iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 13 **Issue:** V **Month of publication:** May 2025

**DOI:** https://doi.org/10.22214/ijraset.2025.70547

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue V May 2025- Available at www.ijraset.com*

# AI Human Fitness Tracker using Computer Vision with MediaPipe

Yashraj Mishra[1], Ankita Jaiswal[2], Anany Shukla[3], Abhishek Verma[4], Humesh Verma[5], Dr. Goldi Soni[6]
*Amity School of Engineering and Technology, Amity University Chhattisgarh*

***Abstract:*** ***In recent years, the integration of artificial intelligence (AI) into health and fitness domains has significantly enhanced personal training and physical wellness monitoring. This research introduces an AI-powered Fitness Tracker system that utilizes computer vision and pose estimation techniques to detect human body posture and accurately count repetitions or steps for various physical exercises. The system leverages MediaPipe for real-time human pose detection, computing joint angles to analyse movements and classify exercises such as push-ups, pull-ups, squats, sit-ups, and walking. It incorporates audio feedback for correct posture recognition and rep completion, enhancing user engagement and form correction. The architecture is modular, consisting of key components: pose detection, angle calculation, exercise classification, and feedback generation. This approach minimizes the need for external sensors or wearable devices, offering a non-intrusive, camera-based solution that is both accessible and scalable. The proposed system aims to assist users in performing workouts with improved accuracy and consistency, promoting a more effective and injury-free fitness routine.***

***Keywords: Artificial Intelligence, Human Pose Estimation, AI-Powered Workout Tracker, Action Recognition, Computer Vision in Fitness, Streamlit Application, MediaPipe [10], Exercise Repetition Counter, Real-Time Video Analysis***

## I. INTRODUCTION

### *A. Background*

With the rise of health awareness and fitness culture worldwide, technology has increasingly become integrated into personal health management. Artificial Intelligence (AI) and Computer Vision have opened new doors for real-time human pose detection and activity monitoring without the need for wearables or physical trainers. These advancements have led to the development of intelligent fitness applications that provide automated feedback, ensuring correct form, counting repetitions, and even alerting users to incorrect movements. Traditional fitness tracking often depends on wearable sensors like smartwatches or fitness bands. While effective, these devices have limitations: they can be expensive, uncomfortable, require regular charging, and may not always accurately detect complex movements or body posture. Moreover, hiring a personal trainer is not always feasible for many users due to financial or time constraints. Thus, there's a growing demand for AI-based fitness systems that can function independently using only a camera and software, delivering real-time insights to users while exercising at home or in a gym.

Computer vision models, especially those utilizing human pose estimation frameworks such as MediaPipe can detect human joint positions and provide meaningful analysis based on limb movements and joint angles. When integrated with rule-based or machine learning algorithms, these models can perform high-level tasks like posture correction, repetition counting, and fitness performance evaluation. This makes them ideal for fitness tracking systems, especially for exercises like squats, push-ups, and walk.

### *B. Problem Statement*

Despite the availability of various fitness tracking solutions, most existing systems either:

- Require physical sensors or wearables that are not accessible to everyone
- Lack real-time feedback, causing delays in correcting improper form, or
- Fail to personalize the exercise monitoring experience to the user’s specific needs.

The primary challenge is to build a real-time, non-intrusive, accurate AI-based fitness tracker that can:

- Track a user's physical movements using only a webcam or mobile camera,
- Count repetitions of exercises (like squats or push-ups) accurately,
- Detect incorrect posture or body alignment using joint angles,
- Provide instant visual or audio feedback with beep sound to help users improve their form,
- Work efficiently on low-resource devices without GPU dependency.

©IJRASET: All Rights are Reserved 

This project aims to solve these challenges by developing an AI Fitness Tracker that combines MediaPipe [10] pose estimation with joint-angle-based logic and audio feedback, eliminating the need for external sensors or high-end computing resources.

## II. LITERATURE SURVEY

The growing interest in intelligent fitness systems has led to significant advancements in real-time pose estimation and human activity recognition. These systems aim to provide automated feedback, correct form detection, and repetition counting, enhancing user experience and safety in physical exercises. Several researchers have focused on vision-based fitness tracking methods due to the non-intrusive and cost-effective nature of camera-based solutions. In [1], the authors proposed a system using OpenPose for real-time pose estimation in fitness applications. Their system effectively identified body joints and angles for exercise classification but faced challenges in outdoor environments due to lighting conditions. Another relevant study [2] utilized deep learning with convolutional neural networks (CNNs) to classify various fitness activities. While the accuracy was promising, the model required a large labeled dataset and high computational resources, making real-time deployment difficult on edge devices. MediaPipe, developed by Google, has been leveraged in multiple studies for lightweight pose detection [3]. Its performance in terms of speed and accuracy enables real-time applications on mobile and embedded systems. The integration of angle-based decision logic using landmark points has proven to be highly efficient for repetition counting and form detection, as explored in [4].

In [5], researchers introduced a yoga posture correction system using pose angles and machine learning, showing that joint angle calculation can be sufficient for identifying incorrect postures without needing full-body skeletal models. Wearable sensor-based systems, although accurate, come with limitations like cost and user discomfort. This was evident in [6], where accelerometer and gyroscope data were used to count reps but required users to wear multiple sensors on different body parts. In contrast, vision-based systems provide a more user-friendly interface. Furthermore, studies such as [7] and [8] emphasize audio-visual feedback integration to improve user motivation and compliance during workouts. These feedback mechanisms enhance engagement and provide real-time corrective actions. Also, there is a section of Human Activity Recognition [9] that uses wearables like smart fit band or watches that detects number of steps or reps taken by users. The proposed AI Fitness Tracker system builds upon the strengths of the above methods by integrating MediaPipe pose estimation, angle-based logic, and audio feedback to deliver a real-time, scalable, and sensor-free solution for fitness monitoring.

## III. METHODOLOGY

The AI Fitness Tracker system uses MediaPipe for human pose detection, integrated with custom logic for repetition counting, posture detection, and real-time feedback, wrapped in a user-friendly Streamlit application. The system is modular, consisting of four main Python files:

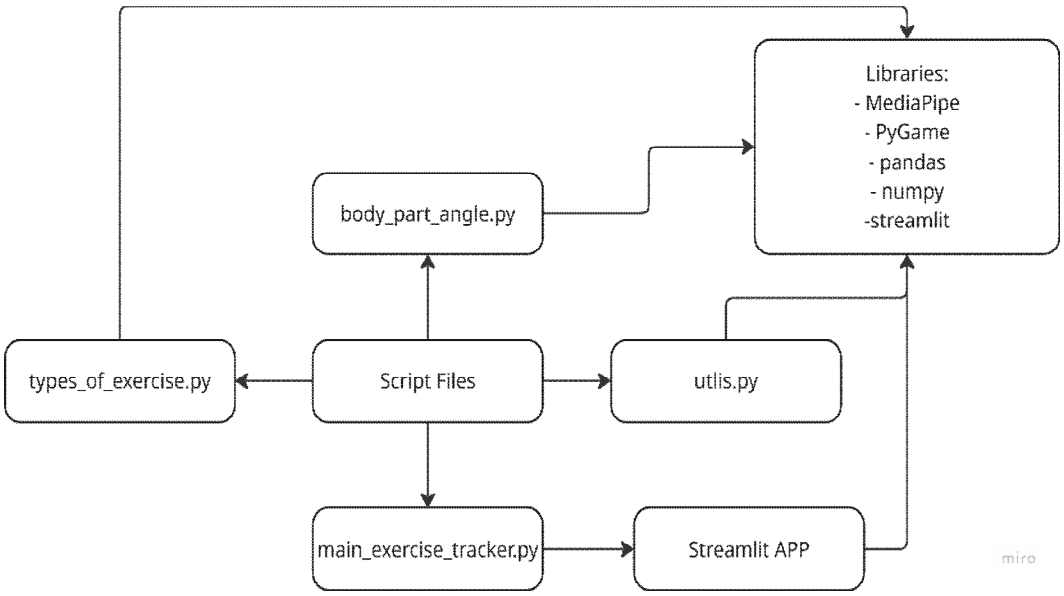



Fig 1 Scripts with Popular Libraries

Below Table describes what are the popular libraries used in the python scripts:

| S.No. | Libraries | Description |
|---|---|---|
| | MediaPipe | Google's MediaPipe enables real-time pose tracking using computer vision, ideal for fitness tracking, exercise monitoring, and posture correction. |
| | Streamlit | Streamlit is an open-source Python framework that creates interactive fitness tracking dashboards quickly with real-time data visualization and user input. |
| | PyGame | Pygame is a Python library used to create interactive fitness games or visualizations, offering real-time graphics, sound, and user input handling. |
| | Opencv-python | OpenCV-Python's cv2 module enables real-time image and video processing, essential for detecting and tracking fitness movements visually. |
| | numpy | NumPy is a fundamental Python library for efficient numerical computations, used in fitness trackers for processing sensor data and calculations. |
| | pandas | Pandas is a powerful Python library used for analysing, cleaning, and processing fitness tracking data through structured data frames and operations. |

*A. Body_part_angle.py*

*1)* Class*:* BodyPartAngle: Purpose is to calculate the angles of various body parts using landmarks.

*2) Method*: angle_of_the_left_arm(self): Calculates the angle of the left arm.
Workflow*:* Detects the positions of the left shoulder, left elbow and left wrist using the detection_body_part function. Calculates the angle between these three points using the calculate_angle function.

*3)* Method: angle_of_the_right_arm(self): Calculates the angle of the right arm.
Workflow: Detects the position of the right shoulder, right elbow and right wrist using the detection_body_part function. Calculates the angle between these three points using the calculate_angle function.

*4) Method*: angle_of_the_left_leg(self): Calculates the angles of the left leg.
Workflow*:* Detects the position of left hip, left knee, and left ankle using the detection_body_part functions. Calculates the angle between these three points using the calculate_angle function.

*5)* Method*:* angle_of_the_right_leg(self): Calculates the angles of the right leg.
Workflow*:* Detects the position of right hip, right knee and right ankle using the detection_body_part functions. Calculates the angle between these three points using the calculate_angle_function.

*6) Method*: angle_of_the_neck(self): Calculates the angle of the abdomen.
Workflow: Detects the positions of the right and left shoulders, right and left hips and right and left knees using the detection_body_part function. Calculates the average positions of the shoulders, mouth corners and hips. Calculates the angle between mouth position, average shoulder position, and average hip position using the calculate_angle function. Then after, adjust the angle to represent the neck.

*7) Method*: angle_of_the_abdomen(self): Calculates the angle of abdomen.
*Workflow*: Detects the positions of right and left shoulders, right and left hips, and right and left knees using the detection using the detection_body_part function. Calculates the average positions of the shoulder position, average hip position, and average knee using the calculate_angle function.

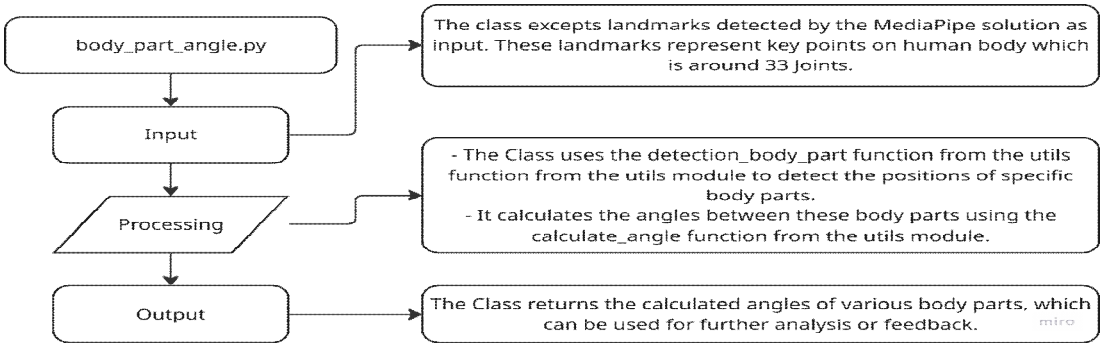



Fig 2 shows the *overall architecture of* body_part_angle.py

*B. Type_of_exercise.py*

*1)* Class: TypeOfExercise(BodyPartAngle): Extends the BodyPartAngle class to include exercise-specific analysis and feedback.

*2)* Method: push_up(self, counter, status): Analyses the push-up exercise.

Workflow*:* Calculates the angles of the left and right arms. Computes the average arm angle. Updates the counter and status based on the average arm angle. Plays a sound when the push-up is correctly performed. Then, returns the updated counter and status.

*3)* Method: display_wrong_posture(self): Displays a message indicating wrong posture. Prints a message indicating wrong posture i.e., status = False.

*4)* Method: pull_up(self, counter, status): Analyses the pull-up exercise.

*Workflow*: Detects the positions of the nose and elbows. Computes the average y-coordinate of the elbows. Updates the counter and status based on the position of the nose relative to the elbows. Plays a sound when the pull-up is correctly performed. Then, returns the updated counter and status.

*5)* Method: squat(self, counter, status): Analyses the squat exercise.

Workflow: Calculates the angles of the left and right legs. Computes the average leg angle. Updates the counter and status based on the average leg angle. Plays a sound when the squat is correctly performed. Then, returns the updated counter and status.

*6)* Method: walk(self, counter, status): Analyses the walking exercise.

Workflow: Detects the positions of the left and right knees. Updates the counter and status based on the relative positions of the Knees. Plays a sound when a step is correctly performed. Then, returns the updated counter and status.

*7)* Method: sit_up(self, counter, status): Analyses the sit-up exercise.

Workflow: Calculates the angle of the abdomen. Updates the counter and status based on the abdomen angle. Plays a sound when the sit-up is correctly performed. Returns the updated counter and status.

*8)* *Method*: calculate_exercise(self, exercise_type, counter, status): Determines which exercise to analyse based on the exercise_type parameter.

*Workflow*: Calls the appropriate exercise analysis method (push_up, pull_up, squat, walk, sit_up) based on the exercise_type. Updates the counter and status accordingly.

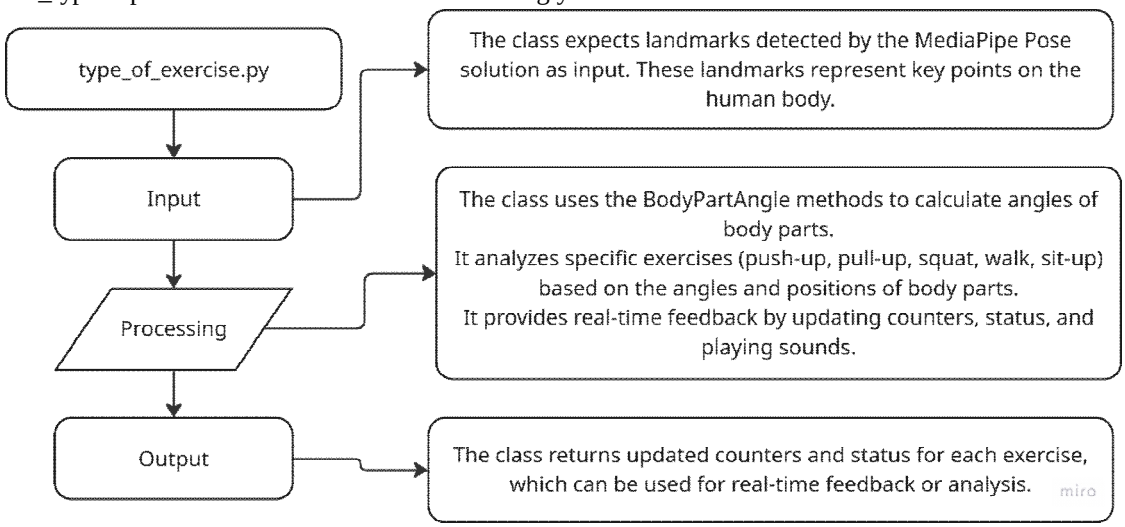



Fig 3 shows the overall architecture of type_of_exercise.py

*C. Utils.py*

*1)* Pose Estimation Initialization: Initializes the MediaPipe Pose solution, which is used for detecting human poses in images or videos.

*2)* Method: calculate_angle(a,b,c): Calculates the angle between three points (a, b, c) in a 2D plane.

Workflow: Converts the input points a, b, and c into NumPy arrays. Calculates the angle in radians between the lines ab and bc using the np.arctan2 function. Converts the angle from radians to degrees. Ensures the angle is within the range [0,180] degrees. Then, returns the calculated angle in degrees.

*3)* Method: detection_body_parts(landmarks): Detects and organizes the coordinates of all the body part data.
Workflow: Initializes an empty Pandas DataFrame to store the body part data. Iterates over all the landmarks in mp_pose.PoseLandmark. For each landmark, retrieves the body part name and its coordinates using the detection_body_part function. Stores the body part name and coordinates in the DataFrame. Then, returns a DataFrame containing the body part names and their corresponding coordinates.

*4)* Method: score_table(exercise, frame, counter, status): Overlays exercise-related information onto a video frame.
Workflow: Uses OpenCV's cv2.putText function to draw text on the frame. Displays the exercise name, counter, and status at specified positions on the frame.

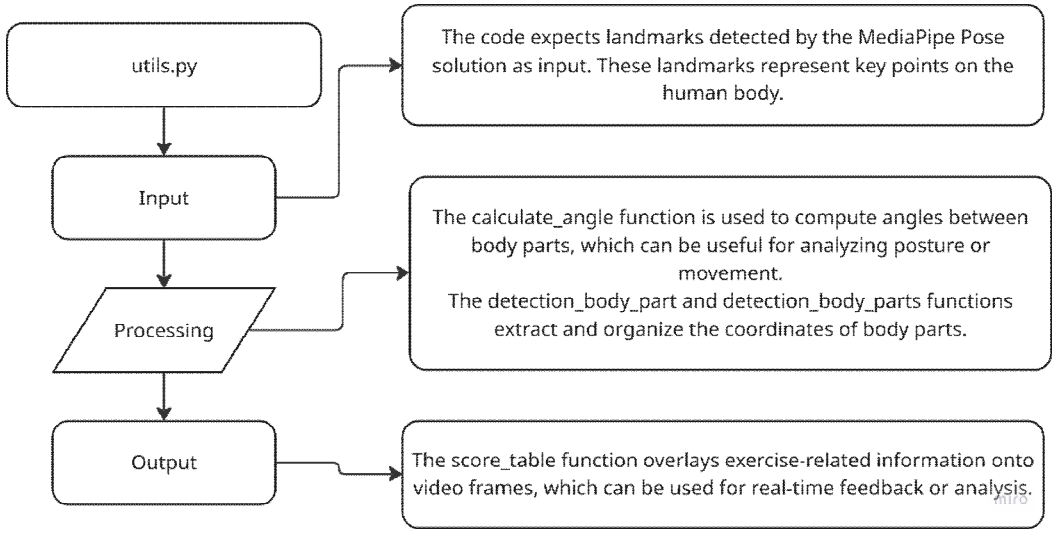



Fig 4 shows the overall architecture of Utils.py:

*D. Main_exercise_tracker.py*

The main script for an exercise tracker application. It utilizes Streamlit for the user interface, OpenCV for video processing, and MediaPipe for pose detection. The application allows users to select an exercise type, choose a video source (webcam or uploaded video), and analyze the exercise in real-time. Below is a detailed explanation of the workflow and architecture of script:

Function: process_frame(frame, exercise_type, counter, status): Processes each video frame to detect poses, calculate body part angles, and analyze the exercise. Workflow steps:

- Initializes the MediaPipe Pose model.
- Converts the frame to RGB for pose detection.
- Processes the frame to detect landmarks.
- Converts the frame back to BGR for rendering.
- Extracts landmarks from the detection results.
- Calculates body part angles using the BodyPartAngle class.
- Analyzes the exercise using the TypeOfExercise class.
- Draws landmarks on the frame.
- Overlays the counter and status on the frame. Then, Returns the processed frame, updated counter, and status.

Function: main_loop(): Main loop for video processing. Workflows:

- Initializes the video capture based on the selected video source.
- Creates a Streamlit image placeholder for displaying the video frames.
- Processes each frame using the process_frame function.
- Displays the processed frame in the Streamlit UI.
- Releases the video capture when done. Then, main_loop function is called when the script is executed

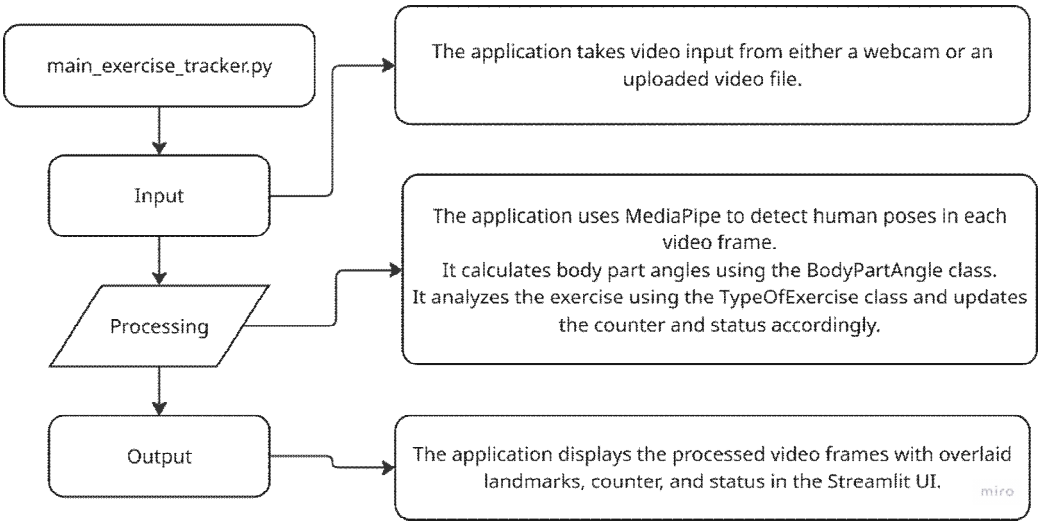



Fig 5 shows the overall architecture of main_exercise_tracker.py:

## IV. RESULTS

The performance of the AI-based Real-Time Exercise Counter system was evaluated using both uploaded video processing and real-time webcam analysis modes. The objective was to determine the accuracy of repetition counting for five key exercises: Walking, Squats, Situps, Pushups, and Pullups. The evaluation was conducted across multiple trials for each activity, with comparisons made between manually counted ground truth values and the system's automated counts.

### *A. Accuracy for Uploaded Video Processing*

The following table summarizes the system's performance on pre-recorded or uploaded exercise videos:

| Exercise | Expected Count or reps | Detected Count or reps | Accuracy (%) |
|---|---|---|---|
| Walking | Ground Truth (21) | System Detected (21) | 100% |
| Squats | Ground Truth (10) | System Detected (9) | 90% |
| Situps | Ground Truth (15) | System Detected (12) | 80% |
| Pushups | Ground Truth (13) | System Detected (13) | 100% |
| Pullups | Ground Truth (10) | System Detected (9) | 90% |
| Average | - | - | 92% |

### *B. Accuracy for Real-Time Video Analysis (Webcam Feed)*

Real-time analysis introduced a minor latency (~3 seconds) due to frame buffering and pose estimation delay. Despite this, the system achieved high performance:

| Exercise | Expected Count or reps | Detected Count or reps | Accuracy (%) |
|---|---|---|---|
| Walking | Ground Truth (21) | System Detected (23) | 90.48% |
| Squats | Ground Truth (10) | System Detected (12) | 80% |
| Situps | Ground Truth (15) | System Detected (14) | 93.38% |
| Pushups | Ground Truth (13) | System Detected (13) | 100% |
| Pullups | Ground Truth (10) | System Detected (8) | 80% |
| Average | - | - | 88.77% |

Note: The accuracy is calculated using two formula:

- Accuracy % (when expected count >= detected count) = (Expected Count/Detected Count) * 100
- Accuracy % (when expected count < detected count), typically, represents the Overcount and false positive case in technical terms = [1- (|Detected-Expected|/ Expected)] * 100

### *C. Observations*

- Pose detection delay in real-time (3 seconds) marginally affected the timing of counter updates.
- Walking detection had the highest consistency due to continuous motion tracking.
- Pullups showed slightly lower accuracy due to pose estimation challenges in vertical limb alignment.
- The uploaded video mode demonstrated higher stability as the model processed frames with fewer real-time constraints.

### *D. Attached Results*

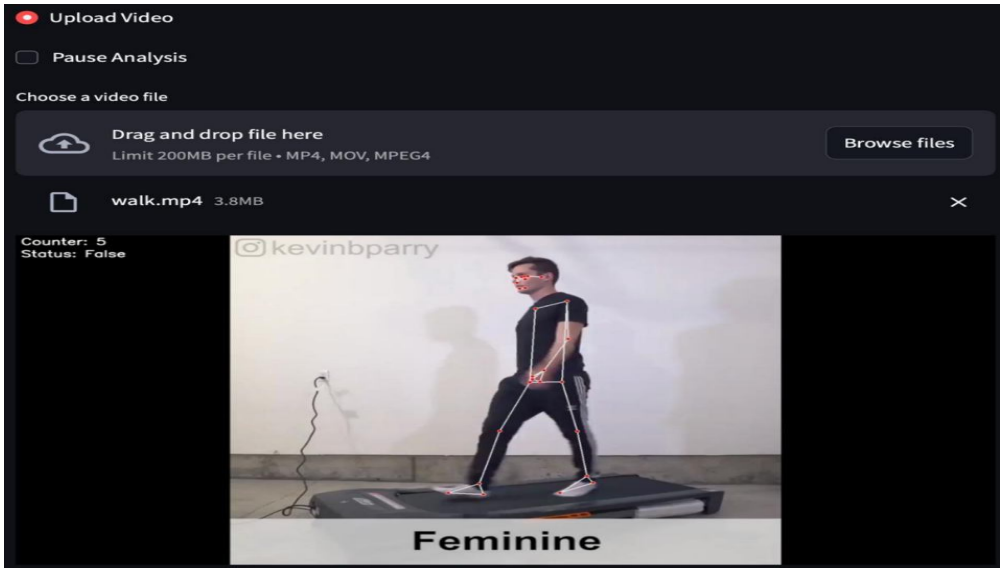
Images showcasing frame-by-frame counter outputs, body keypoint overlays, and visual interface snapshots from the application for each exercise mode are included in this:

*1)* Below image showcase the dashboard of the AI Fitness Tracker using Streamlit Application:

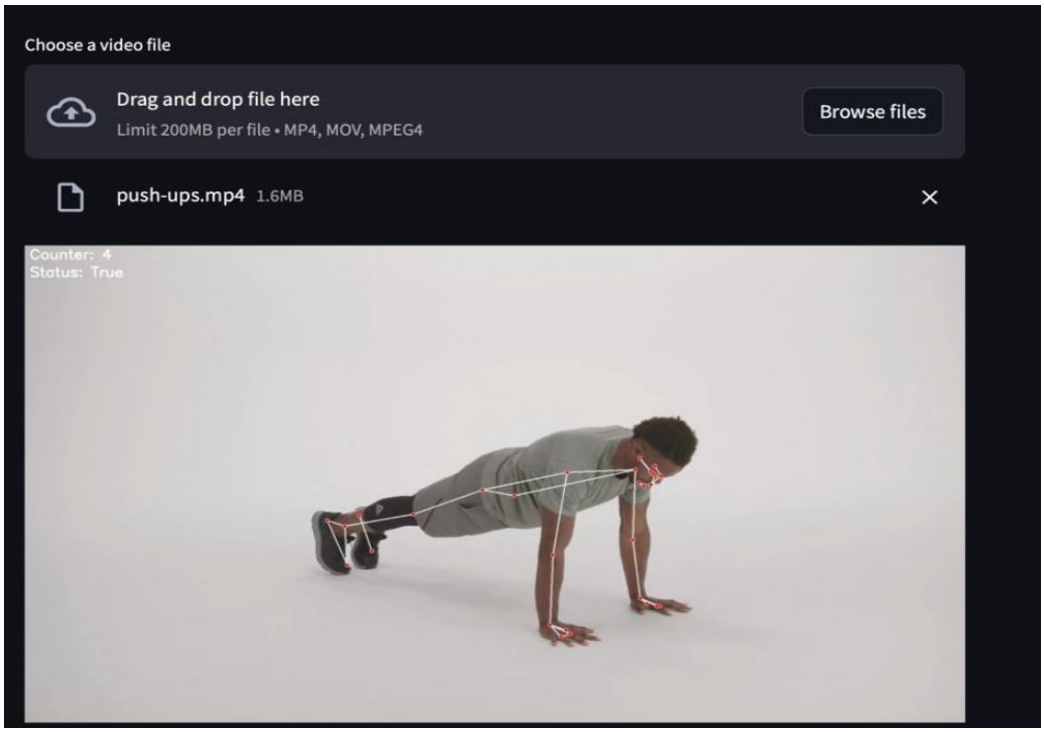



As you can see there are firstly the two options to upload video or process in real time video analyses with selection of types of exercise from 5 options (default is push-ups, walk, pull-ups, sit-ups, squats) and there is a check button in front of the Pause Analyses when its checked then only it starts the video processing and at last you will find out the video upload (drag and drop also upload video manually).

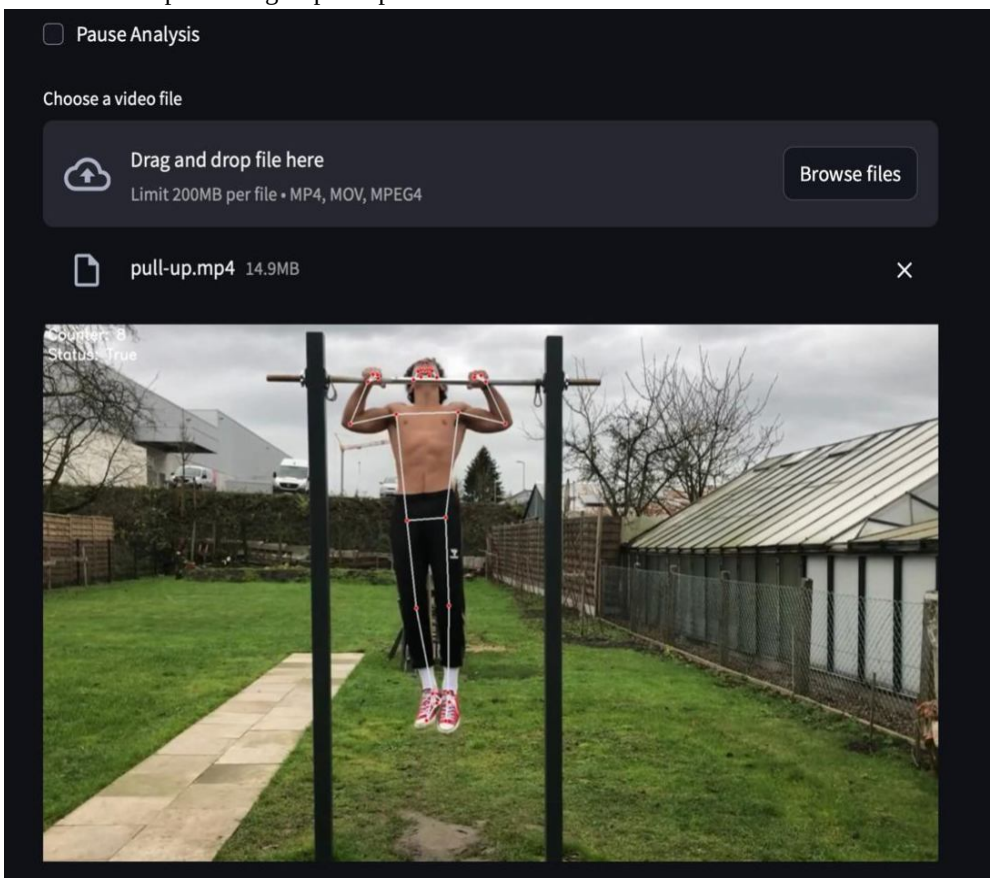
*2)* Below Image shows the video uploaded video processing of Walking:



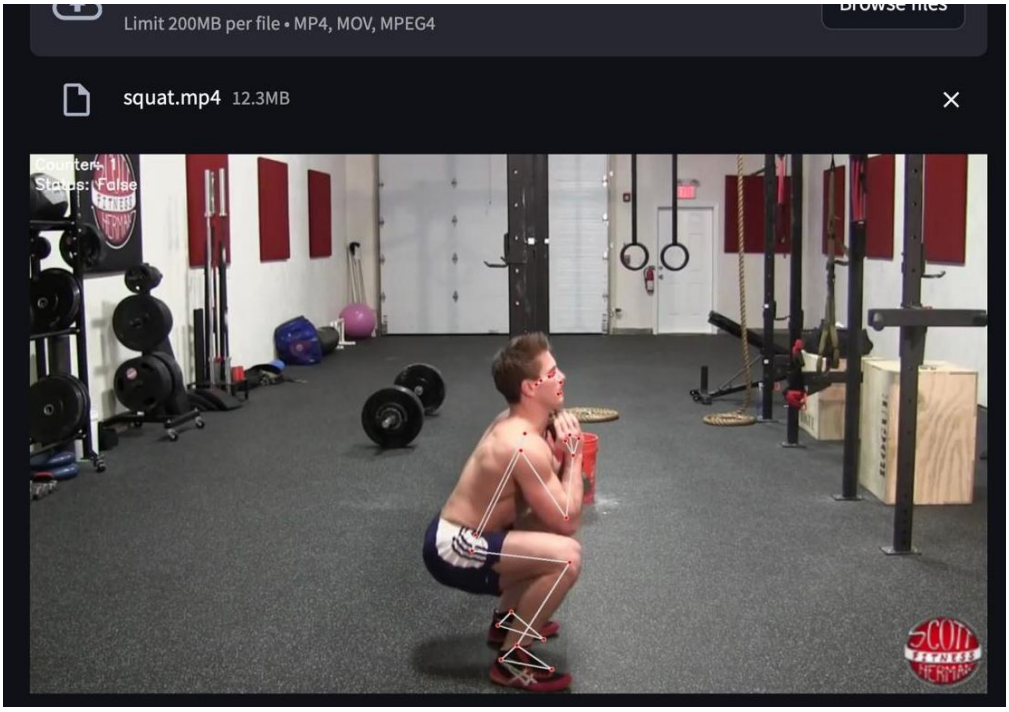
*3)* Below Image shows the video processing of push-ups:



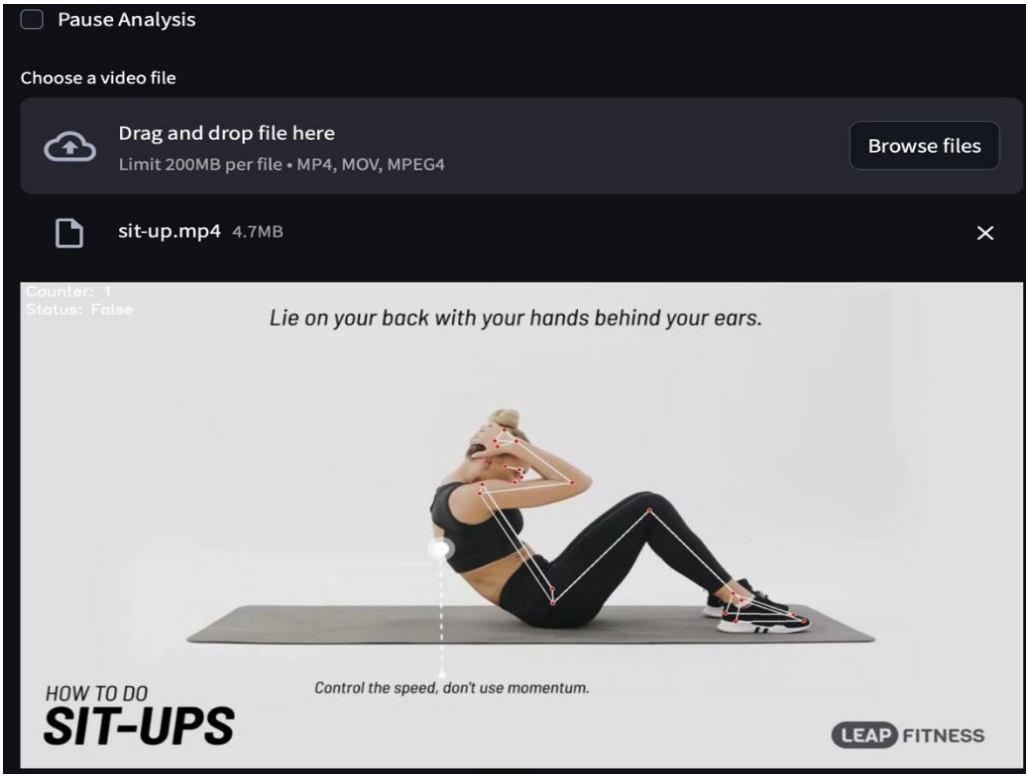

*4)* Below Image shows the video processing of pull-ups



*5)* Below Image represents the video processing of squats:




*6)* Below Image represents the video processing of sit-ups:




As you can see in all the video processing for 5 exercise type, at top left corner there are two elements Counter and Status. Counter counts the number of repetitions or steps performed by user in videos every time when status = True it adds +1 and every time when status = True then it makes beep sound to ensure the counter is add repetitions.

## V. INSIGHTS AND LIMITATIONS

### A. Key Insights

1) *High Counter Accuracy Achieved:* The system demonstrated strong performance across all exercise types in both uploaded and real-time video modes. Particularly, walking and pushups showed the highest counter accuracy due to clear body motion patterns easily captured by pose estimation models.
2) *Real-Time vs Uploaded Video Performance:* While uploaded video processing yielded an average accuracy of 92%, real-time video analysis showed a slightly reduced performance with 88.77% accuracy, primarily due to a 3-second processing delay introduced by frame buffering and pose estimation computations.
3) *Robustness of MediaPipe Pose Estimation:* The integration of MediaPipe's holistic body landmark detection significantly contributed to consistent pose detection across various lighting conditions and angles. However, exercises with vertical motions like pullups presented slightly reduced accuracy due to occasional loss of upper limb landmarks.
4) Scalability and Modularity: The architecture is modular, allowing easy integration of additional exercises and enhanced logic for repetition detection. With minimal modification, the system can be extended to support more complex movement patterns or rehab exercises.
5) Streamlit Integration for Accessibility: The use of Streamlit for frontend deployment allowed for a user-friendly interface, enabling real-time interaction with minimal latency. The web-based deployment potential makes this application accessible on a wide range of devices.

## VI. CONCLUSION

This project successfully implements a real-time AI-powered fitness tracking system capable of accurately counting exercise repetitions through video analysis. Using MediaPipe for landmark detection and custom repetition logic for each exercise, the system provides reliable feedback to users in both offline and real-time scenarios. With an average accuracy of over 90%, the system proves effective for practical use cases such as home workouts, physiotherapy monitoring, and gym supervision. Minor latency issues in real-time processing suggest opportunities for optimization, especially in reducing buffering time and improving model inference speed.

In conclusion, this system lays the groundwork for future enhancements such as:

1) Integration with mobile devices.
2) Personalized fitness feedback using ML models.
3) Support for complex or dynamic exercise routines.
4) Real-time audio/visual feedback mechanisms.

This AI-based exercise tracker represents a promising step toward automated fitness monitoring, encouraging consistent workout habits while reducing the dependency on human supervision.

## VII. FUTURE SCOPE

As this AI-based Exercise Tracker proves effective in recognizing and counting repetitions across multiple exercises, several enhancements and extensions can be pursued to further enrich the system's capabilities, usability, and performance.

### A. Processed Video History and User Tracking

- Exercise History Storage: Implement functionality to save processed videos with timestamps and exercise counts in a structured database or cloud storage.
- User-Specific Logs: Maintain per-user profiles, storing completed workout sessions, total repetitions per exercise, and performance over time.
- Progress Visualization: Generate graphical insights showing trends (e.g., squats over weeks) to motivate consistency and self-assessment.

### B. Model Context Protocol and Efficiency

- Session-Aware Context Protocol: Enable the system to adapt repetition logic based on real-time feedback from prior repetitions in a session. For example, dynamically adjusting thresholds as fatigue sets in.
- Lightweight Models for Edge Devices: Optimize the pose estimation pipeline (e.g., using TensorFlow Lite or ONNX models) to reduce latency and support deployment on mobile phones, Raspberry Pi, or IoT-enabled fitness mirrors.
- Multi-Threaded or GPU Accelerated Processing: Use concurrent video frame pipelines or CUDA acceleration to reduce delay in real-time inference.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

### *C. Enhanced Feedback and Gamification*

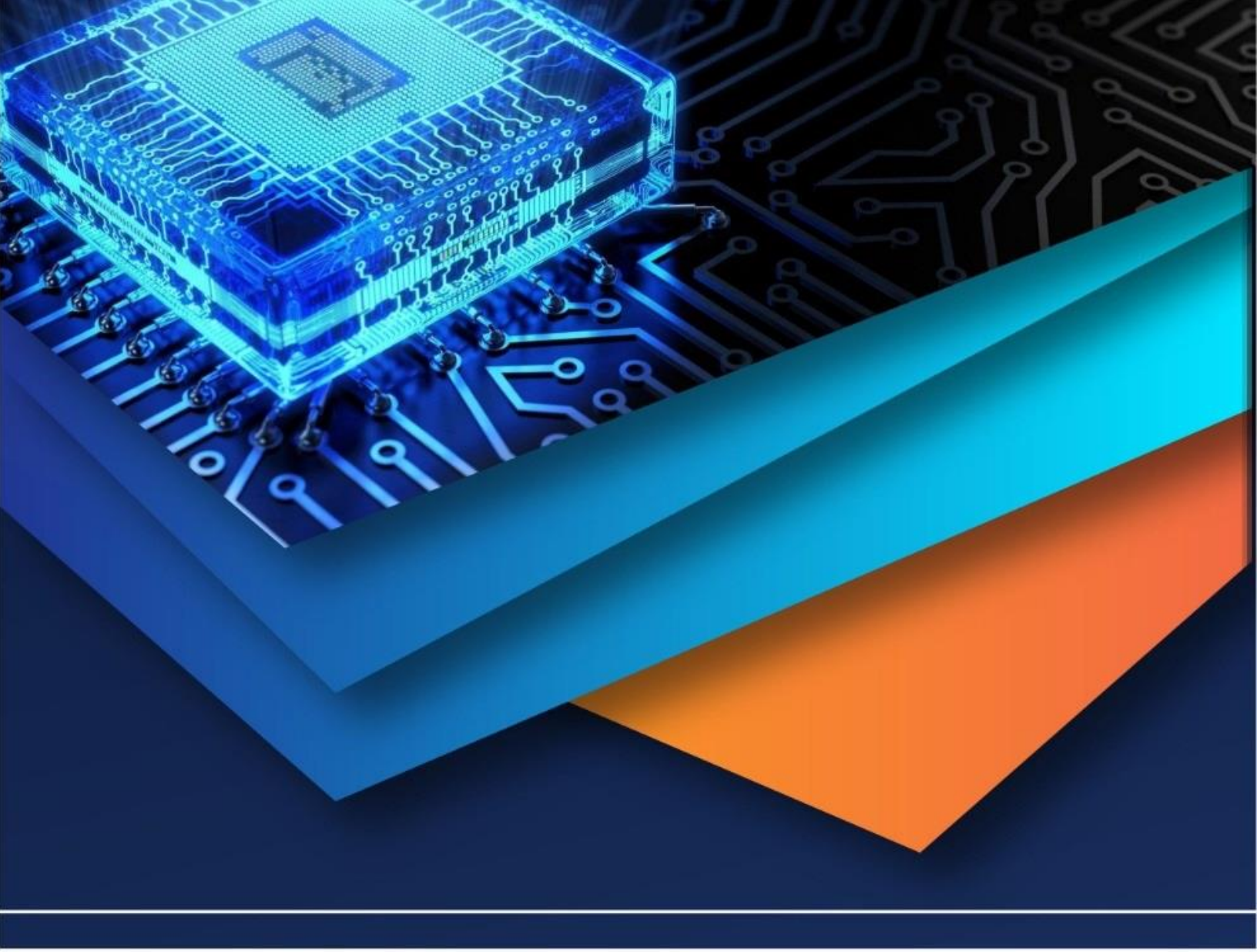
- Audio/Visual Feedback Integration: Provide users with real-time auditory beeps or visual tick marks as confirmation for each successful rep.
- Gamification Layer: Add badges, levels, streak counters, and leaderboards to increase engagement, especially for fitness apps targeting beginners or children.

### *D. Advanced Exercise Support*

- Dynamic Exercise Detection: Integrate a classification layer to automatically detect the type of exercise being performed, removing the need for manual selection.
- Posture Correction: Use pose deviation angles to offer corrective feedback on form, helping prevent injuries and improve results.
- Support for Yoga and Stretching: Extend the system to detect and score static poses (e.g., yoga asanas) using joint angle thresholds.

## REFERENCES

[1] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 172–186, Jan. 2021.
[2] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Gaussian Process Dynamical Models for Human Motion,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 283–298, Feb. 2008.
[3] F. Zhang, X. Zhu, and H. Ye, “MediaPipe Hands: On-Device Real-Time Hand Tracking,” arXiv preprint arXiv:2006.10214, 2020.
[4] R. Pandey and S. Shukla, “Human Pose Estimation for Fitness Tracking Using Joint Angles and Deep Learning,” in Proc. of IEEE Int. Conf. on Smart Technologies and Management, 2021, pp. 225–230.
[5] P. N. Singh and A. Ghosh, “Yoga Posture Correction Using Machine Learning and Pose Estimation,” in Proc. of 2021 International Conference on Computer Vision and Image Processing (CVIP), pp. 1–7.
[6] A. Parate, M. Ganesan, M. Marlin, and J. A. Landay, “Detecting Repetitive Activities Using Accelerometers,” IEEE Pervasive Computing, vol. 12, no. 2, pp. 32–39, Apr. 2013.
[7] Y. Kim and H. Choi, “A Real-Time Exercise Repetition Counting System Using CNN-LSTM,” in 2020 IEEE Int. Conf. on Consumer Electronics (ICCE), Las Vegas, NV, USA, pp. 1–2.
[8] T. Lee and K. Lee, “Smart Trainer: A Real-Time Repetition Counting and Feedback System Using Pose Estimation,” in 2020 IEEE Global Conference on Consumer Electronics (GCCE), pp. 22–25.
[9] Y. Mishra, A. Jaiswal, and G. Soni, "A comparative study on human activity recognition using smartphone dataset through machine learning approach," Int. Res. J. Eng. Technol. (IRJET), vol. 11, no. 7, pp. 813–818, Jul. 2024. [Online]. Available: https://www.researchgate.net/publication/390667823
[10] Google MediaPipe, Available: https://ai.google.dev/edge/mediapipe/solutions/guide



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)