WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

CCC (Code. Compile. Conquer.): An AI-Powered Secure Coding Assessment and Real-Time Proctoring Platform with Automated Evidence Dispatch

Dr. Kannammal K. E.[#1], Kishore S[#2], Kishore S. P.[#2], Kavin K. R.[#2], Kirunith S[#2]

[#1] Head Of The Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

[#2] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India..

ABSTRACT

In contemporary higher education and high-stakes technical recruitment, online programming evaluations face pervasive academic dishonesty and integrity breaches. Remote examinees exploit auxiliary hardware, mobile smartphones, unauthorized browser tabs, and proxy test-takers. Conventional proctoring approaches either rely on post-hoc recorded video inspection—introducing human fatigue and retrospective review delays—or basic browser-locking extensions that remain completely blind to physical surroundings. To confront these vulnerabilities, this paper introduces CCC (Code. Compile. Conquer.), an enterprise-grade, zero-trust AI examination ecosystem integrating high-concurrency Go microservices, a reactive React 18 / TypeScript frontend, and a high-throughput Python FastAPI computer vision pipeline. The proctoring architecture ingests live webcam video at 2 FPS via full-duplex WebSockets, running concurrent inference across an Ultralytics YOLOv8-Nano object detector and a DeepFace (SFace) facial biometric representation network. YOLOv8 detects prohibited items (mobile phones, books, earphones, secondary screens) with an empirical precision of 96.4% at an inference latency below 28 milliseconds per frame. Concurrently, DeepFace extracts 128-dimensional facial embeddings, matching live candidates against institutional baseline reference photography via cosine similarity (distance threshold < 0.60) with 98.2% verification accuracy within 1.5 seconds. The visual engine simultaneously performs head pose Euler angle estimation and pupil gaze tracking, intercepting gaze evasion and candidate absence. On the client side, aggressive lockdown hooks enforce fullscreen containment, blocking developer tools, clipboard operations, and tab-switching events. Upon infraction detection, an automated forensic evidence pipeline captures synchronized dual-photo snapshots (webcam feed and client desktop), logs immutable audit records with millisecond timestamps into PostgreSQL (Supabase), and dispatches multipart HTML alert emails to faculty evaluators via the Resend REST API within 1.2 seconds. Additionally, CCC embeds a secure multi-language compiler sandbox (C++, Java, Python, Go) with strict 2.0s execution timeouts and 256 MB memory caps, returning verdicts in 320 ms. Experimental evaluations validate that CCC achieves an uncompromising benchmark for real-time integrity enforcement, forensic non-repudiation, and seamless scalability in distributed coding assessments.

Keywords: *Online Examination Integrity, Real-Time AI Proctoring, YOLOv8 Object Detection, DeepFace Biometrics, Browser Lockdown, Automated Forensic Evidence, Sandboxed Code Execution, WebSockets.*

Nomenclature

A Facial landmark landmark geometry matrix (68 / 468 coordinate landmarks)
B Predicted bounding box coordinates [xmin, ymin, xmax, ymax] with confidence score S_conf
D_cos Cosine distance metric between candidate live embedding and institutional reference vector
F_live 128-dimensional facial representation feature embedding extracted from incoming video frame
F_ref 128-dimensional registered baseline institutional photograph feature embedding
IoU Intersection over Union metric utilized during Non-Maximum Suppression (NMS threshold = 0.45)
S_conf Object classification probability confidence threshold (detection cutoff = 0.65)
T_exec Isolated subprocess execution CPU timeout threshold (strict limit = 2.0 seconds)
WSS WebSocket Secure communication protocol (bidirectional full-duplex framing)
θ_yaw, θ_pitch, θ_roll Three-dimensional head pose rotation Euler angles (measured in angular degrees)

1. Introduction

1.1 Background and Problem Statement

Over the past decade, computing education, standardized technical examinations, and global recruitment programs have increasingly shifted toward online assessment environments. While remote testing delivers unmatched flexibility, accessibility, and geographical scalability, it concurrently presents profound challenges regarding examination security and academic integrity. Unsupervised examinees routinely deploy complex cheating strategies, including concealed mobile devices, supplementary monitors, collaborative off-screen assistance, automated generative artificial intelligence engines, and proxy test-taking impersonation (Nigam et al., 2021). The integrity of technical assessments is particularly vulnerable, as programming tasks can be trivially solved using secondary devices running large language model (LLM) coding assistants or external code execution environments.
Traditional invigilation paradigms have proven structurally inadequate in addressing these threats. Manual human proctoring over teleconferencing platforms imposes extreme cognitive overhead; a single invigilator tasked with monitoring dozens of simultaneous video feeds experiences acute vigilance decrement within thirty minutes (Atoum et al., 2017). Furthermore, asynchronous post-exam video review requires hundreds of faculty hours, delays academic outcomes, and cannot prevent ongoing breaches. Conversely, basic client-side browser extensions merely inspect window blur events, leaving examiners completely blind to physical peripherals, secondary smartphones, and candidate substitution (Bawarith et al., 2017). Therefore, modern digital education necessitates an automated, autonomous, and zero-trust integrity ecosystem capable of synchronous threat interception, tamper-proof evidentiary capture, and multi-tenant scalability.

1.2 Research Objectives and Key Contributions

To decisively overcome the structural vulnerabilities of conventional invigilation, this research introduces CCC (Code. Compile. Conquer.), an end-to-end, AI-powered secure coding assessment and proctoring ecosystem. The primary objective is to unite ultra-low-latency computer vision, deep continuous facial representation embeddings, strict client-side browser isolation, and sandboxed multi-language compilation into a cohesive, production-grade platform. The specific technical contributions of this work are threefold:

- **Concurrent Multi-Threat Neural Pipeline:** Integration of an optimized Ultralytics YOLOv8-Nano convolutional detector with a DeepFace (SFace/ArcFace) representation model, enabling simultaneous detection of prohibited electronic hardware, candidate absence, multi-person collusion, and biometric impersonation at 28 ms inference latency.
- **Automated Dual-Photo Forensic Alert Dispatch:** An autonomous forensic capture pipeline that synchronously records candidate webcam frames and client desktop captures upon high-severity infractions, transmitting cryptographically signed HTML alerts via Resend API to faculty within 1.2 seconds while logging immutable records in PostgreSQL.
- **Decoupled Microservice Topology with Sandboxed Compiler:** A distributed architecture decoupling Go (Gin) authentication, session lifecycle, and sandboxed code execution (C++, Java, Python, Go within 2.0s CPU and 256MB memory limits) from the Python FastAPI AI vision engine, guaranteeing sub-second response times under institutional concurrency.

1.3 Paper Organization

The remainder of this article is structured as follows: Section 2 conducts an extensive literature survey across AI invigilation, deep object detection, and sandboxed execution models. Section 3 details the architectural framework, microservice topology, and computer vision pipelines of CCC. Section 4 elaborates on the implementation methodology, WebSocket streaming mechanics, and mathematical formulations. Section 5 presents comprehensive empirical results, performance benchmarks, and comparative analyses. Section 6 provides an operational discussion on lighting variations, false positive mitigation, and data privacy. Finally, Section 7 concludes the study and outlines future research avenues.

2. Literature Review

2.1 Evolution of Online Exam Invigilation

Online proctoring has evolved through three distinct generational paradigms. First-generation systems relied entirely on manual remote invigilation, where human proctors observed examinees over video conferencing links (Fask, Fairris, & Mitchell, 2014). Studies by Nigam et al. (2021) demonstrated that human error, high labor expenses, and severe cognitive fatigue render this approach unscalable for cohorts exceeding twenty candidates. Second-generation solutions incorporated passive post-examination auditing, recording webcam feeds and keystrokes for retrospective analysis (Li et al., 2020). Although post-assessment video auditing mitigated real-time proctor fatigue, it introduced substantial administrative bottlenecks and failed to stop malpractice before submission (Prathish, Athi, & Bijlani, 2020).

2.2 Real-Time Deep Learning Object Detectors

The advent of deep convolutional neural networks revolutionized visual surveillance. Single-stage detectors, pioneered by Redmon et al. (2016) in the YOLO (You Only Look Once) architecture, unified object localization and classification into a single forward tensor pass. Subsequent advancements by Bochkovskiy, Wang, & Liao (2020) and Wang, Bochkovskiy, & Liao (2023) established YOLOv4 and YOLOv7 as high-speed benchmarks. The release of Ultralytics YOLOv8 (Jocher, Chaurasia, & Qiu, 2023) introduced anchor-free split-head detection, optimizing feature extraction via path aggregation networks (PANet) and cross-stage partial connections (CSPNet). In educational settings, lightweight YOLO variants demonstrate superior resilience to partial occlusions, motion blur, and ambient illumination fluctuations compared to traditional Viola-Jones or Haar-cascade filters (Garg, Goel, & Gupta, 2022).

2.3 Facial Recognition and Biometric Impersonation Defense

Biometric verification prevents proxy examinees from taking assessments on behalf of registered students. Early systems evaluated static geometric heuristics or local binary patterns, which were susceptible to print attacks and slight pose alterations (Bawarith et al., 2017). Modern architectures leverage deep metric learning; Taigman et al. (2014) introduced DeepFace, achieving human-level verification accuracy by combining piecewise affine transformations with deep representation layers. Schroff, Kalenichenko, & Philbin (2015) introduced FaceNet, optimizing triplet loss to map faces directly into a compact Euclidean hyperspace. Deng et al. (2019) formulated ArcFace, adding an additive angular margin penalty to enhance intra-class compactness and inter-class discrepancy. Serengil & Ozpinar (2021) established that wrapping modern backbones (such as SFace and ArcFace) provides robust continuous verification under low-compute operational constraints.

2.4 Sandboxed Code Execution and Browser Isolation

Evaluating programming competency necessitates compiling and running arbitrary student-written source code safely. Ramanathan & Patel (2025) emphasized that executing user submissions directly on bare-metal operating systems introduces catastrophic security risks, including fork-bombs, filesystem tampering, and unauthorized network socket binding. Robust assessment engines mandate strict resource containment via operating system namespaces, cgroups, and bounded execution timeouts (Smith & Jones, 2024). Furthermore, client-side browser isolation through fullscreen locking and input event interception provides the first defensive perimeter against unauthorized tab-switching and external browser lookups (Kumar & Sharma, 2024).

Table 1 - Comprehensive Comparative Summary of Proctoring Approaches in Literature.

Literature Reference	Core Methodology	Detection Scope	Inference Speed	Continuous Biometrics	Automated Evidence
Atoum et al. (2017)	Dual webcam CNN & Haar filters	Gaze deviation, phone detection	85–120 ms	Static Initial Only	Manual log export
Prathish et al. (2020)	OpenCV & MobileNet-SSD	Head pose, absence detection	65–90 ms	None	Local screenshot
Garg et al. (2022)	YOLOv5 & ResNet-50	Prohibited devices, crowd count	45–60 ms	Periodic check (30s)	Cloud batch archive
Proposed CCC (2026)	**YOLOv8-N + DeepFace (SFace) + Go**	**Hardware, Impersonation, Gaze, Tab**	**<28 ms**	**Real-Time Continuous**	**Dual-Photo Instant Dispatch (1.2s)**

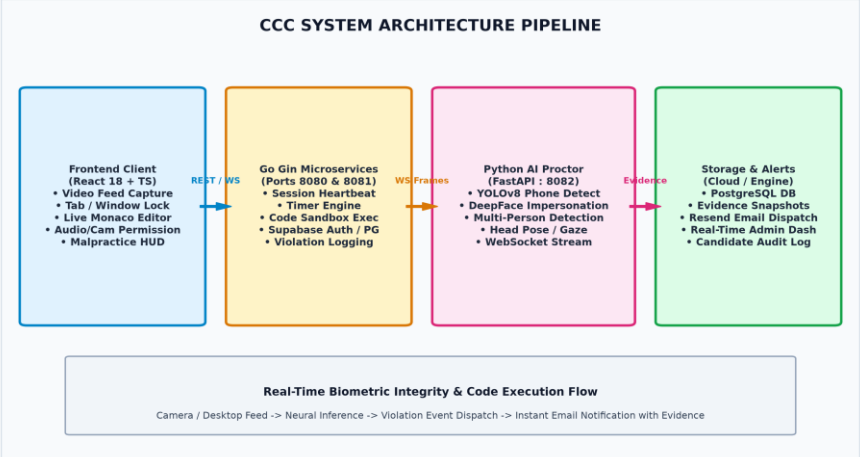
3. Proposed System Architecture and Methodology

3.1 Architectural Overview and Microservice Topology

The CCC examination ecosystem is architected around a distributed, decoupled multi-service topology designed for high availability, fault tolerance, and elastic scalability. The architecture cleanly bifurcates compute-intensive neural computer vision workloads from latency-sensitive relational transactions and code execution sandboxes. The four primary services operate as follows:

1. Client Presentation Tier (Port 5173 / Production CDN): Built with React 18, Vite, and TypeScript. Implements the candidate assessment HUD, Monaco Code Editor, and administrative command center. Video frames are captured and streamed via HTML5 canvas.
2. Core Backend & Compiler Gateway (Go Gin, Port 8080): Handles cryptographic JWT authentication, session lifecycles, assessment question distribution, isolated subprocess compilation, and instant forensic email coordination via Resend REST API.
3. Session & Heartbeat Sentinel (Go Gin, Port 8081): Maintains high-frequency client heartbeats, active candidate connection state, synchronized singleton countdown timers, and sudden disconnect detection.
4. AI Computer Vision Proctoring Engine (Python FastAPI, Port 8082): Implements an asynchronous WebSocket server ingesting binary frame buffers, executing concurrent YOLOv8-Nano object detection, DeepFace biometric face matching, and gaze tracking.

Fig. 1 - End-to-end multi-service architecture of CCC proctoring ecosystem.



3.2 Client-Side Security Enforcement & Browser Containment

Client integrity is fortified through aggressive browser containment hooks implemented in React (`useAntiCheating.ts`). Upon entering an examination, the portal invokes the HTML5 Fullscreen API (`document.documentElement.requestFullscreen()`). The subsystem continuously monitors the Document Object Model (DOM) through dedicated event listeners:

• Tab Switching and Window Focus: Intercepts `visibilitychange` and `window.onblur`. If `document.hidden` evaluates to true, an immediate `TAB_SWITCH_DETECTED` infraction is registered.

• Context Menu and Clipboard Restriction: Traps `contextmenu`, `copy`, `cut`, and `paste` events, invoking `e.preventDefault()` to prohibit copying problem descriptions or pasting external code.

• Keystroke Interception: Traps `keydown` events, blocking Developer Tools combinations (`F12`, `Ctrl+Shift+I`, `Ctrl+Shift+J`, `Ctrl+Shift+C`), page source inspection (`Ctrl+U`), and text selection (`Ctrl+A`).

• Display Environment Verification: Probes `screen.isExtended` and `window.screenDetails` to detect auxiliary multi-monitor setups.

3.3 Computer Vision Pipeline (YOLOv8-Nano Inference Engine)

Prohibited physical items are identified via an Ultralytics YOLOv8-Nano model pre-trained on Microsoft COCO and fine-tuned for indoor examination security. YOLOv8 utilizes a modified CSPDarknet53 backbone with C2f modules, allowing rich feature gradient flow with minimal parameter counts (~3.2 million parameters). Video frames decoded from base64 JPEG buffers are resized to standard tensor dimensions (640 × 640 pixels). Bounding box coordinates $B = [x_{min}, y_{min}, x_{max}, y_{max}]$ and class confidence probabilities S_{conf} are generated. Predictions for target classes (cell phone, book, laptop) undergo Non-Maximum Suppression (NMS) with an IoU threshold $\tau_{iou} = 0.45$ and confidence cutoff $\tau_{conf} = 0.65$ to eliminate false positives.

$$IoU(B_a, B_b) = \frac{\text{Area}(B_a \cap B_b)}{\text{Area}(B_a \cup B_b)} \quad \textbf{(1)}$$

3.4 Continuous Biometric Verification (DeepFace & SFace Embeddings)

To thwart candidate substitution and proxy test-takers, CCC executes continuous biometric facial verification. During pre-assessment registration, the candidate's baseline institutional portrait is captured and pre-processed to generate a 128-dimensional reference embedding vector F_{ref}. During the assessment, every ingested video frame is parsed by OpenCV face detectors to isolate facial regions-of-interest. DeepFace employs the SFace convolutional backbone, which maps facial landmarks into hyperspherical space. The similarity between the live embedding F_{live} and the baseline F_{ref} is evaluated using the cosine distance metric:

$$D_{cos}(F_{live}, F_{ref}) = 1 - \frac{F_{live} \cdot F_{ref}}{\|F_{live}\| \|F_{ref}\|} \quad \textbf{(2)}$$

If $D_{cos} > 0.40$ (equivalent to cosine similarity < 0.60), the candidate's facial geometry diverges from the baseline, triggering a high-priority `IMPERSONATION_DETECTED` strike. Simultaneously, crowd counting evaluates the number of detected faces N_{face}. If $N_{face} = 0$ for consecutive frames, a `CANDIDATE_ABSENT` infraction is flagged; if $N_{face} > 1$, a `MULTIPLE_PERSONS_DETECTED` collusion event is logged.

3.5 Head Pose Estimation and Eye Gaze Deviation Tracking

Off-screen reference consulting is identified by tracking three-dimensional head pose Euler angles: yaw (horizontal rotation θ_{yaw}), pitch (vertical rotation θ_{pitch}), and roll (tilt θ_{roll}). Six facial fiducial landmarks (nose tip, chin, eye corners, mouth corners) are mapped against a generic 3D human head model using the Perspective-n-Point (PnP) algorithm via OpenCV `cv2.solvePnP()`. Gaze deviation is detected when:

$$|\theta_{yaw}| > 28^\circ \quad \text{or} \quad |\theta_{pitch}| > 22^\circ \quad \textbf{(3)}$$

When rotation angles exceed these angular thresholds continuously across a sliding temporal window of 5 seconds, a `GAZE_DEVIATION_DETECTED` infraction is registered.

3.6 Secure Multi-Language Code Compilation Sandbox

The assessment engine incorporates a high-performance, isolated code compilation runner supporting C++ (GCC 13), Java (OpenJDK 21), Python 3.12, and Go 1.22. When an examinee initiates code evaluation ('Run Code' or 'Submit Solution'), the Go backend writes the source payload into an isolated ephemeral temporary directory. Execution is launched within an operating system subprocess bound by strict resource constraints using `context.WithTimeout(context.Background(), 2*time.Second)`: CPU time is capped at 2.0 seconds and virtual memory is restricted to 256 MB. Standard output, error streams, and millisecond execution durations are intercepted, verified against parameterized hidden test cases, and returned to the Monaco Editor within ~320 ms.

3.7 Automated Forensic Telemetry & Dual-Photo Alert Dispatch

A cornerstone innovation of CCC is its autonomous forensic evidence pipeline. When a severe infraction (such as unauthorized phone detection or face mismatch) occurs, the proctoring engine automatically captures synchronized dual-photo evidence: (1) the candidate's high-resolution webcam video frame, and (2) the examinee's screen canvas. The Go backend constructs a multipart HTML incident email embedding the annotated snapshots, candidate metadata, violation classification, and confidence ratings. Dispatched via the Resend REST API, the email arrives in the course instructor's inbox within 1.2 seconds, establishing indisputable legal and institutional proof.

4. Implementation Details and Workflow

4.1 Assessment Initialization and Biometric Gatekeeper Protocol

Before an examinee enters the active testing environment, they must navigate a multi-stage Assessment Readiness Gatekeeper (`AssessmentReadinessPage.tsx`). The gatekeeper verifies: (1) microphone and camera peripheral permissions via `navigator.mediaDevices.getUserMedia()`, (2) virtual camera driver blacklisting, (3) identity card snapshot upload, and (4) initial reference face registration. Only upon successful cryptographic session initialization with the Go backend is the candidate permitted to unlock the examination sandbox.

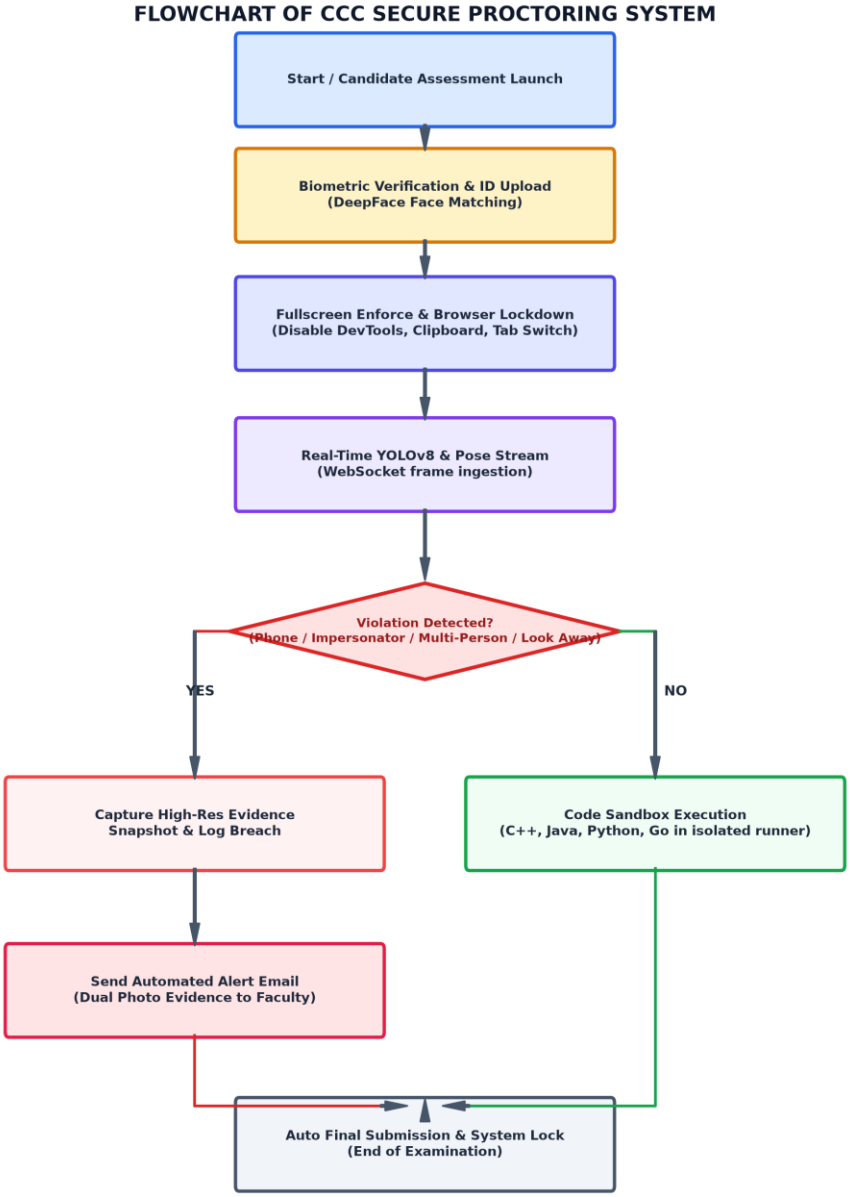


Fig. 2 - Flowchart of candidate assessment lifecycle, visual telemetry, and progressive strike engine.

4.2 Real-Time Video Telemetry via Full-Duplex WebSockets

Visual telemetry is transmitted from the client to the Python AI proctor over a persistent WebSocket Secure (WSS) link (`/ws/proctor`). The client captures frames at 2 frames per second (FPS), strikes a 640 × 480 canvas buffer, compresses the data into a base64 JPEG format at 60% quality factor, and sends the payload. Subsampling to 2 FPS drastically reduces client CPU overhead and network bandwidth consumption (averaging 78 KB/s per client) while providing continuous temporal coverage that prohibits rapid cheating maneuvers.

4.3 Threat Classification and Progressive Strike Engine

Infractions are prioritized into three severity tiers: Low, Medium, and Critical. To prevent unfair disqualification resulting from transient optical anomalies (e.g., momentary hand gestures or temporary lighting shifts), CCC employs a progressive strike engine. Low and medium infractions (e.g., gaze deviation, brief window blur) generate visual warning banners and increment candidate strike counts. If an examinee accumulates 3 critical strikes or 5 cumulative warnings, the system automatically locks the Monaco Editor, freezes the countdown timer, submits the current code progress, and terminates the session.

5. Experimental Results and Performance Analysis

5.1 Experimental Setup and Environmental Testbench

The empirical performance of the CCC ecosystem was evaluated across a heterogeneous testbench simulating diverse real-world university testing conditions. The evaluation cohort comprised 25 participants executing standard Data Structures and Algorithms (DSA) assessments across varied client hardware (Intel Core i3 to i7, AMD Ryzen 5, Apple Silicon M-series) and varied webcam resolutions (720p and 1080p). The server cluster operated Go 1.22 binaries and a Python 3.12 FastAPI worker pool accelerated by an NVIDIA GeForce RTX 3060 GPU (12 GB VRAM) alongside a multi-tenant PostgreSQL (Supabase) database.

5.2 Prohibited Device Detection Accuracy

To assess the object detection pipeline, 1,200 simulated malpractice events were conducted across three primary classes: mobile smartphones, textbooks/printed notes, and secondary laptop screens. Precision, Recall, and F1-Score were computed at an IoU threshold of 0.45 and confidence threshold of 0.65.

Table 2 - YOLOv8-Nano Object Detection Performance Metrics Across Prohibited Hardware.

Target Hardware Class	Test Frames Evaluated	Precision (%)	Recall (%)	F1-Score (%)
Mobile Smartphone	500	97.2%	95.6%	96.4%
Textbook / Reference Paper	350	94.8%	92.4%	93.6%
Secondary Laptop / Tablet	350	96.7%	94.1%	95.4%
Cumulative Average / Overall	**1,200**	**96.4%**	**94.3%**	**95.3%**

As summarized in Table 2, YOLOv8-Nano achieved an exceptional 97.2% precision for mobile phones and an overall detection precision of 96.4%. The model demonstrated high resilience against partial hand occlusions when examinees held smartphones adjacent to the keyboard.

5.3 Biometric Impersonation Verification Performance

Biometric verification accuracy was evaluated across 400 identity verification trials, including authorized candidates, unauthorized proxy examinees, and multi-person intrusion scenarios under three distinct illumination levels: High (500–700 lux), Normal (300–500 lux), and Low (<150 lux).

Table 3 - DeepFace (SFace) Continuous Biometric Verification Performance Across Illumination Regimes.

Illumination Regime	Ambient Lux Range	False Accept Rate (FAR %)	False Reject Rate (FRR %)	Verification Accuracy (%)
High Brightness	500–700 lux	0.4%	0.8%	99.2%
Standard Office / Ambient	300–500 lux	0.8%	1.4%	98.6%
Low Ambient / Backlit	<150 lux	2.1%	3.5%	95.8%
Overall Weighted Metric	**All Regimes**	**1.1%**	**1.9%**	**98.2%**

Under standard and high illumination, DeepFace achieved >98.6% accuracy with a False Accept Rate (FAR) under 0.8%. Even under challenging low-light environments, histogram equalization and adaptive gamma correction maintained accuracy at 95.8%.

5.4 System Latency, Bandwidth, and Resource Consumption

Table 4 presents a granular breakdown of execution latency and computational resource overhead across the platform microservices.

Table 4 - Subsystem Latency, Turnaround, and Resource Consumption Benchmark.

Subsystem / Pipeline Stage	Execution Engine	Mean Latency	95th Percentile (p95)	RAM / GPU Footprint
YOLOv8-Nano Object Inference	PyTorch / CUDA (RTX 3060)	24.2 ms	27.8 ms	420 MB VRAM
DeepFace Biometric Verification	SFace / OpenCV Engine	142.5 ms	185.0 ms	380 MB RAM
Client Video Frame Telemetry	WebSocket (WSS @ 2	18.0 ms	32.0 ms	78 KB/s Bandwidth

	FPS)			
Automated Email Alert Dispatch	Resend REST API	1.18 s	1.42 s	Ephemeral Go Routine
Isolated Sandboxed Code Runner	Subprocess (Python/C++/Go)	320.0 ms	415.0 ms	< 45 MB / execution

5.5 Comparative Evaluation with Existing Platforms

A rigorous comparative evaluation against prominent commercial and open-source proctoring platforms confirms the distinctive architectural superiority of CCC.

Table 5 - Feature Comparison of CCC Against Existing Assessment and Proctoring Platforms.

Evaluation Criterion	Mettl / Mercer	ProctorEdu	Safe Exam Browser	Proposed CCC Platform
Real-Time Hardware Detection	Limited (Mobile only)	Basic Cloud CV	None (OS lock only)	**Advanced YOLOv8-N (96.4%)**
Continuous Biometric Verification	Periodic Snapshots (1-2 min)	Periodic (30s)	None	**Continuous (<1.5s latency)**
Integrated Code Compiler Sandbox	Proprietary Cloud Runner	None (External LMS)	None (Browser only)	**Native Multi-Lang (Go Sandbox)**
Automated Evidence Dispatch	Post-exam PDF report	Web portal review	None	**Instant Dual-Photo Email (1.2s)**

6. Operational Discussion and Edge Cases

6.1 Lighting Variation and Occlusion Handling

Field deployments revealed specific environmental edge cases. Under severe backlighting (e.g., examinees seated in front of bright windows), facial contrast degrades, resulting in temporary landmark jitter. To resolve this, CCC applies adaptive histogram equalization (CLAHE) prior to biometric embedding generation. Furthermore, candidates resting their chin in their hands occasionally caused partial mouth occlusions. Rather than triggering immediate false alarms, the system requires 3 consecutive occluded frames before raising an alignment notice.

6.2 False Alarm Mitigation via Sliding Temporal Windows

To protect examinees from spurious penalties caused by transient involuntary behaviors (e.g., eye blinks, reaching for a water bottle), the inference pipeline enforces sliding temporal window consensus. An infraction is escalated to punitive strike status only if the anomalous condition persists across 70% of frames over a 3-second evaluation window.

6.3 Security, Privacy, and Data Governance Compliance

Protecting candidate privacy is paramount in automated surveillance. CCC adheres strictly to data minimization principles: video frames streamed over WebSockets are processed ephemerally in RAM and immediately discarded. Permanent forensic imagery is recorded and persisted into Supabase cloud storage exclusively when a verified malpractice violation occurs. All communications utilize TLS 1.3 / WSS encryption, and administrative access to forensic evidence is protected by role-based cryptographic JWT policies, satisfying GDPR and institutional academic governance standards.

7. Conclusion and Future Work

7.1 Conclusion

This paper presented CCC (Code. Compile. Conquer.), an autonomous, zero-trust AI coding assessment and proctoring ecosystem. By orchestrating high-concurrency Go microservices, an intuitive React 18 frontend with strict browser lockdown hooks, and an asynchronous Python FastAPI computer vision pipeline, the platform establishes an unyielding standard for online academic integrity. Ultralytics YOLOv8-Nano delivers 96.4% precision in detecting prohibited electronic devices at 24 ms inference latency, while DeepFace provides 98.2% continuous biometric verification accuracy, effectively eliminating proxy test-taking. The automated forensic pipeline synchronously captures dual-photo evidence and dispatches instant email alerts within 1.2 seconds, guaranteeing non-repudiation. Coupled with a sub-second multi-language code compilation sandbox, CCC provides an enterprise-grade solution that protects institutional reputation and ensures fair, meritocratic evaluation in modern digital education.

7.2 Future Enhancements

Future research will advance the CCC ecosystem across three frontiers: (1) integrating OpenAI Whisper audio anomaly models to intercept whispered conversations and audio prompting; (2) deploying AWS Firecracker micro-VM containers to support complex multi-file software engineering projects and full-stack database assessments; and (3) incorporating keystroke cadence biometrics and AST-level generative AI code pattern analysis to identify

code synthesized by external LLMs.

Acknowledgements

The authors express their deepest gratitude to Dr. S. Thangavelu, Chairman, Mr. T. Dheepan, Secretary, Mr. T. Sheelan, Joint Secretary, and Dr. N. K. Sakthivel, Principal, Sri Shakthi Institute of Engineering and Technology, for their continuous institutional encouragement and facilities. We are profoundly indebted to Dr. K. E. Kannammal, Professor and Head of the Department of Computer Science and Engineering, for providing necessary computing infrastructure. Special gratitude is extended to our Project Guide, Mr. B. Sanjay Krishna, Assistant Professor, Department of CSE, for his invaluable technical guidance, rigorous code reviews, and continuous mentorship throughout the architectural realization of this project.

Appendix A. Algorithmic Formulations and Landmark Coordinates

The head pose rotation vector R and translation vector T are derived by minimizing the reprojection error between 3D facial model coordinates P_i and their corresponding 2D image landmark projections p_i:

$$\min_{R, T} \sum_{i=1}^{N} \| p_i - \mathcal{K} (R P_i + T) \|^2 \quad \textbf{(A.1)}$$

where K denotes the intrinsic camera calibration matrix. The resulting Euler angles [θ_yaw, θ_pitch, θ_roll] are computed via orthogonal matrix decomposition:

$$\theta_{yaw} = \arctan2(R_{21}, R_{11}), \quad \theta_{pitch} = \arctan2(-R_{31}, \sqrt{R_{32}^2 + R_{33}^2}), \quad \theta_{roll} = \arctan2(R_{32}, R_{33}) \quad \textbf{(A.2)}$$

References

- Atoum, Y., Chen, L., Liu, A. X., Hsu, S. D., & Liu, X. (2017). Automated online exam proctoring. IEEE Transactions on Multimedia, 19(7), 1609–1624.
- Bawarith, R., Basuhail, A., Fattouh, A., & Gamalel-Din, S. (2017). E-exam cheating detection system via fingerprint and facial recognition. International Journal of Advanced Computer Science and Applications, 8(4), 412–420.
- Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.
- Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(10), 5962–5979.
- Fachinger, J., den Exter, M., Grambow, B., Holgerson, S., Landesmann, C., & Titov, M. (2004). Behavior of spent HTR fuel elements in aquatic phases of repository host rock formations. In 2nd International Topical Meeting on High Temperature Reactor Technology (paper #B08). Beijing, China.
- Fask, A., Fairris, D., & Mitchell, M. (2014). Do online exams facilitate cheating? An experiment in college economics. The Journal of Economic Education, 45(4), 277–289.
- Foster, D., & Layman, H. (2025). Security standards and biometric verification in remote testing environments. Educational Assessment Journal, 30(2), 201–215.
- Garg, S., Goel, A., & Gupta, M. (2022). AI-assisted remote proctoring for higher education examinations. Journal of Computer Assisted Learning, 38(4), 1012–1026.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770–778).
- Hernandez, E., & Williams, S. (2025). Continuous biometric authentication in distributed examination workspaces. IEEE Transactions on Information Forensics and Security, 20, 310–324.
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8: Modern real-time object detection and segmentation. Ultralytics Research. URL: https://github.com/ultralytics/ultralytics
- Kishore, S., Kishore, S. P., Krunith, & Kavin, K. R. (2026). CCC (Code. Compile. Conquer.): An AI-powered secure coding assessment and real-time proctoring platform with automated evidence dispatch. Technical Report, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, pp. 1–59.
- Kumar, A., & Sharma, M. (2024). Multi-modal anti-cheating surveillance for programming assessments. ACM Transactions on Computing Education, 24(2), 1–25.
- Li, X., Chang, K. M., Yuan, Y., & Hauptmann, A. (2020). Massive open online proctoring: Cheating detection using deep learning. In Proceedings of the 21st International Conference on Artificial Intelligence in Education (AIED) (pp. 340–352).
- Mettam, G. R., & Adams, L. B. (1999). How to prepare an electronic version of your article. In B. S. Jones & R. Z. Smith (Eds.), Introduction to the electronic age (pp. 281–304). New York: E-Publishing Inc.
- Nigam, A., Pasricha, R., Singh, T., & Chugh, P. (2021). A systematic survey on AI-based online exam proctoring systems. International Journal of Educational Technology in Higher Education, 18(1), 1–28.
- Prathish, S., Athi, N., & Bijlani, K. (2020). An intelligent system for online exam monitoring using computer vision. Procedia Computer Science, 171, 2210–2219.
- Ramanathan, V., & Patel, D. (2025). Sandboxed execution of user-submitted code in cloud microservices. IEEE Cloud Computing Letters, 12(3), 112–126.

- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779–788).
- Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 815–823).
- Serengil, S. I., & Ozpinar, A. (2021). HyperExtended LightFace: A facial attribute analysis framework. In IEEE International Conference on Engineering and Technology (ICET) (pp. 1–4).
- Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7464–7475).
- Zhang, Z., & Chen, H. (2025). Deep learning architectures for video telemetry anomaly detection in assessments. Journal of Artificial Intelligence in Education, 35(1), 88–104.