Workout Analysis API - Technical Workflow

1. Project Summary
This Flask-based Workout Analysis API accepts exercise videos, uses Gemini AI to analyze exerci
and overrides repetition counts using MediaPipe pose detection. The output is returned as structure

2. Folder Structure
```
APP_MEDIAPIPE/
■■■ app_api.py            # Main Flask API
■■■ .env                  # Gemini API key
■■■ uploads/              # Stores uploaded videos
■■■ requirements.txt      # Python dependencies
■■■ mediapipe_utils/
■   ■■■ rep_counter.py    # Pose-based rep counting logic
■■■ venv/                 # Virtual environment
```

3. How It Works (Step-by-Step)
1. Entry: A client sends a POST request to /analyze with a video file in form-data (key: 'video').
2. app_api.py processes the video, calls Gemini and MediaPipe.
3. rep_counter.py uses angles to count reps from body landmarks.
4. Final JSON includes: exercise_name, reps, calories, form feedback.
5. Exit: API responds with the structured JSON.

4. API Endpoint
POST /analyze
Form-Data Key: video (accepted formats: mp4, avi, mov, webm)

5. Sample Output
```
{
  "exercise_name": "Squat",
  "repetitions": 14,
  "calories_burned": 98,
  "form_analysis": {
    "posture": "Great alignment",
    "range_of_motion": "Full depth",
    "tempo": "Controlled and steady",
    "common_mistakes": ["Knees going too far forward"]
  },
  "performance_score": 8.7,
  "encouragement_and_tips": {
    "positive_feedback": ["Great control during descent!"],
    "improvement_tips": ["Try to keep heels flat on the ground"]
  }
}
```

6. .env Format
```
GOOGLE_API_KEY=your_gemini_api_key_here
```

7. Summary for Node.js Developer
- Flask backend with /analyze endpoint.

- Accepts video, uses Gemini for analysis, MediaPipe for reps.
- Returns clean JSON with scores, tips, and feedback.