✅ Complete Project Summary: PostureGuard AI

PostureGuard AI is an intelligent **real-time posture monitoring system** built using **MediaPipe, OpenCV, and Flask**. It analyzes human posture from **images, videos, or webcam streams** to detect **bad squats or desk sitting posture**, and provides **instant feedback with confidence scores** and analytics.

🔧 How It Works:

1. **Frontend** sends image/video or real-time webcam frames to backend (via REST API or WebSocket).
2. **Backend (Flask)** processes the data:
    - Decodes image using cv2
    - Detects body keypoints using **MediaPipe Pose**
    - Calculates angles, compares with defined thresholds
    - Returns posture status: **good**, **warning**, or **bad**
3. Real-time session stats are calculated: frame count, average confidence, alerts.
4. WebSocket allows **live updates** for streaming camera feeds.

🧠 Key Technical Skills You Demonstrated

Skill	**Description**
Flask	REST API, real-time WebSocket communication
MediaPipe Pose	Human keypoint detection and body landmark tracking
OpenCV	Image/video decoding, preprocessing
Socket.IO	Real-time bi-directional data for live posture monitoring
Python	Backend logic, angle calculations, posture rules
Data Analysis	Tracking session statistics, confidence averaging
Modular Design	Config file, separate analyzer class, structured error handling
CORS, Logging	Security & observability setup for backend

💬 How to Explain It in an Interview

🧩 What was the problem?

Many people unknowingly maintain poor posture while squatting or sitting at a desk, which can lead to injuries or long-term health issues. There's a need for a real-time posture monitoring tool.

🔍 How did you solve it?

I built **PostureGuard AI**, a full-stack application that detects human posture using video or camera feed and provides **real-time feedback**. I used **MediaPipe Pose** for body landmark detection and designed custom posture rules for activities like **squats** and **desk sitting**.

🛠 What tech stack did you use?

- **Backend:** Python, Flask, Flask-SocketIO
- **Posture Analysis:** MediaPipe, OpenCV
- **Real-time:** WebSocket (Socket.IO)
- **Frontend (optional):** Vue/React for camera stream and visualization (if you used it)
- **Deployment:** Render / local server

🚧 What challenges did you face?

- Ensuring smooth real-time performance for live video analysis
- Managing session-wise data (confidence, posture score, frame history)
- Handling poor lighting or low detection confidence from MediaPipe
- Designing posture rules that were strict but realistic (not too sensitive)

🚀 How did you deploy it?

I deployed the Flask backend on **Render / localhost**, enabled CORS, and connected it to a frontend via WebSocket for live feedback.

(If not deployed yet: "It can be easily deployed using Render, Heroku, or Dockerized for production.")

📄 How to Add It to Your Resume

▶️ Project Title: PostureGuard AI – Real-Time Posture Monitoring System

Tech Stack: Python, Flask, MediaPipe, OpenCV, Socket.IO

Description & Resume Bullet Points:

pgsql

CopyEdit

```
- Built a real-time AI-based posture monitoring system using MediaPipe and Flask

- Implemented video and webcam-based posture detection with OpenCV and WebSocket support
```

- Designed custom rule-based analysis for squats and desk posture using body landmark angles

- Enabled live feedback and posture statistics tracking (confidence, good/bad posture count)

- Optimized posture classification using custom angle thresholds and dynamic confidence scoring

- Configured Flask API and WebSocket for image/video upload and real-time camera feed analysis

---

🌐 Deployment & Demo (Optional)

If you **deployed** the backend or created a **video demo**, include:

- ✅ URL or IP (Render, Heroku, Localhost + Ngrok)
- 🎥 Record a demo using OBS and upload to YouTube/Drive
- 📷 Add screenshots in your portfolio/github README