Image-Based Attendance System with Chatbot Querying

Overview

This project provides a complete workflow for automated attendance tracking using facial recognition and allows users to query the attendance data via a chatbot interface powered by the Groq API.

1. **Training:** Learns student faces from images organized in a specific folder structure.
2. **Attendance Marking:** Processes classroom photos, recognizes known students, marks attendance in subject-specific Excel files based on a schedule, and saves annotated images.
3. **Chatbot Querying:** Allows users to ask natural language questions about the attendance records stored in the Excel files.

Features

- Trains a facial recognition model based on provided student images.
- Recognizes multiple faces in captured classroom images.
- Marks attendance automatically based on the current time and a defined weekly schedule.
- Saves attendance records into `.xlsx` files, one per subject.
- Generates annotated images (with bounding boxes and student IDs) for visual verification.
- Organizes processed and original images into dedicated folders.
- Provides a chatbot interface (using Groq) to query attendance details (e.g., "Who was absent on Monday?", "What is Roll Number 101's attendance percentage?").
- Uses ANSI colors for a more user-friendly terminal experience in the chatbot.

Project Structure

```
image-based-attendance-system/
├── attendance_data/          # Output Excel files for attendance
│   └── SUBJECT_NAME.xlsx
├── captured/                 # Place raw classroom images here for processing
│   └── class_photo_1.jpg
├── marked/                   # Stores annotated images after processing
│   └── SUBJECT_NAME/
│       └── DD-MM-YYYY_1.jpg
├── processed_raw/            # Stores original images after successful processing
│   └── class_photo_1.jpg
├── errored_images/           # Stores images that failed during processing
│   └── problematic_image.jpg
├── training_data/            # Input images for training the face model
```

```
│   ├── STUDENT_ID_1/          # Folder name is the student's ID
│   │   ├── image1.jpg
│   │   └── image2.png
│   └── STUDENT_ID_2/
│       └── image1.jpeg
├── .env                       # Stores API keys (e.g., GROQ_API_KEY) - **DO NOT COMMIT**
├── encodings.pkl              # Output of the training process (face encodings)
├── main.py                    # Main script for marking attendance
├── train_faces.py             # Script to train the face recognition model
├── chatbot-LLM.py             # Script to run the attendance chatbot
├── write_attendance.py        # Helper script for writing to Excel (Assumed)
└── README.md                  # This file
```

Prerequisites

- Miniconda or Anaconda
- Git (optional, for cloning)
- Groq API Key (https://console.groq.com/keys)

Setup and Installation

1. **Clone the Repository (Optional):**

   ```
   git clone <your-repository-url>
   cd image-based-attendance-system
   ```

2. **Create a Conda Environment:**

   ```
   conda create -n attendance_env python=3.9 -y
   # Or choose a different Python version (e.g., 3.10, 3.11) if preferred
   ```

3. **Activate the Environment:**

   ```
   conda activate attendance_env
   ```

4. **Install Dependencies:**

   - **Install `dlib`:** `face_recognition` depends on `dlib`. Installation can sometimes be tricky. Follow the official guides or try these common methods:
     - **Using Conda (Recommended):**

       ```
       conda install -c conda-forge dlib
       ```

     - **Using pip (may require build tools):** Ensure you have CMake and a C++ compiler installed. See dlib documentation or face_recognition installation guide.

```
pip install dlib
```

- **Install other Python Libraries:**

```
pip install face_recognition numpy Pillow openpyxl pandas groq python-dotenv
```

5. **Set up Groq API Key:**
   - Create a file named `.env` in the project's root directory.
   - Add your Groq API key to this file:

```
GROQ_API_KEY=your_actual_api_key_here
```

   - **Important:** Add `.env` to your `.gitignore` file to avoid committing your API key.

Workflow / Usage

Follow these steps in order:

Step 1: Prepare Training Data

1. Create the `training_data/` directory if it doesn't exist.
2. Inside `training_data/`, create a subfolder for each student. The **name of the subfolder must be the student's unique ID** (e.g., `101`, `102`, `student_abc`). This ID will be used in the attendance sheets and annotated images.
3. Place several clear photos of each student (showing their face) into their respective ID folder. More images with variations (angles, lighting) generally lead to better recognition.

Step 2: Train the Face Recognition Model

1. Ensure your `attendance_env` conda environment is active.
2. Run the training script:

```
python train_faces.py
```

3. This will process images in `training_data/`, extract facial encodings, and save them into the `encodings.pkl` file in the root directory.

Step 3: Prepare Class Images

1. Create the `captured/` directory if it doesn't exist.
2. Place the photos taken during a specific class session into this `captured/` folder. Supported formats: `.jpg`, `.jpeg`, `.png`.

Step 4: Run Attendance Marking

1. Make sure the `encodings.pkl` file exists (generated in Step 2).
2. Ensure your `attendance_env` conda environment is active.
3. Run the main attendance script:

```
python main.py
```

4. **Output:**
    - The script identifies the current subject based on time and schedule in `main.py`.
    - Processes images in `captured/`.
    - Saves annotated images to `marked/<SubjectName>/`.
    - Moves original images to `processed_raw/`.
    - Moves problematic images to `errored_images/`.
    - Creates or updates `attendance_data/<SubjectName>.xlsx` with the attendance for the current date.

Step 5: Query Attendance Data with Chatbot

1. Make sure the attendance Excel files exist in `attendance_data/` (generated in Step 4).
2. Ensure your `.env` file with the `GROQ_API_KEY` is present.
3. Ensure your `attendance_env` conda environment is active.
4. Run the chatbot script:

```
python chatbot-LLM.py
```

5. **Interaction:**
    - The script will list the available subjects (Excel files).
    - Enter the number corresponding to the subject you want to query.
    - Ask natural language questions about the attendance data for that subject (e.g., "How many times was student 101 present?", "List all students absent on 25-12-2024").
    - Type `quit` to exit the chatbot or select a new subject file.

Configuration

- **`main.py`:**
    - `class_timings`: Modify the list of tuples for lecture time slots.
    - `WEEKLY_SCHEDULE`: Update the dictionary to reflect your class schedule (Subject names must match desired Excel filenames). Use `None` for free slots.
    - Face Recognition Threshold: Adjust `similarity_scores[closest_match_index] < 0.6` (line ~121) for stricter (lower value) or looser matching.
    - Annotation Appearance: Change font size (`ImageFont.truetype` line ~88) and bounding box width (`draw.rectangle` line ~120) as needed.
- **`.env`:**
    - Stores the `GROQ_API_KEY`.
- **`chatbot-LLM.py`:**
    - `attendance_folder`: Change if your Excel files are stored elsewhere.
    - `model`: You can experiment with different models available on Groq (e.g., "llama3-70b-8192") by changing the `model` parameter in the `client.chat.completions.create` call (line ~54). Check Groq documentation for available models.

- `temperature` : Adjust the creativity/randomness of the LLM response (line ~55). Lower values (like 0.2) are better for factual Q&A.

Future Improvements

- Add a GUI for easier interaction instead of CLI.
- Implement more robust error handling and logging across all scripts.
- Optimize face detection/recognition performance (e.g., using batch processing if handling many images).
- Allow querying across multiple subjects in the chatbot.
- Add functionality to calculate and report overall attendance percentages directly.