FACE RECOGNITION

## **Overview**

This system integrates a **face recognition feature** using **FastAPI (backend)** and **React (frontend)**. The system captures images from a webcam, detects faces, and either recognizes existing individuals or registers new ones. The backend leverages **TensorFlow and face\_recognition** libraries for processing, while the frontend provides a **user-friendly UI** to interact with the system.

## **Workflow**

### **1. System Initialization**

* The backend is started using **FastAPI and Uvicorn**, which sets up the API for face recognition and registration.
* The system connects to a **MongoDB database**, where recognized faces and user details are stored.
* Existing face encodings are preloaded from the database for quick recognition.

### **2. Capturing Image (Frontend - React)**

* The frontend accesses the **webcam** to capture an image when the user clicks the “Detect Face” button.
* The captured image is converted into a **blob format** and sent to the backend for processing.

### **3. Face Recognition (Backend - FastAPI)**

* The backend receives the image and converts it into a format suitable for processing.
* A **face detection algorithm** is applied to locate any faces present in the image.
* If **multiple faces** are found, the largest face is chosen.
* The detected face is then converted into a **numerical encoding** for comparison.
* The system **searches the database** for a match.

#### **Scenarios:**

1. **If a match is found:**
   * The backend retrieves the person’s details from the database.
   * The frontend displays the **recognized person’s information**, including **name, age, height, weight, and confidence level**.
2. **If no match is found:**
   * The user is prompted to enter their details (name, age, height, weight).
   * The system **stores the face encoding and personal details** in the database for future recognition.

### **4. Storing New Person’s Data**

* When a new person is added, the **face encoding is stored** in the database along with their details.
* This allows the system to recognize the individual the next time they appear in front of the camera.

## **Key Features**

* **FastAPI Backend:** Provides a high-performance API with real-time processing.
* **Face Recognition:** Uses deep learning to identify and distinguish individuals.
* **Database Integration:** Stores user details and face encodings in **MongoDB**.
* **Webcam Capture:** Allows real-time face detection from a user’s webcam.
* **Automatic Matching:** Compares detected faces with stored data to identify individuals.

## **System Requirements**

### **Backend:**

* Python 3.7+
* FastAPI
* Uvicorn
* OpenCV
* face\_recognition
* MongoDB

### **Frontend:**

* React.js
* Tailwind CSS
* Fetch API for backend communication

## **How to Run the System**

### **Backend**

1. Navigate to the backend folder:

cd backend

2.Start the FastAPI server:

uvicorn main3:app --reload

### **Frontend**

1. Navigate to the frontend folder:

cd frontend

1. Start the React app:

npm start

**Output**





