**DOCUMENTATION OF THE SMART ATTENDANCE SYSTEM**

**Table of Contents**

1. **Overview**
2. **Features**
3. **Installation Guide**
4. **Usage Instructions**
5. **System Architecture**
6. **Modules and Components**
7. **Troubleshooting**
8. **Future Enhancements**
9. **License**

**1. Overview**

The **Smart Attendance System** automates attendance tracking using **real-time face recognition** and **video processing**. It captures faces via a webcam, matches them against a pre-registered dataset, and logs attendance in a SQLite database. The system includes a user-friendly GUI for managing attendance records.

**2. Features**

* **Real-Time Face Detection**: Uses OpenCV and Haar Cascades/CNN.
* **Face Recognition**: Matches faces with images in the known\_faces folder.
* **Attendance Logging**: Stores entries in a SQLite database with timestamps.
* **GUI Interface**: Built with Tkinter for easy interaction.
* **Duplicate Prevention**: Logs attendance once per day per person.
* **Data Management**: Delete entries by name, date, or all records.

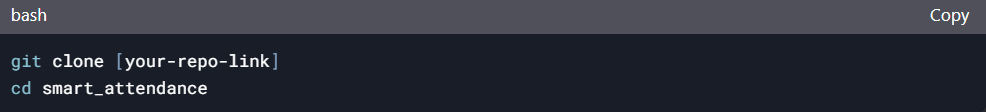
**3. Installation Guide**

**Prerequisites**

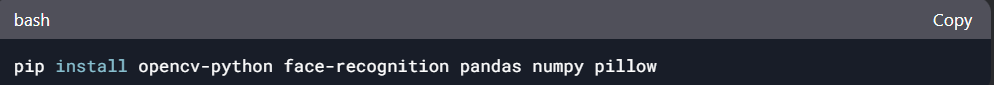
* Python 3.7+
* Webcam

**Steps**

1. **Clone the Project**:



**2.Install Dependencies**:



**3.Folder Structure**:

Copy

smart\_attendance/

├── known\_faces/ # Pre-registered face images

├── create\_dataset.py # Script to register new users

├── database.py # Database operations

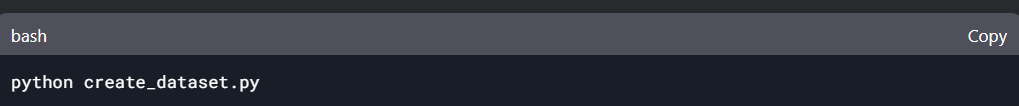
├── gui.py # Main GUI application

└── attendance.db # SQLite database (auto-generated)

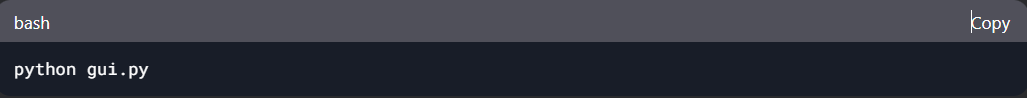
**5.Initialize the Database**:

**4. Usage Instructions**

**1. Register New Users**

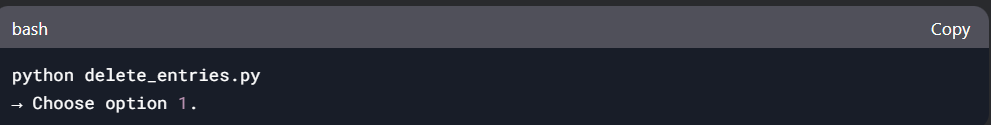
* Enter the person’s name (e.g., john).
* The script saves 20 face images to known\_faces/john\_0.jpg, john\_1.jpg, etc.

**2. Launch the GUI**

**GUI Components**:

* **Live Video Feed**: Displays real-time face detection.
* **Start/Stop Attendance**: Toggle face recognition.
* **Attendance Log**: Shows the last 10 entries.
* **Delete Entries**: Use the delete\_entries.py script or SQL queries.

**3. Delete Attendance Records**

****

* **Delete by Name/Date**:

**5. System Architecture**

**Workflow**

1. **Face Detection**:  
   OpenCV detects faces in the webcam feed using Haar Cascades.
2. **Face Encoding**:  
   face\_recognition generates 128D embeddings for detected faces.
3. **Database Integration**:  
   Logs attendance in attendance.db (SQLite).
4. **GUI**:  
   Displays live video, attendance logs, and controls.

**Flowchart**

Webcam → Face Detection → Face Encoding → Database Match → Attendance Logged

**6. Modules and Components**

**1. `create\_dataset.py**

* **Purpose**: Capture and save face samples for training.
* **Key Functions**:
  + cv2.VideoCapture(): Accesses the webcam.
  + face\_cascade.detectMultiScale(): Detects faces in frames.

**2. database.py**

* **Purpose**: Manage SQLite database operations.
* **Key Functions**:
  + create\_connection(): Connects to the database.
  + mark\_attendance(): Prevents duplicate entries.

**3. gui.py**

* **Purpose**: Main application interface.
* **Key Features**:
  + Real-time video feed with bounding boxes.
  + Attendance log table (last 10 entries).
  + Start/Stop buttons for attendance tracking.

**4. delete\_entries.py**

* **Purpose**: Delete attendance records programmatically.
* **Commands**:
  + Delete all entries, by name, or by date.

**7. Troubleshooting**

| **Issue** | **Solution** |
| --- | --- |
| Webcam not detected | Replace cv2.VideoCapture(0) with cv2.VideoCapture(1) in gui.py. |
| Low recognition accuracy | Use high-resolution, well-lit images in known\_faces. |
| Database errors | Ensure attendance.db exists in the project folder. |
| Dependency issues | Reinstall libraries: pip install -r requirements.txt. |

**8. Future Enhancements**

1. **Liveness Detection**: Prevent spoofing with eye-blink detection.
2. **Cloud Integration**: Store data on AWS/Azure.
3. **Mobile App**: Add a companion app for remote access.
4. **Advanced Analytics**: Generate attendance reports (PDF/Excel).
5. **Multi-Face Support**: Improve handling of multiple faces in a frame.