

IntegrityShield – A Role Based Student Activity Monitoring Platform

Kalluri Jaya Sri Sai, Pinumalla Sai Teja, Tekulapally Rajesh, Thotakura Karthik

¹Department of CSE (Artificial Intelligence & Machine Learning),

ACE Engineering College, Ghatkesar, Hyderabad – 501 301, Telangana, India

ABSTRACT

IntegrityShield is a real-time classroom activity monitoring system designed to support fair and disciplined task execution in digital learning environments such as computer laboratories and online assessments. The system enables students to log in using their roll numbers and provide explicit consent for monitoring during active sessions, while faculty members manage and observe sessions through a web-based dashboard. It tracks task-related activities such as active applications and browser usage, and identifies violations of predefined rules without accessing personal data or capturing screen content, thereby maintaining user privacy. When a violation occurs, the system generates immediate alerts for faculty to enable timely intervention. IntegrityShield follows a client-server architecture with a JavaFX-based desktop application for students, a Spring Boot backend for authentication and validation, and a web interface for faculty control. It incorporates role-based access, secure session handling, and persistent data storage, along with a lightweight classification approach to reduce false alerts. The system provides a practical and scalable solution for maintaining academic integrity while ensuring transparency and user consent.

Keywords: Classroom Monitoring System, Academic Integrity, Activity Monitoring, Role-Based Authentication, JavaFX, Spring Boot, Real-Time Alerts.

1.

INTRODUCTION

Traditional classroom and laboratory environments often lack effective mechanisms for real-time monitoring of student activities, which can lead to unauthorized behavior and reduced academic integrity during lab sessions and online assessments. With the increasing use of digital systems in education, there is a growing need for solutions that can ensure disciplined task execution without disrupting the learning process. IntegrityShield addresses this need by providing a real-time classroom activity monitoring system that enables faculty to supervise student activities during active sessions. The system operates on a client-server model, where a desktop application on student machines tracks task-related activities such as active applications and browser usage, while a web-based dashboard allows faculty to monitor and manage sessions. Unlike traditional monitoring approaches, IntegrityShield focuses on activity-level tracking without accessing personal data or screen content, ensuring transparency and user privacy.

2.

LITERATURE SURVEY

2.1. D. Smith et al. (2024) – AI-Based Classroom Activity Monitoring Systems

This study explores AI-based systems for real-time monitoring of classroom activities, focusing on detecting unauthorized application usage and student behavior. The methodology involves continuous activity tracking using intelligent algorithms to ensure adherence to academic rules. The system significantly improves academic integrity by automatically identifying suspicious actions. However, it raises concerns regarding privacy, especially when sensitive screen data is accessed or monitored without strict safeguards. This work highlights the importance of balancing monitoring efficiency with privacy protection, which is a key consideration in systems like IntegrityShield.

2.1.1. Methodologies and Algorithms

The methodology in AI-based classroom monitoring systems is based on continuous collection of user activity data such as active applications, window focus, and process execution. These systems apply behavior analysis techniques where predefined rules or trained models identify deviations from expected academic activity. Algorithms such as anomaly detection and pattern recognition are used to detect unauthorized actions in real time. In many implementations, lightweight classification models or heuristic-based rules are used to ensure fast processing and low system overhead while maintaining detection accuracy.

2.2. R. Kumar et al. (2023) – Role-Based Access Control in Educational Systems

This research focuses on implementing Role-Based Access Control (RBAC) to manage permissions in educational platforms. The system assigns roles such as student and faculty, ensuring secure and controlled access to resources. The approach enhances system security and prevents unauthorized actions. However, it introduces additional complexity in system design and management. This concept directly supports IntegrityShield's session-based access control mechanism, ensuring that only authorized users can participate in academic sessions.

2.2.1. Methodologies and Algorithms

The methodology in RBAC systems involves defining roles and associating each role with specific permissions. Users are authenticated and dynamically assigned roles based on stored credentials. Authorization algorithms then verify whether a user has permission to access a resource or perform an action. These systems commonly use access control matrices or policy-based evaluation mechanisms to enforce restrictions. This structured approach ensures secure access management and prevents unauthorized operations within the system.

2.3. L. Chen et al. (2023) – Lightweight ML for Activity Classification

The study proposes the use of lightweight machine learning models to classify user activities based on running applications and processes. This approach helps reduce false alerts and improves monitoring efficiency without requiring heavy computational resources. However, the model struggles with detecting new or unknown applications, limiting its adaptability. This research is relevant to IntegrityShield's process monitoring system, where accurate classification of allowed and restricted applications is essential.

2.3.1. Methodologies and Algorithms

The methodology involves collecting features from system processes such as application names, execution patterns, and usage duration. These features are used to train lightweight machine learning models such as decision trees, Naïve Bayes classifiers, or simple neural networks. Feature extraction and preprocessing are important steps to improve classification accuracy. The algorithms classify activities into categories such as allowed or restricted applications. These models are optimized for low latency and minimal resource usage, making them suitable for real-time monitoring systems.

2.4. A. Patel et al. (2022) – Secure Desktop Monitoring for Labs

This paper presents a client-server architecture for monitoring student systems in laboratory environments. The centralized system allows administrators to track activities across multiple devices efficiently. While this improves supervision and control, the system heavily depends on server availability and network stability. This approach aligns with IntegrityShield's centralized monitoring concept, where faculty can oversee multiple students through a unified system.

2.4.1. Methodologies and Algorithms

The methodology is based on a client-server architecture where the client application installed on student machines continuously collects activity data such as running applications, active windows, and user interactions. This data is transmitted to a centralized server using secure communication protocols. The server processes incoming data and applies rule-based or policy-driven algorithms to detect violations. Event-driven mechanisms are used to generate alerts when unauthorized activities are detected. This approach ensures scalability and enables real-time monitoring of multiple users simultaneously.

2.5. J. Anderson et al. (2021) – Privacy-Preserving Monitoring Techniques

The research introduces monitoring techniques that rely on metadata instead of direct screen capture, ensuring user privacy while still tracking activity patterns. This approach minimizes ethical concerns associated with surveillance but provides limited visibility into detailed user actions. The study emphasizes the need for privacy-aware monitoring solutions, which is an important factor in designing systems like IntegrityShield.

2.5.1. Methodologies and Algorithms

The methodology focuses on collecting non-intrusive metadata such as application names, timestamps, and usage frequency instead of capturing sensitive data like screen content. Algorithms analyse this metadata using statistical techniques or lightweight machine learning models to identify unusual patterns or suspicious behavior. Pattern analysis and anomaly detection methods are used to infer user activity while preserving privacy. This approach reduces ethical concerns and ensures compliance with data protection principles while still enabling effective monitoring.

3.

METHODOLOGY

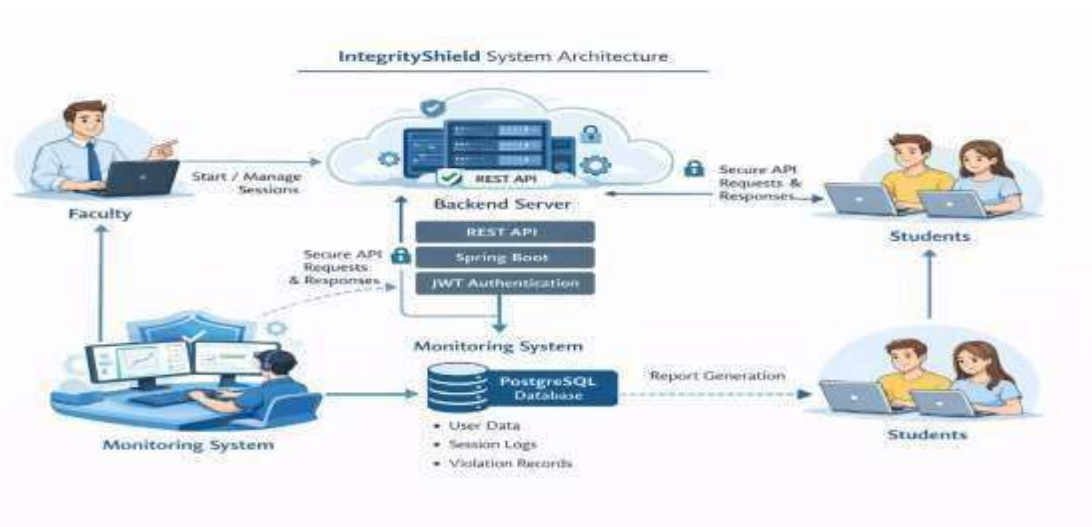


Fig. 3.1 System Architecture

3.1 Architecture Overview

The architecture of IntegrityShield is designed to provide a seamless and efficient framework for monitoring student activities, enforcing session-based access control, and maintaining academic integrity in digital learning environments. At its core, the system begins with the Faculty Control Module, where faculty members initiate and manage academic sessions. This module communicates with the backend server to define session rules, control access, and ensure that only authorized students are allowed to participate. The backend, built using Spring Boot, handles authentication, session validation, and secure communication using JWT-based mechanisms.

Once the session is active, students interact with the system through a desktop application developed using JavaFX. The Student Module is responsible for handling login, session joining, and controlled interaction within the system. Monitoring is activated only after successful session validation, ensuring that tracking occurs strictly within authorized contexts. The Monitoring Module plays a critical role by continuously observing system-level activities, including process execution and active window detection. It identifies unauthorized actions such as opening restricted applications or switching away from permitted environments.

3.2 Algorithm

3.2.1. Data Collection and Preprocessing

The system collects data in the form of user credentials, session information, and system-level activity logs from student machines. When a student logs in, authentication data is securely transmitted to the backend server. During an active session, monitoring components capture real time data such as running processes and active window titles. This data is preprocessed by filtering out system-level or irrelevant processes and normalizing application names (e.g., case handling) to ensure consistent comparison with the allowed applications list. Duplicate and rapid repeated events are controlled using debounce mechanisms to maintain accuracy in monitoring.

3.2.2. Activity Monitoring and Detection

The monitoring system continuously observes student activities using components such as ActivityMonitor, ProcessMonitor, and WindowMonitor. These modules detect currently running applications and active windows in real time. The detected data is forwarded to the backend logic for evaluation. Monitoring is activated only after successful session validation, ensuring that tracking occurs strictly within authorized academic sessions

3.2.3. Violation Detection and Processing

The detected activities are analyzed using a rule-based algorithm implemented in the ViolationService. Each activity is compared against a predefined list of allowed applications managed by the PermissionService. If an application is not present in the allowed list, it is classified as a violation. The system tracks the number of violations per student and applies filtering techniques such as duplicate prevention and debounce handling to avoid false or repeated alerts. Each valid violation is recorded with details such as timestamp, application name, and session context.

3.2.4. Data Storage and Communication

All validated violations and session-related data are transmitted to the backend server using HTTP based communication. The backend, built using Spring Boot, processes and stores this data in the PostgreSQL database. The system maintains structured records for users, sessions, and violations, ensuring traceability and efficient retrieval. Real-time updates are sent to the faculty dashboard, enabling continuous monitoring of student activities.

3.2.5. System Evaluation and Performance

The performance of IntegrityShield is evaluated based on factors such as detection accuracy, system responsiveness, and resource efficiency. The system ensures accurate violation detection through strict rule validation and minimizes false positives using filtering mechanisms. It is designed to operate with low computational overhead, making it suitable for deployment on standard systems. Additionally, real-time communication ensures minimal delay in reporting violations, enabling effective supervision by faculty members.

4

FINAL OUTPUT



Fig. 4.1 Faculty Dashboard

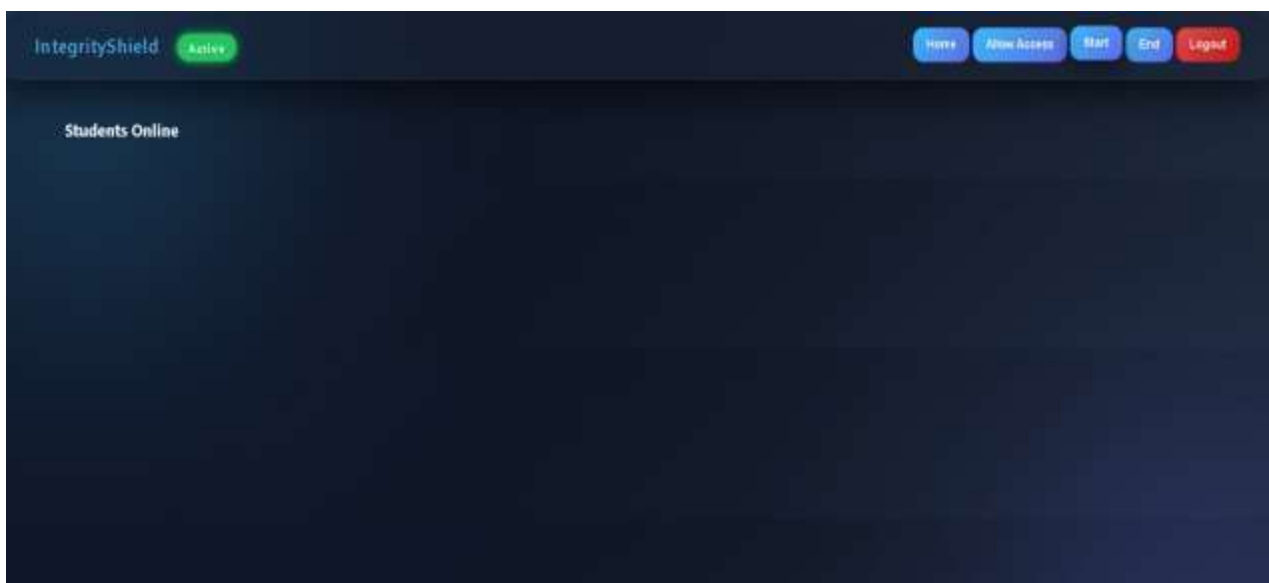


Fig. 4.2 Active Session Page

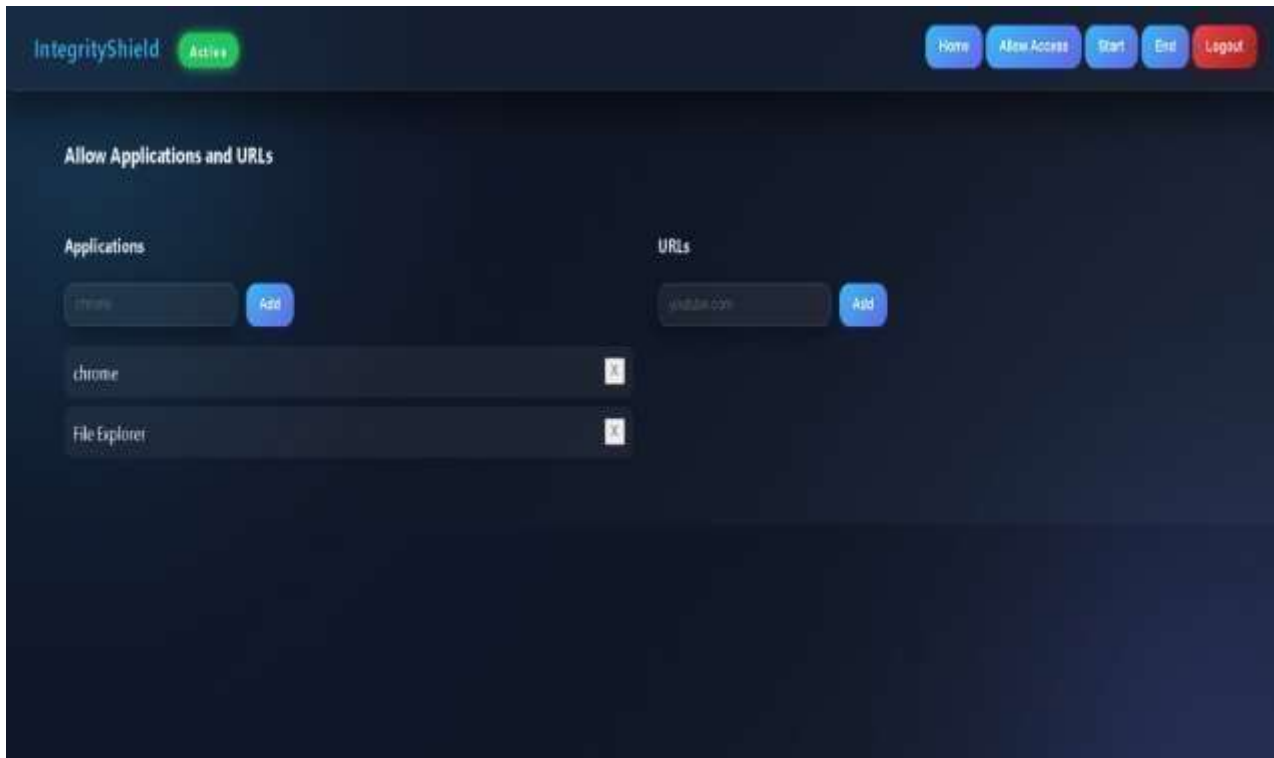


Fig. 4.3 Allowed Applications



Fig. 4.4 Student Login

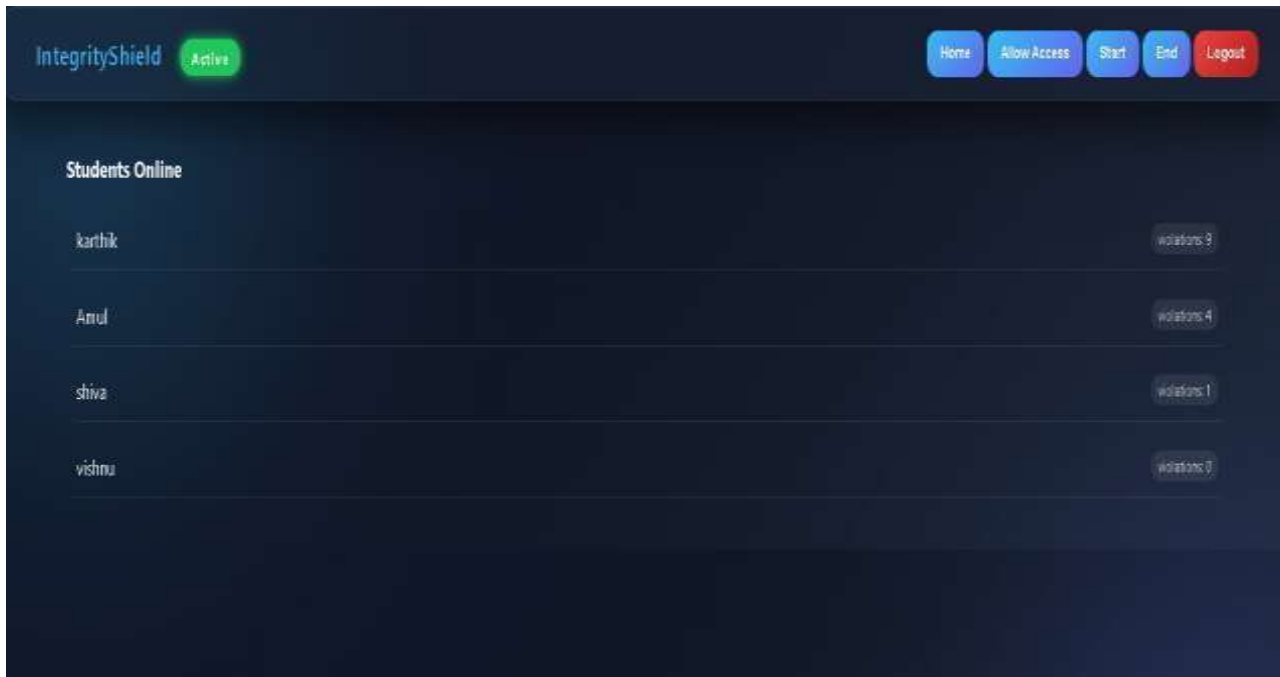


Fig. 4.5 Students Online

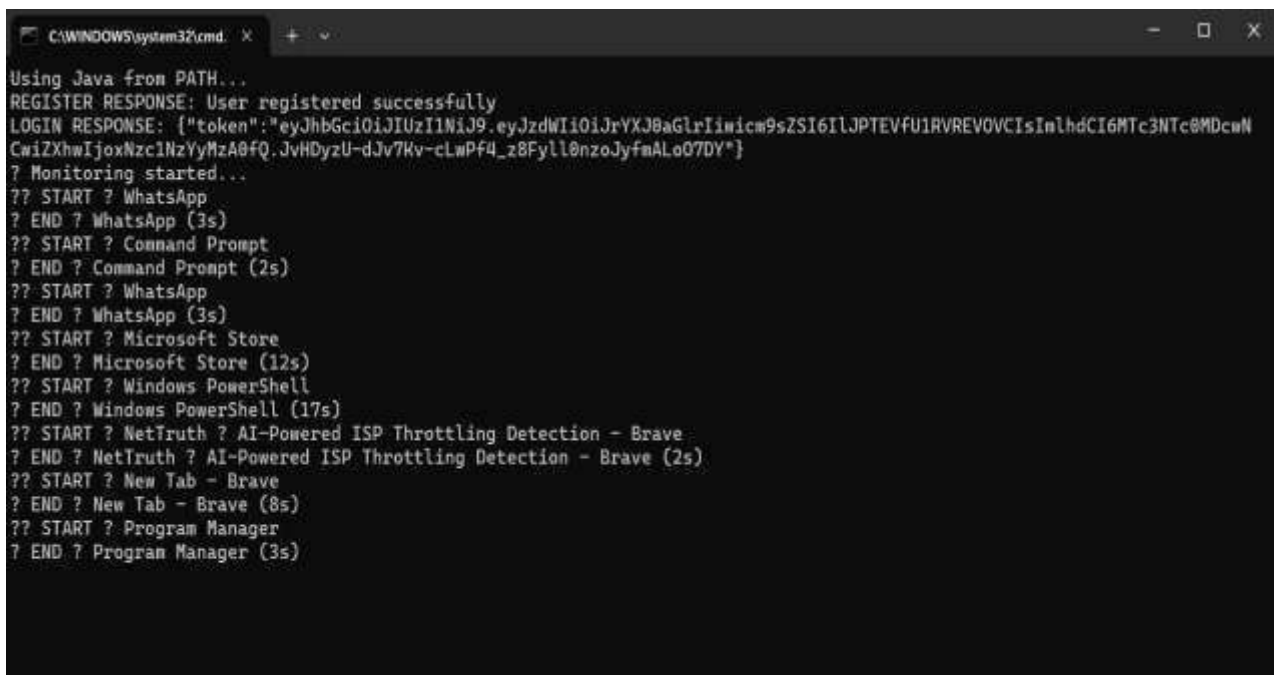


Fig. 4.6 Monitor Logs

5

FUTURE ENHANCEMENTS AND CONCLUSION

5.1 Mobile-Based Faculty Monitoring Dashboard

Objective: Extend the existing web dashboard into a dedicated mobile application (Android/iOS).

Enhancement: Provide real-time access to session status, student activity, and violations through mobile devices.

Benefit: Faculty can monitor sessions remotely without being restricted to desktop systems, improving flexibility and responsiveness.

5.2 AI-Based Behavioral Analysis and Smart Detection

Goal: Improve monitoring intelligence beyond rule-based detection.

Approach: Integrate lightweight machine learning models to analyze behavioral patterns such as frequent tab switching, unusual application usage, and irregular activity timelines.

Enhancement:

- Detect hidden or unknown applications
- Reduce false positives
- Adapt to evolving user behavior

5.3 Real-Time Alert and Notification System

Feature: Introduce instant, multi-channel alert mechanisms.

Enhancement:

- Push notifications to faculty dashboard and mobile app
- Audio/visual warnings for students on first violation
- Escalation alerts for repeated violations

Benefit: Enables immediate intervention and improves system responsiveness.

5.4 Advanced Analytics and Intelligent Reporting

Feature: Provide detailed insights into student activity and session performance.

Includes:

- Violation trends over time
- Student-wise behavioural reports
- Session-based analytics for faculty evaluation
- Risk scoring based on activity patterns

Enhancement: Helps faculty make data-driven decisions and identify high-risk behavior early.

5.5 Real-Time WebSocket-Based Monitoring

Enhancement: Upgrade the system from periodic updates to real-time communication using WebSockets.

Benefit:

- Instant student join/leave updates
- Live violation streaming
- No need for manual refresh

Impact: Makes the system more dynamic and closer to real-time surveillance systems.

5.6 Conclusion:

IntegrityShield is a real-time classroom activity monitoring system designed to address the challenges of maintaining discipline and academic integrity in modern digital learning environments. By integrating system-level monitoring with secure backend technologies, the system effectively tracks student activities, enforces session-based access control, and detects violations in real time. The project demonstrates the practical implementation of a rule-based monitoring approach combined with technologies such as Spring Boot for backend services and JavaFX for the desktop interface. Through continuous tracking of processes and active windows, the system ensures that only authorized applications are used during academic sessions, thereby reducing misuse and improving discipline. IntegrityShield successfully achieves its core objective of providing a reliable, scalable, and efficient monitoring solution. The integration of real-time violation detection, structured data storage using PostgreSQL, and a centralized faculty dashboard enables effective supervision and immediate response to unauthorized activities. Furthermore, the system establishes a strong foundation for future enhancements, including intelligent monitoring, mobile integration, and advanced analytics. By combining automation, security, and real-time feedback, IntegrityShield contributes to creating a transparent and controlled academic environment. In conclusion, the project fulfills its goal of delivering a practical and efficient solution for modern classroom monitoring, ensuring fairness, accountability, and improved academic discipline while providing opportunities.

6

REFERENCES

- [1] Smith, D., Johnson, M., & Lee, K. (2024). AI-Based Classroom Activity Monitoring Systems. *International Journal of Educational Technology*, 12(2), 45–58.
- [2] Kumar, R., Sharma, P., & Verma, S. (2023). Role-Based Access Control in Educational Systems. *Journal of Information Security*, 18(3), 210–225.
- [3] Chen, L., Wang, Y., & Zhao, H. (2023). Lightweight Machine Learning for Activity Classification in Monitoring Systems. *IEEE Access*, 11, 34567–34578.
- [4] Patel, A., Mehta, R., & Shah, D. (2022). Secure Desktop Monitoring for Computer Laboratories. *International Journal of Computer Applications*, 45(6), 112–120.
- [5] Anderson, J., Brown, K., & Wilson, P. (2021). Privacy-Preserving Monitoring Techniques in Digital Systems. *Cybersecurity Journal*, 9(4), 89–102.
- [6] Johnson, M., Lee, S., & Kim, H. (2024). Real-Time Behavioral Analytics in Smart Classrooms. *IEEE Transactions on Learning Technologies*, 17(1), 66–78.
- [7] Singh, P., Gupta, R., & Verma, K. (2023). Blockchain-Based Secure Logging for Student Monitoring Systems. *Journal of Distributed Systems*, 14(2), 130–142.
- [8] Wang, Y., Zhang, X., & Liu, H. (2022). Deep Learning-Based Classroom Surveillance Systems. *Pattern Recognition Letters*, 158, 25–34.