Gym Guardian

GROUP 22:

Harsh Ratnaparkhe	24BCE10892
Rudradeep Chakraborty	24BCE11151
Prathamesh Kapil Dhakad	24BCE10301
Yash Mehta	24BCE10638
Kartikey Karanwal	24BCE10508

SUPERVISOR:

Dr. Dheresh Soni

Introduction

- **The Problem:** The high cost and inconvenience of personal trainers make personalized fitness guidance inaccessible for many.
- **The Solution:** We have developed GymGuardian, a "virtual personal trainer" that uses artificial intelligence to analyze your exercise form in real-time.
- **Our Goal:** To provide an accessible, cost-effective, and convenient way for anyone to get data-driven feedback on their workouts, using only a standard webcam.
- **Value Proposition:** GymGuardian helps users count repetitions accurately, improve their form, and prevent injuries, all from the comfort of their home.

Existing works with limitations

- **Existing Solutions:** Human Personal Trainers: The gold standard, but very expensive and require scheduling.
- **Fitness Apps:** Most are just video libraries or manual workout logs; they don't provide real-time, corrective feedback.
- **Sensor-Based Systems :** Can be very accurate but require expensive, specialized hardware, creating a barrier to entry.(e.g., Wearables, 3D Cameras)
- **The Gap We Fill:** There is a need for a system that is low-cost, requires no special hardware, and still provides intelligent, real-time feedback.

Novelty of the Project

The innovation of GymGuardian lies in its accessibility and intelligent application.

- **Hardware-Agnostic:** It provides advanced biomechanical analysis using only a standard 2D webcam, eliminating the need for expensive 3D sensors or wearables.
- **Markerless Tracking:** The system works automatically by identifying body joints. The user does not need to wear any special markers or clothing.
- **Exercise-Specific Logic:** The system is not one-size-fits-all. It uses unique logic for different exercises, such as angle-tracking for bicep curls and relative-distance tracking for pull-ups.
- **Intelligent Form Correction:** Beyond simple counting, the system includes modules for real-time form analysis, such as the neck and lower back alignment check for push-ups.

Real-Time Usage

1. A user starts the application and aims their webcam.
2. As they perform an exercise (e.g., a squat), the screen shows their live video feed.
3. An overlay on the screen displays.
4. The repetition counter increasing with each valid squat.
5. A progress bar that fills as they go down and empties as they come up, visualizing their range of motion.
6. This immediate feedback allows the user to self-correct their form (e.g., "Knees are bending") mid-workout.

REQUIREMENTS

HARDWARE

A standard PC or laptop.
A webcam
(720p or higher).

SOFTWARE

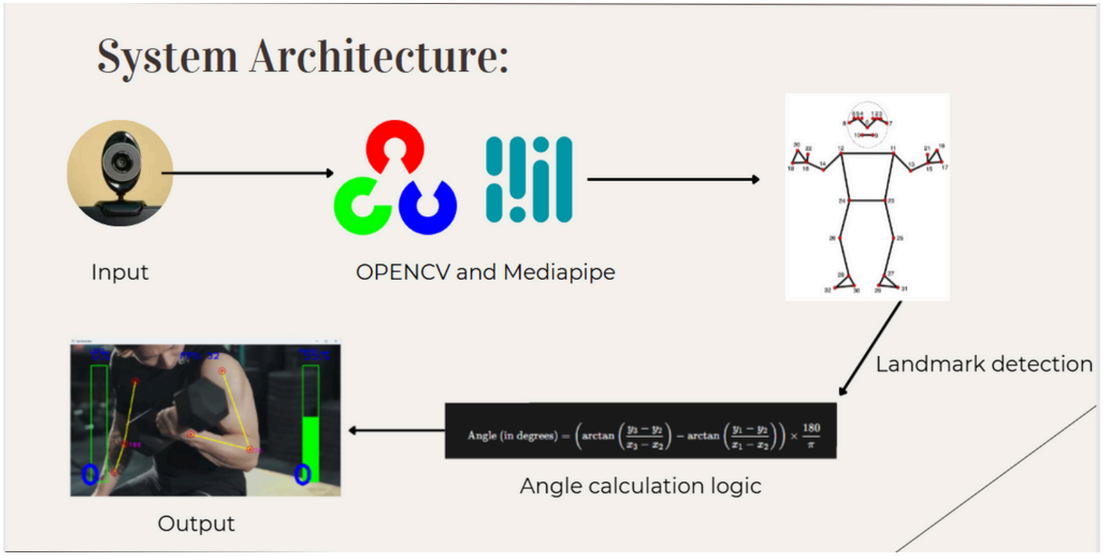
Python 3.8+
OpenCV: Video capture and UI rendering.
MediaPipe: CoreAI-based pose estimation.
NumPy: Numerical calculations and interpolation.

SYSTEM ARCHITECTURE

Layer 1: Input Layer (Data Acquisition)
A block labeled "Webcam" provides a video stream to a block labeled "Image Pre-processing (Resize, Flip)."

Layer 2: Processing Layer (Core Engine)
The frame flows to a large block containing three sub-blocks: "Pose Estimation (MediaPipe)," which feeds data to "Biomechanical Analysis (Angle Calculation)," which in turn feeds data to "Exercise Logic (Rep Counting)."

Layer 3: Presentation Layer (Visual Feedback)
The data from the processing layer (rep count, percentage) flows to a block labeled "UI Rendering (OpenCV)." This block outputs the final image to a block labeled "User Display."



LITERATURE REVIEW

Our review of the field confirmed that while human trainers are highly effective, their cost is a major barrier.
We found that many existing fitness apps are passive video libraries, and more advanced systems require specialized sensors, wearables, or 3D cameras (like the Microsoft Kinect).

Our research identified a significant gap for a low-cost, software-only solution. We selected MediaPipe's pose estimation model as our foundation, as it is a state-of-the-art, real-time, and lightweight model designed specifically for this type of 2D video analysis.

MODULE DESCRIPTION

Our system is designed with five key logical modules:

- **User Feedback & Rendering Module:**

Input: The original frame, rep count, and percentage.
Process: Draws the progress bar, text, and counters onto the frame.
Output: The final video frame displayed to the user.

- **Exercise Logic & Repetition Counting Module:**

Input: The real-time angle/distance value.
Process: Maps the value to a 0-100% scale; uses state logic to track the movement and count full repetitions.
Output: The current rep count and percentage.

- **Video Input & Pre-processing Module:**

Input: Raw webcam video.
Process: Captures, resizes, and flips the video frame.
Output: A clean, standardized frame.

- **Pose Estimation Module:**

Input: A standardized video frame.
Process: Uses the MediaPipe AI model to find all 33 body landmarks.
Output: A list of landmark coordinates (lmList).

- **Biomechanical Analysis Module:**

Input: The lmList of landmark coordinates.
Process: Calculates specific joint angles or relative positions based on the exercise.
Output: A numerical value (e.g., angle, distance).

MODULE WORK FLOW EXPLANATION

The workflow for GymGuardian is a continuous, real-time loop:

First, the Input Layer grabs a single frame from the webcam and prepares it.

Second, the frame is passed to the Processing Layer. The AI model finds the user's joints, the analysis module calculates the current angle, and the logic module checks if a rep has been completed.

Third, the new rep count and percentage are sent to the Presentation Layer, which draws the progress bar and counter on the frame.

Finally, this newly augmented frame is displayed to the user. This entire loop repeats dozens of times per second (as shown by the FPS counter) to create a smooth, interactive experience.

IMPLEMENTATION AND CODING

LANGUAGE: PYTHON

CORE LIBRARIES:

OpenCV: Used for all video capture, window display, and UI drawing (cv.VideoCapture,cv.imshow, cv.putText, cv.rectangle).

MediaPipe: Used for the core AI-based pose estimation.

NumPy: Used for fast numerical calculations, especially linear interpolation (np.interp) to map angles to percentages.

Tkinter: Used for User Interface of the app.

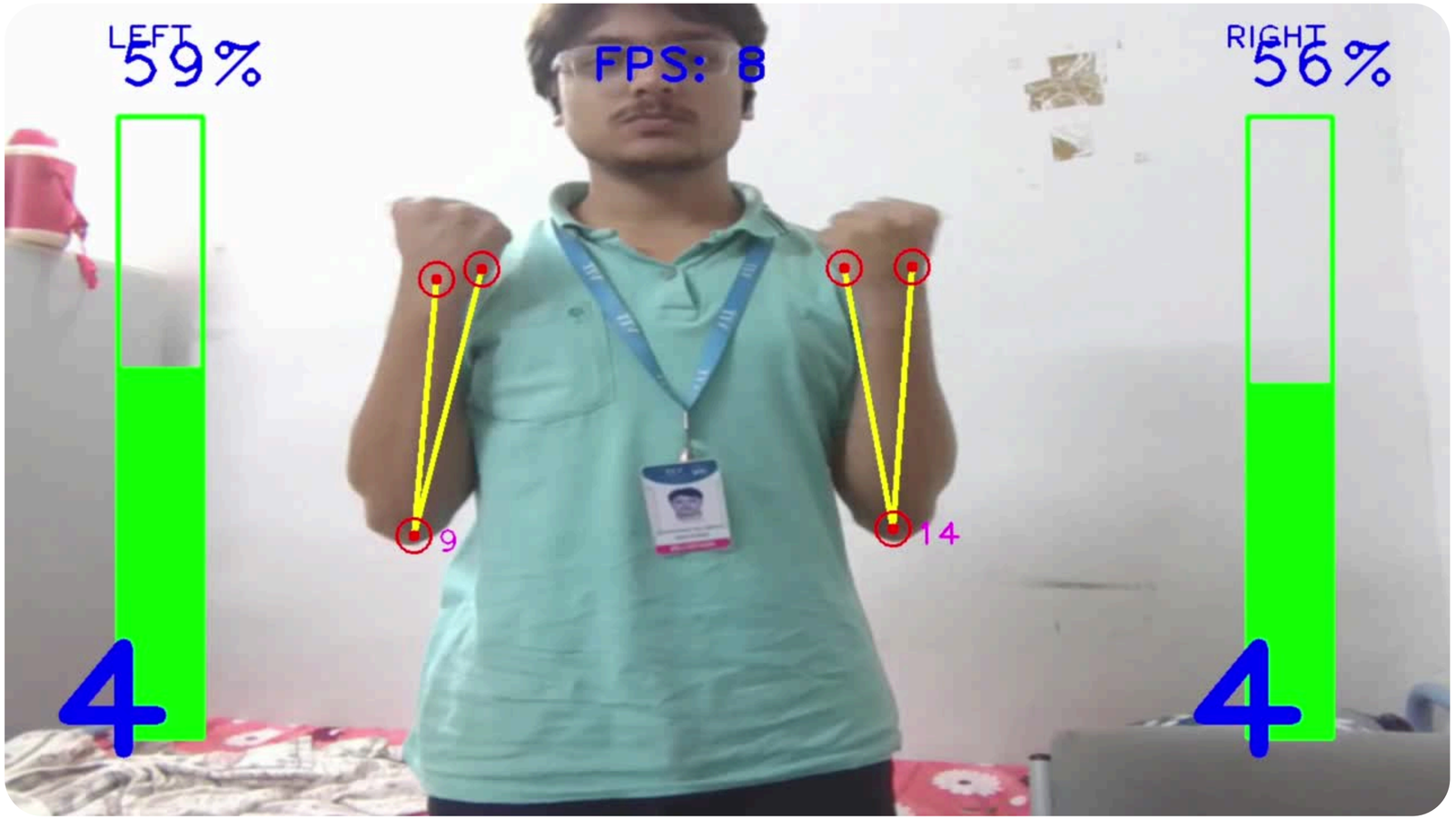
KEY CODE COMPONENTS:

PoseEstimationModule.py: A reusable class we built to make the MediaPipe model easy to use. Its key methods are findPose, findPosition, and findAngle.

Exercise Scripts (e.g., SquatsTrainer.py): Each script imports our PoseEstimationModule and contains the unique landmark IDs, angle thresholds, and counting logic specific to that exercise.

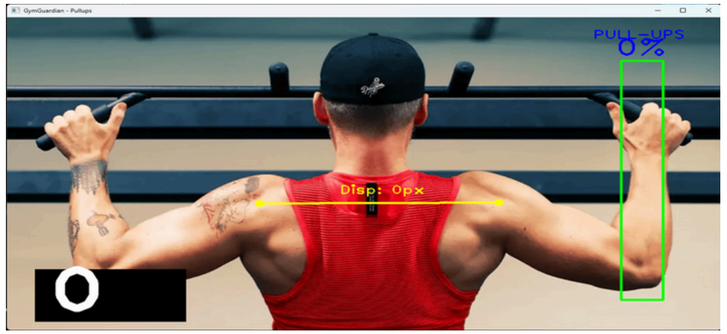
app.py: Contains the main script with the UI and function calling for each trainer modules

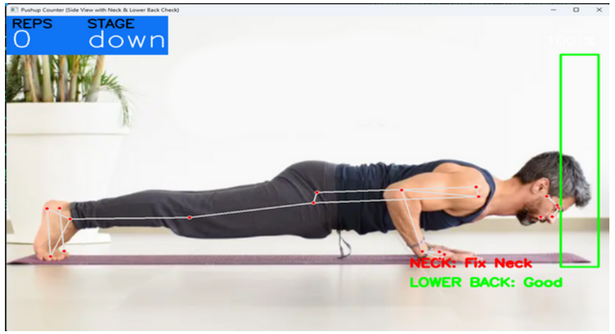
Demo Video



LINK => https://drive.google.com/file/d/1sYNkfymFJXmunAu2XSE80FJLNrahn5Wk/view?usp=sharing

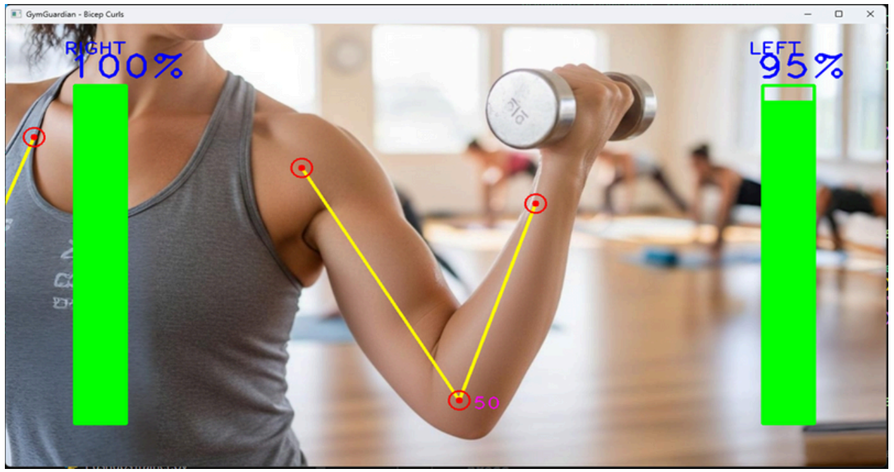
Snap Shots


GymGuardian - Pullups
PULL-UPS
0%
Disp: 0px
0


Pushup Counter (Side View with Neck & Lower Back Check)
REPS
0
STAGE
down
NECK: Fix Neck
LOWER BACK: Good


GymGuardian - Squat Counter
SQUAT
18%
For best tracking, stand at a slight angle to the camera
250
KNEES: GOOD
0


GymGuardian - Squat Counter
SQUAT
18%
For best tracking, stand at a slight angle to the camera
250
KNEES: GOOD
0


GymGuardian - Bicep Curls
RIGHT
100%
LEFT
95%
50

TESTING

WE CONDUCTED SEVERAL TYPES OF TESTING TO ENSURE THE SYSTEM IS ROBUST:

FUNCTIONAL TESTING: WE PERFORMED EACH EXERCISE TO VERIFY THAT REPETITIONS WERE COUNTED ACCURATELY AND THAT THE PROGRESS BAR MAPPED CORRECTLY TO THE FULL RANGE OF MOTION.

Usability Testing: The system was tested under various real-world conditions:

- **Lighting:** Tested in both bright and low-light environments. (Result: Works well, but performance degrades in very dark rooms).
- **Clothing:** Tested with both athletic and baggy clothes. (Result: Works best with clothing that doesn't completely hide the body's joints).
- **Backgrounds:** Tested against both clean and cluttered backgrounds. (Result: The model is robust but works best with a clear background).
- **Performance Testing:** We continuously monitored the FPS (Frames Per Second) to ensure the application runs smoothly and the feedback feels instantaneous.

RESULT AND DISCUSSION

Input: The system takes a standard 2D video stream from a user's webcam.

Output: The system outputs that same video stream, but augmented with real-time data: a rep counter, a percentage, a progress bar, and form cues.

Discussion of Results:

- **Success:** The project is a success. We've proven it is 100% feasible to create a real-time, markerless virtual trainer using only a webcam and open-source libraries.
- **Accuracy:** The MediaPipe model is exceptionally robust. The heuristic (rule-based) counting logic is highly efficient and accurate for the defined exercises.
- **Limitations:** The main limitation is occlusion (when one body part blocks another from the camera's view). As a 2D system, it also cannot detect all form errors (like knee-caving in squats).

CONCLUSION

Summary: GymGuardian successfully achieves its goal of being an accessible, low-cost, and effective AI personal trainer. By leveraging OpenCV and MediaPipe, it provides accurate rep counting and valuable form analysis using common hardware.

Future Work:
Develop more advanced form-checking algorithms.
Add user profiles to track progress and workout history over time

THANK YOU