***Submitted in partial fulfillment of the***

***requirement for the award of the degree of***

**MASTER OF COMPUTER APPLICATION**

****

DEGREE

**Session 2024 – 25**

**By**

***Prince Kumar***

***(24SCSE2030311)***

***Amit Ranjan***

***(24SCSE2030357)***

***Aman kumar***

***(24SCSE2030323)***

***Ashwini Raj***

***(24SCSE2030075)***

**Under the Guidance of**

**Dr. Prashant Johri**

**SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY**

**GALGOTIAS UNIVERSITY, GREATER NOIDA**

**INDIA**

**June , 2025**

 **SCHOOL OF COMPUTER APPLICATIONS AND**

**TECHNOLOGY**

**GALGOTIAS UNIVERSITY, GREATER NOIDA**

**CANDIDATE’S DECLARATION**

I/We hereby certify that the work which is being presented in the project, entitled **“Face Recognition-Based Automated Attendance System”** in partial fulfillment of the requirements for the award of the MCA (Master of Computer Application) submitted in the School of Computer Applications and Technology of Galgotias University, Greater Noida, is an original work carried out during the period of May, 2025 to June 2025, under the supervision of **Dr. Prashant Johri**, Department of Computer Science and Engineering/School of Computer Applications and Technology , Galgotias University, Greater Noida. The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

***Prince Kumar***

***(24SCSE2030311)***

***Amit Ranjan***

***(24SCSE2030357)***

***Aman kumar***

***(24SCSE2030323)***

***Ashwini Raj***

***(24SCSE2030075)***

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

**Dr. Prashant Johri**

.

# CERTIFICATE

This is to certify that Project Report entitled “**Face Recognition-Based Automated Attendance System** ” which is submitted by Prince kumar, Amit Ranjan, Aman kumar, Ashwini Raj in partial fulfillment of the requirement for the award of degree MCA. in Department of School of Computer Applications and Technology, Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree

**Signature of Examiner(s) Signature of Supervisor(s)**

Date: June, 2025

Place: Greater Noida

## ACKNOWLEDGEMENT

**Roll No. Student Name Student Signature**

24SCSE2030311 Prince Kumar

24SCSE2030357 Amit Ranjan

24SCSE2030323 Aman kumar

24SCSE2030075 Ashwini Raj

## Abstract

This face recognition attendance system employs supervised machine learning, utilizing a pre-trained deep convolutional neural network (CNN) from the face\_recognition library to identify individuals in real-time webcam feeds. It detects faces, generates 128-dimensional feature embeddings, and matches them against a pre-loaded database of known faces stored as .jpg or .png images in the known\_faces directory. Recognized names are logged in an Excel file (attendance.xlsx) with date and time, using pandas for data management and openpyxl for file operations, with a 2-second delay to prevent duplicate entries.

A tkinter-based GUI displays the webcam feed, recognized names, and a button to view attendance records, while pyttsx3 provides text-to-speech feedback for attendance events. OpenCV (cv2) handles video capture and frame processing, and PIL converts frames for GUI display. The system integrates these components to deliver an automated, user-friendly attendance tracking solution with robust error handling and real-time feedback.

**TABLE OF CONTENTS**

*DECLARATION ................................................................................................... 2*

*CERTIFICATE……............................................................................................... 3*

*ACKNOWLEDGEMENTS ................................................................................. 4*

*ABSTRACT ........................................................................................................... 5*

**1. Type of Machine Learning Used**

**2. Libraries Used**

**3. Face Recognition-Based Attendance System Architecture**

**4. Code**

**5. Dependencies and Setup Requirements**

**6. How the System Works**

**7. Samples**

**8. Summary**

**1. Type of Machine Learning Used**

The code uses **supervised machine learning** with a **deep learning-based face recognition model**. Here’s a detailed explanation:

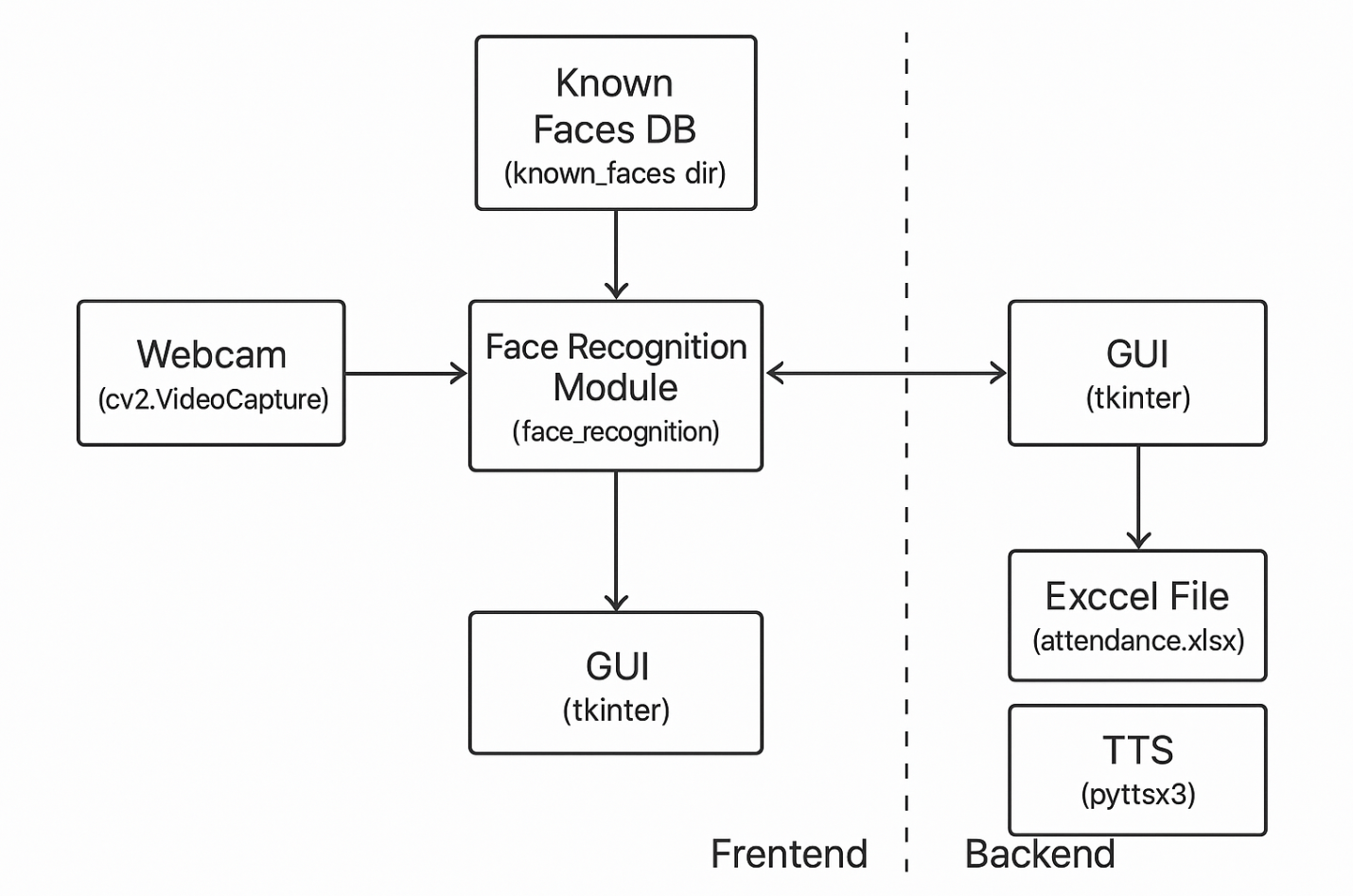
* **Supervised Learning**:
  + The face recognition system relies on a pre-labeled dataset of known faces (images in the known\_faces directory). Each image is associated with a name, serving as a label.
  + The system compares real-time face encodings (numerical feature representations) from the webcam to the precomputed encodings of known faces to classify/identify individuals.
  + This is a **classification task** where the model predicts the identity of a detected face (e.g., "John" or "Unknown") based on trained data.
* **Deep Learning Component**:
  + The face\_recognition library, used in the code, is built on **dlib’s face recognition model**, which employs a **deep convolutional neural network (CNN)**.
  + The CNN extracts facial features (128-dimensional embeddings) from images. These embeddings are compared using a distance metric (e.g., Euclidean distance) to determine matches.
  + The model is pre-trained, meaning the code does not train the CNN but uses its pre-trained weights to generate encodings and perform comparisons.
* **Key ML Operations**:
  + **Face Detection**: Uses a CNN-based model (or HOG + SVM in some configurations of dlib) to locate faces in images or video frames.
  + **Face Encoding**: Generates a 128-dimensional vector for each detected face, capturing unique facial features.
  + **Face Matching**: Compares encodings of detected faces with known encodings using a nearest-neighbor approach to classify the face’s identity.
* **No Explicit Training**:
  + The code does not train a model from scratch. Instead, it uses pre-trained models from the face\_recognition library. The "training" occurs implicitly when known face images are loaded and encoded.

**2. Libraries Used**

The code relies on several Python libraries to handle face recognition, GUI, webcam input, Excel operations, and text-to-speech. Here’s a detailed list of the libraries and their purposes:

* **cv2 (OpenCV)**:
  + Purpose: Handles webcam video capture and image processing.
  + Usage: Captures frames from the webcam (cv2.VideoCapture), processes them (e.g., converts BGR to RGB), and draws rectangles/text on frames for face visualization.
  + Installation: pip install opencv-python
* **face\_recognition**:
  + Purpose: Provides face detection, encoding, and matching functionalities using a pre-trained deep learning model.
  + Usage: Detects faces (face\_locations), generates face encodings (face\_encodings), and compares them (compare\_faces) for identification.
  + Installation: pip install face\_recognition
  + Dependencies: Requires dlib, which may need additional setup (e.g., pip install dlib with a compatible compiler or pre-built wheels).
* **os**:
  + Purpose: Handles file system operations.
  + Usage: Lists image files in the known\_faces directory and constructs file paths.
* **pandas**:
  + Purpose: Manages data storage in Excel.
  + Usage: Reads and writes attendance records to attendance.xlsx using DataFrames.
  + Installation: pip install pandas
* **datetime**:
  + Purpose: Provides current date and time for attendance records.
  + Usage: Records the date (%Y-%m-%d) and time (%H:%M:%S) when marking attendance.
* **pyttsx3**:
  + Purpose: Provides text-to-speech functionality.
  + Usage: Converts attendance messages to speech (e.g., “Attendance marked for John”).
  + Installation: pip install pyttsx3
* **tkinter**:
  + Purpose: Creates the graphical user interface (GUI).
  + Usage: Displays the webcam feed, shows recognized names, and provides a button to view attendance records.
  + Installation: Built-in with Python (no additional installation needed).
* **PIL (Pillow)**:
  + Purpose: Handles image conversion for GUI display.
  + Usage: Converts OpenCV frames to a format compatible with tkinter (ImageTk.PhotoImage).
  + Installation: pip install Pillow
* **zipfile**:
  + Purpose: Handles potential errors with Excel file corruption.
  + Usage: Checks for BadZipFile errors when reading the Excel file.
  + Installation: Built-in with Python.
* **time**:
  + Purpose: Manages timing for detection delays.
  + Usage: Implements a delay (DETECTION\_DELAY) to prevent multiple attendance markings for the same person in a short time.
* **openpyxl** (implicit dependency):
  + Purpose: Backend engine for reading/writing Excel files with pandas.
  + Usage: Required for pandas to interact with .xlsx files.
  + Installation: pip install openpyxl

**3. Face Recognition-Based Attendance System Architecture**

****

**4. Code**

The code implements a face recognition-based attendance system with a GUI, webcam feed, Excel storage, and TTS feedback. Below is a detailed explanation of each section:

**TTS Setup**

python

CollapseWrapRun

Copy

engine = pyttsx3.init()

engine.setProperty('rate', 150)

def speak(text):

print(f"[TTS] {text}")

engine.say(text)

engine.runAndWait()

* Initializes the pyttsx3 engine for text-to-speech.
* Sets the speech rate to 150 (normal speed).
* Defines a speak function to output text as speech and print it to the console.

**Excel Setup**

python

CollapseWrapRun

Copy

EXCEL\_FILE = "attendance.xlsx"

def init\_excel():

if not os.path.exists(EXCEL\_FILE):

df = pd.DataFrame(columns=["Name", "Date", "Time"])

df.to\_excel(EXCEL\_FILE, index=False, engine='openpyxl')

print(f"[INFO] Created new Excel file: {EXCEL\_FILE}")

init\_excel()

* Defines the Excel file name (attendance.xlsx).
* Creates an empty Excel file with columns Name, Date, and Time if it doesn’t exist.
* Uses pandas with the openpyxl engine to manage Excel operations.

**Mark Attendance in Excel**

python

CollapseWrapRun

Copy

def mark\_attendance\_excel(name):

now = datetime.now()

date = now.strftime("%Y-%m-%d")

time\_str = now.strftime("%H:%M:%S")

try:

df = pd.read\_excel(EXCEL\_FILE, engine='openpyxl')

except zipfile.BadZipFile:

print("[ERROR] attendance.xlsx is not a valid Excel file.")

messagebox.showerror("File Error", "attendance.xlsx is corrupted or not a valid Excel file.")

return

except Exception as e:

print(f"[ERROR] Could not read Excel file: {e}")

return

if not ((df["Name"] == name) & (df["Date"] == date)).any():

new\_entry = pd.DataFrame([[name, date, time\_str]], columns=["Name", "Date", "Time"])

df = pd.concat([df, new\_entry], ignore\_index=True)

df.to\_excel(EXCEL\_FILE, index=False, engine='openpyxl')

print(f"[INFO] Marked attendance for {name} at {time\_str}")

speak(f"Attendance marked for {name}")

else:

print(f"[INFO] {name} already marked today.")

* Records attendance in the Excel file with the person’s name, date, and time.
* Checks if the person has already been marked for the current date to avoid duplicates.
* Handles errors (e.g., corrupted Excel file) and provides TTS feedback when attendance is marked.

**Load Known Faces**

python

CollapseWrapRun

Copy

known\_faces\_dir = 'known\_faces'

known\_encodings = []

known\_names = []

for file in os.listdir(known\_faces\_dir):

if file.lower().endswith(('.jpg', '.png')):

path = os.path.join(known\_faces\_dir, file)

image = face\_recognition.load\_image\_file(path)

encodings = face\_recognition.face\_encodings(image)

if encodings:

known\_encodings.append(encodings[0])

known\_names.append(os.path.splitext(file)[0])

else:

print(f"[WARNING] No face found in {file}")

* Loads images from the known\_faces directory (must contain .jpg or .png files).
* Uses face\_recognition.load\_image\_file to read images.
* Generates face encodings using face\_recognition.face\_encodings.
* Stores encodings and corresponding names (derived from filenames) in lists.
* Warns if no face is detected in an image.

**GUI Setup**

python

CollapseWrapRun

Copy

root = tk.Tk()

root.title("Face Attendance System")

root.geometry("600x500")

canvas = tk.Canvas(root, width=600, height=400)

canvas.pack()

name\_label = tk.Label(root, text="No student recognized", font=('Arial', 14))

name\_label.pack(pady=10)

def open\_attendance():

try:

df = pd.read\_excel(EXCEL\_FILE, engine='openpyxl')

messagebox.showinfo("Attendance", df.to\_string(index=False))

except Exception as e:

messagebox.showerror("Error", f"Could not open the attendance file:\n{str(e)}")

attendance\_button = tk.Button(root, text="View Attendance", command=open\_attendance, font=('Arial', 12))

attendance\_button.pack(pady=20)

* Creates a tkinter GUI window (600x500 pixels).
* Adds a canvas to display the webcam feed (600x400 pixels).
* Adds a label to show the recognized name (initially “No student recognized”).
* Adds a button to view the attendance Excel file in a message box.

**Webcam Feed and Face Recognition**

python

CollapseWrapRun

Copy

cap = cv2.VideoCapture(0)

detection\_memory = {} *# {name: last\_detected\_timestamp}*

DETECTION\_DELAY = 2 *# seconds*

def update\_frame():

ret, frame = cap.read()

recognized\_name = "No face recognized"

if ret:

rgb\_frame = frame[:, :, ::-1]

face\_locations = face\_recognition.face\_locations(rgb\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)

now = time.time()

for face\_encoding, face\_location in zip(face\_encodings, face\_locations):

matches = face\_recognition.compare\_faces(known\_encodings, face\_encoding)

face\_distances = face\_recognition.face\_distance(known\_encodings, face\_encoding)

if matches and any(matches):

best\_match\_index = face\_distances.argmin()

if matches[best\_match\_index]:

name = known\_names[best\_match\_index]

top, right, bottom, left = [v \* 4 for v in face\_location]

cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

cv2.putText(frame, name, (left + 6, bottom - 6),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2)

last\_detected = detection\_memory.get(name, 0)

if now - last\_detected > DETECTION\_DELAY:

mark\_attendance\_excel(name)

detection\_memory[name] = now

recognized\_name = name

else:

recognized\_name = f"{name} (waiting...)"

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame\_image = Image.fromarray(frame\_rgb)

frame\_photo = ImageTk.PhotoImage(frame\_image)

canvas.create\_image(0, 0, anchor=tk.NW, image=frame\_photo)

canvas.image = frame\_photo

name\_label.config(text=recognized\_name)

root.after(10, update\_frame)

* Initializes the webcam using cv2.VideoCapture(0).
* Implements a detection delay (DETECTION\_DELAY = 2 seconds) to prevent multiple attendance markings for the same person.
* In the update\_frame function:
  + Captures a frame from the webcam.
  + Converts the frame to RGB (required by face\_recognition).
  + Detects faces (face\_locations) and generates encodings (face\_encodings).
  + Compares detected face encodings with known encodings to identify matches.
  + Draws a green rectangle and name on the frame for recognized faces.
  + Marks attendance if the detection delay has elapsed.
  + Converts the frame to a tkinter-compatible image and updates the GUI canvas.
  + Updates the name label with the recognized name or status.
  + Schedules the next frame update every 10ms using root.after.

**Main Loop and Cleanup**

python

CollapseWrapRun

Copy

update\_frame()

root.mainloop()

cap.release()

cv2.destroyAllWindows()

* Starts the webcam feed loop (update\_frame).
* Runs the tkinter event loop (mainloop).
* Releases the webcam and closes OpenCV windows when the GUI is closed.

**5. Dependencies and Setup Requirements**

To run this code, you need:

* **Python**: Version 3.6–3.9 (some libraries like dlib may have compatibility issues with newer versions).
* **Libraries**: Install via pip:

bash

CollapseWrapRun

Copy

pip install opencv-python face\_recognition pandas pyttsx3 Pillow openpyxl

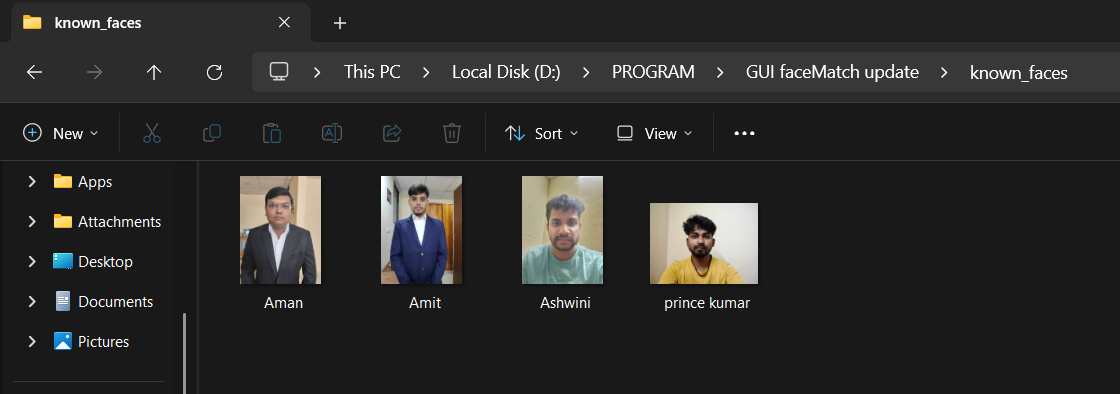
* **dlib Setup**: face\_recognition depends on dlib, which may require:
  + A C++ compiler (e.g., Visual Studio on Windows, g++ on Linux).
  + Pre-built wheels: pip install dlib --verbose (or use cmake for custom builds).
* **Directory Structure**:
  + A known\_faces directory with .jpg or .png images named after individuals (e.g., John.jpg).
* **Hardware**:
  + A webcam (accessed via cv2.VideoCapture(0)).
  + Sufficient CPU/GPU for face recognition (dlib’s CNN is computationally intensive; a GPU accelerates it if available).
* **OS Compatibility**:
  + pyttsx3 works on Windows (via SAPI), Linux (via eSpeak), and macOS (via NSSpeechSynthesizer).
  + Ensure openpyxl is installed for Excel operations.

**6. How the System Works**

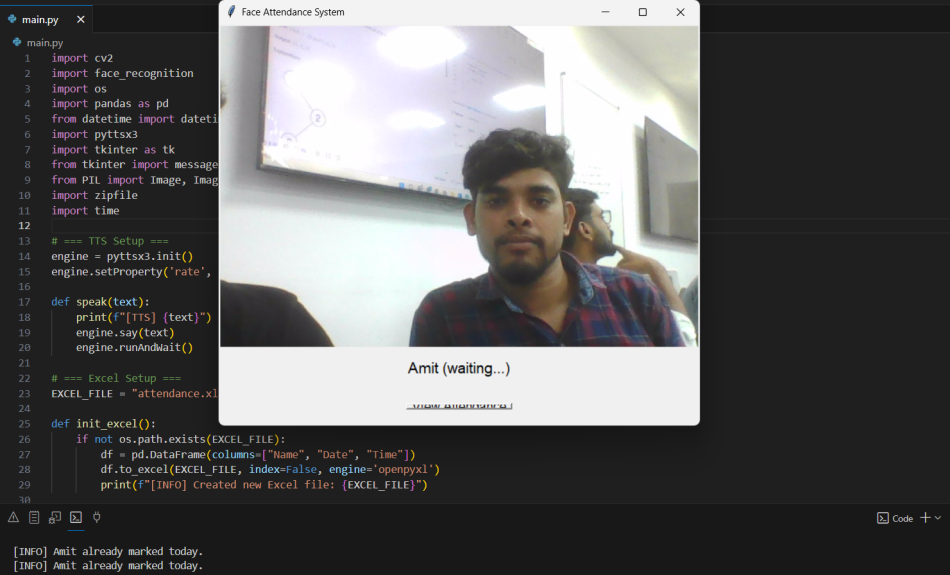
1. **Initialization**:
   * Loads known face images and generates their encodings.
   * Creates an Excel file (attendance.xlsx) if it doesn’t exist.
   * Sets up the GUI and webcam.
2. **Real-Time Processing**:
   * Captures webcam frames continuously.
   * Detects faces, generates encodings, and matches them against known encodings.
   * Displays recognized names and draws face boxes in the GUI.
   * Marks attendance in Excel and speaks the result if a face is recognized (with a 2-second delay to avoid duplicates).
3. **User Interaction**:
   * The GUI shows the webcam feed and recognized names.
   * Users can click “View Attendance” to see the Excel data in a message box.

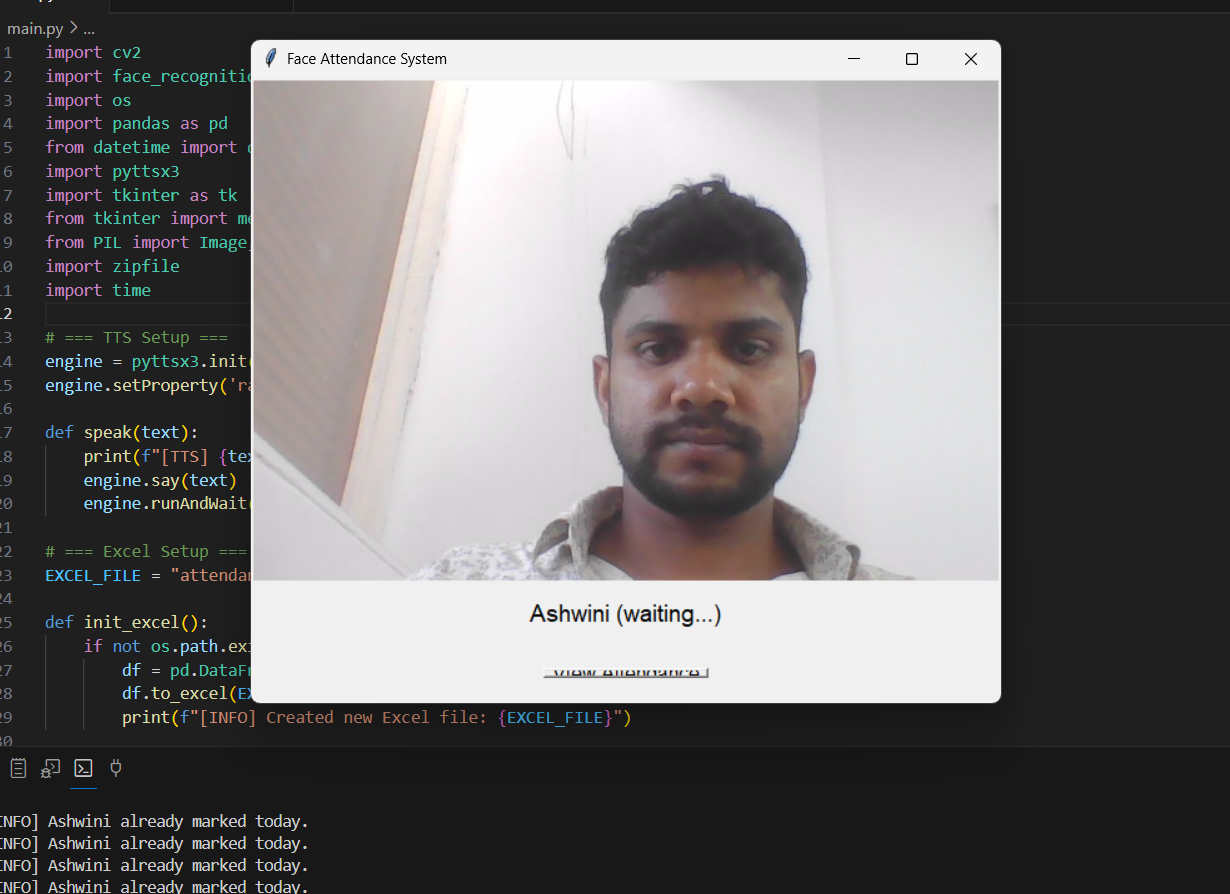
**7. Samples**

**7.1 Building and Managing a Face Recognition Database for Attendance Systems**

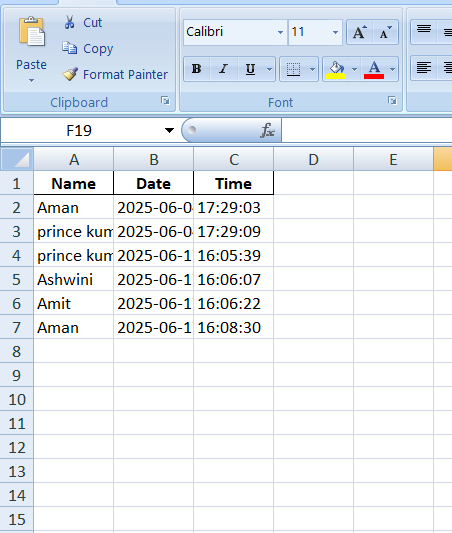
****

**7.2 Optimizing Face Recognition with Detection Delays for Accurate Attendance**

****

****

**7.3Data Management and Storage in Automated Attendance Systems**

****

**8. Summary**

* **Machine Learning**: Supervised learning with a pre-trained deep learning CNN (via face\_recognition) for face detection and recognition.
* **Libraries**: cv2, face\_recognition, os, pandas, datetime, pyttsx3, tkinter, PIL, zipfile, time, openpyxl.
* **Code Functionality**: A real-time face recognition attendance system with GUI, Excel storage, and TTS feedback.
* **Requirements**: Python, specific libraries, a known\_faces directory with labeled images, and a webcam.

If you have specific questions about any part of the code or need help with setup, let me know!