ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-1/578-586
Thodishetty Akshitha *et. al.*, /International Journal of Engineering & Science Research
Full Length Article

Classroom Monitoring System For Exam and Behavior Using Deep Learning

Minhaj Begum[1], Thodishetty Akshitha[2], Alladi Charitha[3], Jammuladinne Mokshitha Rohini[4]
[1]Assistant Professor; Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India.
[2,3,4]B.Tech Students; Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India.
Mail Id; mokshithareddyy0@gmail.com[4], akshithathodishetty1@gmail.com[2], alladicharitha05@gmail.com[3]
Accepted 29-03-2026

Abstract
The increasing adoption of digital examination systems and smart classroom environments has created a demand for reliable automated monitoring solutions that ensure academic integrity. This work presents a web-based classroom monitoring and online proctoring system that integrates deep learning and computer vision techniques for real-time supervision. The system is developed using Flask for backend operations, OpenCV for video processing, YOLOv8 for object detection, MediaPipe for facial analysis, and PostgreSQL for structured data storage.The proposed solution continuously analyzes live video streams to identify potential malpractice indicators, including the presence of mobile phones, books, multiple individuals, absence of a student, abnormal head movements, poor lighting conditions, and suspicious motion patterns. Upon detecting violations that exceed predefined thresholds, the system captures evidence images, logs events with timestamps, and triggers email alerts to administrators.A role-based interface enables administrators to manage students, subjects, examinations, and malpractice records, while students can securely access and participate in scheduled assessments. Additionally, the system supports offline classroom monitoring, extending its usability beyond online examinations. By combining real-time analytics with centralized data management, the system enhances monitoring accuracy, reduces reliance on manual invigilation, and ensures traceable and consistent supervision.

Keywords: *Deep Learning, Online Proctoring, Classroom Monitoring, YOLOv8, MediaPipe, OpenCV, Flask, PostgreSQL, Real-Time Detection, Malpractice Analysis*

Introduction

The rapid digitization of education has significantly transformed examination and classroom environments. However, maintaining fairness and integrity during assessments remains a critical challenge. Traditional monitoring methods rely heavily on human invigilators, who cannot consistently observe every student or detect brief suspicious behaviors in real time. This limitation motivates the development of automated monitoring systems capable of continuous and objective supervision.This paper introduces a deep learning-based classroom monitoring system designed to enhance both online examination proctoring and offline classroom supervision. The system integrates a web-based interface with a real-time video analysis engine and a structured database for evidence management. Students interact with the system through a controlled dashboard to attend scheduled exams, while administrators oversee operations through dedicated management interfaces.The core of the system is a computer vision pipeline that processes live video streams. OpenCV is used for frame acquisition and preprocessing, YOLOv8 enables real-time object detection, and MediaPipe Face Mesh facilitates facial landmark tracking and head pose estimation. The system identifies multiple forms of suspicious behavior, including unauthorized object usage, absence from the camera frame, repeated head movements, and environmental anomalies such as low lighting or camera obstruction.When abnormal patterns persist beyond defined thresholds, the system records the event, stores visual evidence, and notifies administrators through email alerts. This mechanism ensures transparency and supports post-examination review. Furthermore, the inclusion of an offline monitoring mode allows the system to be used in physical classrooms for behavior analysis and supervision.

Purpose of the Project

The primary objective of this work is to develop an intelligent monitoring system capable of automatically observing student behavior during examinations and classroom sessions. The system aims to detect suspicious activities in real time, maintain verifiable records, and notify authorities

without requiring continuous human supervision. This contributes to improving fairness, minimizing malpractice, and enabling reliable audit mechanisms.

Existing System

Conventional examination monitoring approaches depend largely on manual invigilation or basic webcam verification techniques. While these methods provide a minimal level of supervision, they lack scalability and consistency. Human observers may overlook short-duration or subtle suspicious actions due to fatigue or limited attention. Additionally, traditional systems often fail to maintain detailed records or provide immediate alerts, reducing their effectiveness in large-scale environments.

Proposed System

The proposed system is a role-based web application integrated with a deep learning-driven monitoring engine. It utilizes YOLOv8 for detecting objects such as mobile phones and books, MediaPipe for facial landmark analysis, and OpenCV for real-time video processing. PostgreSQL is used to manage user data, examination schedules, and malpractice logs.The system continuously evaluates video frames and identifies predefined suspicious conditions. When a violation persists beyond a configurable threshold, an evidence image is stored, the event is logged in the database, and an alert is sent to the administrator. Separate dashboards for students and administrators enable secure exam participation and efficient system management.

Literature Survey

Automated monitoring in educational environments has evolved significantly with advancements in artificial intelligence and computer vision. Early examination systems primarily relied on manual supervision or basic webcam-based verification, which offered limited effectiveness in detecting complex or short-lived suspicious behaviors. These approaches were constrained by human limitations such as fatigue and lack of scalability.

The emergence of deep learning introduced more sophisticated detection capabilities. Object detection models such as SSD and Faster R-CNN improved detection accuracy but often required substantial computational resources, limiting their suitability for real-time applications. The development of the YOLO (You Only Look Once) family of models addressed this challenge by enabling high-speed object detection with competitive accuracy, making them suitable for live monitoring systems. Recent advancements like YOLOv8 further enhance detection performance while reducing latency.In parallel, facial analysis techniques have gained importance in behavior monitoring. Facial landmark detection methods allow systems to identify key facial features and track movements over time. MediaPipe Face Mesh extends this capability by providing dense facial point mapping, enabling accurate head pose estimation and gaze analysis. These features are essential for identifying behaviors such as frequent head turning, looking away from the screen, or absence from the camera.Modern monitoring systems also emphasize event logging and alert generation. Instead of merely detecting anomalies, these systems document suspicious activities with timestamps, visual evidence, and confidence metrics. This approach enhances transparency and supports post-event analysis. Real-time alert mechanisms further improve responsiveness by notifying administrators immediately when violations occur.Practical deployment of such systems requires consideration of additional factors, including latency, hardware efficiency, scalability, and user interface design. Real-time processing demands low-latency computation, especially when handling multiple video streams simultaneously. Efficient database integration is also essential for managing user data and maintaining logs.Web-based dashboards have become a standard component of modern systems, providing administrators with centralized monitoring capabilities. Features such as live video feeds, alert notifications, and visual indicators improve usability and decision-making. The integration of these components results in a comprehensive monitoring framework that combines detection, analysis, storage, and user interaction.

Requirement Analysis

Functional Requirements

The functional requirements describe the core operations that the system must perform to support automated examination monitoring and classroom supervision. The platform is designed to manage user interactions, process real-time video input, and generate meaningful outputs such as alerts, logs, and dashboards. The system provides secure registration and login mechanisms for both students and administrators, ensuring role-based access to functionalities. Administrators are responsible for managing subjects, scheduling examinations, and reviewing malpractice records, while students can access scheduled exams and participate through a controlled interface.A key functionality of the system is real-time video monitoring and streaming, where live camera input is continuously analyzed using computer vision techniques. The system detects suspicious activities such as the presence of unauthorized objects, multiple individuals, or abnormal behavior patterns. When such activities are identified, the platform captures evidence images, logs the events with timestamps, and generates alerts through email notifications. Additionally, the system offers dashboard-based visualization for reviewing student activity,

examination details, malpractice logs, and performance results, thereby enabling efficient monitoring and decision-making.

Non-Functional Requirements

The non-functional requirements define the quality attributes that ensure the system performs efficiently and reliably under real-world conditions. The platform is expected to deliver high accuracy in detecting suspicious behavior while maintaining low latency in real-time processing. Security is a critical aspect, and the system implements role-based authentication and secure data handling to protect sensitive information.Reliability and scalability are also essential, as the system must support continuous monitoring over extended durations and handle multiple users simultaneously. Usability is ensured through intuitive web interfaces that allow both administrators and students to interact with the system without complexity. Maintainability is achieved through modular design, enabling easier updates and debugging. Furthermore, traceability is maintained by storing detailed logs and evidence for every detected event, and portability ensures that the system can operate on commonly available hardware with standard software configurations.

Computational Resources

Software Requirements

The system is implemented using a combination of programming languages and libraries that support web development, database management, and computer vision tasks. Python serves as the primary programming language for backend development, while HTML, CSS, and JavaScript are used to design the user interface. Flask is employed as the web framework to handle routing, session management, and page rendering. OpenCV is utilized for video capture and frame processing, while YOLOv8 is used for object detection tasks. MediaPipe provides facial landmark detection and head pose estimation capabilities. NumPy supports numerical computations required for motion and brightness analysis, and psycopg2 facilitates interaction with the PostgreSQL database. Flask-Mail is used to enable automated email notifications for detected violations. The development environment is managed using Visual Studio Code.

Hardware Requirements

The system is designed to operate on standard computing hardware to ensure accessibility and ease of deployment. A processor equivalent to Intel i5 or higher is recommended to support real-time video processing. A minimum of 4 GB RAM is required for smooth execution of the application and associated libraries. Additionally, at least 100 GB of storage is suggested to accommodate application files, database records, and stored evidence images generated during monitoring.

System Architecture

The architecture of the proposed system follows a modular design approach, ensuring clear separation of responsibilities among different components. The system begins with user interaction through web interfaces and camera input, which are then processed by the backend application and computer vision modules. The processed data is stored in a structured database, and relevant outputs such as alerts and dashboards are generated accordingly.This architecture integrates three major layers: the presentation layer, the application layer, and the data layer. The presentation layer consists of web-based interfaces for students and administrators. The application layer, implemented using Flask, manages business logic, authentication, and communication between components. The data layer, powered by PostgreSQL, stores user information, examination data, and malpractice records. This separation improves scalability, maintainability, and system performance.

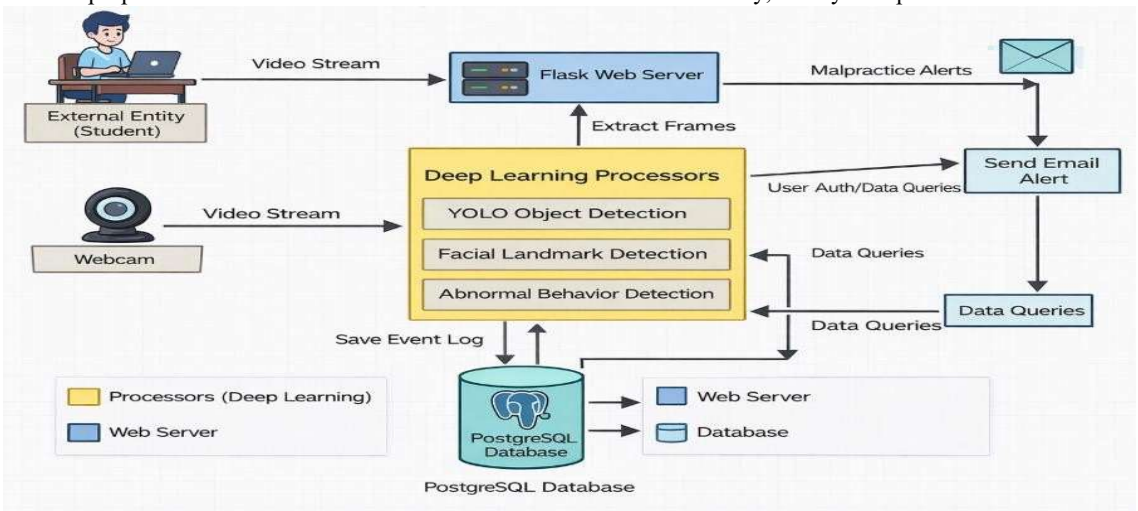


Fig 1; System Architecture

Technical Architecture

The technical implementation of the system is centered around a Flask-based web server connected to a PostgreSQL database and a real-time video analysis engine. The frontend utilizes HTML templates rendered dynamically to display dashboards, forms, and monitoring interfaces. Backend routes handle authentication, exam scheduling, data retrieval, and result storage.The video processing pipeline captures live frames using OpenCV and applies YOLOv8 for detecting objects such as mobile phones, books, and persons. MediaPipe Face Mesh is used to analyze facial landmarks and determine head orientation and visibility. The processed frames are streamed back to the user interface, allowing real-time visualization. When suspicious behavior is confirmed, evidence images are stored, database records are updated, and email notifications are triggered using Flask-Mail. This tightly integrated architecture ensures seamless interaction between all system components.

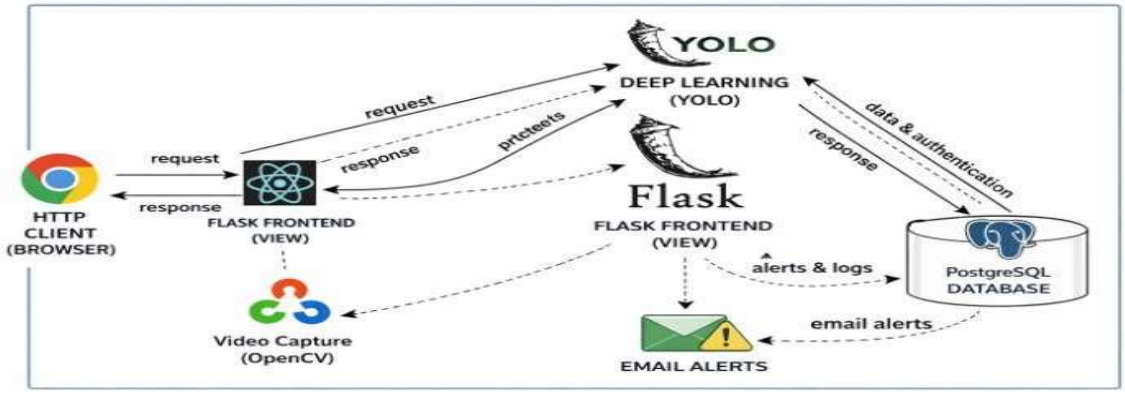


Fig 2 ;Technical Architecture

UML Diagrams

The system design is further represented using standard UML diagrams to illustrate both structural and behavioral aspects. The use case diagram identifies key actors, including students and administrators, and outlines their interactions with the system. The class diagram represents the static structure by defining system entities such as users, exams, and logs along with their relationships.Sequence diagrams are used to describe the chronological flow of interactions between system components, particularly during processes such as login, exam participation, and malpractice detection. Activity diagrams illustrate workflows, including exam execution and monitoring processes, while data flow diagrams provide a high-level representation of how data moves through the system. Together, these diagrams offer a comprehensive understanding of system functionality and design.

Methodology

The methodology adopted for this project follows a structured and implementation-oriented approach. The system is divided into independent modules responsible for authentication, exam management, video processing, alert generation, and database operations. This modular structure ensures flexibility and simplifies system maintenance.

The monitoring process begins with capturing live video frames using OpenCV. These frames are analyzed using YOLOv8 for object detection and MediaPipe for facial landmark tracking. Additional computations are performed to detect environmental conditions such as low lighting and abnormal motion. Instead of reacting to every individual frame, the system uses threshold-based logic to identify repeated suspicious patterns, reducing false positives.Once a violation is confirmed, the system captures an evidence image, records the event in the database, and sends an email alert to the administrator. The collected data is made available through administrative dashboards for further analysis and decision-making. The system also maintains examination results and correlates them with detected events, enhancing transparency.To ensure performance and scalability, the system follows a service-oriented design where each component operates independently but communicates through defined interfaces. Efficient resource management techniques are applied to handle continuous video streams without degrading performance. Security measures such as session management and access control are implemented to protect system data. Overall, the methodology balances automation with human oversight to achieve reliable monitoring.

Modules

The system is divided into several functional modules, each responsible for a specific aspect of the application. The monitoring interface module provides user-facing pages for interaction, including dashboards and live video streams. The authentication module manages login sessions and enforces role-based access control. The video capture and analysis module processes live frames

and detects suspicious activities using deep learning models.

Implementation

System Implementation

The implementation of the system integrates multiple technologies to achieve real-time monitoring and web-based interaction. Flask serves as the backbone of the application, managing routing, session control, and dynamic page rendering. OpenCV is used to capture and process video frames, enabling continuous monitoring through webcam input.YOLOv8 is employed for object detection, allowing the system to identify unauthorized items such as mobile phones and books, as well as count the number of individuals present in the frame. MediaPipe Face Mesh is used to analyze facial landmarks and determine head orientation, which helps in identifying suspicious behavior such as looking away from the screen.PostgreSQL is used for storing structured data, including user details, examination records, and malpractice logs. The psycopg2 library facilitates database connectivity and query execution. Flask-Mail is integrated to send email alerts whenever violations are detected, ensuring timely communication with administrators. NumPy supports numerical computations required for motion and brightness analysis, while HTML and Jinja templates are used to render user interfaces.

Algorithmic Workflow

The system operates through a sequence of interconnected processes. Initially, users register and log in to the platform, after which administrators can schedule exams and manage subjects. During an examination session, live video is captured and analyzed in real time. The detection module identifies suspicious activities based on predefined conditions, and if a violation is confirmed, the system logs the event and sends an alert.

Testing

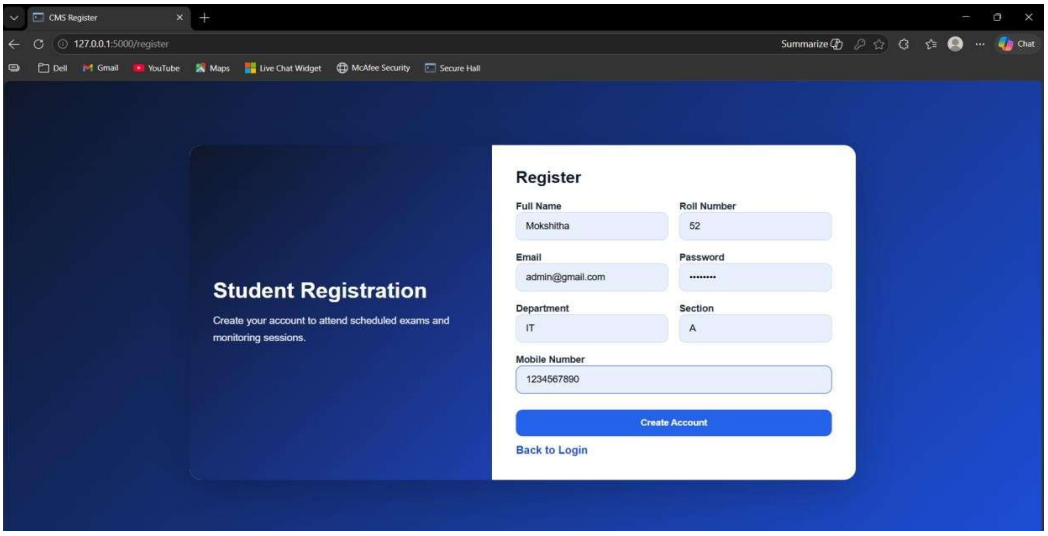
Test Cases

Testing is conducted to verify that the system performs all required functions correctly. Various test cases are designed to evaluate features such as user authentication, exam scheduling, real-time monitoring, and evidence generation. Each test ensures that the system responds appropriately to valid and invalid inputs, maintains data integrity, and performs reliably under different conditions.

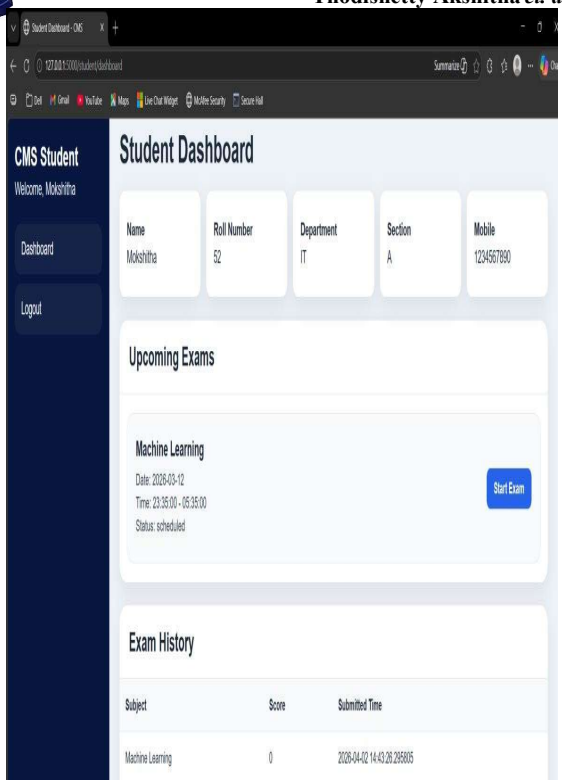
Validation

Validation ensures that the system meets both functional and non-functional requirements. Input validation is implemented in login and scheduling forms to prevent incomplete or incorrect data submission. The system verifies that all required fields are provided and displays appropriate error messages when necessary.During monitoring, the system validates that evidence images and log entries are generated for confirmed violations. This ensures traceability and supports auditing processes. Overall, validation mechanisms enhance system reliability, accuracy, and user experience.

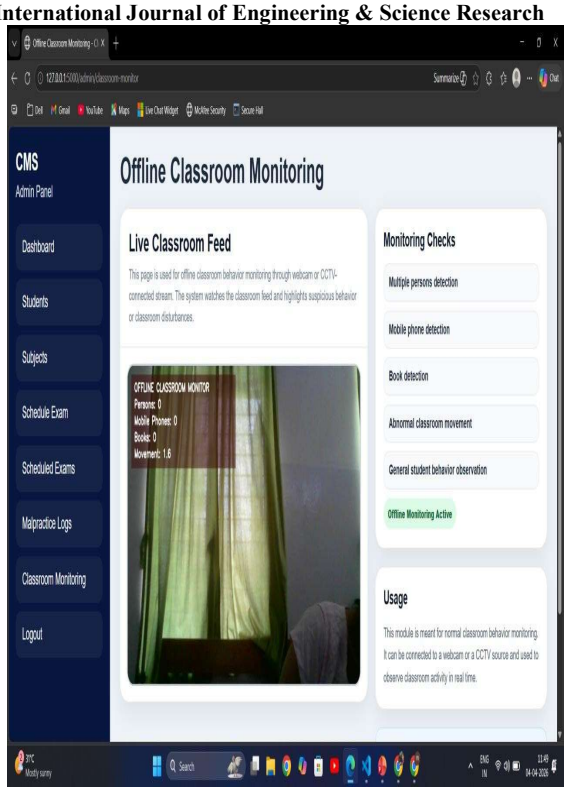
Screenshots



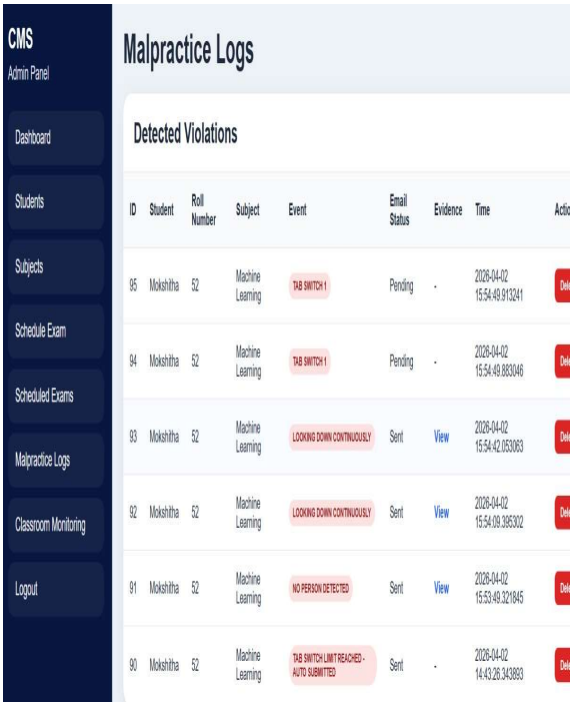
Register Page



Student Dashboard Page:

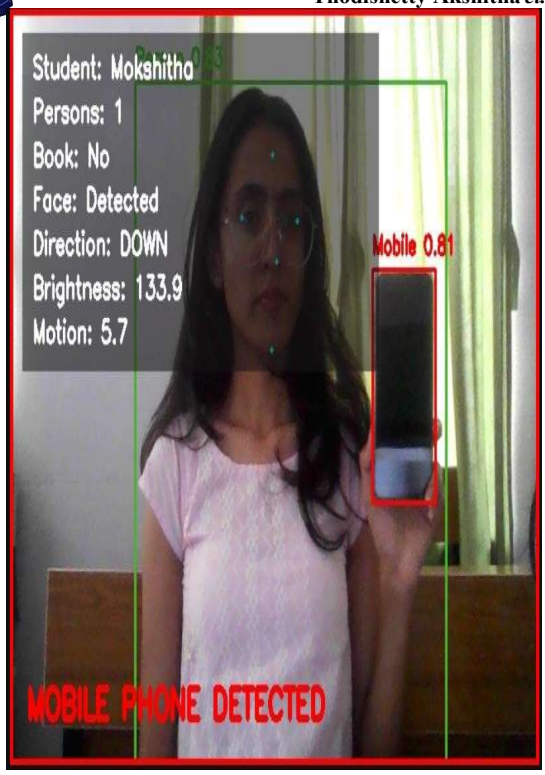


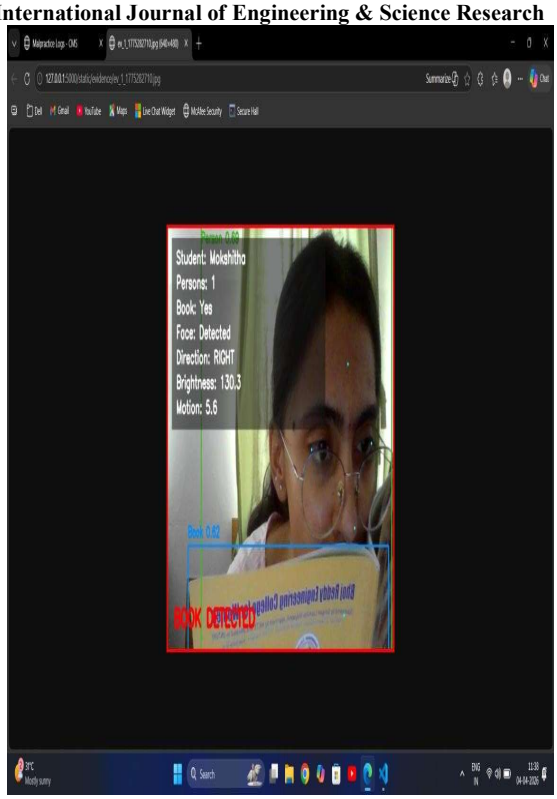
Offline Classroom Monitor:



Malpractice Logs

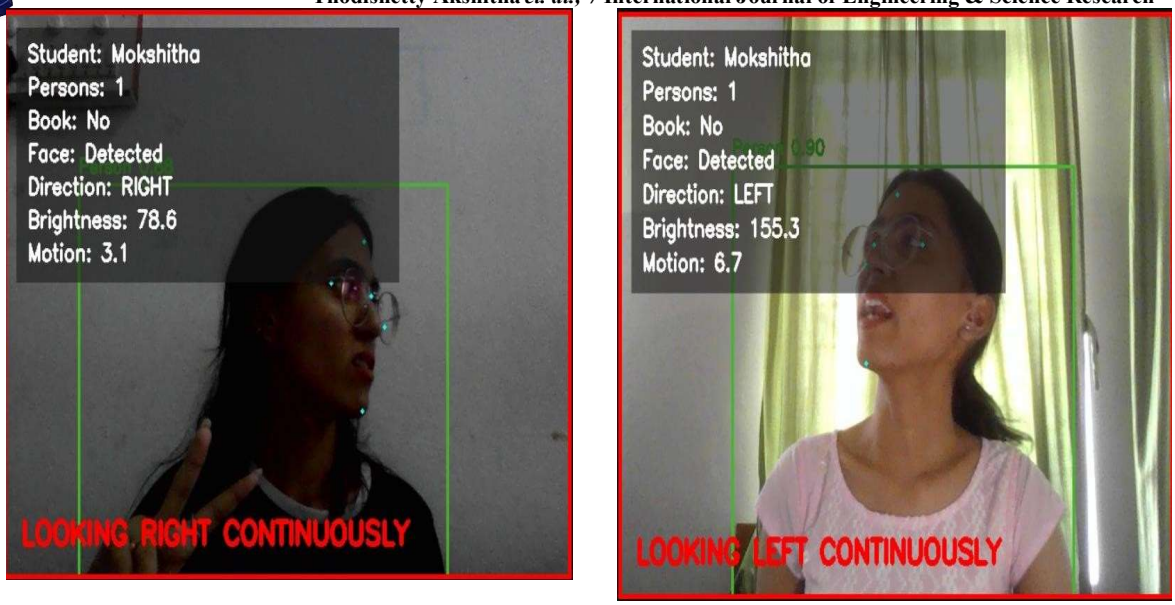
Email notification


Student: Mokshitha
Persons: 1
Book: No
Face: Detected
Direction: DOWN
Brightness: 133.9
Motion: 5.7
Mobile 0.81
MOBILE PHONE DETECTED


Student: Mokshitha
Persons: 1
Book: Yes
Face: Detected
Direction: RIGHT
Brightness: 130.3
Motion: 5.6
Book 0.62
BOOK DETECTED


Student: Mokshitha
Persons: 2
Book: No
Face: Detected
Direction: DOWN
Brightness: 116.8
Motion: 5.3
Person 0.89
MULTIPLE PERSONS DETECTED


Student: Mokshitha
Persons: 0
Book: No
Face: Not Clear
Direction: UNKNOWN
Brightness: 127.6
Motion: 6.3
NO PERSON DETECTED



Evidence image

Conclusion

The proposed Classroom Monitoring System for Exam and Behavior Using Deep Learning presents an effective and intelligent solution for ensuring academic integrity in modern examination environments. By combining deep learning techniques with real-time computer vision and a web-based interface, the system enables continuous and automated observation of student activities. It successfully identifies multiple forms of suspicious behavior, including the use of unauthorized objects, the presence of additional individuals, irregular head movements, and environmental inconsistencies such as poor lighting.The integration of advanced technologies such as YOLOv8 for object detection, MediaPipe for facial landmark analysis, and PostgreSQL for structured data management contributes to the system's reliability and performance. The incorporation of automated alert mechanisms further enhances responsiveness by notifying administrators immediately when violations are detected. This proactive approach improves monitoring efficiency compared to traditional methods that rely heavily on manual supervision.Furthermore, the system offers a scalable and flexible framework suitable for both online examinations and offline classroom environments. The ability to capture evidence and maintain detailed logs ensures transparency and supports post-examination analysis. By reducing human dependency and minimizing observational errors, the proposed solution provides a consistent and objective monitoring process. Overall, this work demonstrates how deep learning can be effectively applied to transform conventional examination practices into more secure, efficient, and technology-driven systems.

Future Scope

Although the proposed system achieves reliable performance, there are several opportunities for further enhancement to improve its robustness and adaptability. Future developments may focus on incorporating more advanced deep learning models to increase detection accuracy and reduce false positives, especially under challenging environmental conditions such as occlusions or varying illumination.

The integration of audio analysis can extend the system's capabilities by enabling detection of suspicious conversations or verbal cues during examinations. Additionally, supporting multiple camera inputs would allow wider coverage and more comprehensive monitoring in large classroom settings. The inclusion of facial recognition techniques can further strengthen security by verifying student identity and preventing impersonation.

Cloud-based deployment represents another promising direction, enabling centralized monitoring, remote accessibility, and improved scalability across multiple examination centers. The system can also be enhanced with adaptive learning mechanisms, allowing models to continuously improve based on newly collected data and feedback.Moreover, the development of mobile applications could provide administrators with real-time alerts and monitoring capabilities on portable devices. Advanced analytics and reporting features may also be introduced to generate insights into behavioral patterns and system performance over time. With these enhancements, the system can evolve into a more intelligent, scalable, and comprehensive solution capable of meeting the

growing demands of digital and remote education environments.

References

[1] W. Alsabhan, "Detection of student cheating in higher education using machine learning and LSTM-based approaches," *Sensors*, vol. 23, 2023.

[2] K. K. Humse, C. K. Mahadevaswamy, and S. K. V., "Centralized mobile phone detection system for examination halls using Arduino Duemilanove," *ACCENTS Transactions on Information Security*, vol. 6, no. 21, 2021.

[3] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "An analysis of spatiotemporal convolutional networks for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] H. Yin, R. O. Sinnott, and G. T. Jayaputera, "Comprehensive survey on video-based human action recognition in team sports," *Artificial Intelligence Review*, vol. 57, 2024.

[5] B. Sun, D. Kong, S. Wang, J. Li, B. Yin, and X. Luo, "A two-stage framework combining generative adversarial networks and knowledge graphs for zero-shot action recognition," *Pattern Recognition*, vol. 126, 2022.

[6] N. Ruiz, E. Chong, and J. M. Rehg, "Fine-grained estimation of head pose without reliance on keypoint detection," in *IEEE CVPR Workshops (CVPRW)*, 2018.

[7] G. Borghi, M. Fabbri, R. Vezzani, S. Calderara, and R. Cucchiara, "Head pose estimation from depth images using face-from-depth techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 3, 2020.

[8] Z. Shah, A. Khan, and A. Khan, "Automated detection of mobile phones in large-scale baggage screening systems," in *International Conference on Advances in Emerging Computing Technologies (AECT)*, 2020.