

SMARTPROCTOR: WEB-BASED PRACTICAL EXAMINATION MONITORING AND EVALUATION SYSTEM

Prof. R. V. Chatse¹, Saloni Badhe², Shruti Chavan³, Suhani Durgawale⁴, Sakshi Gaikwad⁵

^{2,3,4,5} Student, Department of Information Technology, SVPM's College of Engineering, Malegaon BK, Maharashtra, India. ¹ Assistant Professor, Department of Information Technology, SVPM'S College of Engineering Malegaon BK, Baramati, Maharashtra, India

Abstract - The rapid expansion of online education has necessitated the development of secure, scalable, and reliable platforms for conducting practical examinations. This paper presents an enhanced SmartProctor framework, a web-based practical examination monitoring and evaluation system designed using lightweight and deterministic methodologies. The proposed system integrates browser event-listener-based tab locking, real-time activity monitoring, and webcam-based face presence verification to ensure examination integrity without relying on complex behavioral inference models. A structured violation logging and scoring mechanism is introduced to provide transparent and auditable analysis of suspicious activities. The evaluation module employs test-case-based automated grading with support for partial correctness and error handling. The system is implemented using React.js, Node.js, and MongoDB within a modular client-server architecture to ensure scalability and efficient performance. Experimental observations demonstrate reliable detection of violations with minimal computational overhead. The proposed approach provides a privacy-aware and institutionally deployable solution for secure online practical examinations.

Key Words: Online Practical Examination System, Browser-Based Monitoring, Rule-Based Proctoring, Automated Code Evaluation, Privacy-Aware Assessment.

1. INTRODUCTION

The rapid growth of online education platforms has significantly transformed assessment practices in higher education. Web-based examination systems offer scalability, flexibility, and reduced administrative overhead; however, ensuring academic integrity and reliable evaluation in distributed environments remains a major challenge. The absence of physical invigilation increases the risk of impersonation, unauthorized collaboration, and misuse of digital resources.

Practical and programming-based examinations are particularly vulnerable due to their reliance on open computing environments. Common violations such as tab switching, window focus loss, and copy-paste operations are difficult to control using conventional systems. Existing

online proctoring solutions often rely on computationally intensive techniques that raise concerns related to privacy, deployment complexity, and system overhead.

Recent approaches emphasize the use of deterministic and rule-based monitoring mechanisms as an effective alternative. Browser event-listener-based techniques enable real-time detection of suspicious activities with minimal computational cost and improved transparency. In addition, automated evaluation of programming tasks using predefined test cases ensures objective and consistent grading.

Motivated by these observations, this paper presents an enhanced SmartProctor system, a web-based platform for monitoring and evaluating practical examinations. The system integrates browser-level activity tracking, webcam-based presence verification, automated code evaluation, and structured reporting within a unified framework. Furthermore, a violation logging and scoring mechanism is introduced to provide transparent and auditable assessment of student behavior.

The proposed system demonstrates that secure and scalable online practical examinations can be achieved without reliance on complex or intrusive techniques, thereby offering a privacy-aware and institutionally deployable solution.

2. Problem Statement

Development of a secure online practical examination system to ensure academic integrity in remote environments. The system addresses challenges such as unauthorized activities, lack of real-time monitoring, and unreliable evaluation. It provides browser-based tracking, webcam verification, and automated assessment for transparent examination management.

3. Literature Survey

Several research studies have focused on improving the security and reliability of online examination systems. Early approaches relied on manual or video-based invigilation, which were difficult to scale and prone to human error. Recent works have explored browser-based monitoring techniques to detect activities such as tab

switching and copy-paste operations. Some systems incorporate automated evaluation using test-case-based assessment to ensure consistent grading. However, many existing solutions lack integration between monitoring and evaluation modules, leading to fragmented system designs. Additionally, complex approaches often introduce higher computational overhead and raise concerns related to transparency and deployment feasibility. Recent studies highlight the effectiveness of lightweight and rule-based monitoring mechanisms as a scalable alternative. These findings motivate the development of an integrated, efficient, and privacy-aware online practical examination system.

4. MOTIVATION

The rapid transition toward online education has exposed several critical limitations in existing examination systems, particularly in ensuring academic integrity, reliable monitoring, and fair evaluation in remote environments. The motivation for this work arises from the need to develop a lightweight, scalable, and integrated solution that addresses these challenges effectively while maintaining transparency and usability.

The key motivating factors are as follows:

A. Need for Academic Integrity in Remote Environments

In the absence of physical invigilation, online examinations are highly vulnerable to misconduct such as tab switching, unauthorized resource access, impersonation, and collaboration. Ensuring integrity in such distributed environments requires robust yet non-intrusive monitoring mechanisms that can operate reliably in real time.

B. Limitations of Existing Complex Systems

Many existing proctoring solutions rely on computationally intensive techniques, increasing system overhead, hardware dependency, and deployment cost. Such complexity limits their scalability and makes them impractical for institutions with limited infrastructure or large student populations.

C. Lack of Integrated Monitoring and Evaluation

Current systems often separate monitoring and evaluation processes, resulting in fragmented workflows. This lack of integration reduces efficiency and makes it difficult to correlate student behavior with academic performance, thereby affecting the overall effectiveness of the assessment process.

D. Requirement for Transparent and Explainable Assessment

Most systems lack structured violation analysis and clear reporting mechanisms, making it difficult for educators to

interpret student behavior and justify evaluation decisions. A transparent and auditable system is essential to ensure fairness, consistency, and trust in the examination process.

E. Demand for Scalable and Privacy-Aware Solutions

With increasing concerns regarding data privacy and user acceptance, there is a strong need for systems that minimize intrusive monitoring while maintaining effectiveness. Lightweight, rule-based approaches can provide a balance between security, performance, and privacy, making them suitable for large-scale institutional deployment.

E. Need for Real-Time Monitoring and Immediate Feedback

Existing systems often lack real-time responsiveness in detecting and reporting violations, which can delay corrective actions. A system capable of real-time monitoring and instant feedback can enhance examination control, improve user awareness, and reduce the likelihood of repeated violations during the examination process.

5. System Architecture

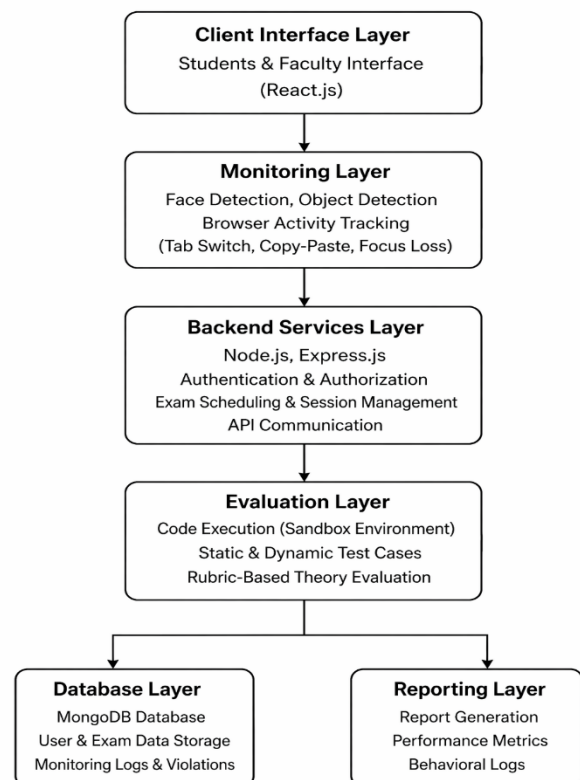


Fig 1: System Architecture

1. Client Interface Layer

This is the topmost layer, responsible for user interaction. It is developed using React.js, providing a

responsive and dynamic interface for both students and faculty members. Students can log in, access the exam dashboard, read questions, write code using the integrated editor, and submit responses. Faculty members can create exams, upload questions, and review results. This layer ensures smooth user experience and real-time interaction with the system.

2. Monitoring Layer

The Monitoring Layer is responsible for maintaining examination integrity through continuous and real-time supervision; it employs browser-level event listeners to capture activities such as tab switching, window focus transitions, and clipboard interactions, ensuring detection of unauthorized actions; additionally, webcam-based presence verification is performed to confirm candidate availability and detect anomalies such as absence or multiple individuals; all monitored events are processed using deterministic rule-based logic, timestamped, and forwarded to the backend for further analysis and storage.

3. Backend Services Layer

The Backend Services Layer acts as the central coordination and control component, implemented using Node.js and Express.js; it manages secure authentication and role-based authorization, handles exam scheduling and session lifecycle management, and facilitates structured API communication between frontend, monitoring, and evaluation modules; the layer also performs request validation, data aggregation, and synchronization of monitoring logs with evaluation results, ensuring system consistency, reliability, and scalability under concurrent user access.

4. Evaluation Layer:

The Evaluation Layer is responsible for automated and objective assessment of student submissions; it executes programming code within a sandboxed environment to isolate execution and prevent system-level risks, validates outputs against predefined static and dynamic test cases, and assigns scores based on correctness and completeness; for theoretical responses, rubric-based evaluation criteria are applied to maintain consistency; the layer also handles compilation errors, runtime exceptions, and partial outputs, ensuring fair and comprehensive grading.

5. Database Layer:

The Database Layer utilizes MongoDB as a NoSQL database to support flexible and scalable data storage; it maintains structured collections for user credentials, examination configurations, submissions, monitoring logs, and violation records; efficient indexing and querying mechanisms enable rapid data retrieval and real-time updates; this layer ensures data persistence, integrity, and scalability for

handling large volumes of examination data across multiple sessions.

6. Reporting Layer:

The Reporting Layer generates structured and comprehensive reports summarizing both academic performance and behavioral analysis; it includes detailed performance metrics, violation logs with timestamps, and final computed results; the reports provide insights into student behavior during examinations and support transparent evaluation and auditing processes; faculty members can access these reports through a secure interface, enabling efficient review, decision-making, and record maintenance.

6. METHODOLOGY

The proposed SmartProctor system adopts a structured and deterministic methodology to ensure secure, scalable, and transparent online practical examinations. The methodology integrates browser-based monitoring, webcam-based presence verification, automated evaluation, and structured reporting within a unified framework. Each stage of the methodology is designed to minimize computational overhead while maintaining reliability and fairness in assessment.

I. Authentication and Session Initialization

The system initiates the examination process through secure user authentication using role-based access control, ensuring that only authorized students and faculty members can access the platform; upon successful verification, a dedicated examination session is established with predefined parameters such as exam duration, question set, and evaluation criteria; session management mechanisms enforce strict access control by preventing multiple concurrent logins, maintaining session integrity, and activating monitoring components prior to exam commencement to ensure a controlled and secure environment.

II. Browser Activity Monitoring

The monitoring process is implemented using an event-driven approach that captures browser-level interactions in real time; JavaScript event listeners are configured to detect activities such as tab switching (visibility change), window focus transitions and clipboard operations (copy and paste); each captured event is evaluated against predefined rule-based conditions to identify violations, ensuring deterministic and immediate detection; the lightweight nature of this approach minimizes computational overhead while maintaining high responsiveness and accuracy in identifying suspicious behavior.

III. Webcam-Based Presence Verification

To ensure continuous candidate presence, the system activates webcam monitoring at the start of the examination session; video frames are periodically captured and processed to verify the presence of a single individual within the frame; abnormal conditions such as absence, multiple individuals, or prolonged inactivity are detected and recorded as violations; the verification process is designed to operate efficiently without employing complex behavioral inference techniques, thereby preserving user privacy while maintaining effective supervision.

IV. Automated Code Evaluation

Programming submissions are evaluated using a test-case-based approach. Code is executed in a controlled environment, and outputs are compared with expected results. Marks are assigned proportionally based on test case success, ensuring consistent and objective grading.

V. Violation Logging and Data Synchronization

All detected violations from browser monitoring and webcam verification are systematically logged with precise timestamps and associated session identifiers; the backend services layer aggregates and synchronizes this data in real time, ensuring consistency across system components; the logs are stored in a structured format within the database, creating a comprehensive and auditable record of user behavior throughout the examination session.

VI. Report Generation and Data Management

The final stage of the methodology involves the generation of comprehensive examination reports that integrate both performance and behavioral data; the reporting module processes evaluation results and violation logs to produce structured outputs containing marks, violation summaries, timestamps, and overall assessment outcomes; these reports are securely stored and made accessible to authorized faculty members through a dedicated interface, facilitating efficient review, auditing, and record management; the integration of monitoring and evaluation data ensures a holistic assessment of both academic performance and examination behavior.

VII. Real-Time Alert and Intervention Mechanism

The system incorporates a real-time alert mechanism to notify administrators when suspicious activities exceed predefined thresholds. Alerts are triggered based on continuous evaluation of violation scores and behavioral patterns. This enables immediate intervention, such as warning the student or flagging the session for manual

review. The mechanism ensures proactive monitoring rather than post-exam analysis alone.

VIII. Face Detection and Identity Consistency Verification

To enhance authentication reliability, the system applies periodic face detection to verify that the same individual remains present throughout the session. Captured frames are compared with the initial registered identity to ensure consistency. This approach reduces impersonation risks while maintaining minimal computational overhead through interval-based processing.

IX. Secure Execution Environment (Sandboxing)

All programming code is executed within a sandboxed environment to ensure system security and isolation. The sandbox restricts unauthorized operations such as file access, network calls, or system-level commands. This prevents malicious code execution and ensures fairness by providing a uniform execution environment for all users.

X. Time Management and Session Control

A synchronized timing mechanism is implemented to enforce strict examination duration. The system continuously tracks elapsed time and automatically submits responses upon timeout. Session control also ensures that disruptions such as page refresh or connectivity loss do not compromise exam integrity, by restoring session states where applicable.

7. Results

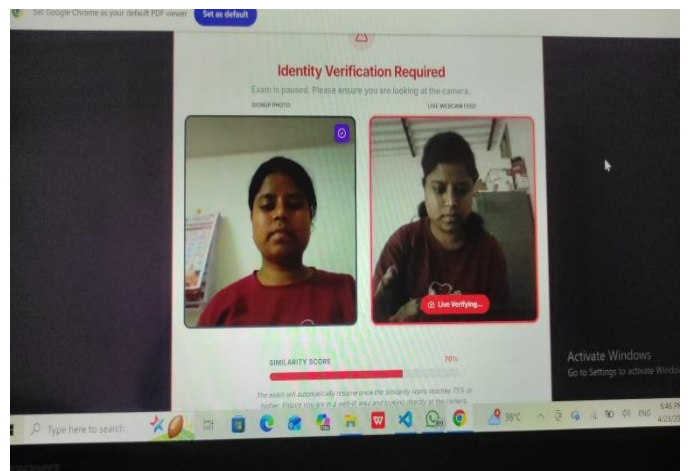
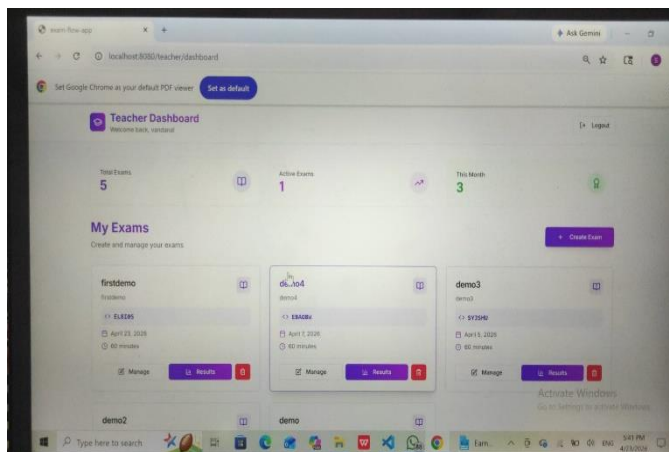
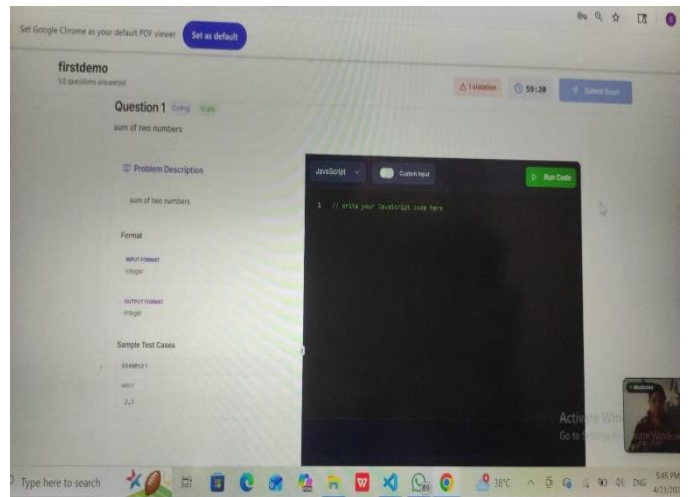
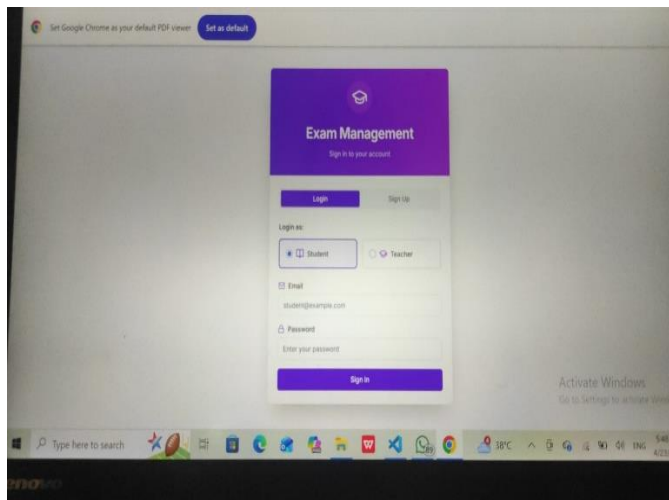
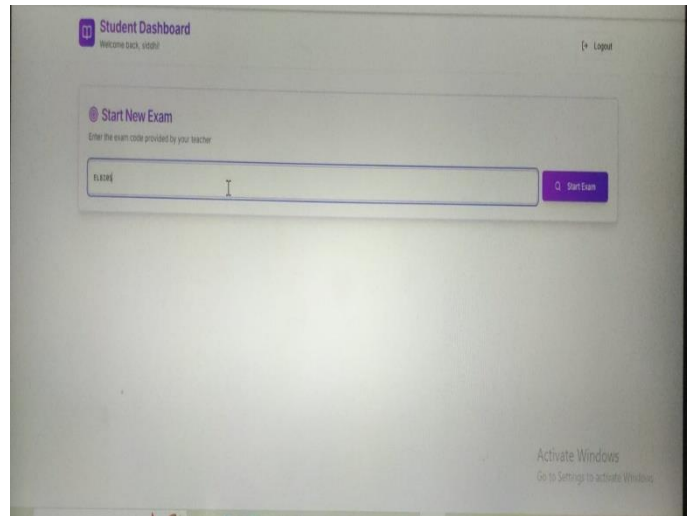
The implemented system demonstrates effective performance in conducting secure and automated online practical examinations. The user interface provides seamless interaction for both students and faculty, enabling efficient exam creation, participation, and result visualization. The monitoring mechanism successfully detects and logs violations such as tab switching and identity mismatches in real time, ensuring examination integrity.

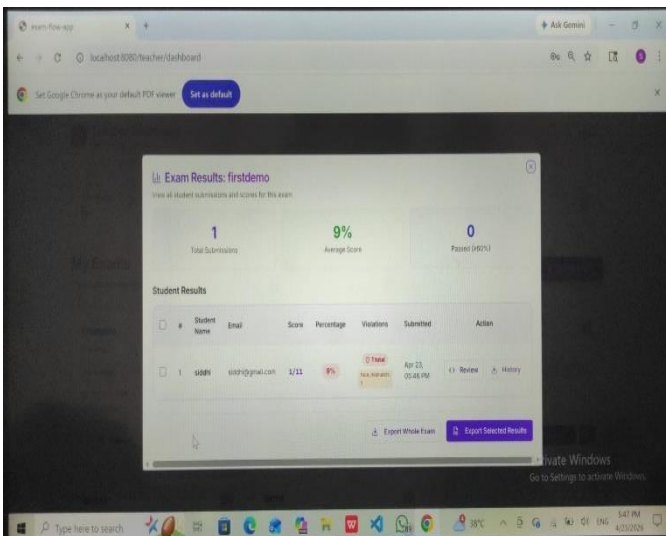
Experimental observations indicate that the system maintains low latency during exam execution and supports smooth code submission and evaluation. The automated evaluation module accurately assesses programming responses using predefined test cases, ensuring consistent and objective grading. Additionally, the integration of violation tracking with performance metrics enables comprehensive analysis of student behavior.

The reporting module generates structured outputs, including total submissions, average scores, violation counts, and individual performance details. The results

demonstrate that the system effectively combines monitoring and evaluation within a unified framework, providing reliable, transparent, and scalable assessment suitable for real-world academic environments. Furthermore, the system exhibits high usability and stability under continuous operation, ensuring minimal disruption during examination sessions.

The modular architecture also facilitates easy extensibility and future enhancements without affecting core system performance.





8. CONCLUSION

A structured and efficient approach for conducting secure online practical examinations has been presented. The system integrates browser-based monitoring, webcam-based verification, automated evaluation, and structured reporting within a unified framework. The use of rule-based and event-driven mechanisms ensures transparent, low-overhead, and reliable detection of examination violations. The evaluation process provides objective and consistent grading through test-case-based assessment. The overall system demonstrates scalability, robustness, and suitability for real-world academic deployment. Its modular design enables easy maintenance and future enhancements while ensuring system stability. Future improvements may focus on optimizing performance under large-scale concurrent usage and enhancing monitoring capabilities while maintaining privacy and efficiency.

REFERENCES

- [1] M. Labayen, "Online Student Authentication and Proctoring System," University of Navarra, Technical Report, 2021.
- [2] T. Singh, R. Sharma, and P. Mehta, "Enhancing Academic Integrity in Online Assessments Using Object Detection Models," *Procedia Computer Science*, vol. 215, pp. 324–331, 2023.
- [3] A. Ahmed and S. Hussain, "Unauthorized Activity Detection During Online Examination Using Browser Monitoring," in *Proc. IEEE International Conference on Computational Intelligence*, 2020, pp. 301–306.
- [4] M. Pawar and S. Bhattacharya, "Automated Assessment of Programming Assignments Using Test Case Evaluation," *IEEE Transactions on Education*, vol. 61, no. 3, pp. 220–228, 2018.