Explanation of the Code in Simple Steps (submain.py)

Simple Explanation of the Face Recognition Streamlit Code for a Student Researcher

This Python code is a **Face Recognition System** that:
✅ Reads **face coordinates from a CSV file**
✅ Captures **live face data** from a webcam
✅ Compares the captured face with **stored records**
✅ Finds the **closest match** and retrieves the **guest's name** from a **database**
✅ Opens the **corresponding guest's image**

Let’s go through it **step by step**:

Step 1: Import Required Libraries

The code imports several essential libraries:

- **cv2 (OpenCV)** – For face detection and image handling.
- **streamlit** – To create a simple web-based interface.
- **pandas** – To read and process face data from a CSV file.
- **sqlite3** – To retrieve guest names from a database.
- **numpy** – To calculate the closest match.
- **os** – To open the correct guest image file.

Step 2: Read Face Data from a CSV File

🔹 The function read_csv(file_path) reads a **CSV file** that contains stored face coordinates:

- X (X-position of the face in pixels)
- Y (Y-position of the face in pixels)
- Width (Face width in pixels)
- Height (Face height in pixels)
- Guest_ID (Unique guest identifier)

📌 **Purpose:** This CSV file acts as a **reference database** for previously recorded faces.

Step 3: Capture a Face from the Webcam

🔹 The function capture_face() uses **OpenCV** to:

1. **Open the webcam**
2. Detect a **face** using Haarcascade XML

3. **Draw a rectangle** around detected faces
4. Wait for the user to **press "Space"** to capture the face
5. Extract **X, Y, Width, and Height** of the detected face

📌 **Purpose:** It allows **real-time face detection** and stores the position and size of the detected face.

Step 4: Compare Captured Face with Stored Data

🔹 The function find_closest_match(test_X, test_Y, test_Width, test_Height, csv_data) performs **face matching** by:

1. Calculating the **difference** between the captured face and each stored record
2. Finding the **closest match** using Euclidean distance
3. Returning the **best-matching guest record**

📌 **Purpose:** Ensures the system can **identify people based on previously stored facial data**.

Step 5: Retrieve the Guest Name from the Database

🔹 The function get_guest_name(guest_id, db_file="guests.db") connects to the **SQLite database** and searches for the **guest name using the guest's ID**.

📌 **Purpose: Links** the detected face to **stored guest information**.

Step 6: Open the Guest's Stored Image

🔹 The function open_guest_image(guest_name, image_folder="captured_faces/"):

1. Searches for an image file named **"Guest_Name.jpg"**
2. Opens the image if it exists
3. Displays an **error message** if no image is found

📌 **Purpose:** Shows the **actual image of the guest** for final verification.

Step 7: Create the Web App with Streamlit

🔹 The main() function builds a **simple web interface** where:

1. The user clicks **"Capture Face for Recognition"** → Captures face coordinates
2. The system **compares** captured data with stored records
3. The **closest match** is displayed along with the guest’s name
4. Clicking **"View Guest Image"** opens the stored guest photo

📌 **Purpose:** Allows **non-technical users** to perform **face recognition** through an easy-to-use **web interface**.

Step 8: Run the Application

```
if __name__ == "__main__":

    main()
```

🔹 This ensures the **Streamlit app starts** when the script is run.

How the System Works (Step-by-Step Flow)

🔵 **Step 1:** Read stored face data from **CSV file**
🔵 **Step 2:** Start webcam, **detect and capture face**
🔵 **Step 3:** Compare **captured face** with stored records
🔵 **Step 4:** Retrieve the **guest name** from the database
🔵 **Step 5:** Show the **guest's stored image**

What You Need to Run the Code

📌 Install required Python libraries:

bash

CopyEdit

```
pip install streamlit opencv-python pandas numpy
```

📌 Ensure the following files exist:
✓ haarcascade_frontalface_default.xml (for face detection)
✓ face_data.csv (for stored face coordinates)
✓ guests.db (SQLite database with guest names)
✓ captured_faces/ (Folder containing stored guest images)

Expected Use Case

🔹 Click **"Capture Face for Recognition"** → Detects & records face data
🔹 Compares with stored records and **displays the closest guest match**
🔹 Click **"View Guest Image"** → Opens the guest's stored photo

---