Discifit: On-Device Automated Exercise Repetition Counting for Gamified Mobile Fitness

Prof. Imteyaz Shahzad[1], Amaan Sheikh[2], Waquas Khan[3], Mhd. Hamza[4], Sheikh Kabir[5], Raymeen Pathan[6]
[1,2,3,4,5,6]*Dept. of Computer Science & Engineering, Anjuman College of Engineering & Technology, Nagpur, India*

***Abstract*—Not many people move much these days. At the same time, eyes stay glued to screens more than ever. One answer comes in the form of an app called Discifit - it ties phone fun to real-world movement. Play only follows sweat. A built-in system tracks workouts using just the phone's camera. No extra gear needed. Nothing sent online. All processing happens right where it lands. Motion begins with basic body tracking tech - think BlazePose or MoveNet. That data flows into a BiLSTM model trained to spot timing patterns in motion. Joint angles get measured through geometry rules when possible. If visuals blur or lag hits, optical flow steps in quietly. Everything runs under one steady logic frame - like a silent referee watching reps. From earlier studies using camera and motion sensors for AERC, the idea for combining both approaches grow naturally. Built for Android and iOS, the method takes shape through practical steps laid out clearly here. Testing follows a set structure meant to keep results consistent. Speed versus size versus precision - each choice tips the balance differently when recognizing counts. Past work points strongly toward aiming higher than 95% correct counts as a solid goal.**

Counting workout reps automatically finds motion patterns through body positioning tech. Instead of manual tracking, movement between frames helps detect each rep using pixel shifts. Body joint data feeds into a two-way memory network that learns timing in exercises. Smartphones handle calculations locally so user privacy stays intact during analysis. While screens often distract, they can also encourage activity when used thoughtfully. Games layered onto fitness routines shift attention from duration to engagement. Training models directly on gadgets avoid constant internet needs for feedback loops.

***Index Terms*—automated exercise repetition counting, pose estimation, BiLSTM, optical flow, mobile health, screen-time management, gamification, on-device machine learning.**

I. INTRODUCTION

Spending too much time glued to phones often starts with just checking updates. Yet one glance pulls users into endless loops of grim headlines and tense videos. Feeds shaped by silent algorithms keep eyes locked longer than intended. Many people now log three or even four hours daily swiping through posts without purpose. This habit grows each year despite growing awareness. Sitting motionless for long stretches reshapes how bodies function over time. Sleep patterns twist out of rhythm when screens glow late into night. Minds feel heavier, nerves tighter, after marathon sessions online. Focus thins like worn fabric, harder to gather once scattered. What feels harmless slowly chips away at balance.

Right now, too many people around the world stay still for far too long. Sitting all day - stuck behind screens, riding buses, or lounging while staring at phones - burns fewer calories, slows down metabolism, yet opens doors to weight gain, heart issues, even joint decay over time. These aren’t separate struggles; the gadget pulling us into motionless hours could also nudge movement instead, provided someone builds it with purpose.

Midway through a workout, your phone watches you. Each movement tracked without internet keeps the game going later. Instead of swiping screens mindlessly, effort unlocks time. That shift runs on a system mixing old ideas with new tricks - camera sees reps, software checks form, feedback locks in counts. Research already points this way, yet gaps remain when signals drop or light fades. Testing must push every edge: shaky hands, dim rooms, rushed squats. Proof lies not just in code but how well it holds up when life gets messy.

II. PROBLEM STATEMENT AND RESEARCH OBJECTIVES

2.1 Core Problem

When tired, people often miscount reps. Focusing on screen cues makes it worse. One option relies on wearables - accurate yet clunky because of extra gear. Another uses cloud-based video analysis; this creates lag and needs constant internet. Privacy concerns pop up too. What works now does not fit what Discifit aims for - a smooth, ready-anytime app experience.

2.2 Research Objectives

1. Conduct a structured review of prior AERC and mobile-health literature to identify methodological strengths and limitations.
2. A mix of pose tracking feeds into the system first. After that, time-based predictions sort actions using past frames. Geometry rules step in when shapes need checking. If movement data gets messy, flow logic takes over instead. All parts connect smoothly inside one working design.
3. Start by setting up the system on both Android and iOS, relying on built-in tools made for each. On one hand, TensorFlow Lite handles the job smoothly for Android devices. Meanwhile, Apple's Core ML does the heavy lifting neatly on iPhones and iPads. Each platform runs the model through its own engine, tuned directly into the operating system. This approach keeps things efficient without extra layers getting in the way.
4. Start by setting up a test method that checks how close counts are to correct values. Instead of strict correctness, allow scores to pass if they're just one number too high or low. Track average mistakes across every frame using absolute differences. At the same time, measure delay for each individual frame during processing. Power used through the entire run gets recorded alongside timing data. Wrap these five measures together into the main scoring approach.
5. Start by looking closely at where speed meets limits. Then, think about what slows things down when features grow. Picture how changes might play out over time instead of rushing fixes now. Build steps forward based on real hiccups, not guesses. Move ahead only after weighing each compromise carefully.

2.3 Scope and Constraints

- Target platform: consumer smartphones without specialised sensors or attachments.
- Start with squats; these build leg strength over time. Push-ups come next - solid for chest and arms when done right. Pull-ups follow, working the back through full motion. Other moves fit here too, if needed later on.
- Running right there inside your phone, every bit of machine learning stays put. Nothing that can identify you ever exits the device. Your details remain exactly where they started - on the hardware itself.
- When emergencies happen, phone access stays live without fail. Critical alerts keep moving even if systems struggle. Safety calls push through every time, no matter what else is happening.
- Most adults should be able to handle the physical demands of exercise without risk. A reasonable level of effort keeps things both doable and secure for many. Staying within safe limits means fewer injuries over time. The goal is activity that challenges gently yet stays out of danger zones. Push too hard, safety slips away - stay steady, it holds firm.

III. LITERATURE REVIEW

3.1 Vision-Based On-Device AERC

Pūioio came from Sinclair, Kautai, and Shahamiri [1][2], standing as the closest match so far. Running only on a phone's front camera, it never reaches out to remote servers. Every image goes through five steps one after another: first extracting keypoints using BlazePose, then checking joint angles to spot movement shifts. Motion flow helps judge overall direction, while a state-driven logic links movements into full cycles. A number count appears once reps are confirmed. With two people tested under strict conditions over 450 actual attempts, it counted right nearly every time - just shy of 98.89%. Yet on a different set of 670 recorded tries, results dipped near 86%, mainly due to blocked views and shifting camera positions. Those flaws revealed how shaky the motion-from-frame-to-frame method really is, especially when history between distant frames gets ignored. Interestingly, its creators meant the motion

flow part to step in quietly whenever pose detection slipped - a choice later reused and built upon here.

What if high-end gear lets pose models hit their limits? Alatiah and Chen [7] showed just that. Using OpenPose, they pulled keypoints from workouts caught on several cameras. After that, a convolutional network labeled each exercise and tallied reps. Nearly perfect classification popped up - 99.96%. Most lifting rounds were counted right within one rep, hitting the mark 91% of the time. Yet, needing bulky GPUs and many anchored cameras kills phone use. Still, skeletal sequences can clearly carry rich timing clues when learned well.

One step ahead, Shuaib and team [4] tested a different approach - using motion sensors built into two Android watches, one on the wrist, another on the ankle. Instead of combining signals, they fed movement patterns directly into a neural network after recording 54 people doing nine CrossFit moves. Results showed nearly perfect detection, hitting 99.96% correct labels while staying within plus-or-minus one rep for 91% of exercise rounds. Even so, needing two gadgets and heavy computer processing later means it won't run live on phones alone. Still, what stands out is how well timing-based models adapt to various physiques - if shown enough variety during learning.

Riccio [3][8] stands out when it comes to architectural relevance in earlier efforts. Instead of stacking methods, they fed 30-frame sequences of normalised joint angles - stripped of scale and orientation influence - into a BiLSTM, hitting over 99% accuracy on four workouts. Their setup was built with mobile use in mind, leaning on a combination of BlazePose and BiLSTM [9]. By generating synthetic data, they avoided massive real-world recording sessions; however, minimal testing of delay directly on devices means how smoothly it runs on average phones remains unclear.

Though it needs the phone strapped to the forearm, which limits everyday use, Mühlbauer and team's MiLift setup still stands out. Their method uses basic signal processing - autocorrelation on IMU signals - to count reps during ten upper-body workouts. Accuracy hits about 93.6%, tested across 69 people. That number sets a solid benchmark, one reached without any machine learning at all.

3.2 Digital Wellbeing and Mobile Health Context

Balance in using phones does not stay fixed, Vanden Abeele notes - it shifts. Tools meant to help us work or connect can also pull attention away. One gadget holds both workout targets and tempting apps. This mix sparks inner push and pull. The struggle gets named motivational conflict. Screen time locks step in, tied to movement. Physical effort opens doors otherwise closed.

One study led by Biedermann and team [15] showed that tools like app blockers, screen time trackers, or reminder alerts can cut down on digital distractions - for a while at least. While results fade over months, adding game-like features might help keep people involved after the first few weeks wear off.

One study by Tsourgiannis and team [16] tested a game-style phone app meant to support students' mental wellness. It showed clear drops in worry levels, along with better self-reported mood, once custom objectives, device sensors, and feedback graphs worked together. The approach shaped how Discifit built its interface - using brief activity goals, on-screen count trackers, daily check-in chains, and milestone icons to tap into similar drivers of motivation seen earlier. That project, called SSResilience, became a quiet blueprint beneath these choices.

Table 1 — Summary of Related AERC and Mobile Wellness Systems

Table 1: Comparative overview of automated exercise repetition counting methods and relevant mobile health system

Study / Year	Method	Device / Scenario	Exercises / Dataset	Accuracy	Key Limitations
Sinclair et al. (2023)	BlazePose + angle threshold + optical flow + FSM	Smartphone front camera	Squats, push-ups, pull-ups (450 reps)	Real-world: 98.89%; pre-recorded: 86.0%	Needs full-body view; no temporal ML; sensitive to occlusion

Alatiah & Chen (2020)	OpenPose keypoints + CNN classification & counting	Multi-camera PC (weightlifting/CrossFit)	4 exercises (snatch, squat, clean, press)	Classification: 99.96%; ±1-rep: 91%	Not mobile; requires GPU; offline processing
Shuaib et al. (2019)	CNN on smartwatch IMU data	Two Android smartwatches (wrist + ankle)	9 CrossFit exercises; 54 participants	Recognition: 99.96%; ±1-rep: 91%	Requires wearable sensors; offline GPU training
Mühlbauer et al. (2018) — MiLift	Autocorrelation + IMU feature extraction	Phone strapped to arm	10 upper-body exercises; 69 subjects	Classification: 93.6%; ±1-rep: 93%	Requires arm holster; vision-free (no form check)
Riccio (2024)	BiLSTM on pose angle sequences	Front camera (web/mobile demo)	4 exercises (squat, push-up, shoulder press, bicep curl)	Classification: >99%	Tested on synthetic data; not fully validated on-device
This Work (2026) — Discifit AERC	Hybrid: pose estimation + BiLSTM + angle FSM + optical flow	Smartphone front camera (Android & iOS)	Squats, push-ups, pull-ups (extendable)	Target: >95% overall accuracy	Needs adequate lighting; requires training data for generalization

IV. PROPOSED HYBRID AERC ARCHITECTURE

Picture shows the setup in five steps, linked together to work at once. One-piece feeds into the next, using different kinds of sensors plus smart rules. If body positioning isn't clear, backup methods step in without breaking rhythm. Angle checks and movement tracking run parallel, feeding results separately. Their outputs meet inside a control unit that decides the count. Even if one-part stumbles, others keep score reliably. Design allows hiccups without stopping the flow. Each layer knows when to trust itself - or look elsewhere.

Figure 1 lays out how pieces connect across stages. Low certainty triggers alternate paths automatically. Motion patterns help confirm what angles alone might miss. Together, they form a steady stream of decisions. No single point kills progress if it hesitates. Signals merge only after passing individual tests. System keeps moving even under shaky inputs.

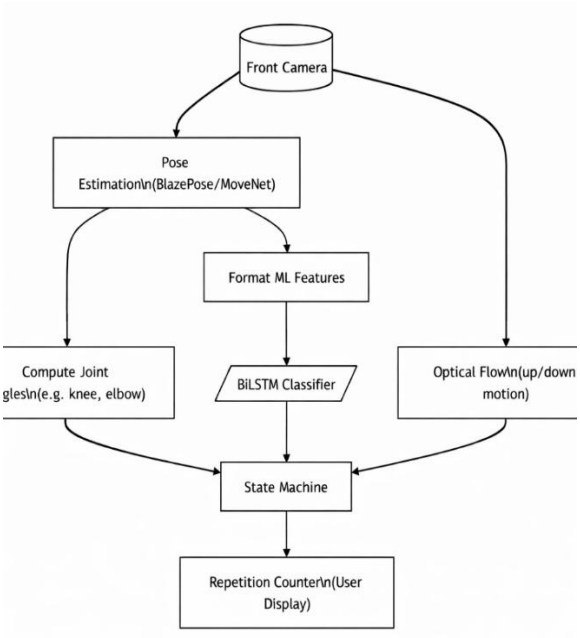

Figure 1: Pipeline for the proposed on-device AERC system.

The front camera feeds into pose estimation; the resulting keypoints are processed in parallel by a

BiLSTM classifier, a joint-angle phase detector, and an optical-flow motion analyser. All three signals converge in the state machine, which drives the repetition counter displayed in the UI.

4.1 Stage 1 — Pose Estimation

Every new image from the camera goes straight into a small built-in system that spots body joint positions - focusing on main limbs needed for specific movements. Instead of one fixed choice, two options get tested: MoveNet-Lightning weighs just 4.7 MB and hits over 30 images each second using a standard phone's graphics chip. Then there is BlazePose-Lite, larger at 20 MB but covering more points on the body, processing each shot in about 15 to 30 thousandths of a second through regular processors. A middle path exists too - MoveNet-Thunder fits between them. Which model runs depends entirely on what the device can handle, decided while operating.

4.2 Stage 2 — Feature Extraction

Midway through processing, the system calculates two kinds of features from identified points. Knee bend, hip opening, or arm angle show up next - each shaped by how limbs connect and shift during movement. Instead of raw positions, scaled-down coordinate values appear, capturing posture while adjusting for size differences. These numbers travel in chunks, sliding along every 20 to 30 image snapshots. Time stretches here just enough for what comes later - recognition takes shape only when motion history feeds it.

4.3 Stage 3 — Temporal ML Classifier (BiLSTM)

Tiny two-way memory cells chew through filled data chunks. Because they glance back plus forward, guesses hold up even when dots flicker or vanish mid-motion. Exercise kind pops out alongside movement stage - no button jabs needed between routines. Inside, twin learning layers run with 64 internal signals each. Squeezed into 8-bit form, the whole thing fits in about 200 thousand bytes. Phones finish a pass in under ten ticks of the clock.

4.4 Stage 4 — Angle Thresholding and Optical-Flow Fallback

From the start, alongside machine learning processing, a fixed rule-based system checks joint angles every frame using preset limits - like spotting a squat's lowest point when knees bend past 120 degrees. These rules run instantly without needing prior examples, firing off fast alerts if body posture seems clear. If detection quality drops - say, due to blocked views, quick actions, or dim light - an alternate method kicks in. Instead of joints, it tracks overall up-down movement across images taken one after another. Rising patterns mark upward effort; falling ones show descent. Much like earlier work by Sinclair and team [1], this backup keeps counts going even under tough conditions, avoiding complete breakdowns.

4.5 Stage 5 — State Machine and Repetition Counter

One state follows another in strict order - Start, then Descend, then Bottom, next Ascend, finally Top. When phase cues arrive - from the BiLSTM, the angle judge, or motion tracker - they get ranked, only the top one moves forward. Only when every stage fires in sequence, from first to last, does the tally rise by exactly one. Short wobbles near a boundary won't fool it; each rep must hit Bottom fully, otherwise nothing registers. The number appears instantly on screen, tucked beside moving dots that trace body joints.

V. MOBILE IMPLEMENTATION

5.1 Platform Strategy

One app works on Android and also runs smooth on iPhones. Flutter handles what users see, letting parts be shared across systems. Heavy math jobs shift to special tools made by each phone maker. Phones with fast chips tap into their own smart engines when able. This move cuts down delays plus saves power over time.

5.2 Quantisation and Optimization

Every neural network output gets turned into 8-bit integers through TensorFlow Lite's after-training method. Because of this shift, models shrink to about one-fourth the original size when stored as floating points - speed goes up on machines missing special decimal math units, yet performance dips just under one percent most times. Running separately, the posture detection part uses either GPU power or NNAPI support. Meanwhile, the smaller bidirectional LSTM task sticks to the central processor, keeping system resources from clashing.

5.3 Model Trade-off Summary

Table 2: Performance trade-offs for candidate on-device models.

Model	Size	Latency	Accuracy (est.)	Remarks
MoveNet-Lightning	4.7 MB	~5–10 ms	~95% key point coverage	Very fast; reduced lower-body detail
BlazePose-Lite	20 MB	~15–30 ms	~98% key point accuracy	Slower; tracks more joints
MoveNet-Thunder	12 MB	~10–20 ms	~97% key point accuracy	Balanced speed/accuracy
BiLSTM Classifier	0.2 MB	~5 ms	>99% (on seen data)	Captures temporal context
1D-CNN Classifier	0.5 MB	~10 ms	~98%	Simpler architecture; lower latency

Latencies are measured on contemporary smartphones using GPU or NNAPI acceleration for pose models. Classifier accuracy refers to repetition-phase classification on held-out test data.

5.4 User Interface and Wellbeing Features

Live movement lines show up on screen with joints marked in colors while reps get counted loud and clear. When practice kicks in, distractions shut down completely thanks to a quiet setting active only during workouts [12][15]. Streaks grow day by day when goals add up gradually - bars fill, milestones light up, small wins stack quietly [16]. Time spent moving turns into tokens used later to block certain apps till enough activity logs in through linked tools.

VI. EVALUATION PLAN

6.1 Dataset Collection

Some folks will do squats, push-ups, or pull-ups - maybe lunges or sit-ups too - for a fresh batch of movement data. Between thirty and fifty people are signing up, trying each move twenty to thirty times. They'll record these in everyday spots: dim rooms, bright ones, odd camera views, all sorts of clothes and shapes included. Two trained eyes will count every rep separately, then a third steps in if they don't match. This agreed number becomes the right one. Clips from online sources like Kaggle fill gaps at first, helping shape how models learn and tune themselves early on.

6.2 Evaluation Metrics

Exactly right counts show how often guesses hit the true number. This measure tracks full matches between what was said and what actually exists. Hits land when predictions line up perfectly with real values. Correct tallies appear each time estimated amounts equal actual ones. Precision here means nailing the number without error. Success comes if the forecast fits reality completely.

One step off still counts right? That measure tracks how often guesses land just a spot away from the real number. It follows earlier work like [1] and [6] did it - no surprises there. Close enough means within plus or minus one. The score grows each time that happens.

On average, how far off predictions are from real numbers - counted without direction across every group. This measure adds up differences but ignores minus signs, then splits by total count. Every set contributes equally to the final value shown.

One measure track how fast each frame processes, using typical phones like a Google Pixel or iPhone at mid-tier levels. Median delays appear alongside numbers showing worst cases during testing periods. Timing comes from real device trials focused on responsiveness under normal conditions. Ninety-five percent of results fall within a certain range seen across repeated runs. Speed reflects average lags plus occasional spikes found in everyday usage scenarios.

Midway through testing, power consumption got logged alongside processor and graphics chip activity. Tools built into the system tracked those metrics nonstop for ten minutes straight. Every session ran without breaks to keep results consistent. Measurements came directly from the device's own monitoring features. Performance load appeared clearly once numbers were compiled after each run.

6.3 Baselines and Ablation Studies

One test uses the first version of Pūioio [2], so results line up with current methods in device-based sound recognition. Instead of comparing directly, another setup swaps out the BiLSTM and adds thresholds plus optical flow - this shows what the time-based classifier actually brings. Where possible, a third option includes data from an outside motion sensor, setting a

limit on how well camera-free techniques might perform.

One by one, parts like optical flow get turned off to see how things change. Changing the size of the BiLSTM's memory state happens separately from adjusting how long the input windows last. Instead of MoveNet, BlazePose steps in during some runs. Some tests run lean models with fewer bits; others stick to full precision. Each tweak reveals what shifts in count correctness or processing demand. Every piece gets its moment under scrutiny, showing how much it weighs on speed and right answers.

6.4 Expected Results

Most likely, the system will count right more than 95 times out of 100 when light is good and people are fully seen. Since it uses a BiLSTM model,

Closing the difference between Sinclair et al.'s live test score - nearly 99% - and their earlier dataset outcome, which lagged near 87%, comes down to missing time-based cues in recordings. Real-time accuracy could climb past 98%. Each frame must process within fifty milliseconds across pose detection, data shaping, and classification on standard hardware. That pace supports a minimum speed of twenty images each second.

VII. LIMITATIONS AND FUTURE WORK

Picture-based motion tracking has limits. If someone moves out of view, the system struggles. Poor light makes it harder to spot body points too. From a sideways angle, markers get tricky to pin down. Overhead shots? Same issue. Baggy clothes tend to hide limbs, which trips up detection now and then. Bright walls behind you, or shadows swallowing details - either mess with precision. Right now, only three full-body actions work reliably. Moves like jumps or deep squats need clearer examples before they function well. Each new type might demand its own fine-tuned settings later on. Accuracy drops when form gets messy - or space gets tight.

Heat builds up fast on basic phones with little memory and no dedicated AI chip, especially if used too long. That heat slows things down while chewing through power at a faster rate. Slower movement in the scene means less need for heavy processing. Adjusting how often frames update could ease the demand, something next steps aim to test out.

Next steps involve checking movement quality live using joint angle info already in the system - then giving instant corrections. Some work will adjust personal limits based on individual bodies, borrowing methods from Sinclair and team [1]. Phone sensors might join the mix to track quick motions more reliably. Over time, how limiting screen access affects workout habits and social media use among actual Discifit users could also get measured.

VIII. CONCLUSION

A fresh look at how exercise tracking works inside the Discifit app reveals a mix of movement sensing and smart timing tricks. Instead of relying on just one method, it pulls pieces from earlier attempts - borrowing motion backup ideas from Sinclair's team, timeline awareness from Riccio's models, posture checks like those used by Alatiah and Chen, plus inspiration from Tsourgiannis on keeping users engaged. The result tries to hit more than 95 percent precision without needing the cloud, running live on everyday phones.

Most days, people check their phones without thinking. What if that habit could help them move more? That idea drives Discifit. It depends heavily on how well the AERC part works. Mistakes in counting reps break confidence fast. When users notice errors, they stop believing the rewards are fair. Five steps make up the process behind the scenes. Each step backs up the one before it. If something fails, the next stage tries to fix it. This setup keeps things working even when conditions get messy. Trust grows slowly. Consistency matters most. Small breakdowns add up unless handled quietly. Confidence sticks around only when results feel honest.

REFERENCES

[1] Sinclair, K. Kautai, and S. R. Shahamiri, "Pūioio: On-device real-time smartphone-based automated exercise repetition counting system," *arXiv preprint arXiv:2308.02420*, 2023, doi: 10.48550/arXiv.2308.02420.

[2] R. Riccio, "Real-time fitness exercise classification and counting from video frames," *arXiv preprint arXiv:2411.11548*, 2024.

[3] Z. Shuaib, P. Shaw, *et al.*, "Recognition and repetition counting for complex physical exercises with deep learning," *Sensors*, vol. 19, no. 3, Art. no. 714, 2019, doi: 10.3390/s19030714.

[4] T. Alatiah and C. Chen, "Recognizing exercises and counting repetitions in real time," *arXiv preprint arXiv:2005.03194*, 2020.

[5] M. Mühlbauer, *et al.*, "MiLift: Efficient smartwatch-based workout tracking using automatic activity recognition," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1352–1365, 2018.

[6] M. M. P. Vanden Abeele, *et al.*, "Digital wellbeing as a dynamic construct," *Communication Theory*, vol. 31, no. 8, pp. 731–750, 2020, doi: 10.1093/ct/qtaa024.

[7] "Building your academic research digital identity: A step-wise guide," Springer, 2023.

[8] D. Biedermann, J. Schneider, and H. Drachsler, "Digital self-control interventions for distracting media multitasking—a systematic review," *Journal of Computer Assisted Learning*, vol. 37, no. 5, pp. 1217–1231, 2021, doi: 10.1111/jcal.12565.

[9] Tsourgiannis, *et al.*, "Can gamified behavioural change mental health mobile apps reduce students' anxiety and improve well-being? An efficacy study," *Exploratory Digital Health Technologies*, vol. 3, Art. no. 101170, 2025, doi: 10.37349/edht.2025.101170.