IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AI-BASED ONLINE PROCTORING SYSTEM FOR SECURE CODING ASSESSMENTS

[1]Dr. G. Janaka sudha, [2]Mithun S

[1]Associate Professor, [2]B.Tech Student

[1]Department of Computer Science and Engineering,

[1]Sri Venkateswara College of Engineering, Sriperumbudur, Tamil Nadu, India

Doi: https://doi.org/10.56975/ijcrt.v14i7.309256

Abstract-Online examinations and remote coding assessments require proctoring systems that can verify candidate authenticity, detect suspicious behaviour, preserve evidence and support fair human review. This paper presents a multimodal AI-based online proctoring system designed for secure coding assessments. The proposed system integrates captcha-less behavioural login analysis, OTP verification, a browser-based coding interface, MTCNN face analysis, MediaPipe FaceMesh gaze estimation, YOLOv8 object detection, audio activity monitoring, cooldown-based violation aggregation and automated integrity-report generation. Unlike conventional webcam-only monitoring, the system represents a candidate session as a multimodal stream and applies mathematical decision rules to authentication, visual attention, object presence, audio activity and event acceptance. The implementation is analysed using project source code and generated violation logs. A case-study session containing 100 accepted events demonstrates the complete monitoring pipeline and shows that gaze deviation, face absence, multiple-face presence and prohibited-object events can be captured in a structured evidence model. The paper further discusses computational complexity, deployment security, privacy, limitations and validation requirements for institutional use.

Keywords: Online proctoring, AI proctoring, coding assessment, computer vision, MTCNN, MediaPipe, YOLOv8, gaze tracking.

I. INTRODUCTION

Online examinations have become a normal part of higher education, recruitment and placement-oriented programming tests. They reduce travel, allow flexible scheduling and make assessment available to geographically distributed candidates. However, the same flexibility weakens the physical supervision that examination halls traditionally provide. A candidate may leave the camera view, receive help from another person, consult unauthorized materials, use a mobile phone or automate access to the platform. In a coding assessment, these risks are especially important because a correct program output does not prove that the candidate independently wrote the solution.

Manual online invigilation is one response to this problem, but it is expensive and difficult to scale. A human reviewer must watch long recordings and correlate visual behaviour with examination events. Rule-based tools such as tab-change detection or full-screen enforcement are useful but incomplete, because many forms of malpractice occur outside the browser state. Webcam recording alone also produces large quantities

of evidence without automatically highlighting the moments that require attention. These limitations motivate intelligent proctoring systems that can monitor multiple behavioural channels and summarize evidence in a reviewer-friendly format. The proposed work focuses on an AI-based online proctoring system for secure coding assessments. The system is designed around a practical requirement: it must not merely detect one suspicious signal, but must support the full examination workflow from login to report generation. The prototype therefore includes behavioural login screening, OTP verification, a web coding interface, real-time webcam and microphone monitoring, structured logging and automated report generation. The design is intended to assist human evaluators rather than replace them.

This paper extends a project implementation into a research formulation. It defines a candidate session as a multimodal signal, introduces mathematical decision functions for authentication and proctoring events, describes algorithms for monitoring and aggregation, and evaluates the generated logs from the implemented prototype. The research question is: how can a modular AI system combine authentication, visual attention analysis, prohibited-object detection and audio monitoring to produce reliable evidence for online coding assessments?

The contributions of this paper are fourfold. First, it proposes a complete system architecture that integrates secure access control, coding assessment and proctoring in one workflow. Second, it formulates detector outputs using equations for behavioural risk, face presence, gaze deviation, object confidence, audio energy and cooldown-based event acceptance. Third, it provides pseudocode for authentication, real-time monitoring and report generation. Fourth, it presents a case-study evaluation using project-generated logs and discusses the limitations that must be addressed before high-stakes deployment.

II. RELATED WORK

Automated proctoring research has evolved from simple webcam recording toward intelligent behaviour analysis. Early systems focused on identity verification, browser restrictions and manual review. These systems provided a baseline level of control, but they were limited when suspicious behaviour occurred outside the browser window or when the reviewer needed a concise explanation of why a session was flagged. Computer vision has become central to modern proctoring. Face detection confirms that a candidate is visible, while multiple-face detection can indicate the presence of another person. Gaze and head-pose estimation provide a proxy for attention direction. If a candidate repeatedly looks away from the screen during a test, the behaviour may indicate consultation of external material. Nevertheless, gaze alone is not sufficient because natural eye movement, reading habits, camera placement and lighting conditions can cause false positives.

Deep learning models such as CNNs extract spatial information from face and eye regions. Recurrent models and attention mechanisms can extend this analysis across time, allowing a system to distinguish brief natural movement from sustained suspicious behaviour. In practice, however, high-capacity temporal models require labelled data, careful calibration and greater computational resources. A prototype deployed on ordinary laptops may therefore combine lightweight landmark geometry with rule-based temporal aggregation as a practical first step. Object detection has also been studied for examination monitoring. Detectors such as YOLO can identify prohibited objects including books and mobile phones. The advantage of object detection is interpretability: a detected phone or book is easier for a reviewer to understand than a black-box anomaly score. The limitation is sensitivity to camera angle, occlusion and confidence thresholds. A robust system should therefore store evidence and avoid issuing final judgements solely from object confidence.

Audio monitoring provides another independent signal. Speech or unusual noise during a remote test may indicate collaboration, but background noise varies widely across homes and laboratories. Energy thresholds and peak amplitudes can be used for lightweight monitoring, while speech-to-text models such as Whisper can support keyword detection. Audio evidence raises stronger privacy concerns than visual metadata, so deployment must carefully define consent, retention and access policies.

Recent surveys on AI-based online proctoring identify multimodal monitoring, privacy and explainability as open challenges. The proposed system follows this direction by combining login behaviour, face status, gaze state, object presence and audio activity. Its distinguishing feature is the integration of these signals with a coding assessment workflow and a report generator rather than presenting detectors as isolated modules.

Table I. Summary of Proctoring Techniques

Technique	Strength	Weakness
Browser rules	Simple and fast	Cannot observer physical environment
Face detection	Confirms candidate visibility	Sensitive to lighting and camera angle
Gaze analysis	Estimates attention direction	Requires calibration
Object detection	Finds phones / books	Affected by occlusion
Audio monitoring	Captures collaboration cues	Privacy and background noise issues
Proposed fusion	Combines independent evidence	Requires careful threshold validation

III. PROBLEM FORMULATION AND SYSTEM MODEL

Let a candidate examination session be represented by a tuple S = {A, V, M, C, R}, where A denotes authentication features, V is the video stream, M is the microphone stream, C is the coding activity and R is the resulting evidence report. The proctoring objective is not to directly classify a candidate as guilty or innocent, but to produce a structured sequence of suspicious events with enough metadata for human verification. At time t, the webcam frame is denoted by F_t and the audio chunk is denoted by X_t. A detector module transforms these signals into event candidates. These candidates are then filtered through policy thresholds and cooldown rules to avoid repeated counting of the same sustained behaviour. The accepted events form a log L that is later summarized into a report.

$$S = \{A, V_t, M_t, C_t, R\}, \quad t = 1, 2, \ldots, T \qquad (1)$$

$$L = \{e_{i,t} \mid accept(e_{i,t}) = 1\} \qquad (2)$$

$$Risk(S) = (1/T) \sum_t \sum_i w_i e_{i,t} \qquad (3)$$

In (3), w_i is the severity weight assigned to the i-th violation category. The risk score is an indicator for review priority, not an automatic disciplinary decision. This distinction is essential because AI detections can be affected by camera quality, background noise and user posture.

IV. PROPOSED ARCHITECTURE

The system follows a layered architecture. The first layer is the candidate browser interface. It collects login details, mouse trajectory, duration and user-agent information before forwarding the candidate to OTP verification. The second layer is the secure assessment layer, which contains the coding problem, editor, language selector and test execution controls. The third layer is the proctoring engine, which coordinates AI modules and maintains session state. The proctoring engine is implemented as a stateful controller. It initializes the camera, starts audio monitoring, loads AI models when monitoring begins and exposes status information to the web interface. Face detection, gaze estimation, object detection and audio monitoring are treated as independent detectors. Each detector produces event candidates, and the engine decides whether to

accept, suppress or log them based on thresholds and cooldown rules. The evidence layer stores JSON records, timestamps, metadata and visual artifacts where applicable. This separation between detection and evidence is important because it allows the report generator to summarize events without destroying the raw audit trail. A reviewer can inspect the high-level event distribution and then return to stored evidence when a decision requires more context.

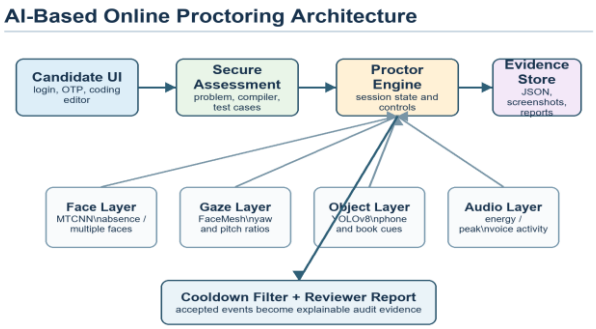

Fig. 1. Proposed Ai-Based Online Proctoring Architecture

Module	Method	Primary Output
Authentication	Email allow-list, OTP, mouse path, duration and user-agent	human / limited / bot
Face layer	MTCNN with confidence filtering	FACE_ABSENT, MULTIPLE_FACES
Gaze layer	MediaPipe FaceMesh landmark geometry	GAZE_AWAY
Object layer	YOLOv8n with examination-relevant COCO classes	OBJECT_DETECTED
Audio layer	Energy, peak amplitude and optional Whisper	VOICE_DETECTED
Reporting	JSON records, screenshots, charts and templates	integrity report

Table Ii. Implemented System Modules

V. METHODOLOGY

A. BEHAVIOURAL AUTHENTICATION

The login workflow uses a captcha-less behavioural risk model. Let d_m represent mouse path distance, t_l represent login duration and u represent the user-agent string. Suspicious user-agent tokens include headless, bot, spider, crawler, curl, postman and selenium. The login score r is computed as a weighted sum of indicator functions.

$$r = \alpha\, I(d\_m < \tau\_m) + \beta\, I(t\_l < \tau\_l) + \gamma\, I(u \in U\_b) \qquad (4)$$

$$class(r) = human \text{ if } r < \theta\_1;\ limited \text{ if } \theta\_1 \le r < \theta\_2;\ bot \text{ otherwise} \qquad (5)$$

This formulation is intentionally lightweight. It does not attempt to perform biometric identification; it only rejects or limits sessions that resemble automated access. A low-risk session proceeds to OTP verification, adding a second factor before the coding assessment begins.

B. FACE PRESENCE AND MULTIPLE-FACE DETECTION

The face module uses MTCNN with keep-all detection enabled. For a frame F_t, the detector returns bounding boxes b_k and confidence values c_k. The valid face set B_t contains detections whose confidence is above the configured threshold τ_f.

$B_t = \{b_k \mid c_k \geq \tau_f\}$ (6)
$FACE\_ABSENT_t = I(|B_t| = 0)$ (7)
$MULTI\_FACE_t = I(|B_t| \geq 2)$ (8)

A face absence event indicates that the candidate has moved away from the webcam or that the camera cannot observe the candidate. Multiple-face detection indicates that another person may be present in the candidate environment. Both conditions require review context because occlusion, lighting or accidental background movement may affect detection.

C. GAZE AND HEAD-POSE ESTIMATION

The gaze layer uses MediaPipe FaceMesh landmarks. Let n_x be the nose-tip x-coordinate, m_x be the midpoint of the outer eye corners and d_e be the inter-eye distance. The normalized yaw ratio y_t estimates horizontal gaze or head direction.

$y_t = (n_x - m_x) / d_e$ (9)
$p_t = (n_y - m_y) / \max(\varepsilon, c_y - m_y)$ (10)
$g_t = Center \text{ if } |y_t| \leq \tau_y \text{ and } \tau_u \leq p_t \leq \tau_d; \text{ Away otherwise}$ (11)

The pitch ratio p_t compares the nose-tip position with the eye midpoint and chin landmark. The thresholds τ_y, τ_u and τ_d map the normalized geometry to center, left, right, top and down states. In strict proctoring mode, stable non-center states are treated as gaze-away candidates.

D. OBJECT DETECTION

The object module uses YOLOv8n to identify prohibited materials. In the implemented prototype, the forbidden set contains book and cell phone classes. For each detected object o_j with class c_j and confidence q_j, the event is generated if the class belongs to the forbidden set and q_j exceeds threshold τ_o.

$O_t = \{o_j \mid c_j \in \{book, phone\} \wedge q_j \geq \tau_o\}$ (12)
$OBJECT_t = I(|O_t| > 0)$ (13)

Object detection is computationally heavier than landmark processing, so the implementation rate-limits YOLO inference. This is a practical design choice for ordinary laptops because continuous high-resolution object inference can reduce frame rate and degrade the responsiveness of the assessment interface.

E. AUDIO ACTIVITY DETECTION

The audio module processes microphone chunks in a separate thread. For chunk X_t of length N, the normalized energy E_t and peak amplitude P_t are computed. The voice/noise candidate is raised when either measure crosses its threshold.

$E_t = (1/N) \Sigma_{n=1}^{N} X_t[n]^2$ (14)
$P_t = \max_n |X_t[n]|$ (15)
$VOICE_t = I(E_t \geq \tau_E \vee P_t \geq \tau_P)$ (16)

Optional speech transcription can be added using Whisper to inspect suspicious words. However, because audio is sensitive personal data, the default research interpretation should treat audio events as review prompts rather than direct evidence of malpractice.

F. COOLDOWN-BASED EVENT AGGREGATION

A sustained behaviour such as looking away may be detected in many consecutive frames. Counting each frame as a separate violation would inflate the session risk score. The engine therefore stores the last accepted timestamp for each event type and accepts a new event only after a cooldown interval Δ_i.

$$accept(e_i,t) = I(t - last_i \geq \Delta_i) \quad (17)$$
$$N_i = \Sigma_t\, accept(e_i,t) \quad (18)$$

The aggregation step improves interpretability. Instead of reporting hundreds of nearly identical frame-level detections, the system reports behavioural episodes. The reviewer can then inspect when each episode occurred and whether the evidence supports a concern.

VI. ALGORITHMS

ALGORITHM 1: BEHAVIOURAL LOGIN AND OTP VERIFICATION

Input : email, mouse path M, login duration t_l, user-agent u
Output: login decision d
1 if email is not in authorized list then reject
2 compute mouse distance d_m from M
3 r <- alpha I(d_m < tau_m) + beta I(t_l < tau_l)
4 if user-agent contains suspicious token then r <- r + gamma
5 if r >= theta_2 then d <- bot and deny access
6 else if r >= theta_1 then d <- limited and restrict session
7 else generate OTP and send verification email
8 if candidate enters valid OTP then d <- human
9 else reject session

ALGORITHM 2: REAL-TIME MULTIMODAL PROCTORING

Input : authenticated session S, webcam stream V, microphone stream M
Output: violation log L
1 initialize camera, counters, cooldown map and report state
2 lazily load MTCNN, FaceMesh, YOLOv8 and audio monitor
3 start audio thread and register callback for VOICE events
4 while session is active do
5 capture frame F_t
6 B_t <- face_detector(F_t)
7 g_t <- gaze_tracker(F_t)
8 O_t <- object_detector(F_t)
9 C <- build candidates from face, gaze, object and audio
10 for each candidate e_i in C do
11 if accept(e_i,t) = 1 then
12 capture evidence and append metadata to L
13 update last_i and violation count
14 end if
15 end for
16 end while

ALGORITHM 3: INTEGRITY REPORT GENERATION

Input : violation log L, evidence directory E
Output: HTML/DOC report R
1 group events by violation type
2 compute N_i, severity labels and timeline statistics
3 sort events by timestamp
4 generate frequency chart and timeline chart
5 attach screenshots or evidence references if available
6 render evaluator-facing template using report fields
7 save final report and preserve raw JSON log

VII. IMPLEMENTATION DETAILS

The prototype is implemented primarily in Python. Flask and Flask-CORS provide routing, API responses, HTML template rendering and video streaming. OpenCV manages webcam capture, frame annotation and JPEG encoding for the live feed. The AI modules are loaded lazily inside the monitoring thread so that heavy initialization does not block web-application startup. Face detection is implemented using facenet-pytorch MTCNN. The configuration enables detection of all visible faces and filters results using a minimum confidence threshold of 0.75. Multiple-face detection is performed by counting high-confidence boxes. This simple rule is suitable for a prototype because it is interpretable and easy to audit.

Gaze estimation uses MediaPipe FaceMesh. Instead of requiring a large labelled gaze dataset, the prototype uses normalized facial geometry. Eye landmarks, nose-tip position and chin position are used to compute yaw and pitch ratios. This approach is lightweight and fast, although it requires calibration for camera position and user posture.
Object detection uses the Ultralytics YOLOv8n model. The prototype restricts attention to examination-relevant COCO classes such as book and cell phone. The threshold is set to 0.25 to avoid missing partially visible objects, while rate limiting is used to keep inference cost manageable.

Audio monitoring uses PyAudio at a 16 kHz sample rate. The energy threshold is 0.00048 and the peak threshold is 0.6 in the current configuration. A background thread processes audio chunks and invokes a callback when a voice/noise event is detected. Optional Whisper support can transcribe recent buffers if a deployment policy permits it.
The reporting layer uses JSON logs, screenshots, Jinja templates and generated visual summaries. Each violation record stores a timestamp, violation type and metadata. The report generator maps raw event types to human-readable labels and severity levels, then produces timeline and frequency summaries for review.

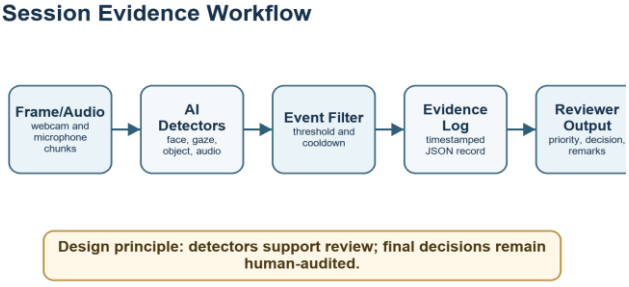

Fig. 2. Evidence generation workflow from detector signal to reviewer output.

Important configuration parameters include camera source 0, target capture resolution 640 x 480, target frame rate 30 FPS, YOLO maximum 15 FPS, face threshold 0.75, object threshold 0.25, audio sample rate 16 kHz and alert cooldown of one second per violation type.

Table Iii. Configuration Parameters

Parameter	Value	Purpose
Camera	source 0, 640x480, 30 FPS	Webcam stream
Face threshold	0.75	Filter weak detections
Gaze threshold	yaw 0.06, pitch up 0.16, pitch down 0.55	Map landmarks to gaze state
Object threshold	0.25, max 15 FPS	Detect books / phones
Audio	16 kHz, energy 0.00048, peak 0.6	Voice / noise detection
Cooldown	1 second	Avoid duplicate event inflation

VIII. EXPERIMENTAL EVALUATION

The evaluation uses the generated logs and reports from the implemented prototype. The latest available session log contains 100 accepted violation events after cooldown filtering. The goal of this evaluation is functional rather than benchmark-based: it verifies whether the system can execute the full workflow from detection to evidence logging and report generation.

Table Iv. Output Event Distribution After Cooldown Filtering

Event	Count	Share	Reviewer Meaning
Gaze away	43	43.0%	Attention repeatedly moved away from screen
Object detected	27	27.0%	Possible phone/book or external material
Face absent	20	20.0%	Candidate not confidently visible
Multiple faces	10	10.0%	Another person may be present
Voice detected	0	0.0%	No threshold-crossing speech/noise

The event distribution shows that gaze-away events are the most frequent category in the case-study session. This is expected because gaze direction can change many times during a test and because strict proctoring treats sustained non-center gaze as suspicious. Object detections form the second largest group, indicating that the YOLO module was active and contributed independent evidence. Face absence events account for a smaller but significant share of the log. These events indicate periods where no valid high-confidence face was visible. Multiple-face events appear less frequently, which is consistent with the fact that they require two or more high-confidence faces to be detected simultaneously. No voice event appears in the latest log, suggesting that audio alerts are not artificially inserted but depend on threshold crossings.

A complete accuracy evaluation would require labelled ground-truth sessions. For each frame or time interval, an annotator would label whether the candidate was visible, whether another person was present, whether gaze was away, whether a prohibited object was visible and whether speech occurred. The prototype logs could then be compared to labels to compute precision, recall, F1-score and false-positive rate. For a future benchmark, the test dataset should include normal behaviour, intentional gaze shifts, temporary face absence, multiple-person scenes, mobile-phone visibility, book visibility, background noise and spoken collaboration. It should also include variation in lighting, camera height, skin tone, spectacles and room layout. Without this diversity, an apparent accuracy score would not generalize to real deployments.

Even without labelled accuracy, the case study validates key engineering requirements. The system starts monitoring, initializes AI modules, accepts event candidates, applies cooldown filtering, stores JSON metadata and generates reports. This end-to-end validation is important because many academic proctoring models stop at detector design and do not demonstrate integration with assessment workflow and evidence reporting.

IX. EXTENDED VALIDATION PROTOCOL

A publication-quality evaluation of an online proctoring system should go beyond a single demonstration session. The system must be tested under controlled normal conditions and controlled violation conditions. A normal session should include ordinary coding behaviour such as reading the problem statement, thinking without typing, compiling code, debugging failed test cases and briefly looking at the keyboard. These behaviours should not be treated as automatic malpractice. A violation session should include clearly defined events. For face absence, the candidate may move completely out of frame for a known interval. For multiple-face events, another person may enter the camera view. For gaze-away events, the candidate may read from an off-screen note for several seconds. For object events, a phone or book may be placed at different distances and orientations. For audio events, speech may occur at different volumes and background noise levels.

The ground truth for such an evaluation should be annotated using time intervals rather than only frame labels. A behaviour such as looking away is naturally an interval event; marking only individual frames can overstate small timing disagreements between the detector and human annotation. Therefore, event-level matching should allow a tolerance window around the start and end of each labelled interval. Let G_i be the set of ground-truth intervals for event type i and P_i be the set of predicted intervals after cooldown aggregation. A predicted interval should be considered a match if its temporal intersection with a ground-truth interval exceeds a threshold λ. This permits small detector delays while still penalizing unrelated predictions.

$$IoU_t(g,p) = |g \cap p| / |g \cup p| \qquad (19)$$
$$match(g,p) = I(IoU_t(g,p) \geq \lambda) \qquad (20)$$

Using interval matching, precision, recall and F1-score can be computed for each event category. This is more suitable than raw frame accuracy because the number of normal frames is usually much larger than the number of violation frames, which can produce misleadingly high accuracy even for a weak detector.

$$Precision_i = TP_i / (TP_i + FP_i) \qquad (21)$$
$$Recall_i = TP_i / (TP_i + FN_i) \qquad (22)$$
$$F1_i = 2\, Precision_i\, Recall_i / (Precision_i + Recall_i) \qquad (23)$$

False positives are especially important in proctoring because an incorrect flag can create stress for candidates and extra work for reviewers. A false-positive analysis should record the cause of each incorrect event, such as poor lighting, a reflective spectacle lens, a background object, another person briefly passing behind the candidate or environmental noise. This error taxonomy is more useful than a single scalar score. False negatives are also important because they represent behaviours that the system failed to detect. For example, a phone placed below the camera frame may not be visible, or a second person may speak without appearing in the webcam view. These cases show that online proctoring should be understood as risk reduction rather than perfect prevention.

The validation protocol should include device variation. A laptop webcam, an external webcam and a low-light camera can produce different face and object detection behaviour. Similarly, microphone gain and background noise differ across devices. A deployment that works only on the developer machine is not sufficient for institutional use.

User diversity should also be included. Face detection and gaze estimation may behave differently across skin tones, face shapes, hairstyles, spectacles and seating positions. A responsible evaluation must measure whether error rates are consistent across groups. If they are not, the system should be recalibrated or redesigned before deployment. The proposed project can support such validation because it already stores timestamped JSON logs and report artifacts. A future dataset builder can align these logs with manually annotated ground truth. The same report generator can then be extended to include precision, recall, event latency and reviewer decision statistics.

X. ABLATION AND SENSITIVITY STUDY DESIGN

An ablation study is necessary to determine the contribution of each module. The complete system combines authentication, face, gaze, object and audio signals. Removing one module at a time can reveal whether the remaining modules still identify the target behaviour and whether the removed module causes many false positives. The first ablation removes behavioural login analysis while retaining OTP verification. This experiment tests whether the mouse-path, duration and user-agent features add value against automated login attempts. If bot-like sessions pass OTP generation more frequently without this layer, the behavioural check is justified even though it is lightweight. The second ablation removes gaze estimation. This version can still detect face absence, multiple faces, objects and audio events, but it cannot identify attention shifts. Comparing reviewer usefulness between the full model and the no-gaze model would indicate whether gaze events provide meaningful prioritization during code assessment.

The third ablation removes object detection. This reduces computational load but eliminates explicit phone and book evidence. The trade-off is important because YOLO inference is the heaviest component in the prototype. If object events account for many meaningful flags, rate-limited object detection is preferable to removing it entirely. The fourth ablation removes audio monitoring. This experiment is useful in privacy-sensitive deployments where microphone access may not be allowed. The system should still function with webcam-only proctoring, but reviewers would lose evidence about collaboration through speech. A configurable deployment should allow institutions to enable or disable audio based on policy. Sensitivity analysis should vary thresholds such as τ_f for face confidence, τ_o for object confidence, τ_y for gaze yaw and Δ_i for cooldown. Lower thresholds increase detection sensitivity but may raise false positives. Higher thresholds reduce false positives but may miss subtle events. The optimal setting depends on the assessment risk level and the expected review capacity.

Table V. Suggested Ablation Experiments

Experiment	Removed Component	Expected Observation
A1	Behavioural login	more automated sessions reach OTP stage
A2	Gaze layer	attention-related events disappear
A3	Object layer	lower compute cost but no phone/book flags
A4	Audio layer	privacy improves but speech cues disappear
A5	Cooldown	duplicate violation counts increase

The cooldown ablation is particularly important. Without cooldown filtering, a single sustained gaze-away episode could produce many repeated events and exaggerate the severity of the session. With cooldown enabled, the report better reflects behavioural episodes rather than frame-level noise.

XI. REVIEWER WORKFLOW AND EVIDENCE INTERPRETATION

The proctoring system is designed to assist a human reviewer. A reviewer-facing workflow should begin with a session summary that lists candidate identity, test time, total duration, total violations and event distribution. The reviewer should then inspect high-severity events first, followed by repeated moderate events and finally low-severity context.Each violation should include a timestamp, type, severity and metadata. For face absence, metadata can include face count and detector confidence. For gaze-away, metadata can include gaze direction and normalized yaw/pitch values. For object detection, metadata should include detected class and confidence. For audio, metadata should include energy, peak amplitude and any permitted transcription indicator.

The reviewer should have three possible decisions for each event: accepted, rejected or inconclusive. Accepted means the reviewer agrees that the event indicates suspicious behaviour. Rejected means the event is explained by benign context. Inconclusive means evidence is insufficient. Storing reviewer decisions would allow the system to learn which event types are most useful and which thresholds need tuning. A session-level decision should not be based only on event count. Some events are more serious than others. Multiple-face detection or phone detection may deserve higher severity than a brief gaze shift. A weighted score can prioritize review, but final action should consider the pattern of events, examination rules and candidate explanation. The report should also preserve raw logs. Summaries are useful for quick inspection, but raw JSON records provide auditability. If a candidate appeals a decision, the institution should be able to reconstruct why a session was flagged and which human reviewer accepted or rejected each event.

XII. COMPLEXITY AND PERFORMANCE DISCUSSION

The computational cost of the system depends on the active detectors. Face and gaze analysis operate on each selected frame, while object detection is rate-limited because YOLO inference is more expensive. If T frames are processed, the total visual cost can be written as the sum of face, gaze and object costs.

$$C_{total} = T(C_{face} + C_{gaze}) + T_o C_{yolo} + T_a C_{audio} \quad (24)$$

Here, T_o is the number of frames on which object detection is executed and T_a is the number of audio chunks. Since T_o can be made smaller than T by rate limiting, the system can reduce computational

load while preserving object evidence. Audio processing is lightweight compared with video inference because it uses simple statistics unless transcription is enabled.

$$T_o = min(T, f_{yolo} D) \quad (25)$$

In (20), f_yolo is the configured maximum object-detection rate and D is session duration. This design allows deployment on ordinary student laptops while keeping the architecture compatible with GPU acceleration. In a production deployment, model inference could be optimized using batching, quantization or hardware acceleration. The memory cost of the system is dominated by loaded neural-network models and frame buffers. Lazy loading helps because the web server can start without immediately allocating all model resources. During a live session, the current frame, recent audio buffer, detector state and violation log must remain available. Older evidence can be written to disk to avoid unnecessary memory growth.

Network cost depends on whether video is streamed to a remote reviewer or processed locally. The prototype processes webcam frames in the local application context and stores evidence artifacts. A cloud deployment could centralize processing, but it would increase bandwidth requirements and privacy risk. A hybrid deployment could process lightweight signals locally and upload only accepted evidence events. Latency matters because a proctoring system should provide timely status updates. Face and gaze detections are suitable for frequent evaluation, while object detection can be evaluated at a lower frequency. If the coding interface becomes slow, candidate experience suffers and the assessment itself may be affected. Therefore, proctoring inference should never starve the code editor or test execution workflow.

XIII. DEPLOYMENT ARCHITECTURE

A production deployment should separate the candidate interface, proctoring service, code-execution service, database and reviewer dashboard. The candidate interface should run in the browser and communicate with the backend through authenticated HTTPS requests. The proctoring service should manage webcam and microphone permissions, detector configuration and event submission. The code-execution service should be isolated from the proctoring service. Candidate programs may contain accidental infinite loops, excessive memory use or malicious operations. Running such code directly on the web server is unsafe. A secure platform should execute submissions inside containers with CPU, memory, network and filesystem limits. The database should store users, exams, submissions, violation records, evidence references and reviewer decisions. Evidence files should be encrypted at rest, and access should be controlled by roles such as candidate, evaluator, administrator and auditor. The system should log every access to sensitive evidence.

A reviewer dashboard should present event summaries and allow filtering by severity, event type and timestamp. It should also allow reviewers to mark events as accepted, rejected or inconclusive. These decisions can later be used to estimate detector usefulness and to retrain or recalibrate threshold policies.
Deployment should also include monitoring for system health. Camera failure, microphone denial, model loading failure and report-generation errors should be reported as technical issues rather than academic violations. Separating technical failures from suspicious behaviour protects candidates from being penalized for system problems.

Table Vi. Production Deployment Components

Component	Responsibility	Security needs
Candidate UI	login, coding, camera permission	HTTPS and session control
Proctor engine	AI inference and event logging	model integrity
Code sandbox	compile and run submissions	container isolation
Database	records and review decisions	encryption and access control
Dashboard	human review	audit logs and role permissions

XIV. MULTIMODAL FUSION AND DECISION SUPPORT

A central design question in proctoring is how to combine different detector outputs. A naive approach would issue an alert whenever any detector fires. This maximizes sensitivity but may overwhelm reviewers with low-value events. A stricter approach would require multiple detectors to fire at the same time. This reduces false positives but may miss important behaviours that are visible in only one modality, such as a phone appearing in view without gaze deviation. The proposed framework treats detector outputs as evidence channels. Each channel contributes an event count, severity and confidence. A decision-support score can be computed as a weighted combination of normalized event frequencies. The score is used to prioritize review rather than determine guilt.

$$Score(S) = \Sigma_i \lambda_i \, normalize(N_i) + \Sigma_i \mu_i \, severity_i \qquad (26)$$
$$normalize(N_i) = N_i / max(1, D) \qquad (27)$$

In (26), λ_i controls the contribution of the event frequency and μ_i controls the contribution of severity. D denotes session duration. A longer session naturally provides more opportunities for events; normalization prevents long tests from being unfairly penalized only because they contain more frames. The score can also include temporal concentration. A session with ten suspicious events spread over two hours may be less concerning than a session with ten events clustered around a difficult problem. Temporal clustering can be measured by dividing the session into windows and computing event density in each window.

$$density_k = (1/|W_k|) \Sigma_{t \in W_k} \Sigma_i e_{i,t} \qquad (28)$$
$$Burst(S) = max_k \, density_k \qquad (29)$$

The burst score identifies short periods of unusually high suspicious activity. This can help reviewers inspect the most relevant segment first. For example, a burst of gaze-away and object-detection events during a coding problem may deserve more attention than isolated events during setup.

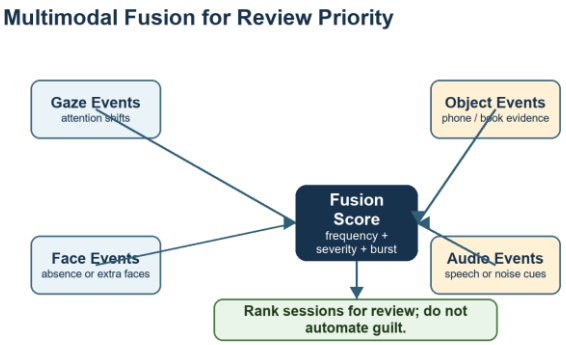

Fig. 3. Multimodal fusion channels used for reviewer prioritization

Table Viii. Sample Review Output Table

Session Signal	Evidence Combination	Priority	Suggested Action
High gaze burst	Gaze away + object event within 10 s	High	Inspect screenshots and code timeline
Camera absence	Face absent without audio	Medium	Check setup and visibility evidence
Extra person	Multiple faces repeated	High	Escalate for manual review
Brief gaze shift	Single gaze event only	Low	Treat as normal unless repeated

A further improvement is cross-modal consistency. If gaze-away and object-detection events occur close together, the combined evidence may be stronger than either event alone. If face absence occurs at the same time as audio speech, the interpretation may differ: the candidate may have moved away and spoken, or the camera may have temporarily failed. The system should report these combinations without automatically assuming misconduct. Cross-modal rules can be expressed using temporal proximity. Two events e_i and e_j are associated when their timestamps differ by less than a window δ. These associations can be shown in the report as related events. This makes the review process more efficient because the evaluator sees behavioural context rather than isolated alerts.

$$assoc(e_i,e_j) = I(|time(e_i) - time(e_j)| \leq \delta) \qquad (30)$$

Table Vii. Example Fusion Rules For Review Priority

Condition	**Interpretation**	**Priority**
Repeated gaze away	Possible external reference	Medium
Object + gaze away	Possible use of material	High
Multiple faces	Possible collaboration	High
Face absent + audio	Candidate may be away and speaking	review
Single brief gaze	Normal thinking or keyboard use	low

XV. FAILURE CASES AND MITIGATION STRATEGIES

Any AI proctoring system must be evaluated not only by successful detections but also by failure cases. A common failure case is poor lighting. Low light may reduce face confidence and create false face-absence events. The mitigation is to check camera quality at session start and warn the candidate before the test begins. The system should distinguish a technical camera problem from suspicious disappearance.

Another failure case is camera placement. If the laptop is placed too low or too high, gaze estimation may classify normal screen reading as looking up or down. A calibration stage can ask the candidate to look at the center, left, right, top and bottom of the screen before the test. The measured landmark ratios can then be used to personalize thresholds.

Spectacles and reflective surfaces can affect eye and landmark detection. Rather than penalizing users, the system should record confidence values and reduce reliance on gaze when landmarks are unstable. This is one reason multimodal fusion is valuable: if gaze is unreliable, object and face evidence can still contribute. Background objects can cause false object detections. A book on a shelf or a phone far behind the candidate may be detected even if it is not used. The report should include object confidence and, where possible, screenshot evidence so the reviewer can inspect context. A deployment may also define a clean-desk setup step before the exam begins.

Audio failure cases include family speech, traffic noise, fan noise and microphone gain variation. A room calibration step can record a few seconds of background noise and adapt energy thresholds. If audio is disabled by policy, the system should continue operating with visual channels and mark audio as unavailable rather than suspicious.

Model loading failures are also important. If MTCNN, MediaPipe, YOLO or PyAudio fails to initialize, the system should log a technical exception. It should not silently continue as though monitoring is complete. A robust deployment must expose monitoring health to the candidate and evaluator.

Network interruptions may affect OTP delivery, report upload or remote dashboard updates. Local buffering can prevent event loss during temporary disconnection. Once the connection resumes, buffered event records can be synchronized with the server. Timestamps should be generated locally and stored consistently to avoid ordering errors.

Code-execution failures should be separated from proctoring failures. If the compiler or sandbox fails, the candidate should not be flagged for misconduct. A production platform should have separate status categories for assessment errors, proctoring warnings and integrity violations.

XVI. COMPARATIVE DISCUSSION

Compared with webcam-only monitoring, the proposed system provides structured interpretation. Webcam-only recording captures evidence but does not identify when a reviewer should look. The proposed system adds event detection and summary reports, reducing the amount of raw video that must be inspected. Compared with browser-lockdown systems, the proposed system observes the physical environment. Browser lockdown can prevent tab switching or copy-paste behaviour, but it cannot detect a phone beside the laptop or another person assisting the candidate.

The proposed object and face modules address this limitation. Compared with single-modality eye tracking, the proposed system is more robust because gaze is only one channel. Eye movement can be ambiguous: a candidate may look away while thinking or reading from the keyboard. Combining gaze with object, face and audio evidence provides richer context. Compared with fully supervised human proctoring, the proposed system reduces repetitive monitoring effort. A human proctor can still make the final decision, but automated logs and summaries help focus attention on important moments. This hybrid approach is more practical for large classes and placement tests. Compared with black-box anomaly detection, the proposed rule-and-module design is interpretable. Each event type has a clear source and threshold. Although future deep temporal models may improve accuracy, interpretability remains important because candidates and institutions need understandable reasons for flags. The main disadvantage of the proposed system is the need for calibration. Because it runs in varied home environments, thresholds for lighting, gaze, object confidence

and audio noise may require adaptation. This is not unique to the proposed system; it is a general challenge for remote proctoring.

XVII. DATASET DESIGN FOR FUTURE BENCHMARKING

A future dataset for this project should be designed around coding-assessment scenarios rather than generic webcam behaviour. Participants should solve programming problems while following scripted normal and suspicious behaviours. The dataset should include screen activity, code-submission timestamps, webcam frames, audio indicators and manual annotations. The normal class should include behaviours that are easily mistaken for violations: looking at the keyboard, looking away while thinking, drinking water, adjusting seating position, reading a long problem statement and debugging silently. Including these behaviours is necessary to avoid a model that flags normal examination behaviour.

The suspicious class should include controlled examples of another person entering the frame, a phone appearing near the keyboard, a book being opened, the candidate leaving the camera view, repeated off-screen reading and audible collaboration. Each event should be annotated with start time, end time, event type and severity. The dataset should also include environmental variation. Lighting should vary from bright to dim. Cameras should include built-in laptop webcams and external webcams. Participants should include different appearances and seating positions. Audio should include quiet rooms, moderate background noise and speech from different distances. Evaluation should be reported per event type and per environment. Aggregate scores may hide important weaknesses. For example, object detection may perform well in bright rooms and poorly in dim rooms; gaze estimation may perform well for centered cameras and poorly for low-angle cameras. Reporting subgroup results is essential for fairness. Because face and audio data are sensitive, dataset governance must be planned from the beginning. Participants should consent to specific uses, retention periods and anonymization procedures. Public release may use metadata and blurred evidence rather than raw video, depending on institutional ethics approval.

XVIII. SECURITY, PRIVACY AND ETHICS

Online proctoring processes sensitive signals, including webcam frames, microphone samples and behavioural metadata. A responsible deployment must clearly inform candidates what is monitored, why it is monitored, how long evidence is retained and who can access it. Consent should be explicit, and institutions should provide alternatives where appropriate. The prototype should not be used as a sole decision-maker in high-stakes disciplinary actions. Automated detections can be wrong. For example, gaze estimation may be affected by screen position, a candidate may briefly look away while thinking, a book may appear in the background without being used, and background speech may come from another room. Human review and appeal mechanisms are therefore necessary.

From a software-security perspective, sensitive credentials such as SMTP passwords must be stored outside source code, preferably through environment variables or secret-management services. Communication should use HTTPS. Candidate code execution should occur inside a sandboxed container with CPU, memory, time and filesystem limits. The current project demonstrates proctoring workflow but should be hardened before production deployment. Evidence should be encrypted at rest and access should be role-based. Review actions should also be logged, so that acceptance, rejection or escalation of automated flags becomes auditable. Data minimization is important: the system should store only the evidence needed for integrity review and delete it according to an institutional retention policy.

XIX. REPRODUCIBILITY AND DATA MANAGEMENT

Reproducibility is a major requirement for research publication. A reader should be able to understand which models were used, which thresholds were configured and how the log statistics were computed. The prototype supports reproducibility through a YAML configuration file, modular source files and JSON violation logs. These artifacts make the system easier to inspect than a closed proctoring platform. The configuration should be reported alongside results. For example, the face confidence threshold, object confidence threshold, gaze yaw threshold, pitch thresholds, audio energy threshold and cooldown interval directly affect event counts. Without these values, another researcher cannot reproduce the same behaviour or compare results fairly.

The violation log should use a stable schema. Each record should include a unique event identifier, timestamp, event type, detector metadata, severity, evidence path and session identifier. A stable schema allows downstream analysis in spreadsheets, databases or statistical tools. It also supports auditing because the report can be regenerated from the raw log. Data management must balance reproducibility with privacy. Public datasets should avoid exposing identifiable faces or voices unless participants have explicitly consented. For internal evaluation, raw evidence can be stored securely while anonymized event metadata is used for analysis. This separation allows researchers to compute metrics without unnecessarily sharing sensitive media. Versioning is also important. The model version, configuration version and code commit should be stored with each evaluation. If a threshold changes, event distributions may change. If YOLO or MediaPipe versions change, detector behaviour may also change. Recording versions prevents confusion when comparing reports over time.

XX. LIMITATIONS

The present work is a functional prototype and research implementation. It does not claim benchmark-level accuracy because the evaluation does not use a labelled public dataset. The event distribution demonstrates system functionality, but it cannot establish precision or recall. A controlled dataset is required for statistically meaningful claims. The gaze model uses geometric thresholds rather than a trained personalized gaze estimator. This makes the system lightweight, interpretable and easy to implement, but it also means that users with different camera positions or postures may require calibration. Similarly, YOLO confidence thresholds must be tuned to balance missed detections and false positives.

The audio module uses signal energy and peak amplitude. This is computationally efficient but cannot always distinguish candidate speech from background noise. Optional transcription improves interpretability but introduces privacy and computational concerns. Future work should investigate privacy-preserving audio features and environment-aware threshold adaptation. The current implementation stores logs and reports in files. For institutional deployment, database-backed persistence, role-based dashboards, encrypted storage and reviewer notes would be necessary. The coding environment should also include secure sandboxing and plagiarism analysis so that behavioural evidence can be interpreted alongside code-submission evidence.

XXI. INSTITUTIONAL USE-CASE SCENARIO

Consider a placement-oriented programming test conducted for a final-year undergraduate class. Candidates log in from different locations using personal laptops. The institution needs to verify that each candidate is authorized, ensure that the coding environment is available, monitor the session without requiring one human proctor per candidate and generate evidence that can be reviewed after the test. In this scenario, the candidate first enters the login page. The system checks whether the email address belongs to the authorized list and evaluates behavioural login features. A candidate using a normal browser and natural

mouse movement proceeds to OTP verification. A suspicious automated user-agent or extremely short interaction is denied or placed into limited access.

After verification, the candidate enters the coding interface. The interface presents the problem statement, editor, language selector and test controls. At the same time, the proctoring engine begins monitoring. The camera stream is analysed for face visibility, multiple faces, gaze direction and prohibited objects. The microphone stream is analysed for energy and peak amplitude.

During the examination, the system does not interrupt the candidate for every low-level event. Instead, it records accepted events after cooldown filtering. If the candidate briefly looks away while thinking, the event may be recorded as low or moderate evidence depending on duration and policy. If a phone is detected near the candidate and gaze-away events occur around the same time, the session may receive higher review priority. At the end of the test, the report generator produces an integrity summary. The placement officer or faculty reviewer can inspect event counts, timestamps and evidence artifacts. The reviewer can mark events as accepted, rejected or inconclusive. Only after this human review should any administrative decision be made.

This use-case demonstrates why a complete proctoring system must include more than detection algorithms. Authentication, evidence management, reviewer workflow, secure code execution and auditability are all part of the research problem. A detector that works in isolation is useful, but an integrated system is necessary for practical online assessment. The scenario also shows the importance of proportional response. A session with one brief gaze-away event should not be treated the same as a session with repeated object detections and multiple-face events. The proposed scoring and reporting framework helps reviewers distinguish minor irregularities from patterns that deserve deeper investigation. Finally, the scenario emphasizes candidate trust. If candidates know what is monitored, how evidence is reviewed and how appeals are handled, the system becomes more transparent. Transparency is especially important in educational contexts, where unfair or unexplained automated decisions can damage confidence in the assessment process.

XXII. FUTURE ENHANCEMENT ROADMAP

The first enhancement is personalized gaze calibration. Before the examination begins, the candidate can be asked to look at five screen points: center, left, right, top and bottom. The system can record landmark ratios for each point and derive candidate-specific thresholds. This would reduce false positives caused by camera height, seating position and screen size. The second enhancement is temporal behaviour modelling. The current prototype uses rule-based cooldown aggregation, which is interpretable and efficient. A future version can add CNN-LSTM, temporal convolution or transformer-based sequence models to distinguish brief natural movements from repeated suspicious patterns. Such models should be trained only after a labelled dataset is available.

The third enhancement is secure code execution and plagiarism analysis. Since the target use case is coding assessment, behavioural evidence should be interpreted alongside code evidence. A candidate who triggers suspicious events and submits code highly similar to another candidate may require stronger review than a candidate with isolated visual alerts. Integrating plagiarism detection would make the platform more complete.

The fourth enhancement is adaptive thresholding. Static thresholds are simple but may not fit all environments. The system can estimate baseline lighting, face confidence, microphone noise and gaze stability during a pre-test calibration period. Thresholds can then be adjusted within safe limits while preserving transparency in the final report. The fifth enhancement is reviewer feedback learning. When reviewers mark events as accepted, rejected or inconclusive, those decisions can be stored as labels for future improvement. Over time, the system can identify which thresholds generate unnecessary alerts and which

event combinations are most useful for final decisions. The sixth enhancement is role-based institutional deployment. Faculty members, placement officers, administrators and auditors need different views of the same evidence. A production dashboard should therefore support exam-level summaries, candidate-level reports, reviewer notes, exportable audit logs and retention-policy controls.

The seventh enhancement is privacy-preserving analytics. Instead of storing raw media for all sessions, the platform can store metadata by default and preserve raw screenshots or audio snippets only for accepted events. Differential access policies can ensure that sensitive evidence is visible only when review is required. The final enhancement is large-scale field testing. Laboratory validation is necessary but not sufficient. The system should be tested during real mock examinations with informed consent, diverse devices and varied network conditions. Such testing would reveal usability issues, deployment failures and candidate concerns that may not appear in small controlled experiments.

XXIII. CONCLUSION AND FUTURE WORK

This paper presented a multimodal AI-based online proctoring system for secure coding assessments. The proposed system integrates behavioural login risk analysis, OTP verification, browser-based coding, MTCNN face detection, MediaPipe FaceMesh gaze estimation, YOLOv8 object detection, audio activity monitoring, cooldown-based event aggregation and automated report generation. The work contributes both an implementation and a research formulation. Mathematical equations define the key decision functions, while pseudocode describes authentication, monitoring and report generation. The case-study evaluation using generated project logs demonstrates that the prototype can capture multiple categories of suspicious behaviour and preserve them as structured evidence. Future work will focus on labelled datasets, calibrated temporal models, transformer- or CNN-LSTM-based behaviour modelling, secure containerized code execution, plagiarism detection, encrypted database-backed dashboards and large-scale validation with diverse users and environments. These additions would move the prototype from a strong academic proof of concept toward a deployable institutional assessment platform.

REFERENCES

[1] M. A. Khan and S. S. Islam, 'Online exam proctoring system using artificial intelligence,' International Journal of Advanced Computer Science and Applications, vol. 11, no. 6, pp. 1-8, 2020.
[2] Q. Ji and X. Yang, 'Real-time eye, gaze, and face pose tracking for monitoring driver vigilance,' Real-Time Imaging, vol. 8, no. 5, pp. 357-377, 2002.
[3] A. Krizhevsky, I. Sutskever and G. Hinton, 'ImageNet classification with deep convolutional neural networks,' Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.
[4] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
[5] J. Donahue et al., 'Long-term recurrent convolutional networks for visual recognition and description,' IEEE Conference on Computer Vision and Pattern Recognition, pp. 2625-2634, 2015.
[6] A. Vaswani et al., 'Attention is all you need,' Advances in Neural Information Processing Systems, pp. 5998-6008, 2017.
[7] Y. Li, M. Liu and Z. Zhang, 'Deep learning-based intelligent online exam proctoring system,' IEEE Access, vol. 9, pp. 123-134, 2021.
[8] T. L. Dang et al., 'Auto-proctoring using computer vision in MOOCs system,' Multimedia Tools and Applications, vol. 84, pp. 26187-26213, 2024.
[9] M. Aly, 'Revolutionizing online education: Advanced facial expression recognition for real-time student progress tracking via deep learning model,' Multimedia Tools and Applications, vol. 84, pp. 12575-12614, 2025.

[10] Y. Liu et al., 'Multiple instance learning for cheating detection and localization in online examinations,' IEEE Transactions on Cognitive and Developmental Systems, vol. 16, no. 4, pp. 1315-1326, 2024.
[11] B. Qiao et al., 'Dispelling the fake: Social bot detection based on edge confidence evaluation,' IEEE Transactions on Neural Networks and Learning Systems, vol. 36, no. 4, pp. 7302-7315, 2025.
[12] T. Potluri and V. P. K. Sistla, 'Survey on AI-based automated online proctoring systems,' 2024.
[13] S. A. Shahzad et al., 'AV-Lip-Sync+: Multimodal inconsistency for deepfake detection,' 2025.
[14] D. De Vito et al., 'Using GenAI to assess design patterns in student-written code,' 2025.
[15] G. Bradski, 'The OpenCV library,' Dr. Dobb's Journal of Software Tools, 2000.
[16] A. Bochkovskiy, C. Wang and H. Liao, 'YOLOv4: Optimal speed and accuracy of object detection,' arXiv, 2020.
[17] J. Redmon et al., 'You only look once: Unified, real-time object detection,' IEEE CVPR, pp. 779-788, 2016.
[18] F. Schroff, D. Kalenichenko and J. Philbin, 'FaceNet: A unified embedding for face recognition and clustering,' IEEE CVPR, pp. 815-823, 2015.
[19] A. Radford et al., 'Robust speech recognition via large-scale weak supervision,' International Conference on Machine Learning, pp. 28492-28518, 2023.
[20] M. Sandler et al., 'MobileNetV2: Inverted residuals and linear bottlenecks,' IEEE CVPR, pp. 4510-4520, 2018.