

FitPlay: An Exercise-Gated Gaming System Using Real-Time Pose Detection for Behavioral Screen-Time Regulation

Calvin K J, Adithya Raj N
Abhishek Yadav, Aditya Mallikarjun
Department of Computer Science and
Engineering
East West Institute of Technology
Bengaluru, Karnataka, India

Prof. Sangeeta M R
Professor
Department of Computer Science and
Engineering
East West Institute of Technology
Bengaluru, Karnataka, India

Abstract—Prolonged and uninterrupted digital entertainment can contribute to sedentary behavior among students and young adults. Conventional screen-time management systems generally address this problem by restricting access to applications after a predefined usage limit, without requiring an alternative physical activity. This paper presents FitPlay, an exercise-gated gaming system in which verified physical activity acts as a prerequisite for obtaining additional gaming time.

FitPlay consists of two independently executing applications. A Python-based verification client captures webcam video and uses MediaPipe Pose landmark estimation to recognize squats, push-ups, jumping jacks, and in-place walking. Exercise-specific geometric features and finite-state logic are used to identify movement phases and count repetitions or accumulate verified activity duration. Following successful completion of an exercise target, the client writes a timestamped unlock record to a shared JSON session file using an atomic temporary-file-and-replacement operation. A Unity-based 2D platformer independently polls this file, validates the freshness of the unlock record, and controls game availability using engine-level pause and resume mechanisms.

The game was developed specifically as the reward environment and incorporates a dual-input projectile combat system, physics-based falling-rock and spike-reveal hazards, and a projectile-triggered chain-reaction platform puzzle. The implementation also addresses practical engineering challenges involving cross-platform path resolution, concurrent file access, Unity collision handling, animation timing, and exercise state-machine transitions. Anti-circumvention mechanisms were developed to address stale unlock records, premature termination of the verification application, and expired game sessions.

The resulting system demonstrates the feasibility of connecting consumer webcam-based exercise verification with a separately implemented game through a lightweight local inter-process communication mechanism. The current implementation is primarily a functional engineering prototype; quantitative recognition accuracy, latency, and large-scale behavioral evaluation remain areas for future work.

Index Terms—pose estimation, MediaPipe, exercise recognition, behavioral gating, exergaming, screen-time regulation, Unity, human-computer interaction

I. INTRODUCTION

The widespread availability of digital entertainment has increased the amount of time that users spend interacting with computers and mobile devices. Among students and young

adults, gaming can provide recreation and social interaction, but prolonged uninterrupted sessions can also reduce opportunities for physical movement. Research into digital self-control has examined several mechanisms for regulating technology use, including application blocking, website restriction, goal setting, and usage feedback [1], [2].

Although restriction-based tools can reduce access to distracting applications, they do not necessarily connect continued access with an alternative constructive behavior. A user whose allotted screen time has expired is generally presented with a restriction rather than an opportunity to earn additional access through physical activity.

A separate body of research has investigated exergaming, in which physical movement is incorporated into digital entertainment. Systematic reviews have examined the relationship between active video games, physical activity, exercise behavior, and user engagement [3], [4]. However, many dedicated exergaming systems depend on specialized controllers or motion-sensing hardware.

FitPlay explores a different interaction model. Rather than using exercise solely as a gameplay input, FitPlay uses verified exercise as a prerequisite for obtaining additional gaming time. A user performs a predefined exercise in front of a conventional webcam. Once the required activity has been recognized, the system grants a corresponding amount of game access. When that period expires, gameplay is locked and a new exercise cycle is required.

The system uses computer vision rather than dedicated fitness hardware. MediaPipe Pose provides real-time human body landmark estimation from RGB video and exposes 33 body landmarks that can be used to derive geometric movement features [5], [6]. Previous work has demonstrated the use of pose estimation for exercise posture analysis using conventional cameras [7].

A central design decision in FitPlay was to implement the reward environment as an original 2D platformer instead of simply unlocking an existing third-party application. This required the development of player movement, projectile combat, physics-based hazards, animation systems, environmental interactions, and puzzle mechanics. Consequently, the system

represents a complete feedback loop between physical activity verification, session management, and interactive entertainment.

The principal contributions of this work are:

- A software-only architecture for exercise-gated gaming using a conventional webcam and real-time pose estimation.
- Exercise-specific geometric and finite-state verification methods for squats, push-ups, jumping jacks, and in-place walking.
- A lightweight inter-process communication mechanism based on an atomically updated local JSON session file.
- Timestamp-based session validation and redundant locking mechanisms designed to resist specific bypass scenarios observed during development.
- An original Unity 2D platformer containing dual-input projectile combat, physics-based environmental hazards, and a projectile-triggered chain-reaction platform puzzle.
- Documentation of implementation defects and their resolutions across the computer-vision, synchronization, and game systems.

The remainder of this paper discusses related work, system architecture, exercise verification, session synchronization, anti-circumvention mechanisms, game implementation, evaluation, limitations, and future work.

II. RELATED WORK

A. Digital Self-Control

Digital self-control tools have been investigated as mechanisms for helping users regulate their interaction with digital technologies. Lyngs *et al.* examined a large collection of digital self-control tools and identified multiple intervention mechanisms including blocking, restriction, goal setting, and related strategies [1].

A systematic review by Biedermann *et al.* examined digital self-control interventions designed to reduce distracting media multitasking and identified different approaches for changing digital media behavior [2]. These studies provide the behavioral background for FitPlay, but FitPlay changes the intervention model by making continued game access conditional on a verified physical activity.

FitPlay is not intended to replace conventional parental-control or digital-wellbeing tools. Instead, it investigates an alternative model in which restriction is combined with a behavioral reward mechanism.

B. Pose-Based Exercise Recognition

Human pose estimation provides a practical method for recognizing body movement without requiring wearable sensors or dedicated motion controllers. MediaPipe Pose is capable of estimating body landmarks from RGB video and is designed for real-time body-pose tracking [5].

BlazePose provides the research foundation for real-time body-pose tracking used by the MediaPipe ecosystem [6]. Chen and Yang demonstrated the use of pose estimation for exercise posture correction in their Pose Trainer system [7].

FitPlay uses a related landmark-based geometric approach, but its objective is different. Instead of primarily evaluating exercise form, the system determines whether sufficient exercise activity has occurred to authorize game access.

C. Exergaming

Exergaming combines physical movement and interactive digital entertainment. Street *et al.* reviewed exergaming programs designed to increase health and exercise behaviors among adults [3]. More recent systematic work has investigated the effect of active video games on physical activity and related outcomes [4].

Research specifically involving college populations has also investigated active video games as a mechanism for increasing physical activity [8]. These studies establish the relevance of combining physical activity and interactive entertainment.

FitPlay differs from conventional exergaming in the location of the exercise requirement within the interaction loop. Exercise is performed before access to the game is granted rather than being necessarily used as the primary gameplay control mechanism.

III. SYSTEM ARCHITECTURE

FitPlay consists of two independently executing applications: a Python-based exercise-verification client and a Unity-based 2D game client. The applications communicate through a shared JSON session file rather than a network connection.

As shown in Fig. 1, the system is divided into three logical components: the Python verification client, the shared session-file bridge, and the Unity game client.

A. Exercise Verification Client

The Python verification client performs the following sequence:

- 1) initializes the local session database;
- 2) captures webcam frames;
- 3) detects body landmarks;
- 4) evaluates the selected exercise;
- 5) counts repetitions or accumulates verified duration;
- 6) calculates the earned gaming time;
- 7) writes an unlock record to the session file; and
- 8) launches or resumes the Unity game.

OpenCV is used for webcam acquisition while MediaPipe Pose is used for body-landmark estimation.

B. Unity Game Client

The Unity application periodically reads the shared session file. When a valid unlock record is detected, the game enters the playable state and starts the reward timer.

When the timer expires, the Unity client writes a locked state, disables gameplay input, pauses game simulation, and initiates a new exercise-verification cycle.

Unity's engine-level time scaling is used to pause the game instead of terminating the executable. This preserves the current scene and player state.

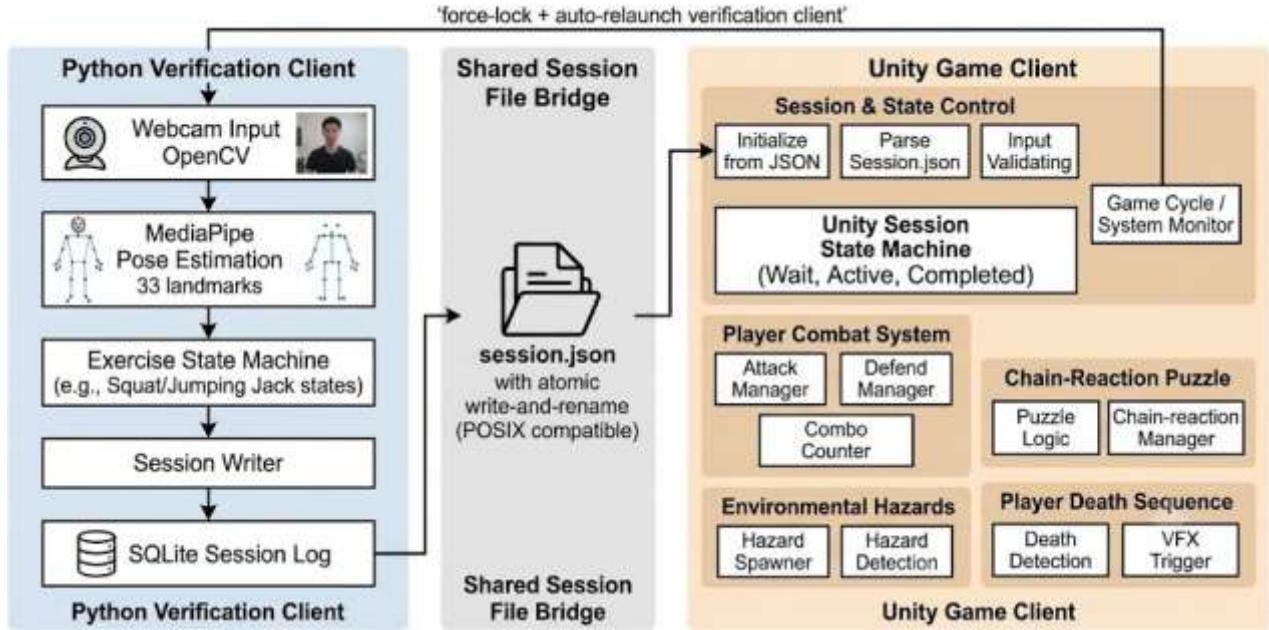


Fig. 1. Overall architecture of FitPlay showing the Python exercise verification client, shared JSON session-file bridge, and Unity game client. The Python client performs webcam-based pose estimation and exercise verification, while the Unity client validates session state and controls gameplay.

C. Session File

The session file stores the current state of the system and the timestamp associated with the state transition. A simplified representation is:

```
{
  "status": "unlocked",
  "minutes": 5,
  "timestamp": 1727580000
}
```

The actual implementation contains the additional information required by the application.

IV. POSE DETECTION AND EXERCISE VERIFICATION

The FitPlay verification pipeline converts webcam frames into body landmarks and subsequently evaluates exercise-specific geometric conditions.

As illustrated in Fig. 2, the detected body landmarks provide the geometric information required for exercise verification.

A. Pose Landmark Processing

For each webcam frame, MediaPipe Pose produces body landmarks represented using normalized image coordinates and visibility information [5]. Relevant landmarks are selected according to the exercise being performed.

For angle-based exercises, three landmarks define two vectors. For landmarks A , B , and C , where B is the target joint, the angle is calculated as

$$\vartheta = \cos^{-1} \left| \frac{(A - B) \cdot (C - B)}{\|A - B\| \|C - B\|} \right| \quad (1)$$



Fig. 2. Real-time pose estimation during exercise verification. Detected body landmarks are used to calculate joint geometry for exercise recognition.

The calculated angle is converted to degrees before comparison with exercise-specific thresholds.

B. Squat Detection

Squats are evaluated using the hip-knee-ankle relationship. The verification system uses two movement states: *UP*, when the knee angle is above the upper threshold, and *DOWN*, when the knee angle is below the lower threshold.

A repetition is registered only after a valid movement transition.

During calibration, the measured standing angle was approximately 178.4° while the measured full-squat angle was approximately 62.9° . Operational thresholds of greater than 160° for the upper state and less than 100° for the lower state were used to provide tolerance against measurement variation.

C. Push-Up Detection

Push-ups use the shoulder-elbow-wrist relationship. The system uses approximately

$$\vartheta_{up} > 155^\circ \quad (2)$$

and

$$\vartheta_{down} < 90^\circ. \quad (3)$$

A repetition is accepted only after the required state transition sequence.

Fig. ?? provides execution-level evidence of a successful push-up verification cycle.

D. Jumping Jack Detection

Jumping jacks require simultaneous movement of the arms and legs. Using only one joint measurement can allow isolated limb movements to produce false positives.

The implementation therefore combines wrist position relative to shoulder position and ankle separation relative to hip separation. A repetition is registered only when both upper-body and lower-body conditions satisfy the open configuration following a closed state.

E. Walking Verification

Walking is treated differently from angle-based exercises because a stationary webcam cannot directly measure the distance traveled through physical space.

FitPlay instead verifies in-place walking using ankle displacement between consecutive frames. If p_t and p_{t-1} represent an ankle position in consecutive frames, displacement is calculated as

$$d_t = |p_t - p_{t-1}|. \quad (4)$$

A rolling average of ankle displacement is compared with a calibrated movement threshold.

Verified activity time advances only when a person is detected, an upright posture condition is satisfied, and sufficient ankle movement is detected. Therefore, the implementation verifies sustained marching or pacing inside the camera field of view rather than literal walking distance.

V. FITPLAY USER INTERFACE

The Python application provides a graphical interface for selecting exercises and monitoring accumulated gaming time.

The interface shown in Fig. 3 displays the total gaming time earned during the day and provides access to the supported exercise activities: squats, jumping jacks, push-ups, and walking.

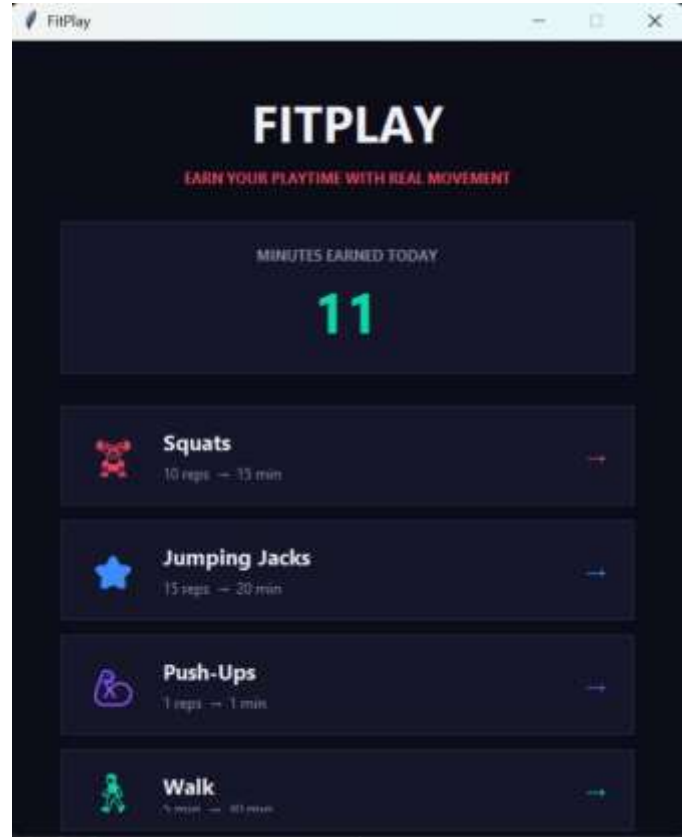


Fig. 3. FitPlay exercise-selection interface displaying earned gaming time and available exercise activities.

VI. INTER-PROCESS COMMUNICATION AND SESSION CONTROL

A. Atomic Session Writes

Concurrent access to the shared session file creates a race condition if one application reads the file while the other is writing it. A direct write could temporarily expose incomplete JSON.

FitPlay therefore uses a temporary-file-and-replacement strategy:

- 1) create a temporary file;
- 2) write the complete JSON record;
- 3) flush the file;
- 4) synchronize the contents; and
- 5) replace the target file.

This approach prevents the Unity reader from observing a partially written JSON document during normal concurrent operation.

B. Session Freshness

A previously valid unlock record should not remain usable indefinitely. Each unlock record therefore contains a timestamp.

The time difference is

$$\Delta t = |t_{\text{current}} - t_{\text{record}}|. \quad (5)$$



Fig. 4. Locked game state displayed after the earned gaming period has expired. The user must complete another exercise session to obtain additional gameplay time.

An unlock is accepted only when

$$\Delta t \leq T_{\text{fresh}}$$
 (6)

where the implementation uses a freshness window of approximately 60 seconds.

Records outside this interval are rejected and the game is returned to the locked state.

As shown in Fig. 4, the game communicates that continued gameplay requires another verified exercise session.

VII. ANTI-CIRCUMVENTION DESIGN

An exercise-gated system introduces security considerations that are not present in a simple application timer. The game depends on a signal generated by another local process, and therefore the system must handle invalid or stale state.

A. Verification Application Termination

A user could close the verification application after leaving the game in a paused state. The verification client therefore resets the session to the locked state when it terminates.

The Unity application additionally disables gameplay input while the game is locked.

B. Replay of an Old Unlock

A previously valid unlock record could otherwise be reused. The timestamp freshness mechanism prevents old records from remaining valid indefinitely.

C. Timer Expiration

When the reward timer expires, the Unity application itself writes a locked record. This results in redundant enforcement:

Exercise → Unlock → Gameplay → Game-Enforced Lock. (7)

These mechanisms should be understood as resistance against the specific bypass scenarios tested during development rather than as complete operating-system-level security. A user with unrestricted control of the host computer could potentially modify files, executables, or processes.



Fig. 5. Gameplay view of the custom Unity 2D platformer developed as the reward environment for FitPlay.

VIII. GAME DESIGN AND IMPLEMENTATION

The reward environment is a custom 2D platformer implemented in Unity.

The game environment shown in Fig. 5 provides the interactive reward following successful exercise verification.

A. Dual-Input Projectile Combat

The combat system contains two independent projectile actions. The primary action launches a horizontal projectile according to the player's facing direction.

A second action controls an upward projectile. When both actions are held simultaneously, the upward attack is selected.

The implementation tracks the two input states independently and evaluates their combination before selecting the projectile type.

B. Falling-Rock Trap

The game contains physics-based falling-rock traps. Initially, the rocks are held in a kinematic state. When the player enters the trigger region, the rocks are released and gravity is enabled.

After collision with the ground or player, the rock stops physical movement, disables its collider, triggers the shatter animation, and destroys itself after the animation finishes.

C. Spike-Reveal Hazard

A separate hazard uses two co-located GameObjects: a permanent spike hazard and an animated rock cover.

The spike remains physically active while the rock cover is rendered above it. After activation, the rock plays its crumbling animation and is subsequently disabled.

D. Projectile-Triggered Chain-Reaction Puzzle

The game includes an environmental puzzle based on a projectile weak-point interaction.

The player must fire a projectile at an invisible trigger volume. When a valid projectile hits the trigger, an associated physics-based block is released.



Fig. 6. Platforming environment containing lava hazards and traversable platforms used in the environmental puzzle system.

After landing, the block stops moving, becomes a stable platform, removes the associated lava tile, and optionally releases the next block.

The lava-based environment shown in Fig. 6 provides the traversal context for the environmental puzzle.

E. Player Death Sequence

The player death system was revised from an earlier physics-based death impulse to an instantaneous visual effect.

The final sequence disables the player renderer, instantiates a smoke-puff effect at the player's last position, and freezes the player's rigidbody.

IX. IMPLEMENTATION CHALLENGES AND RESOLUTIONS

Several implementation defects were identified during development.

A. Exercise State Machine

The original repetition-counting implementation evaluated opposing state conditions independently. This could cause one movement state to overwrite another during the same frame.

The logic was changed to a mutually exclusive transition structure so that only the intended state transition occurs during each evaluation.

B. Cross-Platform Path Resolution

Unity's application data path differs between the Unity Editor and compiled builds. A fixed number of parent-directory operations was therefore unreliable when the project was moved.

The final implementation searches upward from the application's execution directory until the expected shared directory is located.

C. Collision Detection

A falling-rock configuration initially used a collider with its trigger property enabled. The destruction script expected normal collision callbacks, so the required callback was not generated.

The issue was resolved by disabling the trigger property on the rock's physical collider.

D. Camera Boundary Collision

A polygon collider used for camera confinement unintentionally interacted with falling rocks.

Unity's 2D layer collision matrix was used to disable physical interaction between the rock layer and camera-boundary layer.

E. Animation Pivot Alignment

The shatter animation initially exhibited unintended lateral movement because individual spritesheet frames had inconsistent pivot positions.

The sprites were re-sliced using a consistent pivot alignment.

F. Atomic Locking

The original implementation protected the unlock operation with an atomic write but directly wrote the lock state.

The locking operation was changed to use the same atomic helper as the unlock operation.

G. Blocking User Interface

A modal success dialog interrupted the automatic transition from exercise completion to game launch.

The success path was changed to a non-blocking status update while blocking dialogs were retained for genuine errors.

X. EXPERIMENTAL EVALUATION

The current evaluation focuses on functional verification of the complete system and its individual components. It does not claim statistical validation across a large population.

A. Functional Exercise Testing

Each exercise pathway was tested repeatedly after calibration and correction of the repetition state machine. Squat and push-up recognition produced movement transitions consistent with manual observation. The jumping-jack implementation was tested against isolated upper-body and lower-body movements to verify that both body regions were required.

The walking detector was tested using in-place stepping within the webcam field of view.

B. End-to-End Session Testing

The complete workflow was evaluated as

$$\begin{aligned} &\text{Exercise} \rightarrow \text{Verification} \rightarrow \text{Unlock} \rightarrow \text{Gameplay} \\ &\quad \rightarrow \text{Timer Expiry} \rightarrow \text{Lock.} \end{aligned} \quad (8)$$

C. Anti-Circumvention Testing

The following scenarios were tested:

TABLE I
FUNCTIONAL ANTI-CIRCUMVENTION TESTS

Scenario	Expected behavior
Verification application closed	Session returns to locked state.
Stale unlock record	Record rejected after freshness interval.
Game timer expires	Game writes locked state and stops gameplay.
Paused game remains open	Input remains disabled while locked.

D. Game Functional Testing

The following game systems were tested:

- directional projectile attack;
- simultaneous-input upward projectile;
- falling-rock trap;
- spike-reveal hazard;
- projectile weak-point interaction;
- chained falling platforms;
- lava removal;
- player death effect; and
- game pause and resume.

E. Quantitative Evaluation

The current implementation requires additional systematic testing before quantitative performance claims can be made.

TABLE II
EXERCISE RECOGNITION AND MOTION-GATE EVALUATION

Exercise/Test	Test Protocol	Correctly Recognized	Accuracy
Squats	20 repetitions	17/20	85%
Push-ups	20 repetitions	15/20	75%
Jumping Jacks	20 repetitions	19/20	95%
Walking	60-second test	30/60 valid	50%
Overall repetition accuracy	60 repetitions	51/60	85.0%

Table II summarizes the observed exercise recognition performance. Jumping jacks achieved the highest observed recognition accuracy at 95%, followed by squats at 85% and push-ups at 75%. Across the three repetition-based exercises, 51 of 60 repetitions were correctly recognized, corresponding to an observed accuracy of 85.0%.

The walking test revealed a significant limitation in the current motion-verification implementation. During a 60-second test, 30 seconds of standing still were incorrectly credited as valid walking activity. This corresponds to a 50% false-positive rate for the specific motion-gate test. The result indicates that the current ankle-displacement threshold and temporal smoothing mechanism are insufficient for reliably distinguishing sustained walking from stationary posture under the tested conditions.

These results are based on the conducted test cases and should not be interpreted as population-level accuracy esti-

mates. A larger test set with multiple participants, camera positions, lighting conditions, and deliberately performed incorrect movements would be required for a statistically robust evaluation.

XI. DISCUSSION

The implementation demonstrates the feasibility of connecting webcam-based exercise verification with an independent game application without requiring dedicated motion-tracking hardware or a network server.

The file-based communication mechanism is appropriate for a local student-scale system because it minimizes architectural complexity. However, the same design creates an important security limitation: the session file exists on the user's own computer. Timestamp freshness prevents straightforward replay of old records, but it does not provide cryptographic integrity.

The exercise-verification approach is also sensitive to camera placement, lighting, body visibility, and user movement characteristics. Fixed thresholds may therefore behave differently for different users.

Walking presents a separate limitation. The current implementation detects repeated lower-limb movement within the camera's field of view rather than measuring actual distance traveled. Therefore, the system should be described as verifying in-place walking or marching.

The game component demonstrates that the reward can be a substantive interactive environment rather than a placeholder application. However, the current project does not establish that the system produces sustained changes in physical activity or screen-time behavior. Such conclusions require a controlled longitudinal study.

XII. LIMITATIONS

Several limitations should be considered.

First, the current evaluation is primarily functional and was conducted within the scope of an academic software project. A larger participant study is required to evaluate generalization across different users.

Second, exercise recognition depends on camera placement and body visibility. Severe occlusion, poor lighting, unusual camera angles, or multiple people in the frame may reduce reliability.

Third, the thresholds were calibrated empirically rather than learned from a large exercise dataset. This limits their generalization.

Fourth, the walking system does not measure actual walking distance.

Fifth, the local JSON session file does not provide cryptographic protection. The anti-circumvention mechanisms address the specific bypass scenarios tested during development and should not be interpreted as protection against a determined user with full operating-system control.

Finally, the project does not currently establish long-term behavioral effectiveness. Such a claim would require controlled longitudinal evaluation.

XIII. FUTURE WORK

Several extensions can strengthen FitPlay.

An automated first-use calibration module could estimate individual movement ranges and derive exercise thresholds automatically. This would reduce dependence on manually selected constants.

A larger exercise dataset could support machine-learning-based classification in addition to the current geometric state machines. Temporal features could also improve discrimination between complete and partial repetitions.

The walking detector could be extended using gait-cycle analysis, temporal pose features, or depth information.

A stronger session-security model could use cryptographic authentication or operating-system-level protection rather than relying solely on local timestamp validation.

The game could be expanded with additional levels, exercise-based reward scaling, additional combat abilities, and more environmental puzzles.

Most importantly, a controlled user study could evaluate whether exercise-gated gaming affects physical activity, screen-time behavior, perceived motivation, and long-term adherence.

XIV. CONCLUSION

This paper presented FitPlay, an exercise-gated gaming system that connects real-time webcam pose estimation with controlled access to an independently developed 2D platformer.

The system combines MediaPipe-based body landmark detection, exercise-specific geometric analysis, finite-state repetition recognition, atomically written session-state communication, timestamp validation, and Unity-based pause/resume control. Four activity pathways were implemented: squats, push-ups, jumping jacks, and in-place walking.

Unlike a conventional screen-time restriction mechanism, FitPlay makes verified physical activity part of the access condition for subsequent gameplay. The project also implements an original game environment containing dual-input projectile combat, physics-based hazards, and a projectile-triggered chain-reaction platform puzzle.

Development revealed practical challenges involving state-machine design, cross-platform path resolution, concurrent file access, Unity collision callbacks, animation pivots, and user-interface blocking. Resolving these issues resulted in a functional end-to-end prototype.

The current results establish technical feasibility rather than clinical or behavioral effectiveness. Quantitative recognition accuracy, false-positive rates, latency measurements, and longitudinal behavioral outcomes remain important areas for future evaluation.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Sangeeta M R, Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, for her guidance, technical feedback, and support throughout the development of this project.

REFERENCES

- [1] U. Lyngs, K. Lukoff, P. Slovak, R. Binns, A. Slack, M. Inzlicht, M. V. Kleek, and N. Shadbolt, "Self-control in cyberspace: Applying dual systems theory to a review of digital self-control tools," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.
- [2] A. Biedermann *et al.*, "Digital self-control interventions for distracting media multitasking – a systematic review," *Journal of Computer Assisted Learning*, vol. 37, no. 5, pp. 1217–1231, 2021.
- [3] T. D. Street, S. J. Lacey, and R. R. Langdon, "Gaming your way to health: A systematic review of exergaming programs to increase health and exercise behaviors in adults," *Games for Health Journal*, vol. 6, no. 3, 2017.
- [4] J. C. Davis *et al.*, "Exergaming for physical activity: A systematic review," *Journal of American College Health*, vol. 72, no. 7, pp. 2090–2098, 2024.
- [5] Google, "Mediapipe pose," MediaPipe Solutions Documentation, 2023. [Online]. Available: <https://developers.google.com/mediapipe>
- [6] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *arXiv preprint arXiv:2006.10204*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.10204>
- [7] S. Chen and R. R. Yang, "Pose trainer: Correcting exercise posture using pose estimation," *arXiv preprint arXiv:2006.11718*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.11718>
- [8] Y. Zhao, K. G. Soh, H. A. Saad, C. Liu, and C. Ding, "Effects of active video games on physical activity among overweight and obese college students: A systematic review," *Frontiers in Public Health*, vol. 12, 2024.