Posture Detection ML Model – Project Documentation (Python Version)

1. Project Objective

Build a Python-based machine learning system that detects a person's **desk posture** in real time while they work on a laptop or desktop.

The system should analyze a webcam feed and classify posture into categories such as:

- **Good Posture**
- **Slight Slouch**
- **Heavy Slouch**
- **Leaning Left / Right**
- **Head Forward / Too Close to Screen** (optional)

The focus is on using **pre-trained pose estimation models** to extract body keypoints and then designing a **custom ML model or rule-based logic** for posture classification.

2. Scope of the Project

The final system should:

1. Access live video from the webcam.
2. Detect human pose landmarks (shoulders, neck, nose, hips, etc.).
3. Extract meaningful posture-related measurements (angles, distances).
4. Use ML or threshold-based logic to determine posture correctness.
5. Display real-time posture feedback on screen.
6. (Optional) Trigger alerts when posture is consistently bad.

3. Technology Stack

Programming Language

- Python 3.9 or above

Core Libraries

- **MediaPipe** – for extracting pose landmarks

- **OpenCV** – for live webcam capture and visualization
- **NumPy** – for mathematical calculations
- **Scikit-learn** (optional) – for training a posture classifier
- **Flask/FastAPI** (optional) – to provide a REST API for frontend integration

4. System Architecture

4.1 Pose Estimation Layer

Uses a pre-trained model (MediaPipe Pose) to detect body landmarks such as:

- Nose
- Left/Right Shoulder
- Left/Right Hip
- Left/Right Ear
- Eyes, elbows, etc. (only if needed)

This layer returns keypoints with x–y coordinates and confidence values.

4.2 Feature Extraction Layer

Transforms raw pose landmarks into usable posture metrics. Examples:

- **Neck Angle**
  Measures how far the head is bending forward.
- **Back/Torso Angle**
  Indicates slouching or sitting upright.
- **Shoulder Tilt**
  Identifies leaning to one side.
- **Head Distance From Shoulders**
  Detects head-forward posture or proximity to the screen.

These features form the input for the posture classifier.

4.3 Classification Layer

Two approaches:

A. Rule-Based System (Simpler)

You manually define thresholds such as:

- Neck angle above a certain degree → slouch
- Shoulder tilt above a threshold → leaning
- Back angle too far forward → heavy slouch

Useful when you do not have a labeled dataset.

B. Machine Learning Classifier (Recommended)

Train a small ML model like RandomForest or SVM using extracted features.

You create a dataset containing:

neck_angle	back_angle	shoulder_tilt	head_dist	label

Labels:

- 0 = Good
- 1 = Slight Slouch
- 2 = Heavy Slouch
- 3 = Lean Left
- 4 = Lean Right

The trained model will predict posture categories based on real-time input.

5. Dataset Creation (For ML Model)

If opting for machine learning, you need your own dataset.

How to create it:

1. Run the camera and show pose landmarks.
2. Capture posture features every time the user presses a key (like "G" for good posture).
3. Save the computed angles/distances along with the correct label.
4. Collect at least:
    - 150–200 samples of good posture
    - 150–200 samples of slight slouch
    - 150–200 samples of heavy slouch

    - Additional samples for leaning left/right

The dataset is stored in a CSV file.

6. Model Training (ML Option)

Steps:

1. Split dataset into training & testing sets.
2. Train a simple model (RandomForest recommended).
3. Evaluate accuracy using classification metrics.
4. Save trained model for real-time usage.

Note: This project does **not** require deep learning training; pose estimation is already handled by MediaPipe.

7. Real-Time Monitoring Pipeline

Once the model is ready, real-time processing will follow this loop:

1. Capture webcam frame.
2. Detect pose landmarks.
3. Compute posture features.
4. Feed features to classifier (or rule-based logic).
5. Determine posture class.
6. Display posture status on screen (GOOD / BAD / SLOUCH).
7. (Optional) Trigger an alert if posture is bad for a continuous period (e.g., 10 seconds).

8. Optional Extensions

A. Local UI Alerts

- Show colored overlay:
    - Green = Good
    - Yellow = Slight Slouch
    - Red = Heavy Slouch

B. Sound or Notification Alerts

- Play a beep after 10 seconds of slouching.

C. REST API for MERN Integration

Expose an endpoint like:

`/status → { posture: "slouch", neck_angle: 30.5 }`
so a React web app can show notifications.

D. Logging

Store posture logs over time to visualize:

- How long user stays in good posture
- Peak slouch times
- Daily posture score

9. Expected Deliverables

Your friend should produce the following:

1. Documentation

- Explanation of the entire pipeline
- How features are computed
- Threshold values (for rule-based)
- Dataset structure (for ML version)

2. Posture Detection Module

- Webcam and pose detection integration
- Feature extraction logic
- Posture classification logic (rule-based or ML)

3. Optional Components

- Data collection script
- ML training script
- REST API server

- Posture logging/reporting system

10. Summary

This project does NOT require training a pose estimation model.
Instead, it uses:

1. **Pre-trained pose model** → extract body keypoints
2. **Custom logic or ML model** → classify posture
3. **Visualization layer** → show real-time posture feedback