MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

A

REAL-TIME/FIELD BASED RESEARCH PROJECT

On

Hyperfit AI: your smart, adaptive and always evolving personal trainer

Submitted in partial fulfillment of the
Requirements for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

By

G.PURIJAGANNATH	**-**	**23R21A6688**
J.SNEHA	**-**	**23R21A6691**
P.MOHANA	**-**	**23R21A66B2**
K.VAISHNAVI	**-**	**23R21A6697**

Under the guidance of

Mr P.Babu
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

2023-2027

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY (UGC AUTONOMOUS) Affiliated to JNTUH,Approved by AICTE Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING-

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

This is to certify that the project entitled **"Hyperfit AI: your smart, adaptive and always evolving personal trainer** "has been submitted by **G.PURIJAGANNATH(23R21A6688), J.SNEHA(23R21A6691), P.MOHANA (23R21A66B2), K.VAISHNAVI(23R21A6697)** in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering-Artificial Intelligence & Machine Learning from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Internal Guide **Project-Coordinator**

Head of the Department

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS

MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

NATIONAL BOARD of ACCREDITATION

NAAC A

COMPUTER SCIENCE AND ENGINEERING-

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DECLARATION

We hereby declare that the project entitled “Hyperfit AI: your smart, adaptive and always evolving personal trainer ” is the work done during the period from **Feb 2025 to July 2025** and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in **Computer Science and Engineering- Artificial Intelligence & Machine Learning** from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

G.PURIJAGANNATH	**-**	**23R21A6688**
J.SNEHA	**-**	**23R21A6691**
P.MOHANA	**-**	**23R21A66B2**
K.VAISHNAVI	**-**	**23R21A6697**

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING-

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

ACKNOWLEDGEMENT

The satisfaction and euphoria that acconamy the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we now have the opportunity to express our guidance for all of them. First of all, we would like to express our deep gratitude towards our internal guide**, Mr. P.Babu , Assistant Professor, Computer Science and Engineering-Artificial Intelligence & Machine Learning** for support in the completion of our dissertation. We wish to express our sincere thanks to**Dr. K. Sai Prasad, HOD, Department of Computer Science and Engineering- Artificial Intelligence & Machine Learning** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

G.PURIJAGANNATH	**-**	**23R21A6688**
J.SNEHA	**-**	**23R21A6691**
P.MOHANA	**-**	**23R21A66B2**
K.VAISHNAVI	**-**	**23R21A6697**

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS

MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

NATIONAL BOARD of ACCREDITATION

NAAC A

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING-

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

ABSTRACT

This project investigates the potential of Hyperfit AI to empower users with readily available. A Personal AI Gym Trainer leverages artificial intelligence, computer vision, and machine learning to provide real-time, personalized fitness coaching. This system analyzes user movements using pose estimation techniques, ensuring proper exercise form and reducing injury risks. It integrates with wearable devices and IoT sensors to track vital metrics such as heart rate, calories burned, and workout intensity. Natural Language Processing (NLP) enables voice interaction, while AI-powered recommendations customize workout plans based on user progress and goals. Cloud computing ensures seamless data storage and accessibility, allowing users to monitor their fitness journey through mobile and web applications.

LIST OF FIGURES & TABLES

INDEX

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

HyperFit AI is an intelligent pose detection and correction system designed to assist users in performing physical exercises with optimal posture and technique. Leveraging advanced computer vision technologies such as **OpenCV** and **Mediapipe**, the system analyzes human body movements in real time and identifies deviations from the correct form.By accurately detecting key body landmarks and comparing them with predefined ideal poses, HyperFit AI provides instant feedback and corrective suggestions. This makes it an ideal tool for fitness enthusiasts, personal trainers, and rehabilitation professionals who aim to reduce the risk of injuries and enhance workout efficiency. The project bridges the gap between technology and fitness by offering a smart, interactive, and user-friendly platform for posture improvement and body awareness.

1.2 PURPOSE OF THE PROJECT

The purpose of implementing HyperFit AI: Pose Detection and Correction System encompasses various objectives and potential benefits . Below are the primary goals:
Real-Time Feedback: Users receive instant guidance on their exercise form through real-time pose correction and feedback.
Injury Prevention**:** By ensuring proper body alignment, the system helps reduce the risk of workout related injuries.
Performance Enhancement: Correcting posture allows users to maximize the effectiveness of their workouts and achieve better results.

1.3 MOTIVATION

The growing popularity of home workouts and online fitness programs has made fitness more accessible, but it has also introduced a significant challenge—lack of proper supervision. Incorrect posture during exercise is a leading cause of injuries and reduced performance, especially for beginners. While personal trainers can address this issue, they are not always available or affordable for everyone.

CHAPTER2

LITERATURE SURVEY

We conducted a thorough literature survey by reviewing existing systems for hyperfit AI. Research papers, journals and publications have also been referred in order to prepare this survey

2.1 EXISTING SYSTEM

Fitness Tracking Apps: Popular fitness apps like Google Fit, Apple Fitness+, and MyFitnessPal offer activity tracking, heart rate monitoring, and step counting. However, they lack the ability to assess or correct exercise form and posture in real time.

AI Pose Estimation Tools : Technologies like OpenPose, BlazePose (from Mediapipe), and PoseNet provide the foundation for real-time human pose estimation using camera input. These tools detect key landmarks on the human body but require additional development to provide actionable feedback for posture correction.

Virtual Personal Trainer Apps: Some mobile apps and platforms (e.g., Freeletics, Fitify, or Smart Mirror systems) offer guided workouts with virtual trainers. While they provide visual cues, they often lack true real-time form correction or rely on predefined animations rather than dynamic AI-driven analysis.

Physiotherapy & Rehab Systems:AI-assisted rehabilitation systems are used in medical environments to monitor patient movements. These systems focus more on recovery than fitness and are typically expensive and require specialized hardware.

Home Workout Video Guides (YouTube, etc.): Many users follow workout tutorials online without any real-time feedback. These videos are static and cannot adapt or respond to the user's form, often leading to incorrect posture and potential injury.

2.2 LIMITATIONS OF EXISTING SYSTEM

Most existing fitness apps lack real-time pose correction, offering only general workout guidance.

AI pose estimation tools often require technical expertise to interpret results without automated feedback. Virtual trainers and rehab systems are either too generic or costly, limiting accessibility for everyday users.
Home workout videos provide no personalized correction, increasing the risk of injury from improper form.

Lack of Deep Personalization: Most apps provide generic plans that do not account for individual body types, fitness levels, or unique goals.

Inaccurate Data Tracking: Reliance on self-input or basic sensors can lead to misleading metrics, especially for calories burned, heart rate, or workout intensity.

Limited Adaptability: Apps often fail to adjust routines dynamically based on performance, fatigue, or missed sessions.

Lack of Real-Time Feedback: Apps usually cannot give immediate form corrections or respond to live user input like a personal trainer.

Inflexible Scheduling and Time Management: Apps don’t adequately support irregular schedules or allow flexible workout durations.

CHAPTER 3

PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

The proposed system leverages computer vision and AI to detect human body posture during workouts in real time.Using Mediapipe and OpenCV, it tracks key body landmarks to evaluate the correctness of the user's form.If the posture deviates from the ideal, the system provides immediate visual or audio feedback for correction.It is designed to work with standard webcams or phone cameras, ensuring easy accessibility and low cost.The system helps prevent workout-related injuries and improves the efficiency of exercise routines.By offering real-time, intelligent feedback, it acts as a virtual personal trainer for fitness enthusiasts.

3.2 OBJECTIVES OF PROPOSED SYSTEM

The objectives of the proposed system include the following:

Personalized Workout Plans**:** AI analyzes fitness goals, body metrics, and performance data. Generates tailored exercise routines for optimal results.

Real-Time Performance Tracking: Wearable devices with AI monitor heart rate, calories burned . Provides instant feedback and adjustments during workouts.

Nutrition Guidance: Creates customized meal plans based on dietary needs & fitness goals.Tracks macro nutrient intake and suggests adjustments.

Injury Prevention: Analyzes movement patterns to detect improper form or risk of injury.Suggests corrective exercises and safer alternatives.Leverages advanced techniques like hyper parameter tuning, transfer learning, and model pruning.

3.3 SYSTEM REQUIREMENTS

Here are the requirements for developing and deploying the application.

3.3.1 SOFTWARE REQUIREMENTS

- Programming Language: Python 3.8 or later
- Libraries/Frameworks: OpenCV, Mediapipe, NumPy
- IDE: Visual Studio Code / PyCharm
- Pose Estimation Library**:** Mediapipe (for real-time pose detection)
- Visualization: OpenCV (for video capture and feedback overlay)
- Database (Optional): SQLite or Firebase for storing workout data
- Version Control: Git for source code management
- Environment Management: Anaconda (optional, for managing dependencies)
- Testing Tools: Webcam or external camera for real-time input
- Optional UI Framework: Tkinter / Flask (if a GUI or web interface is implemented)

3.3.2 HARDWARE REQUIREMENTS

- **Processor:** Intel i5 or AMD Ryzen 5 and above for smooth performance
- **RAM:** 8 GB minimum (16 GB recommended for real-time processing and multitasking)
- **Graphics Card:** Integrated GPU is sufficient; dedicated GPU (e.g., NVIDIA GTX 1050 or higher) recommended for enhanced performance
- **Camera:** HD Webcam or external camera for real-time pose detection
- **Operating System:** Windows 10/11, macOS, or Linux
- **Display:** 1080p resolution or higher for better visual feedback
- **Optional Devices:** Android/iOS smartphones for mobile version (if extended to mobile)

3.3.3 FUNCTIONAL REQUIREMENTS

The system should capture real-time video input using a webcam or mobile camera.It must detect and track human body keypoints using Mediapipe's pose estimation model.The application should compare detected poses with predefined correct postures.It should provide real-time feedback (visual/audio) to guide users in correcting their form. Optionally, the system can log performance data for tracking user progress over time.

Pose Detection:

- The application must accurately detect human body poses in real time using the device's camera
- Integrate a pose estimation library such as **Mediapipe** to track key body landmarks (e.g., shoulders, elbows, knees).

Pose Evaluation and Correction:

- The system should compare the user's current pose against a predefined correct form for specific exercises.
- It must identify incorrect angles or misalignments and trigger appropriate corrective feedback.

Feedback Mechanism:

- Visual feedback (highlighted joints/lines) should be displayed on-screen to indicate incorrect posture.
- Optional audio feedback (e.g., "Straighten your back") can guide users during workouts in real time.
- Consider haptic feedback for wearable integration (if extended to smart devices).

Exercise Mode Selection:

- Users should be able to choose specific exercises (e.g., squats, push-ups, curls) for pose tracking.
- Each mode should load its corresponding ideal posture data and evaluation logic.

Performance Tracking:

- The system should log user performance (e.g., reps, sets, posture score) during each session.
- Optionally, this data can be visualized over time to monitor improvement.

Offline Functionality:

Core pose detection and feedback should work offline using local processing, without requiring an internet connection.

User Interface and Accessibility:

- UI must be clean and intuitive, with clearly labeled controls and real-time visual overlays.
- Include text-to-speech support for visually impaired users to receive audio cues and feedback.
- Use contrasting colors and large fonts to improve visibility and accessibility

Data Privacy and Security:

- Ensure that any user data (e.g., posture logs or session history) is stored securely.
- Follow privacy regulations and include options for users to delete or manage their data.

Accuracy and Responsiveness:

- The system must provide high accuracy in pose detection and minimal latency in feedback delivery.
- Ensure frame processing is optimized for smooth performance even on mid-range devices.

3.3.4 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of the HyperFit AI system focus on ensuring a high- quality user experience beyond core functionality. The system must deliver real-time performance with minimal lag, offering accurate pose detection and instant feedback. It should be user-friendly and accessible, allowing individuals of all fitness levels, including those with impairments, to engage effectively. The application must also be reliable and portable, operating across different platforms and environments. Additionally, it should ensure data security and maintainability, with a clean code structure that supports future updates. Lastly, efficient resource usage is crucial to maintain smooth operation.

Performance:

- The system must process and analyze video frames with latency below 200 milliseconds.
- Feedback (visual/audio) should be provided instantly without noticeable delay.
- Pose detection should function smoothly at a minimum of 15–30 FPS depending on device capability.

Reliability:

- The system should perform consistently under various lighting conditions.
- It should support a variety of body types, postures, and backgrounds.
- It must not crash or freeze during extended use.

Device Compatibility:

- The HyperFit AI system is designed to be compatible with a wide range of devices to ensure accessibility and usability across platforms. The system supports:
- Desktop and Laptop Computers:Compatible with Windows 10/11, macOS, and Linux systems. Requires a webcam (integrated or external) for real-time pose detection and at least an Intel i5/Ryzen 5 processor with 8 GB RAM.
- Mobile Devices (Optional Extension):Designed to be extendable for Android and iOS platforms. Compatible with smartphones and tablets that support ARCore (Android) or ARKit (iOS) for enhanced camera tracking. Requires camera access and moderate processing power.

Scalability:

- Architecture should support easy integration of additional exercises or modules.
- System should allow deployment on both desktops and mobile platforms.
- Capable of integrating cloud-based services for advanced analytics if required.

Security:

- User workout data must be encrypted during storage and transfer.
- Authentication should be used if user profiles or cloud sync is implemented.
- Compliance with relevant data protection policies (e.g., GDPR) must be ensured.

3.4 CONCEPTS USED IN THE PROPOSED SYSTEM

The proposed **HyperFit AI** system integrates multiple cutting-edge technologies to provide real-time pose correction and feedback for fitness users. Key concepts used in the system include:

Pose Estimation: The core of the system relies on pose estimation technology using **Mediapipe**, which detects key body landmarks from live video. This enables accurate tracking of user posture during exercise.

Computer Vision: Utilizing **OpenCV**, the system processes video frames from the camera, extracts relevant features, and overlays visual feedback on-screen to highlight correct or incorrect posture.

Machine Learning (ML):ML concepts are applied to evaluate the user's pose by comparing detected keypoints with predefined ideal poses for different exercises. This ensures adaptive and intelligent feedback.

Real-Time Feedback Mechanism: The system provides immediate corrective guidance using visual overlays (e.g., highlighting incorrect joints or angles) and optional audio prompts to help users adjust their posture instantly.

User Interface (UI) and User Experience (UX): An intuitive and responsive interface ensures that users can easily interact with the system. Clear buttons, visual indicators, and optional voice support enhance usability during workouts.

Data Logging (Optional): The system can store session data such as repetition counts, posture scores, and user performance locally or in the cloud for future review and progress tracking.

APIs (Optional): If extended with cloud features, APIs can be used to sync workout history, retrieve pose datasets, or integrate with third-party fitness platforms.

3.5 DATA SET USED IN THE PROPOSED SYSTEM

The dataset used in the HyperFit AI system is a crucial component that drives the accuracy and effectiveness of pose detection and correction. The data serves as a reference for detecting human body postures and evaluating correctness during physical exercises. Here's an overview of the key elements in the dataset used for HyperFit AI:

Curated Dataset:

You can create your own dataset by collecting high-resolution videos or images of individuals performing various exercises such as squats, push-ups, bicep curls, etc. This approach gives you complete control over the data quality, environment, camera angle, lighting conditions, and participant diversity. However, it requires significant manual effort for data collection and annotation, including labeling joint positions and categorizing correct vs. incorrect postures.

Commercial Datasets:

Companies that specialize in pose estimation, fitness tracking, or computer vision may offer commercial datasets containing labeled human body keypoints during various exercises. These datasets often include high-resolution videos or images captured from multiple angles and annotated with detailed joint positions and motion dynamics. Such datasets are particularly useful for:

- Training advanced models to detect exercise-specific poses (e.g., squats, lunges, curls)
- Enhancing accuracy in posture classification and rep detection
- Ensuring the system is generalizable across different body types and environments

Open-source Datasets:

There are several open-source pose estimation datasets available that include annotated keypoints and exercise motions. While not all are fitness-specific, many can be adapted to train models for your project. These datasets help reduce development time and offer a good starting point for testing pose detection algorithms. Some reliable sources include:

Third-party APIs:

Several third-party APIs offer pose estimation and fitness tracking capabilities that can be integrated into the HyperFit AI system. These APIs can provide access to pre-trained models, real-time keypoint detection, and even analytics related to human posture and movement. Here are some additional tips for choosing your data sets:

Data Quality:

Ensure that exercise videos/images are high-resolution and clearly show the full body in various posture. Pose annotations (joint angles, body keypoints) must be accurate to ensure effective feedback and training.

Data Format:

Use formats compatible with your development tools—e.g., JSON or CSV for keypoint coordinates, MP4 or PNG for videos/images.Standard pose datasets usually include COCO or MPII formats which are widely supported by pose estimation libraries.

Data Licensing:

When using open-source or commercial datasets, always review licensing restrictions. Some datasets may allow academic use only, while others may permit commercial use with attribution or licensing fees.

Limitations of the HyperFit AI Project

Pose Detection Accuracy:

The accuracy of pose estimation can be affected by factors like poor lighting, occluded joints (e.g., arms hidden behind the body), background clutter, or unusual camera angles. This may lead to incorrect feedback.

Device Requirements:

HyperFit AI relies on devices with sufficient processing power and camera quality. Older smartphones or low-end systems may struggle to run real-time pose tracking smoothly, impacting user experience.

Dataset Bias and Generalization:

Models trained on limited or homogeneous datasets may not perform well across diverse body types, clothing, or movement styles, potentially reducing the system's effectiveness for all users.

Additional Considerations for HyperFit AI:

Regulatory Compliance: If HyperFit AI collects or stores user movement data, especially for personalized fitness analytics, it may need to comply with relevant data privacy regulations such as GDPR or HIPAA (if used in medical rehabilitation contexts).

Target Audience Alignment: Tailoring features to different user groups—such as beginners, athletes, or physiotherapy patients—can improve engagement and effectiveness. Clear, accessible feedback is especially important for non-technical or first-time users.

Over-reliance on Technology: Users may depend too heavily on AI feedback and ignore professional guidance. The system should reinforce that it is a support tool, not a substitute for certified fitness trainers or healthcare providers.

CHAPTER 4
SYSTEM DESIGN

4.1 COMPONENTS OR USERS IN THE PROPOSED SYSTEM

Admin

- Manages and updates the exercise database, including pose templates, correct form guidelines, and common mistakes.
- Oversees user accounts and access permissions (if applicable).
- Monitors system performance, usage statistics, and error reports.
- Ensures timely updates to the AI models and app features based on user feedback or new fitness trends.

Pose Estimation Model/Classifier

- Processes video or image input to detect and track body keypoints in real-time.
- Compares user posture against predefined exercise templates to assess correctness.
- Works with the admin to incorporate new exercises or improve detection accuracy.
- Generates feedback data for the user interface.

End User

- Uses the HyperFit AI app to perform exercises while receiving real-time posture feedback.
- Views visual and audio guidance on correct form and repetitions.
- Accesses personalized workout plans and progress tracking (if implemented).

General Users:

Fitness Enthusiasts**:** Use the app to improve exercise technique and prevent injury.

Personal Trainers: Utilize the app to monitor clients' form remotely or during sessions.

Physiotherapists: Employ the system to assist patients with rehabilitation exercises and ensure proper movement patterns.

Game Engine / Development Platform: Unity 3D:Used to develop the interactive environment and real- time feedback system. Unity's support for AR and computer vision plugins makes it ideal for building pose detection and correction features.

Pose Estimation Framework: Mediapipe / TensorFlow:These frameworks provide the core machine learning models for detecting and tracking human body keypoints in real time, enabling accurate pose analysis.

Programming Language: C# / Python :C# is primarily used within Unity for app logic, UI, and interactions. Python may be used for training and fine-tuning pose estimation models or offline data processing.

User Interface (UI) Design: Unity UI Toolkit**:**Used to create user-friendly interfaces, including exercise instructions, visual cues for pose correction, progress tracking dashboards, and interactive buttons.

Version Control: Git**:**For managing source code versions, collaborating with team members, and maintaining the development lifecycle efficiently.

Supported Devices: Smartphones and Tablets (ARCore / ARKit-enabled):The system supports Android and iOS devices with AR capabilities to deliver real-time pose feedback using the device camera.

4.2 PROPOSED SYSTEM ARCHITECTURE

The proposed HyperFit AI system leverages real-time pose estimation technology to detect and analyze users' body posture during exercise. Using the device's camera and advanced machine learning models, the system captures body keypoints and compares them with ideal exercise poses. It then provides instant, personalized feedback and corrections to help users perform exercises safely and effectively. The system includes a user-friendly interface for visual and audio guidance, progress tracking, and customizable workout plans. By integrating AR elements, HyperFit AI offers an immersive and interactive fitness experience accessible on smartphones and tablets without the need for specialized hardware.

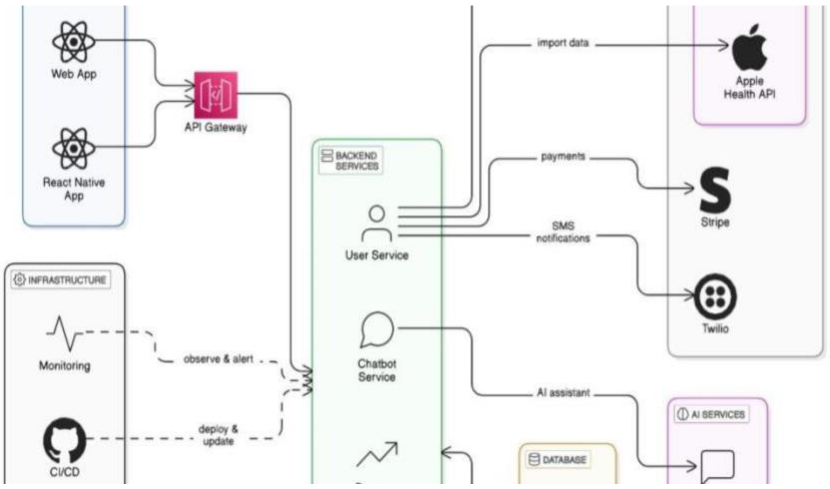


Fig:4.2 Architectural diagram

Frontend:

<u>Web App / React Native App</u>: Provides the user interface for web and mobile platforms.

<u>API Gateway:</u> Routes requests from frontend apps to appropriate backend services.

Backend Services:

User Service: Manages user authentication, profile, and links with external APIs.

Chatbot Service: AI-powered assistant that interacts with users through natural language.

Progress Tracker: Tracks user workout progress, health metrics, and milestones.

Recommendation Engine: Generates personalized workouts and meal plans using AI models.

Database:

PostgreSQL: Central storage for user data, progress records, and recommendation logs.

Data Access: Backend services read/write data to support app functionality and AI processing.

AI Services:

NLP Models: Enable chatbot to understand and respond to user queries naturally.

Recommendation Models: Use user data to predict and suggest suitable fitness and nutrition plans.

Integrations:

Wearable APIs (Fitbit / Apple Health): Import fitness and health data for user tracking.

Stripe & Twilio: Handle payment transactions and SMS notifications respectively.

Infrastructure:

Monitoring: Observes backend service health and sends alerts on failures or anomalies.

CI/CD: Automatically deploys new updates and improvements to the platform.

4.3 UML DIAGRAMS

A UML (Unified Modeling Language) diagram is a visual representation used to model the structure and behavior of a software system. It helps developers, designers, and stakeholders understand how different parts of the system interact with each other. UML diagrams are broadly categorized into structural diagrams (like Class Diagrams) and behavioral diagrams (like Use Case, Sequence, and Activity Diagrams). In the context of Hyperfit AI, UML diagrams can effectively represent its architecture and flow. For example, a Use Case Diagram would show how users interact with features like fitness tracking, AI chatbot assistance, and personalized recommendations. A Class Diagram would define system components such as User, ProgressTracker, and RecommendationEngine, along with their relationships. A Sequence Diagram can illustrate how a user's request for a workout plan flows through the API Gateway to backend services, AI models, and database.

4.3.1 Class Diagram:

A Class Diagram is a type of UML diagram that shows the static structure of a system by representing its classes, attributes, methods, and the relationships between them. It helps in understanding how different parts of the system are connected and organized. In the Hyperfit AI system, a class diagram would include classes like User, ChatbotService, ProgressTracker, and RecommendationEngine, showing how they interact with components like the PostgreSQL database and NLP Models. This helps visualize how user data flows and how personalized recommendations are generated.

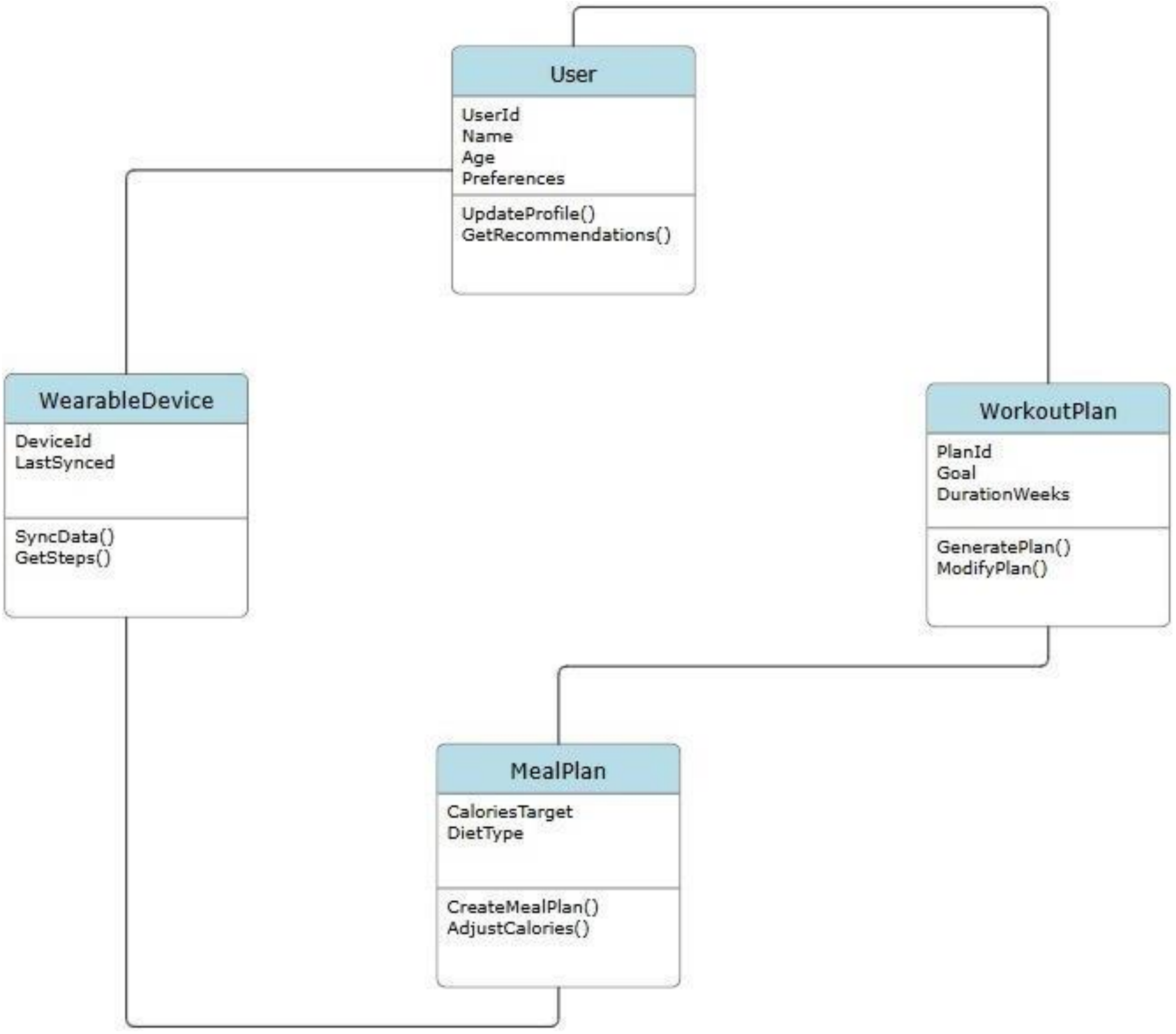


Fig:4.3.1 Class Diagram

4.3.2 Use Case:

A **Use Case Diagram** is a type of UML (Unified Modeling Language) diagram that visually represents the **interactions between users (actors)** and a **system** to achieve specific goals (use cases). It shows **what the system does** (functionality), not **how** it does it.The use case diagram for **HyperFit AI** illustrates the interactions between users (primarily gym-goers) and the system, showcasing the key functionalities it offers. It highlights how users can log in, start workout sessions, and engage with real-time pose detection powered by computer vision tools like OpenCV and MediaPipe. Essential features such as repetition counting, form feedback, and workout summaries are represented as use cases that fall within the system boundary. Optional interactions, such as progress tracking and trainer adjustments, provide additional support for personalized fitness journeys. This diagram helps visualize what services HyperFit AI offers and how users interact with them, ensuring a clear understanding of the system's functional scope.

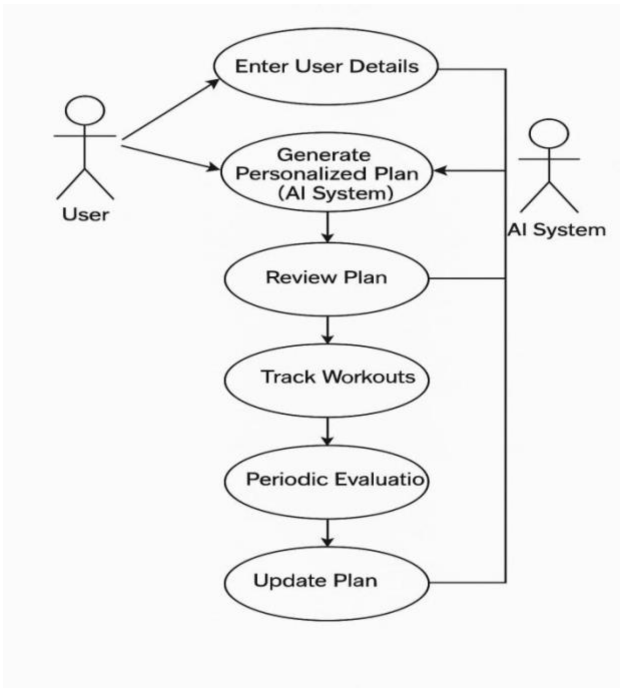
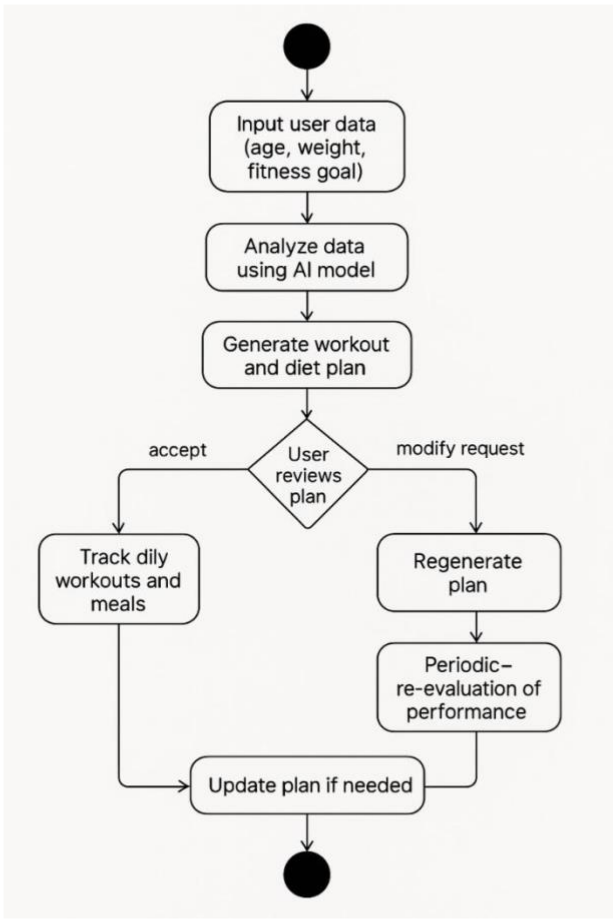


Fig:4.3.2 Use Case Diagram

4.3.3 Activity Diagram:

An Activity Diagram for HyperFit AI represents the flow of actions a user follows while interacting with the system during a workout session. It visually outlines the selecting a workout, and starting the camera for pose detection, to tracking repetitions, providing real-time feedback, and finally displaying a workout summary. The diagram helps identify decision points (like valid pose or not), parallel processes (like rep counting and feedback), and end results, making it useful for understanding the system's workflow and improving user experience.



Input user data (age, weight, fitness goal)
Analyze data using AI model
Generate workout and diet plan
accept
User reviews plan
modify request
Track dily workouts and meals
Regenerate plan
Periodic–re-evaluation of performance
Update plan if needed

Fig:4.3.3 Activity Diagram

CHAPTER 5
IMPLEMENTATION

The implementation of **HyperFit AI** involves integrating Unity with MediaPipe and OpenCV to create a real-time gym tracking system that detects user poses and counts exercise repetitions. The camera feed is processed to extract key body landmarks, enabling analysis of movements like squats or curls. Using Unity's UI system, the app provides instant feedback on form and displays workout statistics. Error handling, user data storage, and performance optimizations ensure a smooth and responsive fitness experience.

1. Unity Setup

- Use Unity as the core platform to build the cross-platform fitness tracking application.
- Set up a 3D/2D gym environment in Unity for real-time visualization and feedback.
- Import necessary Unity packages: AR Foundation (if you plan on adding AR features)

2. Camera Integration

- Access the device camera in Unity using WebCamTexture or Unity Webcam API.
- Display the camera feed in the scene via a UI RawImage or 3D plane.

3. Pose Detection Engine (MediaPipe Integration)

- Integrate MediaPipe via a native plugin or use OpenCV for Unity to handle data transfer between Python (pose model) and Unity.
- Extract 33 key body landmarks in real-time from the camera feed.

4. **Exercise Detection Logic**

Implement logic in C# to:

- Detect poses like bicep curls, squats, pushups using joint angle calculations.
- Count repetitions based on motion patterns (e.g., elbow angle decreasing and then increasing).
- Validate correct form using thresholds (e.g., knee alignment, elbow angles).

5. **Real-Time Feedback System**

- Repetition count
- Current exercise

6. **Error Handling**

Show messages like:

- “Pose not detected”
- “Make sure you're in the camera frame”
- “Incomplete motion – Try again”
- Add fallback animations or default UI when pose data is not available.

7. **User Management & Session Data**

Create a local or cloud-based **user database** (Firebase, SQLite, or JSON).

- User details (name, ID)
- Session logs (date, reps, feedback)
- Exercise history

8. **Performance Optimization**

Minimize CPU/GPU usage by:

- Reducing frame rate of pose detection (e.g., 15 FPS)
- Using lightweight shaders and models in Unity
- Running MediaPipe inference on GPU

9. **Workout Analytics & Summary**

At the end of each session, display:

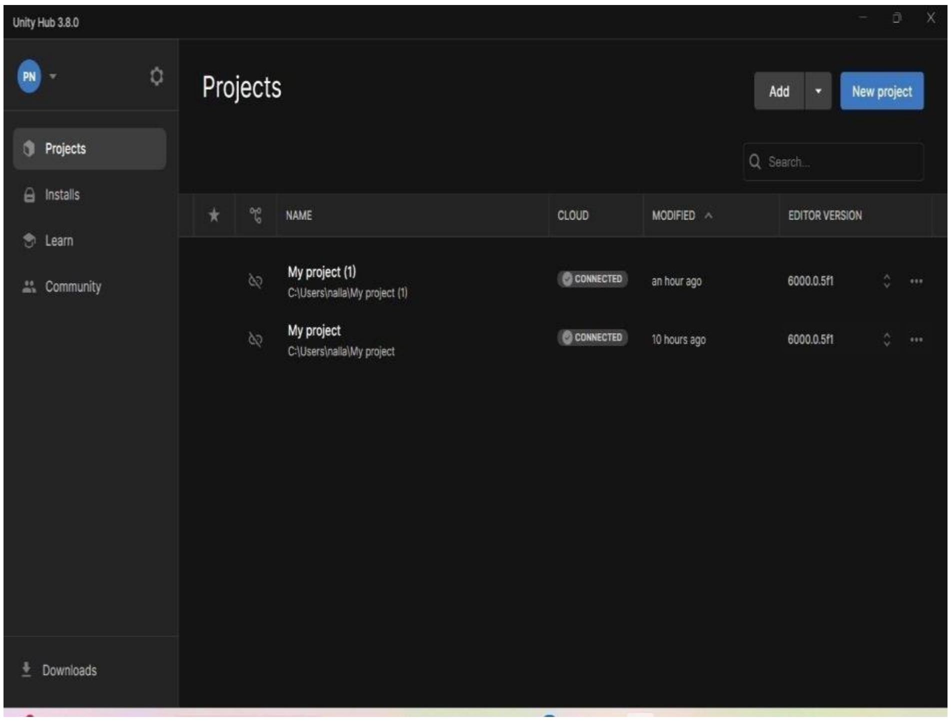
- Total reps
- Duration
- Performance rating (accuracy of form)

10. **AR/Marker (Optional Feature)**

- Use **Vuforia** or **AR Foundation** to place exercise avatars in physical space.
- Detect printed markers (e.g., printed dumbbell images) to start specific workouts.
- Show 3D avatars on top of marker with demo poses.

UI Design: Design a user-friendly UI element within your Unity scene to display the retrieved and matching algorithms to minimize processing time, especially on mobile devices.

Steps involved:

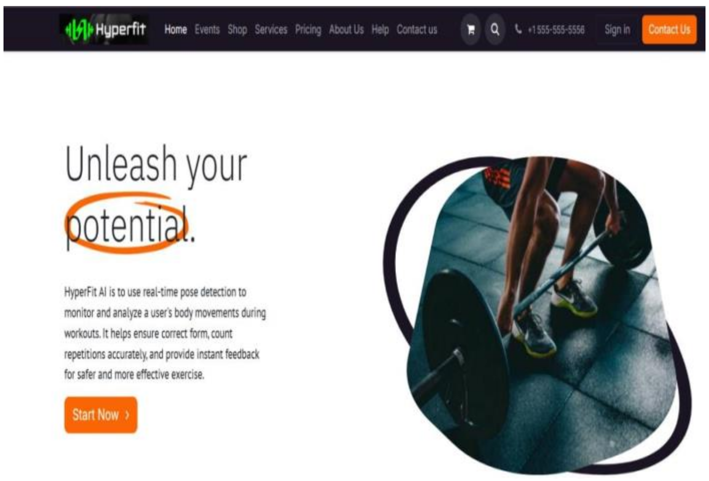


NAME	CLOUD	MODIFIED	EDITOR VERSION
My project (1) C:\Users\nalla\My project (1)	CONNECTED	an hour ago	6000.0.5f1
My project C:\Users\nalla\My project	CONNECTED	10 hours ago	6000.0.5f1

Fig:5.1 Unity interface

import Pose Detection Library package:

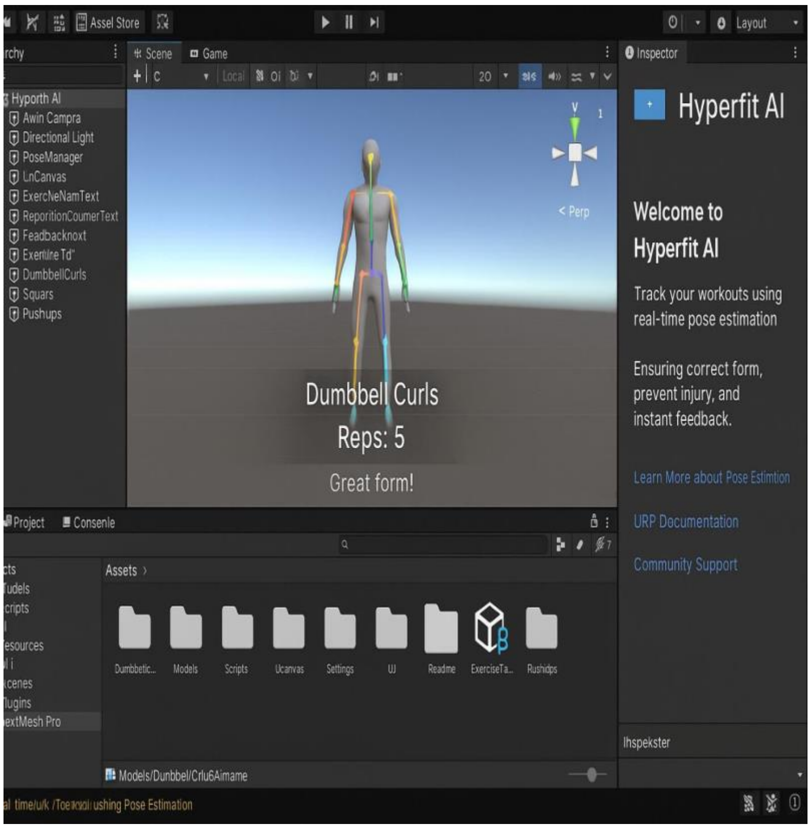
To implement **HyperFit AI**, begin by integrating pose detection capabilities using **MediaPipe** and **OpenCV**. First, set up your Unity project and import the necessary **Python–Unity communication plugin** or use a native plugin to connect MediaPipe with Unity. MediaPipe, known for its accurate real- time human pose estimation, will serve as the core engine for detecting body movements. The application is developed using **C# in Unity**, where the real-time pose data is used to trigger actions like counting repetitions and giving form correction feedback. Instead of markers, HyperFit AI relies on live camera input to detect and analyze the user’s body posture for tracking gym.



Unleash your potential.

HyperFit AI is to use real-time pose detection to monitor and analyze a user's body movements during workouts. It helps ensure correct form, count repetitions accurately, and provide instant feedback for safer and more effective exercise.

Fig:5.2 Hyperfit AI



Dumbbell Curls

Reps: 5

Great form!

Hyperfit AI

Welcome to Hyperfit AI

Track your workouts using real-time pose estimation

Ensuring correct form, prevent injury, and instant feedback.

Learn More about Pose Estimtion

URP Documentation

Community Support

Fig:5.3 Main page

AR Camera: This specialty camera replaces your main camera when working with AR. It captures the real-world video feed and tracks the device's position and orientation.

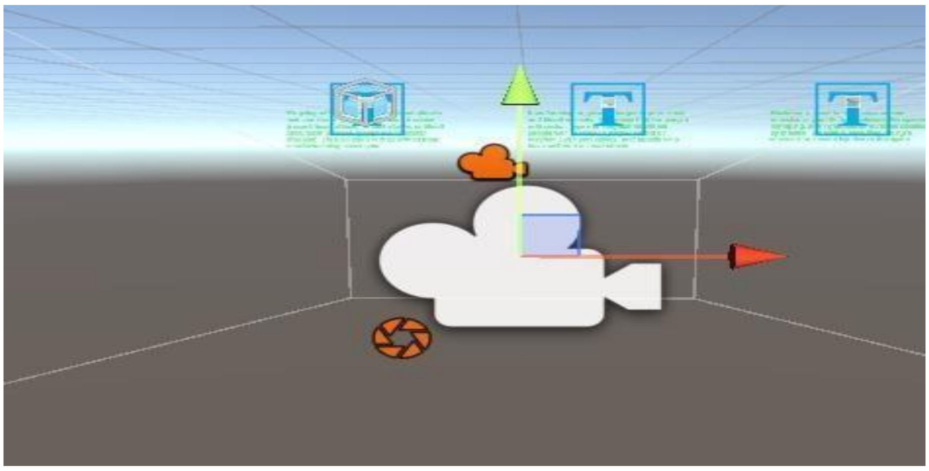


Fig:5.4 AR camera

Pose Direction Lines:

Pose direction lines are visual connectors between key body joints that represent limb orientation and posture. They help track movement and form in pose detection systems like Mediapipe.

Pose Detection

Fig:5.5 Pose Detection Lines

Create Human Posture Detection Prefabs for HyperFit AI

Within Unity, we should create a folder in the project's **Assets** directory to store prefabs related to human posture detection models and feedback UIs.

Goal:

Build reusable prefab components that support posture tracking, including:

- Visual overlays for detected human joints and bones
- UI feedback elements (e.g., "Correct", "Incorrect", "Hold posture")
- Camera input pipeline
- Skeleton model visualization

1. Set Up Pose Detection System

Tools Required:

- Unity
- Mediapipe Unity Plugin **or** a custom pose estimation model (e.g., BlazePose via Barracuda)Steps:

1. **Create a Scene:**
   Go to File > New Scene and name it PoseDetectionScene.
2. Import Pose Detection Package:
   Import **Mediapipe Unity Plugin** or load your trained model via **Barracuda** (Unity ML inference engine).

3. **Set Up Camera Feed:**

- Create a new camera object GameObject > Camera.
- Attach webcam texture script to it to display live input.
- Ensure lighting and layers are adjusted for visibility.

2. Create Skeleton Visualization Prefab (2D or 3D)

For 2D Skeleton:

1. Go to GameObject > UI > Canvas. Under this canvas:
    - o Add UI > Imagefor each joint (e.g., head, elbow, knee).
    - o Create lines between joints using UI > LineRendereror custom UI images.
2. Adjust positioning using the detected coordinates from pose estimation output.
3. Group all joint images into an empty GameObject called HumanSkeleton2D.

3. Create Feedback UI Prefabs

1. Go to UI > Panel. Add text fields or images to show posture status:

- "Perfect Posture"
- "Incorrect Form"
- "Adjust Arm Angle"

2. Name it PostureFeedbackUI.

3. Drag to Prefabs to save.

4. Posture Recognition Logic (Core Script)

Implement a script called PostureAnalyzer.cswhich:

- Takes input from the pose detection model
- Compares joint angles with predefined "correct" posture values
- Outputs posture evaluation to the PostureFeedbackUI

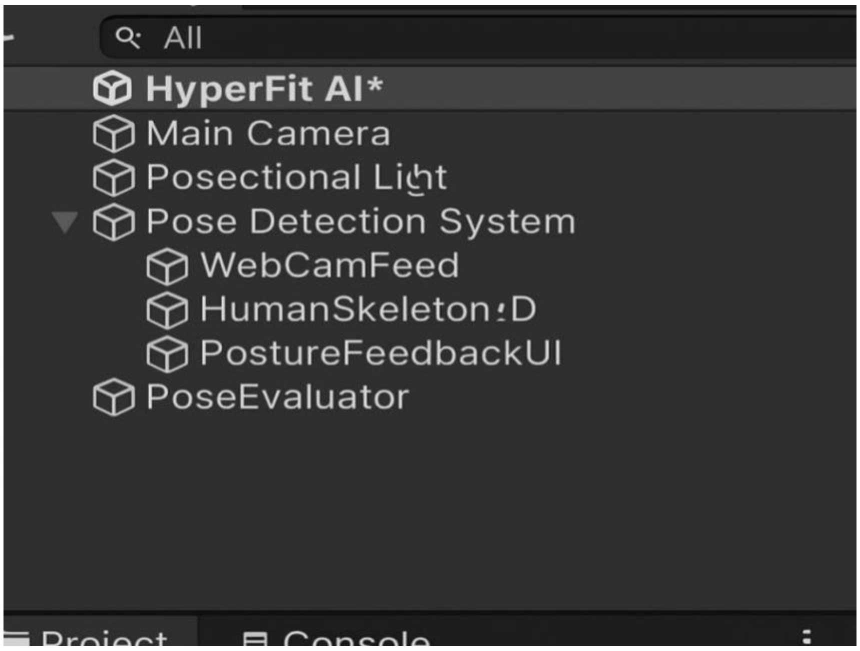


Fig:5.6 InsertionAfter importing the results are:

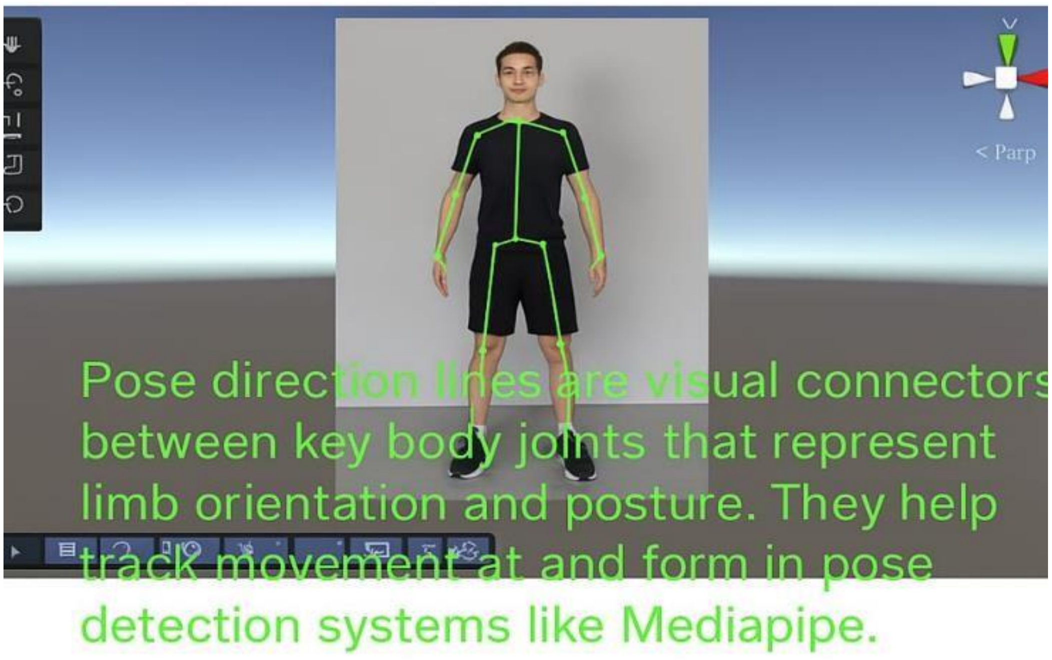

Pose direction lines are visual connectors between key body joints that represent limb orientation and posture. They help track movement at and form in pose detection systems like Mediapipe.

Fig:5.7 Result Before Apk

5.1 Source Code:

AITrainer.py

```
import cv2
import numpy as np import time
import PoseModule as pm
cap = cv2.VideoCapture("AiTrainer/curls.mp4") detector = pm.poseDetector()
count = 0
dir = 0
pTime = 0 while True:
success, img = cap.read()
img = cv2.resize(img, (1280, 720))
# img = cv2.imread("AiTrainer/test.jpg") img = detector.findPose(img, False) lmList =
detector.findPosition(img, False) # print(lmList)
if len(lmList) != 0:
# Right Arm
angle = detector.findAngle(img, 12, 14, 16) # # Left Arm
#angle = detector.findAngle(img, 11, 13, 15,False)
per = np.interp(angle, (210, 310), (0, 100))
bar = np.interp(angle, (220, 310), (650, 100)) # print(angle, per)

# Check for the dumbbell curls color = (255, 0, 255)
if per == 100:
color = (0, 255, 0) if dir == 0:
count += 0.5
dir = 1
if per == 0:
color = (0, 255, 0) if dir == 1:
count += 0.5
dir = 0
print(count)

# Draw Bar
cv2.rectangle(img, (1100, 100), (1175, 650), color, 3)
cv2.rectangle(img, (1100, int(bar)), (1175, 650), color, cv2.FILLED)
```

```
cv2.putText(img, f'{int(per)} %', (1100, 75), cv2.FONT_HERSHEY_PLAIN, 4,
        color, 4)  # Draw Curl Count
cv2.rectangle(img, (0, 450), (250, 720), (0, 255, 0), cv2.FILLED)
cv2.putText(img, str(int(count)), (45, 670), cv2.FONT_HERSHEY_PLAIN, 15,
(255, 0, 0), 25)
cTime = time.time()
 fps = 1 / (cTime - pTime) pTime = cTime
cv2.putText(img, str(int(fps)), (50, 100), cv2.FONT_HERSHEY_PLAIN, 5,
(255, 0, 0), 5)

cv2.imshow("Image", img) cv2.waitKey(1)
```

PoseModule.py

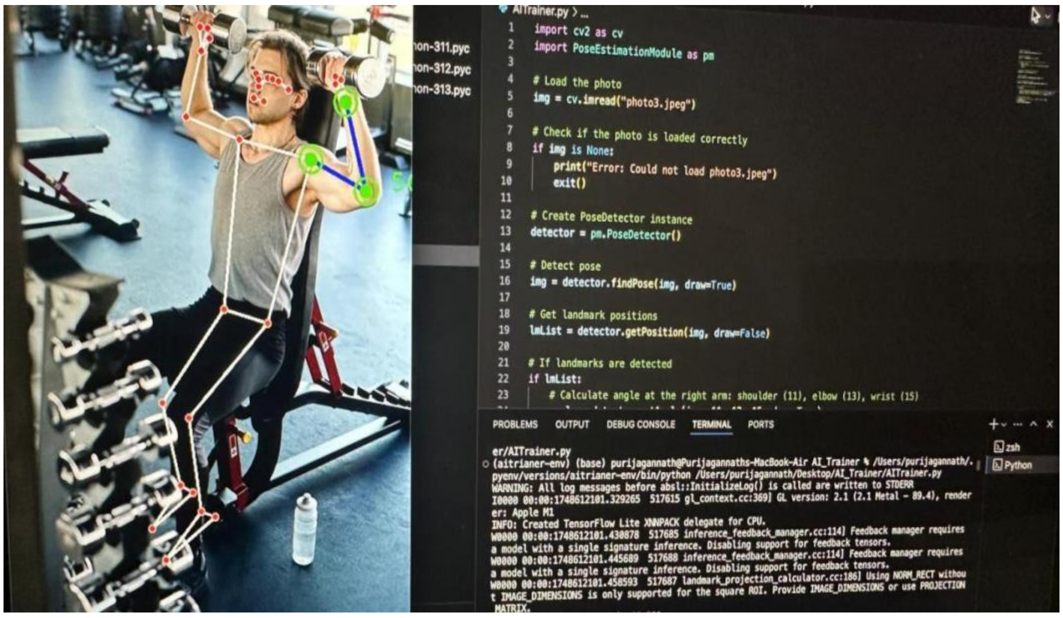
```
import cv2
import mediapipe as mp import time
import math
class poseDetector():

def __init__(self, mode=False, upBody=False, smooth=True, detectionCon=0.5, trackCon=0.5):

self.mode = mode self.upBody = upBody self.smooth = smooth
self.detectionCon = detectionCon self.trackCon = trackCon


self.mpDraw = mp.solutions.drawing_utils self.mpPose = mp.solutions.pose
self.pose = self.mpPose.Pose(self.mode, self.upBody, self.smooth, self.detectionCon,
self.trackCon)

def findPose(self, img, draw=True):
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) self.results =
self.pose.process(imgRGB)
if self.results.pose_landmarks: if draw:
self.mpDraw.draw_landmarks(img, self.results.pose_landmarks,
self.mpPose.POSE_CONNECTIONS)
return img

def findPosition(self, img, draw=True):
self.lmList = []
```

```
 if self.results.pose_landmarks:
for id, lm in enumerate(self.results.pose_landmarks.landmark): h, w, c = img.shape
 # print(id, lm)
 cx, cy = int(lm.x * w), int(lm.y * h) self.lmList.append([id, cx, cy])
 if draw:
         cv2.circle(img, (cx, cy), 5, (255, 0, 0), cv2.FILLED) return self.lmList
 def findAngle(self, img, p1, p2, p3, draw=True): # Get the landmarks
 x1, y1 = self.lmList[p1][1:] x2, y2 = self.lmList[p2][1:] x3, y3 = self.lmList[p3][1:]

 # Calculate the Angle
angle = math.degrees(math.atan2(y3 - y2, x3 - x2) - math.atan2(y1 - y2, x1 - x2))
if angle < 0: angle += 360
 # print(angle) # Draw
 if draw:
 cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3)
 cv2.line(img, (x3, y3), (x2, y2), (255, 255, 255), 3)
 cv2.circle(img, (x1, y1), 10, (0, 0, 255), cv2.FILLED)
 cv2.circle(img, (x1, y1), 15, (0, 0, 255), 2)
 cv2.circle(img, (x2, y2), 10, (0, 0, 255), cv2.FILLED)
 cv2.circle(img, (x2, y2), 15, (0, 0, 255), 2)
 cv2.circle(img, (x3, y3), 10, (0, 0, 255), cv2.FILLED)
 cv2.circle(img, (x3, y3), 15, (0, 0, 255), 2) cv2.putText(img, str(int(angle)), (x2 - 50, y2 + 50),
 cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 255), 2)
 return angle


 def main():
 cap = cv2.VideoCapture('PoseVideos/1.mp4') pTime = 0
 detector = poseDetector() while True:
 success, img = cap.read()  img = detector.findPose(img)
 lmList = detector.findPosition(img, draw=False) if len(lmList) != 0:
 print(lmList[14])
 cv2.circle(img, (lmList[14][1], lmList[14][2]), 15, (0, 0, 255), cv2.FILLED)

 cTime = time.time()
 fps = 1 / (cTime - pTime) pTime = cTime
```

```
cv2.putText(img, str(int(fps)), (70, 50), cv2.FONT_HERSHEY_PLAIN, 3,
(255, 0, 0), 3)

cv2.imshow("Image", img) cv2.waitKey(1)


if _name_____== "_main_": main()
```

CHAPTER-6

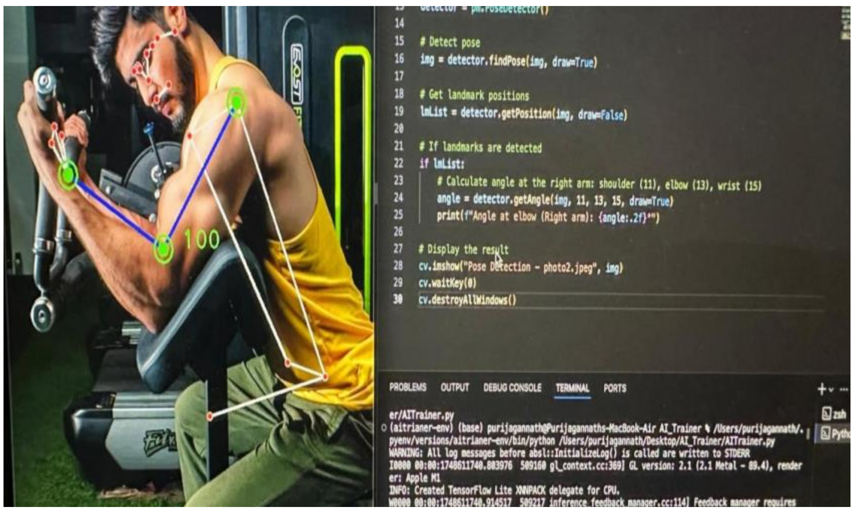
RESULTS



```python
import cv2 as cv
import PoseEstimationModule as pm

# Load the photo
img = cv.imread("photo3.jpeg")

# Check if the photo is loaded correctly
if img is None:
    print("Error: Could not load photo3.jpeg")
    exit()

# Create PoseDetector instance
detector = pm.PoseDetector()

# Detect pose
img = detector.findPose(img, draw=True)

# Get landmark positions
lmList = detector.getPosition(img, draw=False)

# If landmarks are detected
if lmList:
    # Calculate angle at the right arm: shoulder (11), elbow (13), wrist (15)
```

```
er/AITrainer.py
(aitrianer-env) (base) purijagannath@Purijagannaths-MacBook-Air AI_Trainer % /Users/purijagannath/.
pyenv/versions/aitrianer-env/bin/python /Users/purijagannath/Desktop/AI_Trainer/AITrainer.py
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1748612101.329265  517615 gl_context.cc:369] GL version: 2.1 (2.1 Metal - 89.4), render
er: Apple M1
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
W0000 00:00:1748612101.438878  517685 inference_feedback_manager.cc:114] Feedback manager requires
a model with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1748612101.445689  517688 inference_feedback_manager.cc:114] Feedback manager requires
a model with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1748612101.458593  517687 landmark_projection_calculator.cc:186] Using NORM_RECT withou
t IMAGE_DIMENSIONS is only supported for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION
 MATRIX.
```

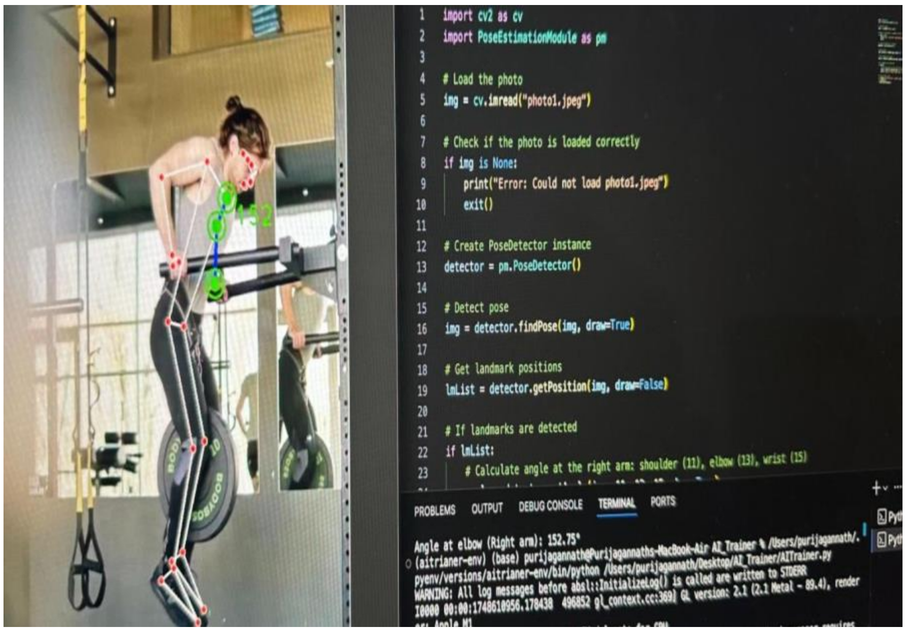
Fig:6.1 Screenshot of Result1



```python
detector = pm.PoseDetector()

# Detect pose
img = detector.findPose(img, draw=True)

# Get landmark positions
lmList = detector.getPosition(img, draw=False)

# If landmarks are detected
if lmList:
    # Calculate angle at the right arm: shoulder (11), elbow (13), wrist (15)
    angle = detector.getAngle(img, 11, 13, 15, draw=True)
    print(f"Angle at elbow (Right arm): {angle:.2f}°")

# Display the result
cv.imshow("Pose Detection - photo2.jpeg", img)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
er/AITrainer.py
(aitrianer-env) (base) purijagannath@Purijagannaths-MacBook-Air AI_Trainer % /Users/purijagannath/.
pyenv/versions/aitrianer-env/bin/python /Users/purijagannath/Desktop/AI_Trainer/AITrainer.py
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1748611740.883976  509160 gl_context.cc:369] GL version: 2.1 (2.1 Metal - 89.4), render
er: Apple M1
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
W0000 00:00:1748611740.914517  509217 inference_feedback_manager.cc:114] Feedback manager requires
```

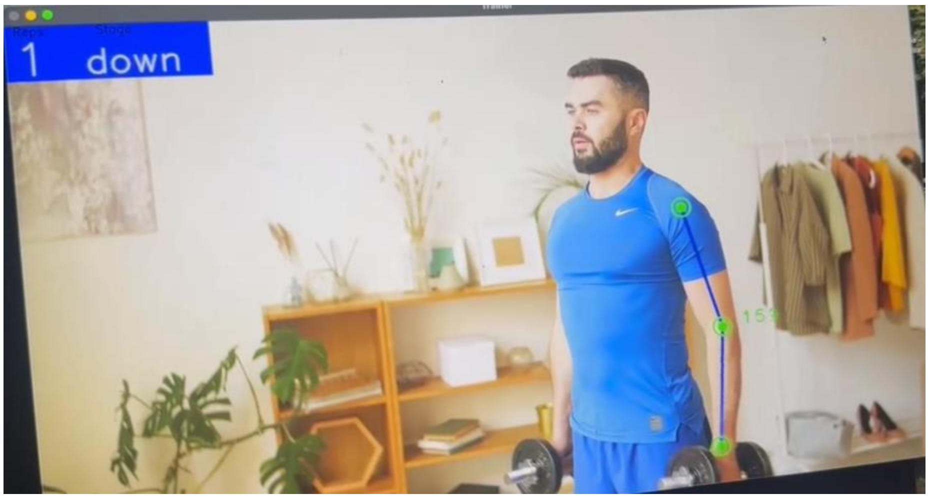
Fig:6.2 Screenshot of Result2



```python
import cv2 as cv
import PoseEstimationModule as pm

# Load the photo
img = cv.imread("photo1.jpeg")

# Check if the photo is loaded correctly
if img is None:
    print("Error: Could not load photo1.jpeg")
    exit()

# Create PoseDetector instance
detector = pm.PoseDetector()

# Detect pose
img = detector.findPose(img, draw=True)

# Get landmark positions
lmList = detector.getPosition(img, draw=False)

# If landmarks are detected
if lmList:
    # Calculate angle at the right arm: shoulder (11), elbow (13), wrist (15)
```

```
Angle at elbow (Right arm): 152.75°
(aitrianer-env) (base) purijagannath@Purijagannaths-MacBook-Air AI_Trainer % /Users/purijagannath/.
pyenv/versions/aitrianer-env/bin/python /Users/purijagannath/Desktop/AI_Trainer/AITrainer.py
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1748610956.178438  496852 gl_context.cc:369] GL version: 2.1 (2.1 Metal - 89.4), render
er: Apple M1
```

Fig:6.3 Screenshot of Result



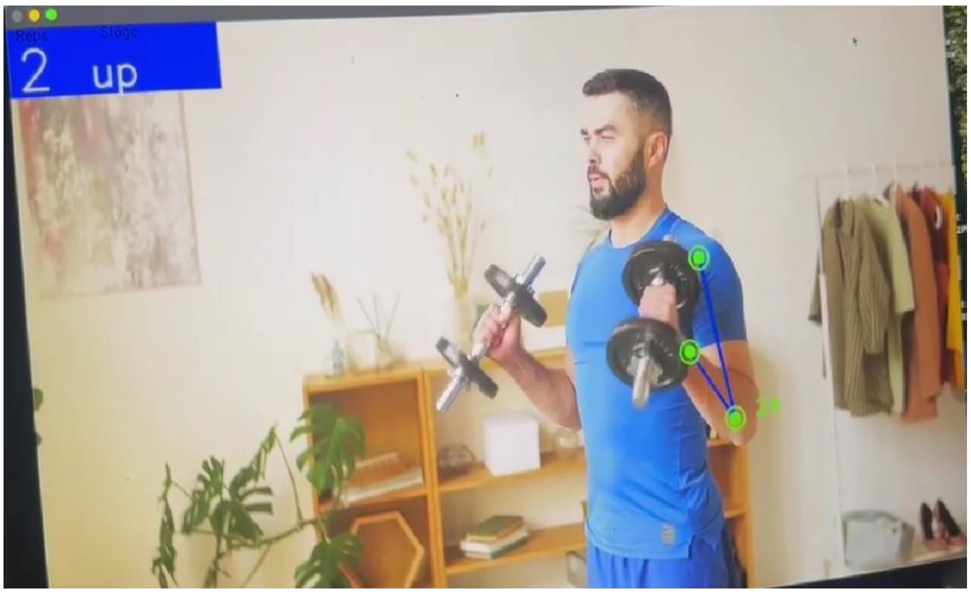


Fig:6.5 Screenshot of Result 5

CHAPTER 7

CONCLUSION

In conclusion, this project successfully demonstrated the potential of Artificial Intelligence (AI) and Pose Estimation in transforming the fitness experience. Developed using Python, OpenCV, and Mediapipe, the Hyperfit AI application tracks human workout movements in real-time with high accuracy. By analyzing body posture during exercises like dumbbell curls, the system provides immediate feedback, helping users maintain correct form and avoid injuries. This user-focused solution promotes self-guided fitness, making professional-level guidance accessible from home. The project sets a strong foundation for future enhancements, such as incorporating a broader range of exercises, personalized workout plans, and voice feedback for hands-free interaction. As AI continues to evolve, Hyperfit AI exemplifies how intelligent systems can redefine personal fitness and wellness.

REFERENCES

[1]. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2019). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), 172-186.

[2]. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Ceze, L., & Kautz, J. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.

[3]. Zheng, C., Zeng, X., Wang, Y., & Hu, B. (2021). AI-powered fitness coaching system based on pose estimation. *Procedia Computer Science*, 183, 316–323.

[4]. Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., & Oliva, A. (2019). Improving human activity recognition through pose estimation. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[5]. Tolooshams, B., & Guo, X. (2021). Smart workout tracker using pose estimation and feedback. *IEEE International Conference on Consumer Electronics (ICCE)*, 1-4.

[6]. Lee, J., Kim, Y., & Han, Y. (2020). Fitness posture evaluation system using 3D pose estimation and feedback. *Journal of Intelligent & Fuzzy Systems*, 38(2), 1781–1788.

[7]. Kim, S., & Kim, J. (2020). Real-time exercise recognition using AI and vision-based pose estimation. *Sensors*, 20(23), 6836.

[8]. Zhang, W., Xu, X., & Liu, X. (2020). PoseTrainer: Real-time posture feedback for exercise training using deep learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(3), 1-19.

[9]. Haque, A., Guo, M., Alahi, A., Yeung, S., Luo, Z., Rehg, J. M., & Fei-Fei, L. (2016). Towards viewpoint invariant 3D human pose estimation. *European Conference on Computer Vision*, 160-177.
[10]. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.