has to block that image so that further student can not capture a new face until teacher unblocks them by this student can not perform any malpractice in attendance in future.

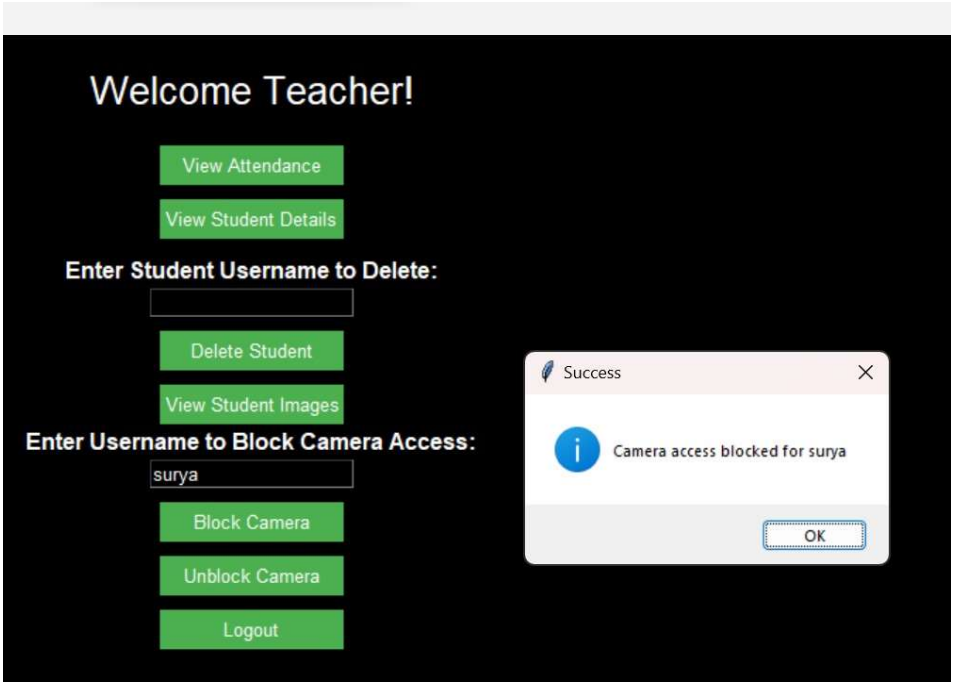


Figure 8:blocking student image

After this student can give attendance by selecting the particular subject and faculty if the image recognition is successful with the image with one which student captured then they will het attendance if not it denies.

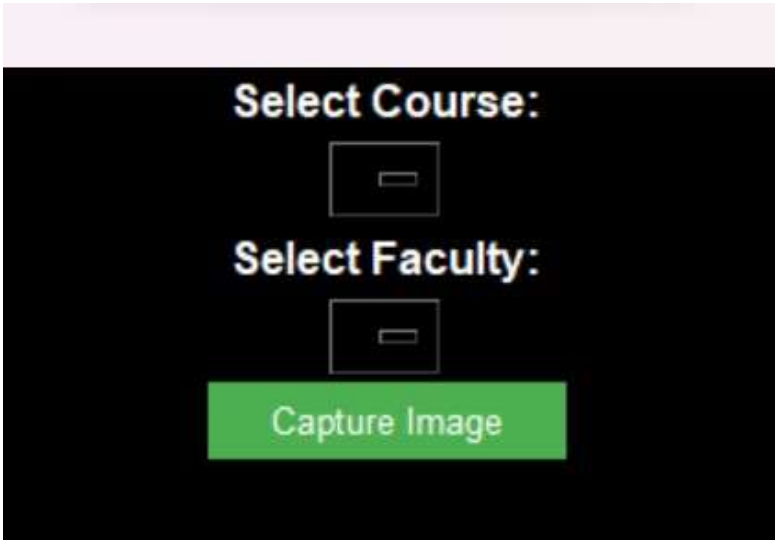


Figure 9:selection of subject and faculty

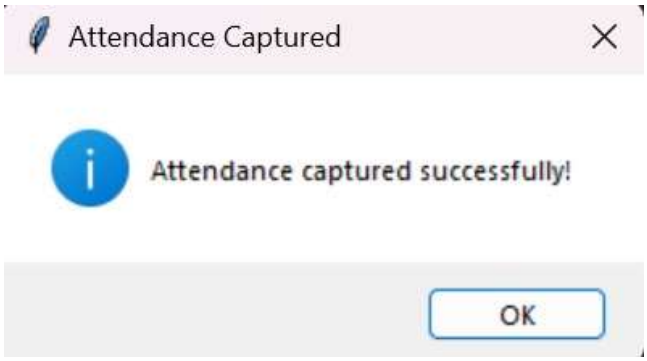


Figure 10:successful attendance

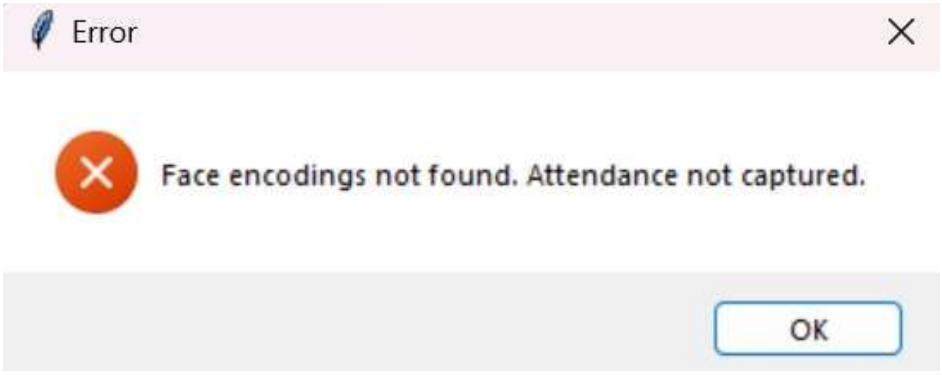


Figure 11:denied attendance

Once students give their attendance teacher can view students attendance in their desk.

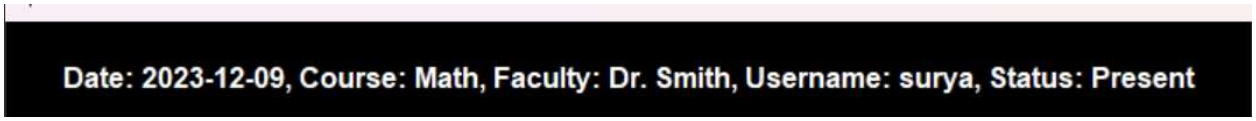


Figure 12:teacher desk attendance details

3. CONCLUSION

To summarize, the introduction of the Student Attendance System with Face Recognition represents a major advancement in the field of attendance management. The project uses Python, Tkinter, OpenCV, and face_recognition to give instructors a powerful toolset for record-keeping while streamlining student attendance monitoring.

The use of facial recognition technology enhances the system's precision and safety, guaranteeing dependable attendance recording. Despite its success, real-time notifications and improved facial recognition algorithms should be explored for future advancements.

This study demonstrates how computer vision and user interface design work well together to solve common problems in educational systems.

4. REFERENCES

1. https://www.esslsecurity.com/face
2. https://buddypunch.com/blog/attendance-system-face-recognition/
3. https://buddypunch.com/blog/attendance-system-face-recognition/