ProctoAI: An AI-Powered Online Examination Proctoring System Using MERN Stack and Real-Time Computer Vision

Avinash Balasani, Pavan Bandaru, Sudhakar Banoth, Vamshi Banoth
Department of CSE (Artificial Intelligence & Machine Learning), ACE Engineering College
Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana, India
Under the Guidance of Mr Avinash , Associate Professor and Head, Department of CSE (AI & ML)

Abstract

The rise of online learning platforms has made it necessary to have strong ways to make sure that academic honesty is upheld during remote tests. This paper introduces ProctoAI, an AI-driven online exam proctoring system created with the MERN stack (MongoDB, Express.js, React.js, Node.js). The system uses TensorFlow.js and OpenCV-based Haar Cascade classifiers to give computers the ability to see in real time. This lets it keep an eye on students' behaviour through a webcam all the time. ProctoAI can find four types of suspicious activity: (1) when the student's face is missing, (2) when there are more than one face, (3) when a mobile phone is used with YOLO-based object detection, and (4) when a browser tab is switched. A rule-based weighted scoring system counts up violations and puts behaviour into one of three categories: Normal, Suspicious, or High Risk. Events that are detected are given a timestamp and saved in MongoDB. Instructors can access them through a special dashboard. Using JSON Web Tokens (JWT), the system lets students and teachers access it safely based on their roles. The tests show that violations are correctly detected, scores are correctly calculated, alerts are sent in real time, and the system works well for multiple users at the same time. ProctoAI is a scalable, automated alternative to human proctoring that makes online tests more fair and reliable while making them less dependent on manual supervision.

Keywords: *AI Proctoring, Online Examination System, MERN Stack, TensorFlow.js, YOLO, Computer Vision, Face Detection, Tab Switching Detection, Secure Authentication, Real-Time Monitoring*

1. Introduction

Events that are detected are given a timestamp and saved in MongoDB, where instructors can see them through a special dashboard. Using JSON Web Tokens (JWT), the system gives students and teachers safe role-based access. The results of the tests show that the system correctly detects violations, calculates scores accurately, sends alerts in real time, and works well with multiple users at the same time. ProctoAI is an automated, scalable alternative to human proctoring that makes online tests more fair and reliable while reducing the need for manual supervision.

People can help each other with their mobile phones or get to resources they shouldn't be able to by switching between browser tabs. Most online exam platforms only focus on delivering questions and collecting answers. They don't have good automated systems to make sure that the exam is fair. Video conferencing for manual proctoring partly fills this gap, but it doesn't work for large groups of candidates because it's not cost-effective or scalable. It's easy for one person to miss small violations when they're watching multiple video streams at once. This problem with scalability means that we need automated, smart monitoring systems that can work all the time, without getting tired. This paper introduces ProctoAI, an AI-driven online examination proctoring platform that incorporates computer vision and machine learning methodologies into a comprehensive web application. The system uses video frames from a webcam that are processed in real time to find suspicious activities. A rule-based scoring system looks at all the violations and gives teachers full behavioral logs through a special dashboard. The main

contributions of this work are: (1) a full-stack MERN implementation of an online exam platform with built-in AI proctoring; (2) a multi-modal cheating detection pipeline that uses face detection, YOLO-based object detection, head pose estimation, eye tracking, and browser activity monitoring; (3) a weighted rule-based violation scoring system; and (4) a teacher dashboard that lets teachers see what students are doing in real time and after the exam.

.

1.1 Research Gap and Motivation

Prior research on AI-driven proctoring has validated the viability of individual detection components in isolation: face detection systems (Viola & Jones, 2001), YOLO-based object detection (Redmon & Farhadi, 2018; Bochkovskiy et al., 2020), browser activity tracking (Gupta & Mehta, 2021), and deep learning behavior analysis (Lee et al., 2023; Verma & Gupta, 2022). But there isn't a single system that combines these modalities into a single, web-deployable, full-stack application that also lets you manage exams and process results. ProctoAI fills this gap by combining multi-modal detection with a MERN application that is ready for production and can be accessed through standard web browsers without the need for plugins.

2. Literature Survey

Sharma et al. (2022) created an automated online proctoring system that used webcam-based face detection and anomaly monitoring. They showed that real-time visual monitoring works well, but they also pointed out that it has trouble finding complex behaviors like using a mobile phone. Lee et al. (2023) suggested a proctoring system based on deep learning that was more accurate than manual monitoring, but it couldn't be used with a web-based exam platform. The YOLO object detection algorithm was created by Redmon and Farhadi in 2018. It made it possible to quickly identify objects in real time. This work is the basis for ProctoAI's mobile phone detection. Bochkovskiy et al. (2020) later released YOLOv5, which was better for real-time use because it was more accurate and faster. Gupta and Mehta (2021) investigated browser activity monitoring, encompassing tab switching and window focus alterations, and suggested a logging system to identify unauthorized resource access. This work worked on its own, but ProctoAI combines both types of monitoring. Viola and Jones (2001) developed real-time face detection utilizing Haar Cascade classifiers, which remain extensively used due to their computational efficiency. Patel and Shah (2022) built a web-based exam system with the MERN stack, but it didn't have AI proctoring features. This showed that the technology stack works.

Smilkov et al. (2019) released TensorFlow.js, which lets machine learning inference happen right in the browser. This cuts out the server round-trip latency that comes with server-side AI processing. Jones et al. (2015) established the utilisation of JSON Web Tokens for stateless, secure web application authentication, which ProctoAI employs for session management. Khan and Ali (2023) introduced a machine learning-driven cheating detection system that employs weighted activity scoring to categorise behaviour, a concept that closely parallels ProctoAI's violation scoring system.

The synthesis of this literature indicates that although individual detection components have been validated in previous studies, no system has effectively integrated multi-modal AI proctoring with comprehensive examination management within a browser-native, scalable architecture. Table 1 shows the main differences between ProctoAI and other systems that are similar.

Table 1. Comparison of Related Online Proctoring Systems

System / Study	Full-Stack Web	Face Detection	Object Detection	Browser Monitoring
Sharma et al. (2022)	No	Yes	No	No
Lee et al. (2023)	No	Yes	No	No
Gupta & Mehta (2021)	No	No	No	Yes
Patel & Shah (2022)	Yes	No	No	No
ProctoAI (Proposed)	Yes	Yes	Yes (YOLO)	Yes

3. System Requirements

3.1 Hardware Requirements

- Processor: Intel Core i5 or higher for real-time processing and AI model inference.
- RAM: Minimum 8 GB to support simultaneous video processing and web application execution.
- Storage: Minimum 256 GB for user data, exam records, and violation logs.
- Camera: High-resolution webcam for capturing clear student video during examinations.
- GPU (Optional): Dedicated graphics card to accelerate AI model inference.
- Network: Stable broadband internet connection for real-time communication and data transmission.

3.2 Software Requirements

- Operating System: Windows 10+, macOS, or Linux.
- Frontend: React.js, HTML5, CSS3, Material-UI.
- Backend: Node.js and Express.js.
- Database: MongoDB with Mongoose ORM.
- AI/ML Libraries: TensorFlow.js (browser-side inference), OpenCV (server-side face detection).
- Object Detection: YOLO for mobile phone detection.
- Authentication: JSON Web Tokens (JWT) and bcryptjs for password hashing.
- Development Tools: VS Code, npm, Git.
- Browser: Google Chrome or any modern browser supporting WebRTC and the Fullscreen API.

4. Methodology

4.1 System Architecture

ProctoAI is built on a client-server model. The React.js front end talks to the Node.js/Express.js back end through RESTful APIs that are protected by CORS policies. MongoDB is the permanent storage for user records, exam data, questions, results, and logs of violations. The backend has modular route handlers for managing users (/api/users), managing exams (/api/exams), processing results (/api/results), and storing exam logs (/api/exam-logs). Most of the time, the AI monitoring pipeline works on the client side. When a student starts an exam, React-Webcam turns on the camera on the device and takes pictures at regular intervals. TensorFlow.js processes these frames locally for real-time inference, which cuts down on server round-trip latency. API calls send detected violation

events to the backend, where they are stored in MongoDB. The Fullscreen Change API and the Visibility Change API are used to record events at the browser level, like switching tabs or leaving fullscreen mode.

4.2 System Modules

The system is made up of ten modules: (1) User Authentication Module — handles registration, login, password hashing via bcrypt, JWT token generation, and role-based access control; (2) Exam Management Module — lets teachers make exams, add MCQ questions, set the length, and schedule exams; (3) Webcam Capture and Monitoring Module — turns on the webcam and sends frames to the AI pipeline; (4) AI-Based Proctoring Module — combines face detection, multiple face detection, mobile phone detection, head pose estimation, and eye tracking; (5) Browser Activity Monitoring Module — keeps track of tab switching, fullscreen violations, and clipboard restrictions(6) Cheating Detection and Scoring Module — uses rule-based weighted scoring to add up violations and sort them by risk level; (7) Data Storage and Logging Module — stores all exam, user, and violation data in MongoDB; (8) Teacher Dashboard Module — gives teachers a way to look at student results and violation logs; (9) Result Processing Module — checks student answers against the correct ones and gives them scores; (10) User Interface Module — gives both students and teachers responsive React-based interfaces for their workflows.

4.3 AI Detection Pipeline

Face Detection: The OpenCV Haar Cascade classifiers look at each frame that is taken to see if the student's face is visible. A lack of a face in a frame sets off a violation record. A multiple-face violation is logged if more than one face is found in a single frame.

Detection of Mobile Phones: A YOLO object detection model looks at each frame to find things that are not allowed. When a mobile phone is found inside the bounding box, a cheating event is logged and the student is told about it.

Head Pose Estimation and Eye Tracking: Facial landmarks are used to figure out where the head is facing and where the eyes are looking. It is considered suspicious behaviour when someone repeatedly looks or turns their head away from the screen.Browser Activity Monitoring: JavaScript event listeners on the document.visibilitychange and fullscreenchange events capture tab switching and fullscreen exit attempts. Each such event increments a violation counter. After two tab-switching violations, the exam is automatically submitted, enforcing zero-tolerance for browser switching.

4.4 Violation Scoring Mechanism

The system employs a weighted additive scoring scheme to quantify suspicious activity. Each event type is assigned a severity weight based on its likelihood of indicating malpractice: looking away from the screen scores +1; no face detected scores +2; tab switching scores +2; multiple faces detected scores +3. Scores accumulate over the exam session to compute a total suspicion score. The score is used to classify student behavior into three risk categories: Normal (low cumulative score), Suspicious (moderate score), and High Risk (high score). Real-time alerts are displayed to students when violations are detected, and all events are logged with timestamps to the database.

4.5 Authentication and Security

User authentication uses JWT tokens with a 30-day expiry, stored in cookies. Passwords are hashed using bcrypt prior to database storage. Route-level middleware validates tokens on all protected API endpoints, enforcing role separation between student and teacher accounts. CORS is configured on the backend to allow only trusted origins (localhost and Vercel deployments), reducing the attack surface for cross-origin requests.

4.6 Data Model

The MongoDB database contains the following primary collections: Users (name, email, hashed password, role), Exams (examName, duration, scheduledDate, createdBy), Questions (exam reference, question text, options array, correctAnswer index), Results (user reference, exam reference, answers array, score, total, timestamps), and ExamLogs (examName, studentName, violationCount, violationTypes, timestamps). This structure supports efficient querying for both student-facing result retrieval and teacher-facing violation review.

5. Results

The ProctoAI system was implemented and functionally tested across the following modules. All test cases were evaluated using black-box functional testing, performance testing, and validation testing.

5.1 Functional Testing Results

User authentication correctly validated credentials, rejected invalid login attempts, and prevented duplicate registrations. JWT token generation and role-based routing functioned as specified. Exam creation by teacher accounts successfully stored questions and exam parameters in MongoDB. Student exam interfaces loaded questions, displayed timers, supported answer navigation, and triggered automatic submission on timer expiry.

Violation detection was validated through two documented test cases. In Test Case 1, a student exited fullscreen mode during the exam; the system displayed a warning overlay with a 5-second countdown and flagged a violation. In Test Case 2, repeated tab switching triggered the violation limit (threshold: 2 violations); the system automatically terminated the exam and displayed a termination screen. Score calculation was verified by cross-checking computed scores against known correct answers across multiple exam sessions, yielding accurate results in all cases tested.

Table 2. Violation Scoring Scheme and Test Outcomes

Violation Type	Score Weight	Auto-Submit Trigger	Test Outcome
Looking away (head/eye)	+1	No	Passed
No face detected	+2	No	Passed
Tab switching	+2	Yes (after 2)	Passed
Multiple faces detected	+3	No	Passed
Fullscreen exit	Warning	Countdown overlay	Passed

5.2 Performance and Validation Testing

Performance testing confirmed that the system handled multiple simultaneous users attempting exams without observable crashes or significant latency. Backend API endpoints for login, exam fetch, and result submission responded within acceptable time bounds during load tests. The system maintained stability across long-duration exam sessions without data loss on submission.

Validation testing confirmed that calculated exam scores matched expected values based on correct-answer keys, violation logs stored correct event types and timestamps, and the teacher dashboard accurately displayed student results and behavioral reports. The system's user interface was validated for responsiveness across desktop environments.

6. Discussion

6.1 Interpretation of Results

The successful functional testing of all core modules validates the feasibility of integrating multi-modal AI proctoring within a production-ready full-stack web application. The weighted scoring mechanism provides a principled, consistent basis for evaluating student behavior that removes subjectivity from violation assessment. The automatic exam termination on repeated tab switching enforces a firm deterrent against browser-based cheating without requiring teacher intervention.

By executing AI inference client-side via TensorFlow.js, ProctoAI avoids the latency inherent in server-side processing pipelines, enabling genuinely real-time monitoring. This architectural choice also reduces server computational load, improving scalability. The use of YOLO for mobile phone detection aligns with the approach validated by Bochkovskiy et al. (2020), who demonstrated high accuracy and speed on similar object detection tasks.

Compared with prior systems, ProctoAI uniquely combines exam management, real-time AI monitoring, browser activity tracking, and violation logging within a single cohesive platform. Prior works such as Sharma et al. (2022) and Lee et al. (2023) addressed monitoring in isolation, while Patel and Shah (2022) provided full-stack exam management without monitoring. ProctoAI bridges both capabilities.

6.2 Limitations

The current system has several limitations. First, quantitative detection accuracy metrics (precision, recall, F1) for the face and object detection components are not reported in the project documentation, making it difficult to characterize false positive and false negative rates. Second, the system does not perform biometric identity verification; while it detects whether a face is present, it does not confirm the identity of the detected individual against a registered photograph. Third, the rule-based scoring thresholds are fixed and may require calibration for different examination contexts. Fourth, the detection pipeline is sensitive to environmental conditions such as poor lighting, webcam quality, and camera angle. Fifth, the system currently targets desktop browsers and has not been validated on mobile platforms.

6.3 Future Work

Future enhancements include: (1) biometric face recognition for identity verification, ensuring the registered student is the exam taker; (2) voice activity detection to identify unauthorized communication; (3) adaptive violation thresholds based on historical data; (4) a mobile application extending accessibility to Android and iOS devices; (5) cloud deployment for large-scale usage; (6) AI-based adaptive question generation tailored to student difficulty levels; (7) integration with existing Learning Management Systems such as Moodle and Google Classroom; (8) multi-language support for non-English speakers; and (9) advanced analytics dashboards providing question-level difficulty analysis and longitudinal performance tracking.

7. Conclusion

This paper presented ProctoAI, an AI-powered online examination proctoring system that integrates real-time computer vision with a full-stack MERN web application. The system addresses the core challenge of maintaining academic integrity in remote examination environments by automating cheating detection through face detection, YOLO-based object detection, head pose and eye movement tracking, and browser activity monitoring. A weighted violation scoring mechanism provides consistent, objective behavioral assessment, while comprehensive logging and a teacher dashboard ensure transparency and accountability. Functional and performance testing confirmed correct operation of all modules. ProctoAI demonstrates that scalable, automated exam supervision is achievable

without specialized software installation, offering a practical and cost-effective alternative to manual human proctoring for modern educational institutions.

References

[1] Bochkovskiy, A., Wang, C., & Liao, H. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[2] Gupta, R., & Mehta, S. (2021). Browser activity monitoring for secure online examinations. International Journal of Cyber Security, 8(1), 55–63.

[3] Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519, Internet Engineering Task Force.

[4] Khan, M., & Ali, S. (2023). Machine learning-based cheating detection system for online exams. [Author, venue — cited in project documentation].

[5] Kumar, A., & Verma, S. (2023). Development of scalable online examination platforms using MERN stack technologies. Journal of Web Engineering, 15(2), 89–102.

[6] Lee, J., Park, H., & Kim, S. (2023). AI-based remote proctoring system using deep learning. Computers & Education, 182, 104–115.

[7] Patel, K., & Shah, D. (2022). Web-based online examination system using MERN stack. [Author, venue — cited in project documentation].

[8] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[9] Sharma, A., Kumar, V., & Singh, R. (2022). Automated online proctoring system using computer vision techniques. [Author, venue — cited in project documentation].

[10] Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., ... & Wattenberg, M. (2019). TensorFlow.js: Machine learning for the web and beyond. Proceedings of SysML 2019.

[11] Verma, P., & Gupta, S. (2022). Deep learning-based student monitoring system for online exams. [Author, venue — cited in project documentation].

[12] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1, I-511.