Automatic Attendance System using Face Recognition

Abstract

The traditional attendance system, based on manual entry or physical signatures, is time-consuming and prone to errors such as proxy attendance. This project introduces an **Automatic Attendance System** using **Face Recognition** technology. The system detects and recognizes faces from classroom images, records attendance in a structured CSV file, and generates annotated output images.

Two separate implementations are developed:

- A **local execution model** using **VS Code**.
- A **cloud execution model** using **Google Colab**.

Both systems use the **DeepFace library** with **VGG-Face embeddings** and cosine similarity for face recognition. The project demonstrates the accuracy, efficiency, and potential of AI-driven attendance automation.

Introduction

Attendance tracking is an essential part of classroom and organizational management. However, traditional methods such as roll-call or paper-based signatures are inefficient. With the advancements in **AI and computer vision**, it is now possible to automate attendance by recognizing student faces directly from classroom images.

This project leverages the **DeepFace library** in Python to implement a **face-recognition-based attendance system**. The recognized faces are logged into an attendance CSV file, with the date, time, and recognition confidence recorded.

Problem Statement

- Manual attendance systems are **time-consuming and error-prone**.
- **Proxy attendance** undermines the credibility of manual systems.
- Need for a **fast, reliable, and automated system** to record attendance.

Objectives

- To develop a **face recognition-based automatic attendance system**.
- To create a solution that works on both **local (VS Code)** and **cloud (Google Colab)** environments.
- To store attendance in a **structured CSV file** for easy management.
- To ensure **high accuracy** using VGG-Face embeddings and cosine distance.

System Requirements

Hardware Requirements

- Computer/Laptop with Webcam or Classroom Camera Images
- Minimum 4GB RAM (8GB recommended)
- Storage: At least 2GB free

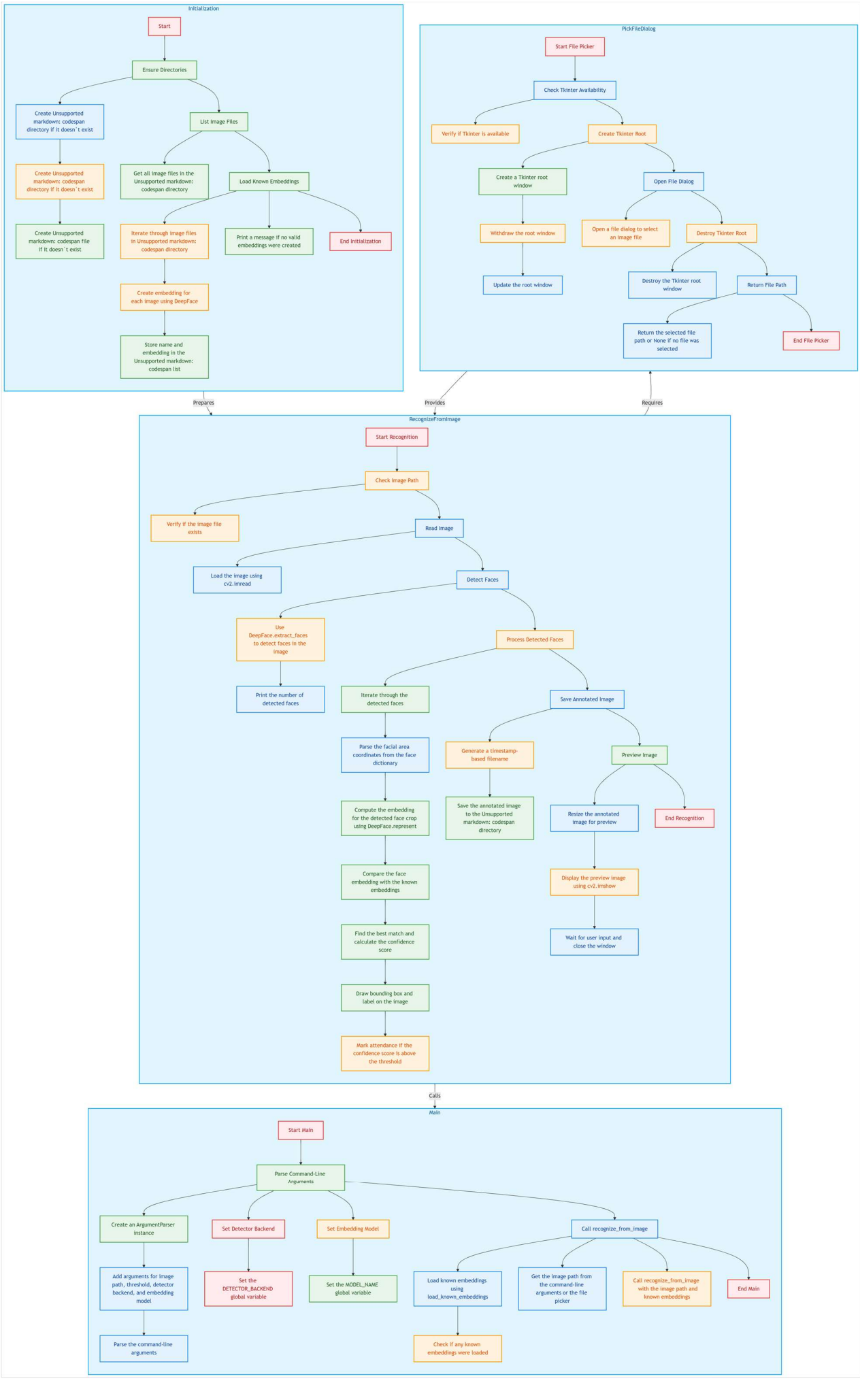
Software Requirements

- **Programming Language**: Python 3.9.x
- **Libraries**: OpenCV, DeepFace, NumPy, Pandas
- **Execution Platforms**:
  - Local: **VS Code**
  - Cloud: **Google Colab**

System Design & Architecture

Process Workflow:

1. Load **known student face images** from a folder.
2. Extract embeddings using **DeepFace (VGG-Face model)**.
3. Capture or upload a **classroom image**.
4. Detect all faces in the image.
5. Compare embeddings with known faces using **cosine similarity**.
6. Mark attendance in **attendance.csv**.
7. Save annotated classroom images with recognized names.


Initialization
Start
Ensure Directories
Create Unsupported markdown: codespan directory if it doesn't exist
List Image Files
Create Unsupported markdown: codespan directory if it doesn't exist
Get all image files in the Unsupported markdown: codespan directory
Load Known Embeddings
Create Unsupported markdown: codespan file if it doesn't exist
Iterate through image files in Unsupported markdown: codespan directory
Print a message if no valid embeddings were created
End Initialization
Create embedding for each image using DeepFace
Store name and embedding in the Unsupported markdown: codespan list
PickFileDialog
Start File Picker
Check Tkinter Availability
Verify if Tkinter is available
Create Tkinter Root
Create a Tkinter root window
Open File Dialog
Withdraw the root window
Open a file dialog to select an image file
Destroy Tkinter Root
Update the root window
Destroy the Tkinter root window
Return File Path
Return the selected file path or None if no file was selected
End File Picker
Prepares
Provides
Requires
RecognizeFromImage
Start Recognition
Check Image Path
Verify if the image file exists
Read Image
Load the image using cv2.imread
Detect Faces
Use DeepFace.extract_faces to detect faces in the image
Process Detected Faces
Print the number of detected faces
Iterate through the detected faces
Save Annotated Image
Parse the facial area coordinates from the face dictionary
Generate a timestamp-based filename
Preview Image
Compute the embedding for the detected face crop using DeepFace.represent
Save the annotated image to the Unsupported markdown: codespan directory
Resize the annotated image for preview
End Recognition
Compare the face embedding with the known embeddings
Display the preview image using cv2.imshow
Find the best match and calculate the confidence score
Wait for user input and close the window
Draw bounding box and label on the image
Mark attendance if the confidence score is above the threshold
Calls
Main
Start Main
Parse Command-Line Arguments
Create an ArgumentParser instance
Set Detector Backend
Set Embedding Model
Call recognize_from_image
Add arguments for image path, threshold, detector backend, and embedding model
Set the DETECTOR_BACKEND global variable
Set the MODEL_NAME global variable
Load known embeddings using load_known_embeddings
Get the image path from the command-line arguments or the file picker
Call recognize_from_image with the image path and known embeddings
End Main
Parse the command-line arguments
Check if any known embeddings were loaded

Implementation

7.1 Implementation in VS Code (Local Execution)

- **Code Overview:**
  - Uses **argparse** for command-line arguments.
  - Loads known faces from a Faces/ folder.
  - Reads classroom image from path or file dialog.
  - Detects faces → Generates embeddings → Matches with database.
  - Logs attendance in **attendance.csv**.
  - Annotates classroom image with names and saves in outputs/.
  - Displays results in a pop-up window.
- **Key Libraries Used:**
  - cv2 – Image processing
  - DeepFace – Face detection & recognition
  - numpy – Embedding calculations
  - pandas – Attendance management
  - tkinter – File dialog

Screenshots:

```
(venv9) PS D:\OEP\Auto Attendance System> python .\attendance_system.py -i "IMG-20250629-WA0043.jpg"
[INFO] Loading known faces from: Faces
[OK] Loaded embedding for 'Achyut'
[OK] Loaded embedding for 'Arpit'
[OK] Loaded embedding for 'Dwarkesh'
[OK] Loaded embedding for 'Pratham'
[OK] Loaded embedding for 'Trush'
[INFO] Processing classroom image: IMG-20250629-WA0043.jpg
[INFO] Detected 4 face(s).
[ATTENDANCE] Marked Achyut at 17:39:10 (conf 32.9%)
[ATTENDANCE] Marked Dwarkesh at 17:39:12 (conf 13.1%)
[INFO] Saved annotated image -> outputs\recognized_20250818_173912.jpg
[INFO] Recognized: Achyut, Dwarkesh
```

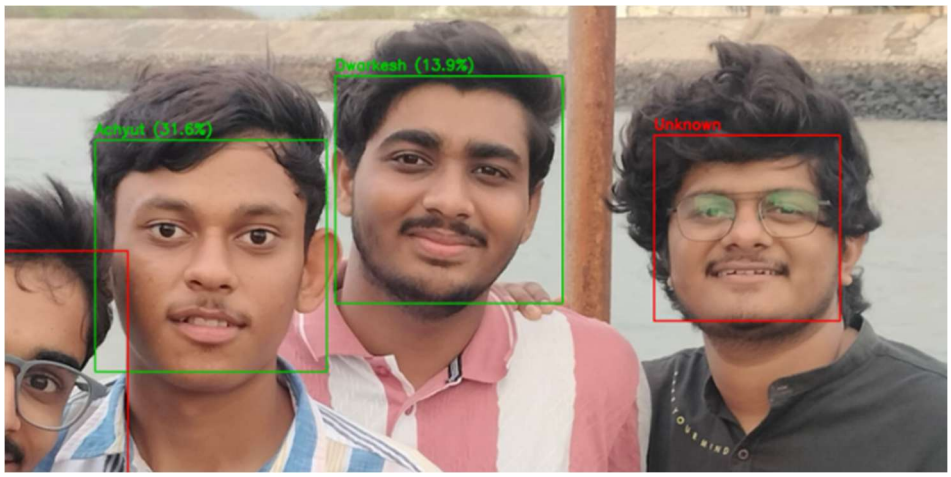
Name	Date	Time	Confidence
Achyut	2025-08-15	19:52:31	0.3292
Dwarkesh	2025-08-15	19:52:33	0.1308
Achyut	2025-08-18	17:39:10	0.3292
Dwarkesh	2025-08-18	17:39:12	0.1308

7.2 Implementation in Google Colab (Cloud Execution)

- **Code Overview:**
  - Removed argparse and tkinter (not supported in Colab).
  - Uses google.colab.files.upload() for classroom image uploads.
  - Displays annotated images using IPython.display.Image().
  - Similar recognition pipeline as VS Code version.
  - Attendance is logged in **attendance.csv**.
  - Outputs are saved in outputs/ folder.
- **Key Libraries Used:**
  - cv2 – Image processing
  - DeepFace – Face recognition
  - google.colab.files – File upload interface
  - IPython.display – Displaying results inline

Screenshots:

```
[INFO] Loading known faces from: Faces
[OK] Loaded embedding for 'Dwarkesh'
[OK] Loaded embedding for 'Trush'
[OK] Loaded embedding for 'Pratham'
[OK] Loaded embedding for 'Arpit'
[OK] Loaded embedding for 'Achyut'
[INFO] Please upload the classroom photo.
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving IMG-20250629-WA0043.jpg to IMG-20250629-WA0043 (2).jpg
[INFO] Processing classroom image: IMG-20250629-WA0043 (2).jpg
[INFO] Detected 4 face(s).
[ATTENDANCE] Dwarkesh already marked today (2025-08-15).
[ATTENDANCE] Achyut already marked today (2025-08-15).
[INFO] Saved annotated image -> outputs/recognized_20250815_155936.jpg
```



Results & Analysis

- **VS Code version** provides **real-time pop-up previews**.
- **Google Colab version** is **portable and requires no local setup**.
- Both systems successfully log attendance with confidence scores.
- Recognition accuracy depends on image quality, face angles, and dataset diversity.

Advantages

- Automated attendance reduces human error.
- Prevents proxy attendance.
- Works both locally and on the cloud.
- Attendance is stored digitally for easy management.

Limitations

- Requires **high-quality images** for accuracy.
- Performance may degrade with **large class sizes**.
- Cannot handle **identical twins or masked faces** effectively.