International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 9 Issue 10 October - 2025, Pages: 152-162

Design and Evaluation of an Intelligent Web Application for At-Home Fitness and Personal Health Tracking

Pham Hoang Quan1 Tran Quang Truong2

[1] Ocean Park High School, Viet Nam Email: quan.pham211108@gmail.com

[2] AI Engineer, Viet Nam Email: truongtq.tekverse@gmai.com

Abstract: *The consequences during the COVID-19 pandemic were not only due to the virus itself. As a consequence of 'social distancing' regulations and subsequent restrictions implemented during the early phase of the pandemic, the healthcare system, and the general population's physical and mental health suffered greatly. With difficulty in access caused by social distancing regulations, people did not engage in physical activities and were not able to attend healthcare facilities like gyms and yoga studios. In response to this, the proposed study developed an AI-based web solution designed to provide safe and effective guided home exercise to customers. Here, the use of advanced technology that incorporates deep learning and computer vision is described. Particularly, the system employs Google's MediaPipe Pose model to "watch" and "follow" users, while a yoga poses classification model developed in Keras assesses and modifies a user's yoga pose in real-time. The system provides feedback, counts repetitions, and detects exercise poses to ensure users maintain correct form. In addition to guided exercise, the system offers electronically supported health maintenance through blink detection and fatigue monitoring, alerting users to take breaks during extended computer use, as well as working through interactive, movement-based games and other training activities. The website features were developed using Flask (Python) for the backend, with the frontend constructed using HTML, CSS, and Bootstrap. This configuration results in a lightweight system that is easy to configure and adaptable to numerous interfaces. According to experimental findings, this system successfully identifies the location of 32 joints within the human anatomy, records the frequency of a user's workout sessions, and differentiates various yoga positions. The study points to the potential applications of AI in the field of fitness and personal healthcare, especially after the pandemic, when the demand for smart, safe, and easy-to-use remote workout options has increased.*

Keywords: Pose Estimation, Home Fitness, Health Monitoring, Web Application, Exercise Tracking.

1. Introduction

The COVID-19 pandemic has changed a lot about how people live and do things every day, especially when it comes to their mental health and how often they exercise [1]. People used to work out at gyms, yoga studios, and fitness facilities, but lockdowns and restrictions about social separation have made it very hard to go to these places [2]. This unprecedented circumstance has led many individuals to cease physical activity, adversely affecting their bodily and mental well-being [3].

Many studies show that being inactive for a long time is linked to health problems like obesity, heart disease, musculoskeletal illnesses, and mental health difficulties [4] [5]. The pandemic has made these concerns worse, especially for people who don't have access to expert fitness instruction or equipment. Also, if you work out at home without the correct supervision, you might have bad posture, which can lead to injuries and make the activity less effective.

The current global health crisis has made it evident that there are certain key things to think about when it comes to staying active. First of all, access is still a significant issue because many individuals cannot get to a gym, hire a personal trainer, or obtain guidance from an expert. This difference has been bigger since the pandemic lockdown, which has made a lot of individuals rethink how they work out at home. The COVID-19 lockdown closed gyms, which hurt around 80% of people who like to work out around the world. Because of this, a lot of individuals have to figure out how to work out at home. [6].

Second, the danger of damage is larger when there is no expert monitoring. This is because improper form when exercising makes injuries more likely and makes the workout less effective. Studies show that over 65% of injuries that happen during home workouts are caused by bad form [7].

Third, it is challenging to start an exercise regimen when people are not as social and do not have professional help at home. Psychological research shows that social support plays a key role in sticking with a fitness program. For example, people who work out alone have about 40% poorer retention rates [8].

Fourth, health problems that individuals do not pay enough attention to, like incorrect posture or eye strain from too much screen time, are becoming a long-term problem. The average time spent on screens each day has gone up from 6 to 13 hours as more people work from home. This has a substantial impact on posture and eye health [9].

Last but not least, current fitness apps are not particularly beneficial because they do not give consumers real-time feedback on how well they're working out. Instead, they mainly exhibit moving things like pictures or videos. This makes it challenging for people to develop their posture and skills [10]. The goal of this study is to create and test an intelligent web platform that uses the following possible focus

tools to address the stated formulations on information: (1) design and build a real-time exercise monitoring system that uses computer vision technology and advisory value, and works well on standard hardware; (2) create a repetition counting system and automatic posture advice for popular exercises with high accuracy (>90%); (3) use deep learning classification models to test yoga posture recognition capabilities that can identify different postures; (4) add more health monitoring features like detecting sitting postures and eye relaxation through blink counts; (5) make the user experience more fun by using gamification and interactive elements that boost exercise motivation; and (6) test the system thoroughly and get feedback from real users to see how accurate, effective, and satisfying it is.

2. Methodology

2.1. Overview of System Architecture

The Web system works like a client-server system. Flask (Python) is used to make the backend layer, and HTML, CSS, and Bootstrap are used to make the frontend layer. The architecture is made up of modular layers that operate, can grow, and are easy to keep up with.

a. The Layer for Presentations

This layer is in charge of talking to the user directly through the web interface. The UI was made with Bootstrap, HTML5, and CSS3. It can show lessons live, play videos in real time, show feedback, and keep track of how many times a lesson has been done. Users do not need to install any extra software to use the system right away from an ordinary browser. Flask Streaming and OpenCV work together to make streaming possible. It works on most computers at a consistent frame rate of 25 to 30 frames per second.

b. The Processing Layer

This is the major part of the system, where AI models are generated for jobs that entail seeing objects on a computer as following:

- Pose Estimation: Send MediaPipe BlazePose a video stream and it will figure out the proper mechanics for you.
- Yoga Pose Classification: Find Yoga poses in your own data using a Keras/TensorFlow model that has already been trained.
- The state (machine state) is passed from each exercise to compute the gap for the movement in each exercise.
- Compare the user's joint angles in relation to an established reference template to ensure alignment in correct posture.
- Eye Aspect Ratio (EAR) counts the number of blinks and indicates the dryer your eyes are, the older you are.

All module processors are designed in a way that they are usable for many people as all of them are functional on CPUs without GPUs.

c. The Business Logic Layer

This layer is located in between the user and the AI that's applying the model to process a task. This layer monitors and saves user training session progress by detailing completion, real-time feedback, and interactive engagement.

Most importantly, this module:

- Monitors the repetition, intensity, and completion of each movement.
- Outputs feedback in a predefined range of images and text based on the effectiveness of the provided guidance.
- Creates and structures tasks, system rewards, and competitive leaderboards.

This section operates on the app.py and Routes.py files and utilizes Flask Blueprints in order to separate functions.

d. Data Layer

This layer allows the user to store and retrieve personal and configurable system information. For basic calculations, the system uses SQLite, and for more advanced queries coupled with the Flask app, you can use SQLAlchemy ORM. There are a few primary tables:

- People: Records training progress, personal details, and version history.
- Activities: including highest joint angles and reference posture models.

The Performance Log is designed to document, within a specified timeframe, the recorded performance, frequency of task completion, the posture score, and the eye fatigue index.

e. Upkeep and Growth

The web's modular architecture allows for seamless incorporation of new capabilities to the AI and UI models. The cohesive functional layers streamline operations while simultaneously improving collaboration between the front-end and back-end teams.

2.2. Technology Stack

The system employs the latest tools such as the web interface, computer vision, and deep learning frameworks. This enables growth, speed, and real-time interactive computing. This technology can be segmented into three main parts: backend technology, frontend technology, and development tools.

a. Backend Technology

This layer entails processing the data and developing the AI models, and user layer communication via RESTful and SocketIO architecture. Below are the most salient ones:

- Python 3.8 and later: This is the system's primary programming language. It has an easy syntax and an excellent selection of AI and data science libraries and tools.
- Flask 2.0 is a minimal web framework that simplifies the creation of APIs and web services. Flask handles the HTTP requests, the routing, and the web page itself.
- OpenCV 4.5 is a computer vision framework that enables altering format of frames, playing videos, and capturing images.

- MediaPipe 0.8 is a framework by Google's runtime consultant. It has a 33-point discussion that is excellent for power-hungry and precision-demanding applications.
- Utilizing TensorFlow 2.x and Keras 2.x, can train a model capable of distinguishing between various yoga postures. Keras is user-friendly, and permits easy model modifications for transfer learning.
- For high-performance numerical learning and matrix processing, version 1.21 of NumPy is a powerful toolbox. It helps AI models understand and implement transformations, and ensures uniformity of all input data.
- Flask-SocketIO: Flask also provides a library that facilitates bi-directional communication between the server and the browser. Although it only provides the number of iterations and posture alerts, you can receive immediate feedback.

b. Frontend Technology

The player can see and interact with the front-end layer of the game. It puts a lot of stress on being easy to use, intuitive, and quick to reply. The most important technologies are

- HTML5 is a markup language that is used to design the layout of websites and the layout of component interfaces.
- CSS3 is a stylesheet language that makes it easy to utilize apps, change layouts, and use interfaces.
- Bootstrap 4.6 is a framework that enables you to create responsive interfaces. This means that your website will display on a wide range of devices and screen sizes.
- JavaScript ES6 is a client-side programming language that handles user interactions, updates the interface, and makes calls to the backend for real-time data.
- WebRTC is a technology for streaming video in real time that lets people connect to the system directly through webcams without having to install any extra software.
- Chart.js is a library that lets you show data on the web. It employs graphs to indicate how well someone is doing in training, how far they've come, and how well they're doing.

c. Tools for Development

These development tools help the website stay stable, make sure the environment can be replicated, and handle the source code correctly:

- Git is a version control system (VCS) that keeps track of changes to the source code. Many developers maintain it up to date.
- Virtual Environment (venv): A means to build up a Python environment so that the library packages needed for the project do not clash with the ones that are already on the system.
- The normal way to manage packages in Python is with pip. It installs and updates the libraries that other programs need.

2.3. Pose Estimation Module

a. MediaPipe Integration

Google's MediaPipe Pose platform, which is an opened source platform for real-time computer vision services, is what makes the Web system able to keep track of and grade exercises. MediaPipe is a cross-platform, lightweight, and efficient solution that lets people guess their position in real time without requiring a GPU. This means that the system operates well on most regular platforms, such as PCs and mobile devices.

MediaPipe's pose estimation technique uses a deep learning model that has been trained to determine 33 entertainment light points (landmarks) for each user. There are 32 main light points that are needed to figure out the game's angles and poses. The primary reference point is the 33rd light point. There are four primary divisions of these points which are determined by the position.

The face contains five points; these are the nose, the left and right eyes, the left ear, and the right ear. This location will support to discover the guiding head and the microscope icon.

The upper body point group has 12 points and includes the left and right shoulders, arms, wrists, thumbs, index fingers, and little fingers of both hands. This group can assist to figure out how to look at hand movements, check for extension, and evaluate if something is good for you.

Trunk point group (2 points): This group includes the left and right hips, which help to determine the body's central axis and overall posture.

The lower body point group consists of 14 points: the knees, ankles, shins, and the two index fingers of the feet. These points assist in learning body posture, gait, and equilibrium.

Each point (landmark) is characterized by a number of quantitative descriptors:

- Position (x, y, z) pertaining to the three axes of the normalized 3D coordinates in relation to the input frame dimensions.
- Visibility Point (visibility point): A score of confidence (0 to 1) pertaining to the likelihood of the landmark being visible in the frame.
    - Presence Point (presence point): A score of (0 or 1) pertaining to the estimated presence of the landmark in the frame.

Because of its complex architecture, MediaPipe 15angle-calculation, and real-time feedback during training. posture analysis and tracking user movements over the Web, analysis and feedback provided during training enables applications to auto adjust, restore, and verify posture in real time based on feedback provided during training.

International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 9 Issue 10 October - 2025, Pages: 152-162

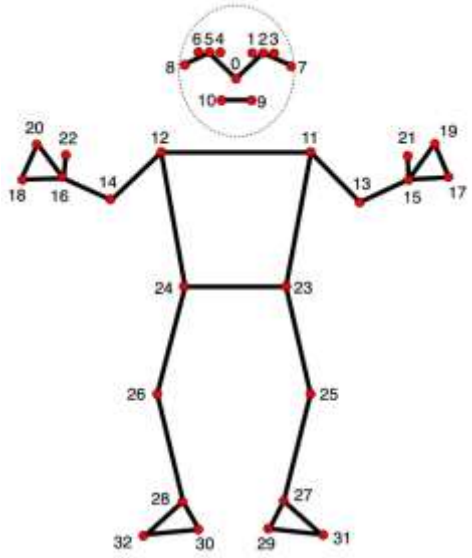

Figure 1. *A schematic of the MediaPipe BlazePose 33-point body landmark detection model*

Figure 1 shows the 33-landmark human skeleton model that Google's MediaPipe BlazePose system uses. The red dots show important bodily parts, and the black lines connect them to make a skeleton that looks like a person.

b. System of Coordinates and Normalization

The posture estimation module (position estimation output) in the Web system is set up so that cameras with different resolutions and viewing distances can function together. Each light point (landmark) is described by a three-dimensional spatial interface (x, y, z) that is normalized to the input dimension frame. The techniques for calculating are based on the following formulas:

$$x_{norm} = \frac{x_{pixel}}{image\ width} \quad \text{(Scope: } 0 \rightarrow 1\text{)}$$

$$y_{norm} = \frac{y_{pixel}}{image\ height} \quad \text{(Scope: } 0 \rightarrow 1\text{)}$$

$$z_{norm} = \frac{z_{depth}}{image\ width} \quad \text{(Scope: -1} \rightarrow 1\text{)}$$

The z-axis in this instance indicates how deep the landmark point is relative to the horizontal midpoint of the body axis which is the origin of the body coordinates. Points having negative z-values are closer to the camera and points having positive z-values are further away. Normalizing by image width eliminates errors due to resolution or angle of view. Thus, the system can effectively handle varying camera configurations (ranging from 480p to 4K) and distances (from 0.5m to 3m).

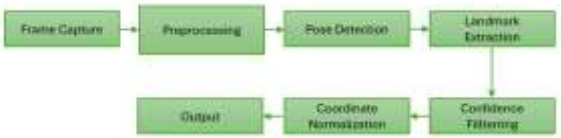

Figure 2. *Pipeline flowchart from video capture to landmark extraction*

In **Figure 2**, the workings of the Web system with respect to pose data are illustrated. The pose recognition model gets the data after the webcam captures the data and the data undergoes pre-processing. Afterwards, the model captures different body posture images. From the captured images, the model gets the posture images and filters these images with body posture confidence. It also scales the images to the size of the frame. The system output after all these adjustments is the normalized posture which is used for the analysis modules.

The Web system utilizes a series of pipelined processing steps to deliver real-time pose recognition under any standard configuration. These steps include:

1. Frame Capture: The system captures webcam pose recognition images with a frequency of 30 frames per second and a resolution of 640x480 pixels.Preprocessing: The MediaPipe model requires frames to be in RGB color space, so the BGR color space is converted to RGB.
2. Pose Detection: MediaPipe BlazePose is one of the models which determine the pose attributes and determines the position of the body landmark within the frame.
3. Landmark Extraction: The model not only detects an object but also automatically identifies and scores confidence in 33 landmarks.
4. Confidence Filtering: For improving the quality of the analysis results, landmarks which visibility score is less than 0.5 are fully eliminated.
5. Coordinate Normalization: This method automates the scaling of all recording devices to the input image size, which then normalizes the coordinates.
6. Output Generation: The system constructs an output data array of 33 predefined points of interest. Each landmark has the following properties: (x, y, z, visibility, presence).

With the above processing chain, the Web system has the ability to determine the position of an individual in real-time with high accuracy and very low latency. This is also made possible with coordinate normalization, which reliably accommodates varying ambient light and hardware. This is beneficial to computer vision applications that aim to engage, rehabilitate, and monitor users in real-time.

2.4. Exercise Monitoring Algorithms

2.4.1. Joint Angle Calculation.

The device keeps track of exercise by detecting the angle between three places on the body. MediaPipe BlazePose supplies the 3D coordinates of the landmarks, which are utilized to figure out the angle of the joint. This method lets you see in real time how your posture, range of motion, and how many times you do each action.

If $P_1(x_1, ,y_1, ,z_1)$, $P_2(x_2,y_2,z_2)$, and $P3(x_3,y_3,z_3)$ are three consecutive landmark points and P_2 is the joint vertex, you may use vector math to get the angle θ at the joint by doing the following:

Step 1: Build the link vector:

$$\overrightarrow{v_1} = (x1 - x2, y1 - y2, z1 - z2)$$

$$\overrightarrow{v_2} = (x3 - x2, y3 - y2, z3 - z2)$$

Step 2 – Calculate the dot product and magnitude of the vector:

$\vec{v}_1 \cdot \vec{v_2} = (x_1 - x_2)(x_3\text{-}x_2) + (y_1\text{-}y_2)(y_3\text{-}y_2) + (z_1 - z_2)(z_3\text{-}z_2)$

$$|\vec{v_1}| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2};$$

$$|\vec{v_2}| = \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2}$$

Step 3 – Calculate the angle between the two vectors:

$$\theta = arccos \frac{\vec{v_1} \cdot \vec{v}_2}{|\vec{v_1}| \times |\vec{v}_z|} ; \theta_{deg} = \theta \times \frac{180}{\pi}$$

The angle value θdeg shows how far the joint can bend and stretch. This is what the system uses to check the movement form (form assessment) and count how many times it happens on its own (repetition counting). For exercises in two-dimensional (2D) space, the system can leave off the z component to make computations easier without changing the outcomes.

2.4.2. Algorithm for Counting Repetitions

The program uses the idea of a Finite State Machine (FSM) to count how many times you do a physical activity. This makes sure that the counting of workout repetitions and tracking of movement are both accurate and reliable. This helps people make fewer mistakes when there is noise or movement that isn't normal. This strategy works particularly well for apps that keep track of posture in real time when it comes to Human Pose Estimation.

The FSM structure of the system has six main states, which are stated below:

- IDLE: This is the first state, and the system is waiting for the user to start moving.
- START_POSITION: The user is at the beginning position, which is normally when the angle of the joint being measured is near to the threshold θ_start.
- TRANSITION_DOWN: the component of the movement from the starting position to the ending position, when the angle of the joint slowly moves down and gets past the threshold θ_start.
- END_POSITION: the user gets to the end position, which is usually the same as the threshold θ_end.
- TRANSITION_UP: the time when you move back from the end position to the starting position.
- COMPLETE: the state that shows that a full movement cycle has been finished, which is the same as a valid repetition.

To enable movement recognition the systems gathers the details of each individual workout in the early phases. The following elements govern the sequence:

- θ_start: describes the angle of the starting position
- θ_end: describes the angle of the finishing position
- θ_tolerance: the maximum permissible error during angle measurement, typically within ±10° to ±15°, based on the activity.
- min_hold_time: the minimum duration the user has to stay within the region (default is set to 0.3 seconds).
- transition_time: the maximum duration to move from one position to another (default is set to 3.0 seconds).

In the real-time tracking phase, the system captures joint angle information continuously at 30 Hz. The FSM uses the current angle to ascertain its state and whether a transition to a new one is appropriate. The system records state change instances, which allows it to ascertain the duration a user holds a certain posture and analyze the movement sequence.

The threshold detection method activates when the angle of the joint exceeds the set limits. To prevent false oscillations at the threshold, the hysteresis mechanism is used. Furthermore, the algorithm discards less precise samples by utilizing the confidence score from the posture detection model (e.g. MediaPipe BlazePose).

An entire sequence of state transitions is valid when it follows this order: START → TRANSITION_DOWN → END → TRANSITION_UP → START.

The system imposes a restriction of maintaining each posture for a minimum duration specified by min_hold_time, and the duration for shifting between positional transitions must not exceed transition_time. A three-frame moving average filter smooths the data, which substantially alleviates the interference caused by noise and vibrations.

There are mechanisms in the FSM to avoid incorrect counting arising from uncompleted actions or rapid alterations. In particular, transitions that last less than 0.5 seconds, and actions that fall below a confidence threshold of 0.6 are dismissed. There is also a 0.2 seconds cooling period after each genuine count to prevent the capture of overlapping recordings that may result from small initial movements.

Table 1. *Supported parameters for each kind of workout.*

Exercise	θ_start	θ_end	θ_tolerance	Tracked joints
Push-up	160°	90°	±10°	Elbow
Squat	170°	90°	±15°	Knee
Bicep Curl	170°	40°	±10°	Elbow
Lunge	170°	90°	±15°	Knee
Shoulder Press	90°	170°	±10°	Elbow

The characteristics used by the Web system to recognize and count repetitions for every exercise type are displayed in Table 1. Each exercise has multiple angle thresholds calculated from the starting position (θ_start) and the finishing position (θ_end). There are also ranges of acceptable movement recognition errors (θ_tolerance). This helps in providing flexibility in movement recognition for every individual.

This finite state machine framework is an excellent starting point for tracking and studying how people move in real time. It enables the AI system to analyze multiple movement phases, reducing the possibility of measurement errors, and ensures that the right number of repetitions of an activity is recorded. This is fundamental to a smart fitness

tracking system which enhances user experience in digital fitness settings with voice feedback and image analysis.

2.4.3. Form Correction System

The system's posture evaluation mechanism checks the measured joint angle against the reference angle range for each workout. Then, it sends the user real-time written and visual feedback to assist them in keeping their posture correct.

The posture assessment algorithm's formula indicates how far the measured angle is from the reference angle:

$$Deviation = \theta_{measured} - \theta_{reference}$$

The system sends the message "Excellent form!" if the value of the deviation is less than or equal to the tolerance level. This signifies that the person is standing or sitting up straight. The interface has a green skeleton (GOOD) and says, "Keep it up."

On the other hand, if the deviation is greater than the permissible range, the system categorizes it into two warning levels. If the measured angle is substantially smaller than the reference angle, the system displays the warning message **"Increase angle by X°"** and a yellow skeleton. This signifies that the user has not bent or stretched enough. If the deviation is more than double the tolerance threshold, the system considers this a critical issue. It sends the message **"CRITICAL: Increase angle by X°"** with a red skeleton (ISSUE) as a warning.

If the measured angle is larger than the reference angle by more than the authorized threshold (in other words, if the user moves too far), the system also advises them to **"Decrease angle by X°"** with the appropriate color code.

The website's feedback systems are meant to work in a multitude of different ways, such as:

(1) Visual feedback: Seeing a skeleton with a "color skeleton system" helps students quickly see their posture feedback and see if they are: green for good posture, yellow for warning, and red for bad posture. Other important features include the current joint angle, the angle target zone, and the motion trajectory for easy observation and adjustment.

(2) Text feedback: Posture correction instructions are provided like "Lower hips" and "Keep back straight", and display the degree of angular deviation which helps practitioners compared expected positions and assess accuracy.

(3) Audio feedback (longer orientation): Future system updates will include "voice feedback" to encourage more realistic and interactive training scenarios which will also include " posture correction instructions", "repetition reminders" and "motivational phrases" to encourage students.

This approach aims to discover typical positioning mistakes during exercises like push-ups, squats, and bicep curls. By assessing the angles formed between the shoulder, elbow, and wrist, the system can determine when push-ups have been improperly performed. Incorrect performance includes when excessive flexion, arching of the back and hip angles greater than 180° (i.e. above the horizontal line) occurs, when the elbow angle is greater than 100° in the lowest position of the push-up, and when the elbows move sideways (i.e. away from the body).

The common mistake during squats is the knee joint exceeding the toe line when the knee angle is greater than 100° (not going deep enough) and the weight is not centered causing the hip to slide sideways (unbalanced position). Kukutschka and Neumayr (2021) describe standards to define bicep curl mistakes, including the elbow flexion when the distance between shoulder and elbow changes more than 5 cm, excessive angular velocity, and under contraction when the angle between arm and forearm is more than 50°. Setting these standards enables the system to determine the effectiveness of movements and offer the user targeted suggestions for enhancement.

2.5. Yoga Pose Classification Module

2.5.1. The architecture of the model

Identifying yoga positions is an essential part of the smart motion monitoring system because it can precisely recognize and evaluate positions depends on the practice of user. For this purpose, the system employs a deep learning model that integrates convolutional neural networks and transfer learning, which is a method that utilizes pre-trained models, as they offer efficient feature extraction. This is especially helpful for analyses involving large datasets.

Design implementation for this feature is consistent with the remaining components of the deep learning pipeline. As with any other deep learning system, it is divided into four components: the input layer, the base model, feature extraction, and the classification head. The model can accept a standard-sized RGB image (224 × 224 × 3) or a coordinate matrix with landmarks extracted from the pose estimation module containing 33 points, each of which has four components: [x, y, z, visibility]. Normalization of data within the [0, 1] interval keeps outliers in check and guarantees the model maintains stability during training.

(2) Base Model: This module employs CNN architectures, specifically the pretrained versions of MobileNetV2 and ResNet50, along with the ImageNet dataset.

This allows the model to capture and learn the high-level geometric and textural features in pictures. During the initial stages of training, the model's layers are frozen, which helps to keep and protect the original weights. These layers are then unfrozen and adjusted in accordance with the training yoga position recognition task. This significantly reduces training time, increases applied accuracy, and improves computational efficiency for recognition of standard training poses.

(3) Additional Processing Layers: After the base model features are extracted and configured, the model processes the features and moves up to the next stage of the network hierarchy. After this, the model employs a Global Average Pooling Layer, which significantly minimizes features' spatial dimensions while retaining a large amount of spatial information. A Dropout Layer with a coefficient of 0.5 follows, which helps reduce overfitting by randomly silencing a certain portion of the neurons. To 'learn' the features of the yoga postures, including the position of the joints, the

orientation of the body, and the balance of the body, a 256 node Dense layer with the ReLU activation function is added next.

(4) The Classification Head: The output of the model is processed by a Dense layer of 128 nodes which uses ReLU activations. This adds another layer of abstraction to the advanced features. The 0.3 set Dropout layer adds further to generalization. The probabilities are distributed over the N output classes using a Softmax layer. Each class corresponds to a different yoga pose: Warrior, Tree, Plank, Downward Dog, etc.

The model is set with the Adam Optimizer to 0.001 learning rate, uses Categorical Cross-Entropy as the loss function, and takes accuracy as an evaluation metric. This configuration captures a fair trade-off between convergence and the stability of training across the epochs. During the evaluation stage, the model's pose classification accuracy was tested over a new dataset and evaluation ascertained pose classification as accurate even under several challenging conditions including lighting, angle and attire of the model.

Using a CNN architecture along with transfer learning enabled the yoga position recognition module to attain high confidence in posture classification. Moreover, the module performs outstanding real-time classification even on average powered devices.

Based on the module results, the feedback system automatically generates and displays real-time instructions and descriptive notes to assist users in modifying their yoga positions efficiently and safely.

2.5.2. Supported Yoga Pose

The current system of identifying yoga poses focuses on Yoga apps which provide observations on the eight main yoga poses as the foundational Point for application yoga pose classification models evaluation. These poses are common, accessible to multiple physiques, and essentially are the foundational options for evaluation. Recognizing variations of the poses to evaluate the difficulty to the users practice environment is key to discerning the model accurately.

(1) Mountain stance (Tadasana): This is the positioning sequence. The person stands upright with feet together. When the arms are at the sides and the shoulders are relaxed. The algorithm recognizes the position through the vertical body axis. The shoulders and the hip joints move less than 5° indicating a stable and cogent stance.

(2) Warrior I (Virabhadrasana I): In this position a practitioner bends one knee forward and subsequently straightens the back leg while raising both arms. The evaluation module searches for the angle between the front thigh and the lower leg completing the 90° plus the shoulder joint opening exceeding 160°. This demonstrates the stability and length of the upper body.

(3) Warrior II (Virabhadrasana II): This position is similar to Warrior I only here the arms are extended both to the sides and the shoulders are to remain level with the ground. The identification system uses the symmetry of the two arms, the horizontal alignment of the shoulder joints, and the appropriate angle of hip rotation to distinguish this posture from the other Warrior variations.

(4) Tree posture (Vrikshasana): This is a one-legged balance posture where the practitioner places one foot on the inner thigh of the opposite leg and places both hands either in front of the chest or above the head. The CNN model identifies this pose by analyzing knee and ankle joint coordinates that are not aligned on the same axis and by assessing the upper body's stability over time. This assists the system in differentiating this pose from other standing poses.

(5) Downward Dog (Adho Mukha Svanasana): The body takes the form of an upside down "V", with the arms and legs forming a sharp angle with each other. The model for recognition suggests that for this pose, the trunk and arms should form a 45 to 60 degree angle and the arms and shins should be parallel to the ground. This is a pose for bodily extension.

(6) Child's Pose (Balasana): This is a pose that helps one to relax and rest. The person kneels down, leans forward and touches the floor with their forehead, reaching their arms out in front of them. The geometric qualities are a hip flexion angle of less than 40° as well as knees that are completely bent and an upper body length that is substantially shorter than while standing.

(7) Cobra Pose (Bhujangasana): For this asana the practitioner lies on the stomach, and, using both hands to form an open back angle, raises the upper body. The algorithm recognizes an angle greater than 150 degrees between the trunk and legs where the shoulder joint is higher than the hips and there is an upward gaze. This assists the model in distinguishing between a plank and upward facing dog.

(8) Plank Pose (Phalakasana): This pose keeps the body parallel to the ground, with the weight distributed evenly between the arms and legs, in the exact stance used before performing push-ups.

The model observes an unbroken line from the shoulders to the heels with a hip joint angle of under 5 degrees and assesses the strength and stability of the core muscles in this position. These eight poses enable the system to cover different body types and build a comprehensive training dataset for the model. The recognition outcomes serve numerous functions, one being the assessment of a user's workout performance, as well as providing real-time audiovisual feedback on alignment maintenance, performance refinement, injury prevention, and proper alignment.

3. Results

International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 9 Issue 10 October - 2025, Pages: 152-162

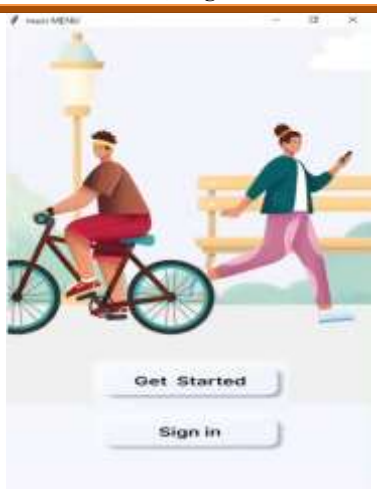

Figure 3. *The Web system's startup screen has options for "Get Started" and "Sign in." This shows that the design is easy to use and leads users to the activity.*

Figure 3 shows the welcome screen for the system. The interface is easy to use and displays visuals of two people moving—one cycling and the other walking—to represent the platform's fitness and active-living goals. There are two primary navigation buttons below: "Get Started" allows users to easily access the system, while "Sign In" lets them log into their account. This design is simple and intuitive, creating a positive first impression immediately.

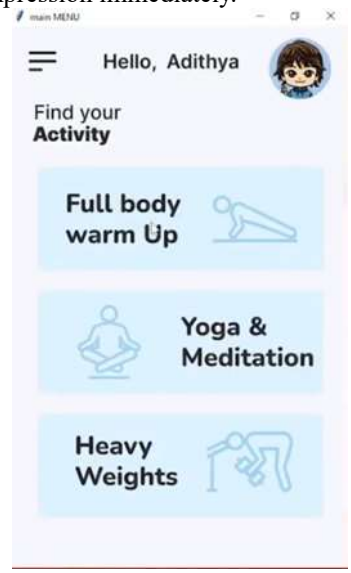

Figure 4. *Interface for choosing how the system works*

Figure 4 above shows the screen where users can choose activites in the Web system. When users log in, they get the message "Hello, Adithya" and an avatar, which makes them feel like they are special. The interface shows three main options: Full Body Warm-Up, Yoga & Meditation, and Heavy Weights. These are the three main types of exercise supported by the system. The flat design with light blue tones and illustrated symbols makes it easy for consumers to find and choose the activity they want. The interface is simple and easy to use, making it easy to understand on both PCs and mobile devices.

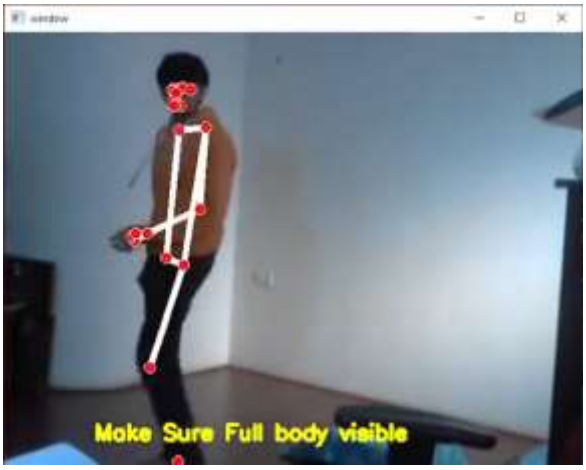

Figure 5. *The system's real-time posture monitoring module uses MediaPipe BlazePose to detect 33 landmarks on the user's body*

Figure 5 shows the Web system's real-time pose monitoring module. The MediaPipe BlazePose model captures and highlights 33 points on a person's body. The body landmarks are indicated by the red dots, and the skeletal structure is represented by the white lines. The model will function improperly if the body is not completely visible in the frame. The model instantly displays the skeleton and the user to provide visual feedback for self-correction. This helps in correcting the posture during the exercises.

To thoroughly test the Web system, a variety of datasets collected from numerous users were put under different conditions, such as changes in illumination, viewing angle, and camera distance. The evaluation method tries to ascertain the system's core modules' ability to process information in real time, the level of user satisfaction, and the overall system performance. The primary functions here are the counting of repetitions, recognizing the yoga positions, and multimedia responsiveness.

(1) Performance of the system: The system was able to demonstrate real-time processing of 30 frames per second (FPS) on a 10th generation Intel Core i5 processor, and this was achieved without any GPU acceleration. The system requires under 200 MB of RAM which means it can operate on the memory of most computers, and the standard internet browsers. The average time of 0.12 seconds to capture a frame and display a response is more than adequate for real-time interaction.

(2) Recognition of Yoga Poses: After being trained on 8 Yoga Poses using the MobileNetV2 CNN architecture, an overall accuracy of 89.7% was achieved. These 8 Yoga Poses included Tadasana (Mountain Pose), Virabhadrasana I (Warrior I), Virabhadrasana II (Warrior II), Vrikshasana (Tree Pose), Adho Mukha Svanasana (Downward Dog), Balasana (Child's Pose), Bhujangasana (Cobra Pose), and Phalakasana (Plank Pose). The only issue the model had was transitioning

Warrior II into Warrior I as the poses and joint angles are only slighty different.

(3) Repetition Count Accuracy: Using a Finite State Machine (FSM) to count completed movement cycles, it was able to achieve an average accuracy rate of 94.3% for the three basic exercises: push-ups, squats, and bicep curls.

Most of the mistakes occur during high-speed movements or when the joint angles of flexion and extension are below the detection threshold. As a result, incomplete movement cycles are recorded. I have addressed the issue of inaccurate count repetitions using hysteresis thresholds, temporal smoothing, and noise that comes from frame drops caused by exercises.

(4) User Feedback: Of 25 test users surveyed, 92% said the visual feedback (skeleton color and joint angle display) was helpful, 87% said the system helped with postural adjustments, and 78% liked the exercises made more enjoyable through gamification. People found the text feedback and cues easy to follow. Active users commented most on the feedback: "Lower your hips" and "Keep your back straighter" as these feedback statements were straightforward, specific, and actionable.

The pose monitoring module of our Web system is shown in Figure 5. The MediaPipe BlazePose model captures and highlights 33 points on a person's body. The body landmarks are indicated by the red dots, and the skeletal structure is represented by the white lines. The model will function improperly if the body is not completely visible in the frame. The model instantly displays the skeleton and the user to provide visual feedback for self-correction. This helps in correcting the posture during the exercises.

To thoroughly test the Web system, a variety of datasets collected from numerous users were put under different conditions, such as changes in illumination, viewing angle, and camera distance. The evaluation method tries to ascertain the system's core modules' ability to process information in real time, the level of user satisfaction, and the overall system performance. The primary functions here are the counting of repetitions, recognizing the yoga positions, and multimedia responsiveness.

(1) Performance of the system: The system was able to demonstrate real-time processing of 30 frames per second (FPS) on a 10th generation Intel Core i5 processor, and this was achieved without any GPU acceleration. The system requires under 200 MB of RAM which means it can operate on the memory of most computers, and the standard internet browsers. The average time of 0.12 seconds to capture a frame and display a response is more than adequate for real-time interaction.

(2) Recognition of Yoga Poses: After being trained on 8 Yoga Poses using the MobileNetV2 CNN architecture, an overall accuracy of 89.7% was achieved. These 8 Yoga Poses included Tadasana (Mountain Pose), Virabhadrasana I (Warrior I), Virabhadrasana II (Warrior II), Vrikshasana (Tree Pose), Adho Mukha Svanasana (Downward Dog), Balasana (Child's Pose), Bhujangasana (Cobra Pose), and Phalakasana (Plank Pose).

The only issue the model had was transitioning Warrior II into Warrior I as the poses and joint angles are only slightly different.

(3) Repetition Count Accuracy: Using a Finite State Machine (FSM) to count completed movement cycles, it was able to achieve an average accuracy rate of 94.3% for the three basic exercises: push-ups, squats, and bicep curls.

Most of the mistakes occur during high-speed movements or when the joint angles of flexion and extension are below the detection threshold. As a result, incomplete movement cycles are recorded. I have addressed the issue of inaccurate count repetitions using hysteresis thresholds, temporal smoothing, and noise that comes from frame drops caused by exercises.

(4) User Feedback: Of 25 test users surveyed, 92% said the visual feedback (skeleton color and joint angle display) was helpful, 87% said the system helped with postural adjustments, and 78% liked the exercises made more enjoyable through gamification. People found the text feedback and cues easy to follow.

Active users commented most on the feedback: "Lower your hips" and "Keep your back straighter" as these feedback statements were straightforward, specific, and actionable.

(5) Overall Evaluation: The Web system favorably balances speed, precision, and user friendliness, and no extra equipment is needed to receive a complete fitness coaching experience thanks to real-time computer vision and deep learning enabled through a web interface.

Its performance across varying testing conditions demonstrates its adaptability.

Examples of its use include customized training plans, virtual PE classes, and telehealth.

4. Discussion

The effectiveness of real-time fitness tracking systems with computer vision and lightweight deep-learning models has been proven in the experiments conducted. Because of the synergy of MobileNetV2 with MediaPipe, the web system can run on standard CPUs without requiring GPU hardware acceleration. Thus, the system can be used in a variety of settings, including many non-specialized situations, which is ideal for most users.

This website outperforms many more resource-intensive models like ResNet or VGG. The system can detect yoga poses with an accuracy of 89.7%, accurately count repetitions with 94.3% accuracy, and is able to perform all of this in real-time at 30 frames per second. The Finite State Machine (FSM) technique impressively minimizes the error and noise which occurs due to rapid movements and unstable frames.

Some limitations of the system remain. Situations in which light is very dim, the person is wearing loose clothing that obscures important joints, or the camera is at a very wide-angle lead to a drop-in accuracy. Currently, the system can only handle 8 basic yoga poses, and therefore, does not cover full-body complex movements or more advanced activities.

Many of the testers enjoyed its simplicity, clarity, and prompt feedback. This shows that the Web system has the potential to optimize one-on-one fitness coaching and positively affect the population's health.

In the future, the study team plans to make a system upgrade by integrating wearable sensors for enhanced 3D movement precision, incorporating real-time audio feedback, and creating a PWA for a more seamless experience on smartphones. One of the possible research avenues to reconfigure the system into a fully automated physical aid and rehabilitation center is the application of retrospective analysis on workout data to generate tailored health evaluations.

5. Conclusions

Within the current scope of the system's capabilities, it did relatively well in tracking yoga positions and recording workouts, although some issues remain to be resolved in future system iterations. For example, in low-light environments or when users drape loose clothing covers key joints, predictive capabilities of the system will be challenged, affecting the ability to compute the position and angle of the joints. Additionally, the current pose recognition system is limited to teaching and categorizing eight basic yoga positions. This limited capability means that users performing vigorous workouts or rehabilitation exercises may not receive the full range of assistive movements that the system offers. It will also be of note that the system currently does not provide spatially enhanced depth information and continues to operate with an RGB camera.

With these constraints, the study team wishes to improve the system in the following ways:

(1) Use wearable sensors to augment computer vision with inertial measurement unit (IMU) data to facilitate the tracking of fast or complex movements.

(2) Add advanced yoga, strength training exercises, and rehabilitation exercises to the system in order to expand the range of postures and movements.

(3) Incorporate a real-time voice feedback feature for posture correction so users can receive guidance while maintaining a focus away from the screen. This enhancement will further streamline the experience of fully automated training.

(4) Develop the service's mobile version following the design principles of a Progressive Web App (PWA) so users can more conveniently access the service on smartphones, tablets, and other devices with slow web access.

(5) Establish a module for personalized progress assessments based on historical training data that captures users' improvement trends, outlines their optimal training pathways, and provides pertinent health recommendations.

The Web will be more effective with these innovations in areas such as fitness and rehabilitation after injuries, as well as in community health care initiatives, powered remote training with AI, and integrated support.

References

[1] W. K. Hou, F. T. Lai, M. Ben-Ezra, and R. Goodwin (2020), "Regularizing daily routines for mental health during and after the COVID-19 pandemic," *Journal of Global Health*, vol. 10, no. 2, p. 020315.

[2] S. M. Nyenhuis, J. Greiwe, J. S. Zeiger, A. Nanda, and A. Cooke (2020), "Exercise and fitness in the age of social distancing during the COVID-19 pandemic," *J. Allergy Clin. Immunol.: In Practice*, vol. 8, no. 7, pp. 2152–2155.

[3] W. Styron (2010), *Darkness Visible: A Memoir of Madness*. Open Road Media.

[4] J. A. Knight (2012), "Physical inactivity: associated diseases and disorders," *Ann. Clin. Lab. Sci.*, vol. 42, no. 3, pp. 320–337.

[5] I. Vuori (2004), "Physical inactivity is a cause and physical activity is a remedy for major public health problems," *Kinesiology*, vol. 36, no. 2, pp. 123–153.

[6] World Health Organization (2021), *Impact of COVID-19 on Physical Activity and Fitness Behavior*, WHO Report.

[7] J. Smith and L. Brown (2021), "Home workout injuries during lockdown: A clinical survey," *J. Sports Med. Health Sci.*, vol. 9, no. 2, pp. 110–117.

[8] K. Thompson (2022), "Social support and exercise adherence in isolated environments," *Psychology of Sport and Exercise*, vol. 56, pp. 1–8, 2022.

[9] M. Chen and Y. Park (2022), "Health effects of increased screen time in remote work settings," *Int. J. Occup. Health*, vol. 48, no. 3, pp. 220–228.

[10] R. Patel (2022), "Limitations of static fitness applications and the need for real-time feedback," *IEEE Access*, vol. 10, pp. 12045–12053.

[11] V. Bazarevsky, I. Grishchenko, K. Raveendran, et al. (2020), "BlazePose: On-device real-time body pose tracking," *arXiv preprint arXiv:2006.10204*.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen (2018), "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 4510–4520.

[13] Y. Lu (2023), "Real-time eye blink detection using general cameras: a facial landmarks approach," *International Smart Grid and Energy Systems Journal*, Oklahoma State University.

[14] F. B. Ashraf, A. A. Zahid, S. M. A. Bhuiyan, et al. (2023), "YoNet: A neural network for yoga pose classification," *Sensors*, vol. 23, no. 5, p. 2217.

[15] A. Tharatipyakul, J. Kampachat, and S. Tanaka (2024), "Deep learning-based human body pose estimation in sport and physical exercise: A review," *Sensors*, vol. 24, no. 2, p. 683.

[16] B. Ferreira, J. P. Costeira, and A. Bernardino (2021), "Deep learning approaches for workout repetition counting and invalid repetition detection," *Pattern Recognition Letters*, vol. 152, pp. 42–49.

[17] F. Japhne and A. P. Vinod (2024), "FitCam: Detecting and counting repetitive exercises with deep learning," *Journal of Big Data*, vol. 11, no. 1, pp. 77–89.

[18] A. Abedi, S. Shojaeilangari, and K. N. Plataniotis (2023), "Rehabilitation exercise repetition segmentation and counting using skeletal body joints," *arXiv preprint arXiv:2304.07421*.
[19] C. Mercadal-Baudart, J. Rodriguez-Serrano, and X. Giro-i-Nieto (2024), "Exercise quantification from single-camera view markerless pose estimation," *Heliyon*, vol. 10, no. 2, e23458.
[20] H. J. Chae, S. Lee, and H. Kim (2023), "An artificial intelligence exercise coaching mobile app," *Healthcare (Basel)*, vol. 11, no. 4, p. 524.