Journal of Artificial Intelligence and Capsule Networks (ISSN: 2582-2012)
www.irojournals.com/aicn/

An In-Browser Proctoring System Using YOLO-Based Object Detection and Gaze Analysis

Ayush Sharma Kaundinya[1], Pramish Adhikari[2], Mohan Bikram KC[3], Pratikshya Shrestha[4]

[1, 2, 3]Gandaki College of Engineering and Science, Pokhara University, Pokhara, Nepal

[4]Assistant Professor, Gandaki College of Engineering and Science, Pokhara University, Pokhara, Nepal

Email: [1]ayushsharmakaundinya@gmail.com, [2]pramishadhikari30@gmail.com, [3]kcmohan64@gmail.com, [4]pratikshyashrestha@gces.edu.np

Abstract

The development of remote learning has increased requirements for effective online evaluations, but the lack of physical supervision can lead to inappropriate actions such as invisible support, illegal collaboration, and the use of restricted resources. This work represents a web-based automated proctoring solution that uses a camera and a lightweight backend system to reduce the need for constant monitoring by avoiding complicated client-side implementation. The proposed system consists of three major components such as a YOLOv8s, a Dlib application and a web-based system. The YOLOv8s vision module detects human faces and restricted objects (such as mobile phones and books), a Dlib facial recognition application estimates eye and head direction to identify behavioral changes and the web-based monitor detects the behaviors like switching tabs, reducing the window size and exiting full-screen mode. A rule-based layer analyzes these signals to determine the type of violation and takes necessary action such as providing real-time notifications and expanding procedures for repeated errors. The detected activities will be saved as structured recorded files in a MySQL database for transparency and scientific evaluation with suitable supporting links shown on administrator dashboards for monitoring after the evaluation. This experimental evaluation and

DOI: https://doi.org/10.36548/jaicn.2025.4.005
Received: 10.11.2025, received in revised form: 11.12.2025, accepted: 27.12.2025, published: 08.01.2026

dashboard examples demonstrate that the system will monitor proctoring actions in real-time and provide accurate, reviewable data suitable for remote conditions.

Keywords: Online Proctoring, Real-Time Monitoring, YOLOv8, Gaze Estimation, Computer Vision, Academic Integrity, Browser-Based System, Cheating Detection

1. Introduction

When online education first began, computer-based tests were widely used to evaluate student performance. However, online tests cannot provide the same level of real-time, in-person monitoring as classroom-based exams. This lack of control increases the chance of inappropriate behavior, such as accessing restricted devices, copying notes or textbooks, and asking others for answers. Professional proctoring solutions help with this issue, but they require subscriptions, continuous human monitoring and complete access to a student's device. This creates problems related to privacy, cost, reduced acceptability and accessibility. As a result, there will be an increased demand for automated systems that require minimal setup and depend on standard technology. Recent advances in computer vision and deep learning have enabled real-time analysis of student behavior using techniques such as face identification, object localization, and eye tracking. This system offers a lightweight, web-based automated proctoring solution that includes facial detection, eye tracking and the identification of restricted devices like phones and books. The goal is to provide accurate and dependable remote tests while keeping the system accessible and reducing dependence on external proctoring services.

1.1 Problem Statement

With remote exams routine in many courses, lecturers have less control over the conditions in which students attend their tests. The examiner has slight visibility of students' behaviors, such as using a phone, copying from a book, or getting help from someone nearby. Existing proctoring technologies aim to address this issue, but most are challenging to implement widely. Few systems depend on live supervisors viewing video feeds; others require separate apps with extensive access to the student's computer, and some use expensive licenses to operate. Web-based tests require only a webcam and can detect obvious risk situations, such as the presence of multiple faces, a visible phone or book, and attention continuously shifting

away from the screen. This study proposes a real-time, automated proctoring system to ensure equitable online exams.

1.2 Objective

A web-based automated proctoring system develops a lightweight model that uses YOLOv8 and eye tracking to identify student behaviours in online tests.

2. Related Work

Dilini et.al. [1] proposed an internet browser-based system for proctoring using eye-gaze information to identify cheating during online examinations. This is achieved through a webcam plug-in, which records eye pupil movements and then transmits the data to a pre-trained One Class Support Vector Machine (OCSVM) using normal gaze patterns. There are notable differences between the current eye pattern and the reference model, particularly if they are irregular, a signal is triggered that enables monitoring of the individual candidate during an online examination, whereby a human monitor/proctor is not required. Patil et al. [2] designed an AI-based proctoring solution to ensure the integrity of online exams through student behavior observation. The proctored system makes use of computer vision techniques like OpenCV and MediaPipe for eye-tracking and hand movements. As soon as it detects any unusual student behavior, it captures a photo. This photo will be analyzed later by an invigilator. From this study, the vision-based proctoring systems are capable of performing a considerable amount of proctoring tasks.

Motwani et al. [3] explored an automated method for monitoring online exams with minimal human invigilation. The article highlights an AI proctoring system with computer vision to detect irregular activity during exams. The system is designed to record such activity as images or videos for later verification. The work indicates that automated monitoring can enhance the security and administration of online exams. The system is an affordable and flexible alternative to human invigilation. They argue that automated monitoring systems can eliminate cheating during exams. Kaddoura and Gumaei [4] proposed a deep-learning system that detects cheating in online exams in real time. They aimed to improve earlier vision-based methods that relied on hand-crafted image features. The system handles two input types at once: it uses convolutional neural networks to learn visual features from the exam video, and a Gaussian-based discrete Fourier transform to extract features from the audio stream. It was

trained and tested on a public dataset containing various cheating behaviours and achieved high accuracy with a processing time suitable for live use.

Ong et al. [5] described a cheating detection system for online exams using a clustering algorithm to detect similar behaviour among candidates. In this study, the work used an online exam analysis system focusing on facial movements, eye movements, and human gestures. They created a custom-made dataset using 50 candidates who were made to cheat as well as perform normal activities during an exam. These activities were later mapped into a feature map after a clustering analysis allowed normal patterns to be distinguished from cheating ones. They attained an approximate accuracy of 83% in detecting cheating during online exams using unsupervised learning. Sridhar and Rajshekhar [6] proposed an AI-based proctoring solution for online exams, taking over a major part of the supervision process to prevent cheating. They implemented behavior analysis models for monitoring students throughout the entire online test and were capable of storing evidence whenever the system flagged a case of cheating. It was designed as a secure, cost-effective solution, enabling institutions to rely less on human proctors for the supervision process. They believed AI could provide viable, scalable, and trustworthy solutions for online test supervision.

Gaikwad [7] presented an unsupervised machine-learning proctoring system for online exams that works without a live invigilator. The software observes students through their webcams, flags behavior or objects that seem improper, and logs each flagged event so it can be checked later. The study found that this approach provided useful accuracy under normal test conditions and argued that unsupervised models can support large-scale online exams when regular in-person supervision is not possible. Naik et al. [8] designed an online invigilation system that employed the use of Dlib and YOLO for monitoring the activities of the candidates. The system was designed in such a way that it utilized the webcam and microphone to monitor the activities of the candidates through their eye movements, as well as the entry of another candidate within the camera range. Additionally, the system was able to detect unauthorized devices. If the system detected this activity, it alerted the candidates without the need for human invigilators.

Singh et al. [9] described an online exam proctoring system that uses YOLO object detection to support academic integrity in online exams. Their model combines video and audio to track head direction, catch attempts to fool the system, monitor mouth movement, detect mobile phones, and count the number of people visible on camera. The system can spot extra

individuals or prohibited devices in real time with pre-trained YOLO weights and automatically stop the exam when a violation is detected. The system provides a practical, automated way to make online assessments secure and consistent. Yulita et al. [10] created a deep learning solution capable of identifying cheating during online exams, responding to the challenges presented by online classes during the COVID-19 pandemic. Their solution monitors students' activities using the webcam and a deep neural network to identify questionable activities, achieving an F1-score of 84.52%. As a web-based application in the Indonesian language, the solution provides educators with a useful and credible platform to monitor academic integrity during online exams. From this research, the effectiveness of deep learning in designing a proctoring solution that is scalable and suitable for online education can be assessed.

Abbas & Hameed [11] examined the online proctoring system for exams that utilizes AI and deep learning, specifically the attempt to identify potential cheating signals. They conducted a thorough analysis of 41 studies published between 2016 and 2022 and examined the specifics of each study, including the methods used, the algorithms employed, the data sources, and the techniques utilized for cheating detection. Their analysis also points out several vulnerabilities exhibited by the current studies, such as inadequate training environments and slow adoption of new technologies. This article provides clear insight into the current situation regarding automated online proctoring and the fields that require improvement. Zuo et al. in [12] developed an online cheating detection system using a modified YOLOv8 model and the addition of an attention mechanism. This system observes candidates remotely and marks unusual observations automatically so that the human proctor does not have to continually monitor the screen. The detection accuracy achieved on recorded videos of exam sessions was 82.71% and was very hardware-efficient. It appears that by integrating object detection models and attention models in proctoring systems, scalability, affordability, and sufficient accuracy can be achieved.

Hylton et al. [13] studied whether webcam monitoring during online exams affects cheating. They ran an experiment with two groups: one group took a webcam-proctored exam, and the other took the same test without video monitoring. The average scores were almost the same, but students in the proctored group finished more quickly and said they felt they had fewer chances to cheat. The authors concluded that visible webcam use can influence behavior and help maintain exam integrity. Garg et al. [14] proposed a CNN-based "virtual exam

controller," incorporating students' camera capabilities for surveillance purposes during online exams. This tool adopts the application of Haar Cascade Classifiers and deep learning models for the detection and identification of faces within the video stream. This approach enables the identification of situations such as the presence of more than one face within the video and other potential suspicious activity during the exam. If such activity is detected, the software will be programmed with predetermined parameters for sending notifications as a strategy for ensuring the elimination of cheating and making online exams more standardized.

Ahmad et al. [15] describe an online proctoring system that uses deep-learning methods to keep track of students during remote exams. Their system employs HOG-based face detection, OpenCV face recognition, and eye-blink checking to spot static or spoofed images, as well as object detection to identify items such as phones or extra laptops. Tested on the FDDB and LFW datasets, it achieved around 97% accuracy for face detection and 99.3% for face recognition, suggesting that biometric monitoring can be used reliably in automated exam supervision. Erdem and Karabatak [16] examined whether deep learning models and other machine learning algorithms can identify cheating activity during online examinations. The work tested their model on 129 datasets from different examination scenarios. The authors' results showed that, with a regression task of determining cheating activity, they reached 80.9% accuracy with their 5-layer DNN model and 96.9% accuracy with their 10-layer model for a yes/no task. However, with multi-class cheating labels, they achieved a maximum accuracy of 97.7% with XGBoost. The authors also applied SHAP and LIME techniques to examine which features of activity were most meaningful.

3. Proposed Work

This study explains how the web-based online proctoring system detects student misbehavior during online exams. The proposed tool is designed for easy access and a simple implementation process. Students need a built-in camera and a current web browser with no additional software to install or configure. This method uses real-time computer vision to automate invigilation while minimizing security risks and human involvement.

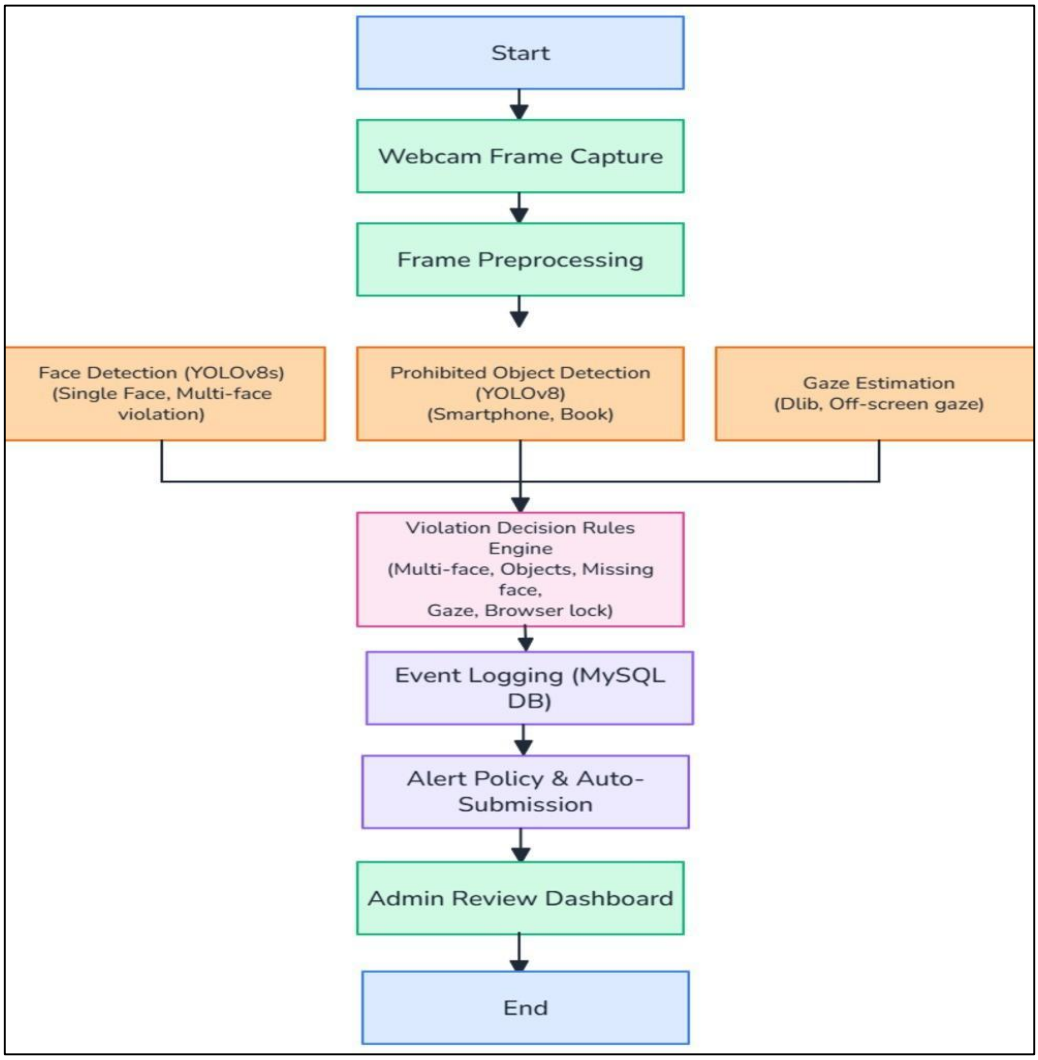

Figure 1. System Workflow

Figure 1 represents the proposed system's workflow demonstrating how the detection models handle input data and transfer it from the backend to the frontend.

3.1 Datasets

The multi-class object detection dataset used in this work had three different classes of objects: Phone, Face and Book. The Mobile Phone Detection Computer Vision Model dataset [18] had 23,749 images with class label 2, which were further divided into 20,772 training images, 1,984 validation images, and 993 testing images. For faces, the AiFaceAttribute 2.0 Computer Vision Dataset [17] contained 18,258 images with class label 0, which were divided into 16,318 training images, 1,581 validation images, and 354 testing images. For books, Book Detection with YOLO Computer Vision Dataset [19], Books Computer Vision Dataset [20], and BookData Computer Vision Model [21] were combined for a total of 9,711 images with

class label 1, further divided into 7,695 training images, 1,178 validation images, and 838 testing images. The total dataset comprised 44,785 training images, 4,743 validation images, and 2,185 testing images. All bounding boxes were annotated using the YOLOv8 format, which ensured consistent annotation for the three classes of objects.

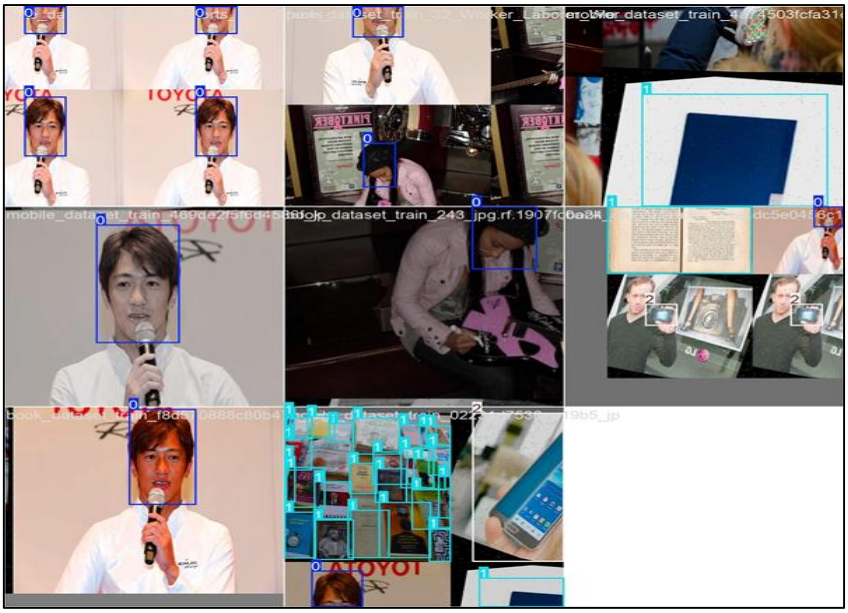

Figure 2. Sample Dataset

Figure 2 presents example images from the dataset including faces, students, and banned items, and illustrates the range of object sizes, viewing angles, and exam conditions used to train and test the model.

3.2 Data Preprocessing

Preprocessing and augmentation were specified independently for each category because the photographs do not have the same form or substance. For the Phone and Face datasets, all images were resized to 416 × 416 using stretch resize. This provides the detector with a fixed square input and retains as many pixels as possible on the real item, rather than creating empty padding around it. From each original image, we created three augmented copies using horizontal flip (probability 0.5), vertical flip (probability 0.5), random rotation between −26° and +26°, exposure change between −25% and +25%, and salt-and-pepper noise on 4% of the pixels. These processes replicate common variations in webcam broadcasts, such as slight camera rotation, mirrored views, uneven brightness, and sensor noise.

Preprocessing for the Book dataset was specific to each subset. In the first subset (5,311 images), we applied rotation‑based augmentation: every image was rotated by 0°, 90°, −90°, or 180° with equal probability to make the model less sensitive to how a book is oriented on the desk. The second subset (1,872 pictures) was handled differently. We first corrected the image orientation using EXIF metadata, then resized each image to 640 × 640 with stretch resize and did not add further augmentation. The increased size maintains fine features such as page borders and covers while maintaining the detector's predetermined input size. Constant preprocessing throughout the train, validation, and test splits ensures accurate bounding boxes and reliable detection of faces, phones, and books in noisy, low-resolution webcam footage with varying views and illumination.

Figure 3. Dataset Preprocessing

Figure 3 shows sample photos from the dataset, comparing the originals to the preprocessed versions and demonstrating the data prepared for accurate model training and testing.

3.3 Model Architecture

In this study, the YOLOv8s variation serves as the primary object-detection model. It consists of three main components: a backbone that analyzes each video frame to collect visual data, a neck that integrates features at various sizes and a detecting head that generates the final item predictions. The backbone can detect both large targets, such as a student's upper body and small objects, such as phones and books. The neck connects these multi-scale properties

together allowing the model to locate items of various sizes depending on their appearance in the camera view. The detecting head provides a class name and bounding box for each item, allowing the system to monitor faces, people, phones, and books simultaneously in real time.

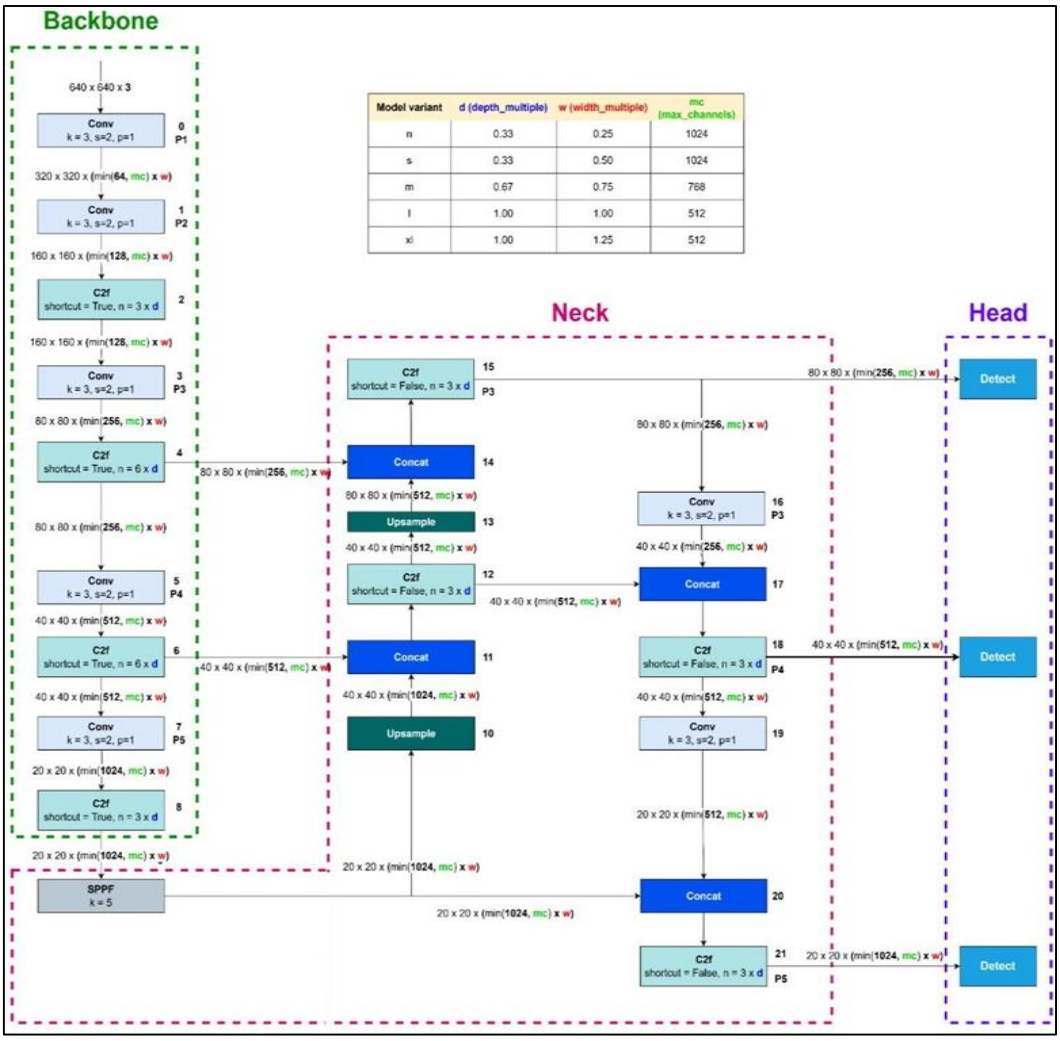

Figure 4. Visualisation of the YOLOv8 Architecture Including Backbone, Neck, and Detection Head Adapted from Hidayatullah et al. [22].

Figure 4 illustrates the YOLOv8s architecture used in the proctoring system, explaining the received video frames processed to extract features, integrate data across several scales, recognize students and restricted items in real time.

3.4 Model Training and Optimization

The model was trained for 150 epochs with a batch size of 8, using images resized to 640 × 640 pixels. We started from pretrained YOLOv8s weights and fine-tuned them with an Adam optimizer (learning rate 0.01, momentum 0.937, weight decay 0.0005). Automatic

mixed precision was turned on to speed up training. To improve generalization, we used several data-augmentation steps, including horizontal flips, scaling, translation, random erasing, RandAugment, and mosaic augmentation during the first 10 epochs. Validation was carried out throughout training. All experiments used a single GPU with 8 data-loader workers, and the final model runs fast enough to detect faces and prohibited items in real time during online exams.

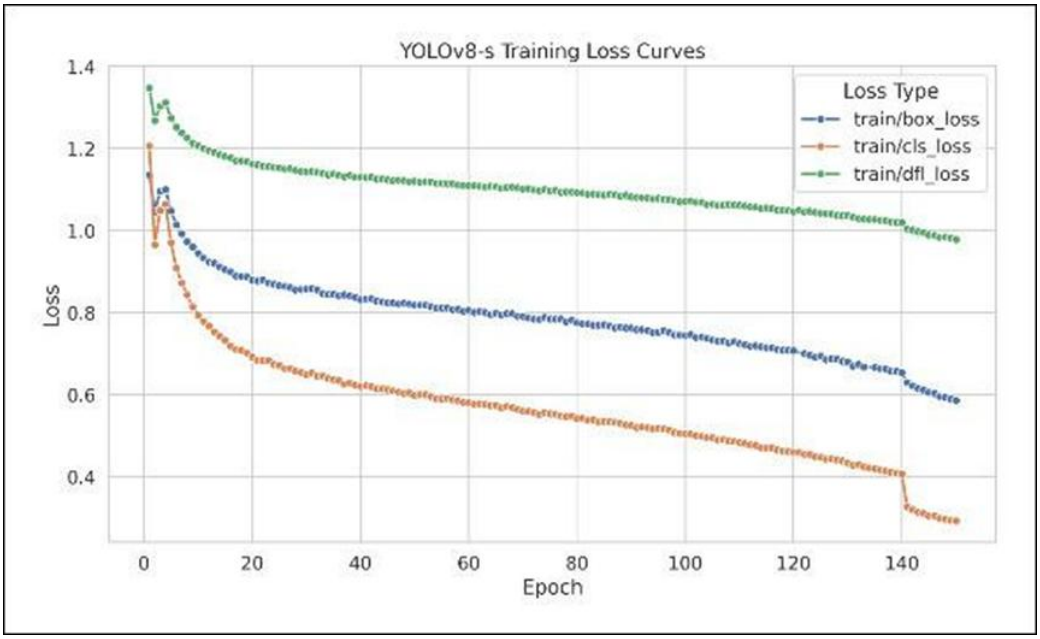

Figure 5. Training Loss Curves Over Epochs

Figure 5 shows the classification and localization loss during YOLOv8s training. The steady drop in these curves indicates that the model improves over time and settles into a stable solution.

3.5 Frame-Based Frontend–Backend Pipeline

The proctoring system has a framework-based client-server architecture. In the web browser, the candidate's camera records photos at various times, which are reduced to image frames. These frames are transmitted to the backend one by one via a continuous WebSocket connection. On the server, a WebSocket handler uses OpenCV to decode the base64-encoded JPEG and convert it to a BGR 'numpy' array. The frame is delivered to an internal detection pipeline that runs object-detection models and checks the set of proctoring rules. Any unusual behavior, along with the required information is recorded in a database, and an overview of that behavior and any rule violations is created. The report is sent back to the client via the

same WebSocket channel allowing for near real-time monitoring without requiring a continuous video stream.

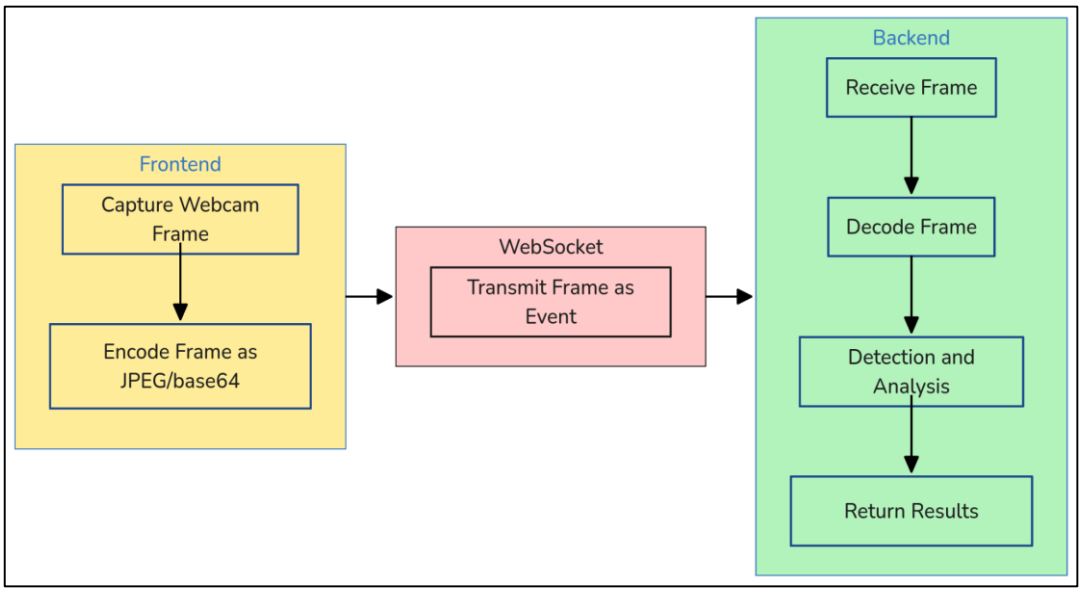

Figure 6. WebSocket-Based Frame Transfer and Backend Analysis Pipeline

Figure 6 depicts the whole frame flow in the proctoring system from frontend capture, encoding, and transmission over WebSocket to backend receipt, decoding, analysis, and result production.

3.6 Gaze Behavior Estimation Module

The eye-tracking system is the second stage in a frame-based pipeline used for other proctoring checks. The backend decodes the JPEG picture for each camera frame received via the WebSocket channel and uses a YOLOv8 detector to recognize faces and other relevant items. YOLOv8's face bounding boxes reduce the focusing areas, which are then processed using a dlib facial-landmark model. This model generates a set of important areas on the face with a significant concentration of points near the eyes. The system detects whether the individual is directed at the screen or away in that frame based on the position of certain eye landmarks, such as the iris' placement inside the eyelid area and the balance between the left and right eyes. Every frame generates an eye identifier as well as the appropriate landmark coordinates. These results are included in the structured proctoring system provided directly to the website. If off-screen looking is detected, it will be saved as suspicious actions in the database. The module combines YOLOv8 for face localization with dlib for landmark

estimation, creating a simple and effective gaze-based monitoring mechanism within the proctoring system.

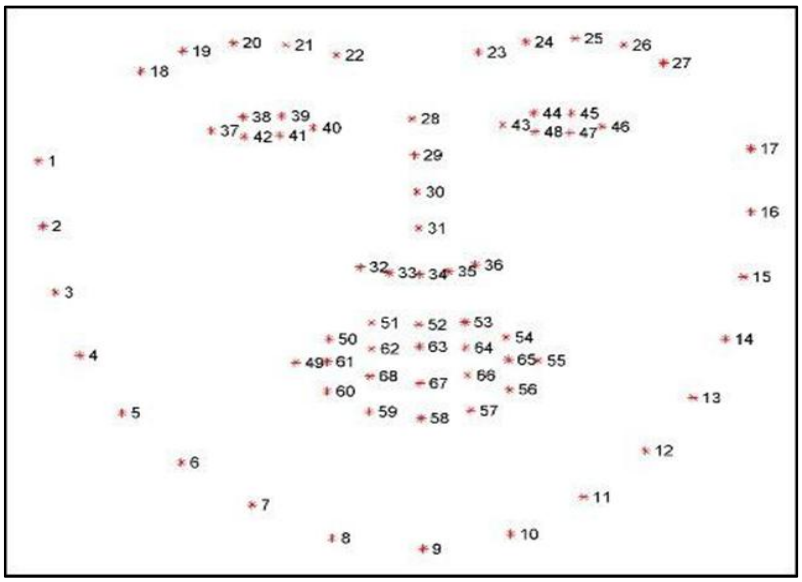

Figure 7. Visualisation of the 68 Facial Landmark Configuration from the iBUG Facial Point Annotations Dataset, Adapted from PyImageSearch [23].

Figure 7 shows the distribution of 68 facial landmarks over the main areas of the face. These points provide the geometric data required to identify the eyes and estimate gaze direction in the proposed system.

3.7 Event-Driven Alert Logging

In this proposed system, the inputs received from detecting outputs and activities on the browser used by the candidate during the examination are moderated through several predetermined rules. Once a rule has been violated, it activates a specific response while collecting details of the occurrence in a MySQL database, which can be viewed later through an admin interface for each examinee. This is shown in Table 1.

Table 1. Proctoring Event Types, Trigger Conditions, and Recorded Evidence

Violation Type	Triggering Condition	Logged Data Stored in Database	System Action
Multiple Faces in Frame	More than one face detected in consecutive video frames	User ID, time of event, violation type, reference to the stored frame	Warning flag raised and event recorded

Face Not Visible / Blank Screen	Face not detected for longer than a defined time limit	User ID, time of event, violation type, reference to the stored frame	Warning flag raised and event recorded
Smartphone Detected	YOLOv8-based detector identifies a mobile phone above a set confidence level	User ID, time of event, violation type, reference to the captured frame	Alert generated and event recorded
Book / Unauthorised Material	YOLOv8-based detector identifies a book or printed material on the desk	User ID, time of event, violation type, reference to the captured frame	Alert generated and event recorded
Suspicious Gaze Direction	Estimated gaze remains directed away from the screen beyond a set limit	User ID, time of event, violation type, reference to the captured frame	Alert generated and event recorded
Browser Focus Lost	User switches tab, minimises the window, or exits full-screen mode	User ID, time of event, violation type	Warning issued and event recorded
Repeated Browser Violations	More than three browser-focused warnings during a single exam session	User ID, time of event, violation type	Automatic exam submission and final log stored

All events related to proctoring are logged in the suspicious_events table. For every single event, the system logs the candidate's username, the test ID associated with the event, the timestamp of the event, the nature of the violation, and, if captured by the webcam, a reference to the image captured. Foreign-key links to the users and tests tables ensure that every record is tied to a specific candidate and exam, supporting both real-time alerts during the test and structured post-exam review with visual evidence.

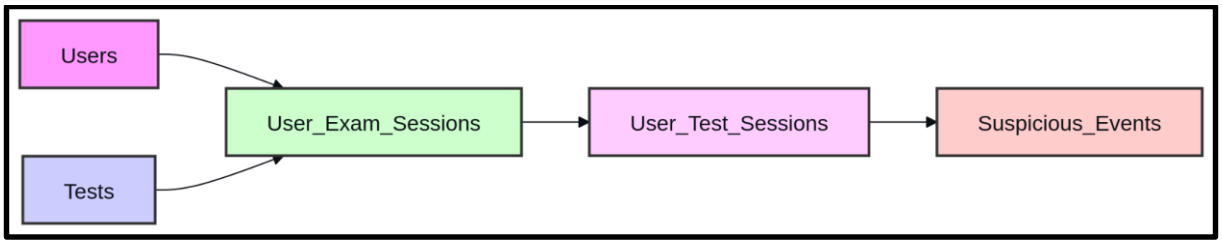

Figure 8. Flow of Entities in Proctoring Database

Figure 8 illustrates how the database tables relate to one another, connecting user, test, and session records with the logged events. It shows that each suspicious event is linked to a

particular candidate and test, which makes it straightforward to trace, inspect, and review proctoring incidents.

3.8 Handling Latency and Detection Failures

The proctoring system maintains an active WebSocket connection and follows an identified sample time period. At each sample interval, the web page takes and submits the current webcam image for analysis. The sample rate is set in advance for the whole session; it doesn't change during runtime. There is no adaptive frame rate. The interval is chosen before implementation, balancing response and computation or network load. On the backend, every image is processed as soon as it comes and replaces the previous one without being queued. This reduces buffering, lowers the possibility of irrelevant frames being examined when the network slows down, and allows the server to return the result over the same channel, maintaining the monitoring dashboard and warning logic up to date. Short, temporary losses of the face signal are handled with a small grace period. When a received frame does not contain a detectable face, the system starts or refreshes a per-user timer instead of immediately counting a violation. A "no face detected" event is logged only if the absence continues beyond a short threshold, typically several consecutive samples or roughly three seconds. This rule filters out spurious alerts caused by brief lighting changes, quick head movements, or momentary network delays.

3.9 Escalation Policy and Cooldown Period

The system has a gradual monitoring method in which suspicious behaviors are initially recorded, followed by user warnings, and finally exam cancellation with manual review if violations continue. A maximum of five warnings is allowed per candidate for all monitored rules. A 10-second relaxation has been established after each logged violation to prevent frequent triggering by temporary errors, while similar occurrences are ignored. When the warning limit is reached, the examination session is immediately ended and the attempt is marked for a post-exam manual review by an administrator. This technique maintains a balance between robustness and false positives, allowing for immediate action in confirmed cases of illegal activity.

3.10 Thresholds

The system's rule levels are defined effectively using validation data instead of fixed heuristics. Detection confidence levels are established by analyzing precision-recall trade-offs across various operational points to balance sensitivity and false positives. Face and head position levels have been adjusted using recorded landmark deviations to differentiate between normal and suspicious motions. Temporal thresholds are used to guarantee that only continuous violations are highlighted, improving durability to temporary detection noise. The final limits were designed to maintain a balance between effective memory (to prevent missed errors) and proper precision to avoid false alerts. Table 2 represents the performance based on confidence.

Table 2. Per Class Performance based on Confidence

Confidence Threshold	Face(P/R/ mAP@0.5)	Book(P/R/mAP@0.5)	Mobile(P/R/mAP@0.5)
0.5	0.91 / 0.72 / 0.83	0.63 / 0.83 / 0.63	0.97 / 0.92 / 0.95
0.6	0.94 / 0.65 / 0.80	0.65 / 0.80 / 0.62	0.97 / 0.91 / 0.95
0.7	0.97 / 0.51 / 0.74	0.67 / 0.76 / 0.61	0.98 / 0.89 / 0.94
0.8	0.99 / 0.22 / 0.60	0.68 / 0.67 / 0.58	0.99 / 0.83 / 0.91

4. Results and Discussion

The automated proctoring system was evaluated in controlled tests that simulated distant exam sessions. All experiments were conducted in a regular web browser with a consumer-grade camera, which is the common configuration for students to use at home. The examination assessed facial recognition, restricted object detection, and eye tracking. Tests were repeated under various lighting conditions, backdrops, and viewing angles to see if the secure system was maintained. We also verified that notifications were delivered on time and that each incident was recorded in the logs.

4.1 Model Performance Evaluation Curves

Training and evaluation will demonstrate the proctoring detection model (for faces, mobile devices, and books) as it evolves over time. Figure 7(1) shows that the accuracy, recall, and F1-score values increase significantly during the first few epochs before slowing down, indicating that the model understands the key concepts and achieves a level of stability and

consistency. However, the precision level remains higher than the recall level, indicating that the model is more cautious with alarms and is less prone to false positives. Figure 7(2), the Precision-Recall Curve, verifies this by showing that the precision is high even when the recall level is still satisfactory. This is important for the proctoring system since phones and books must be identified, but student behavior shouldn't be taken as false alarms. Figure 7(3) shows the same trend for mAP values, as mAP@0.5 and mAP@0.5:0.95 display comparable progress patterns and reach stability after a sudden increase.

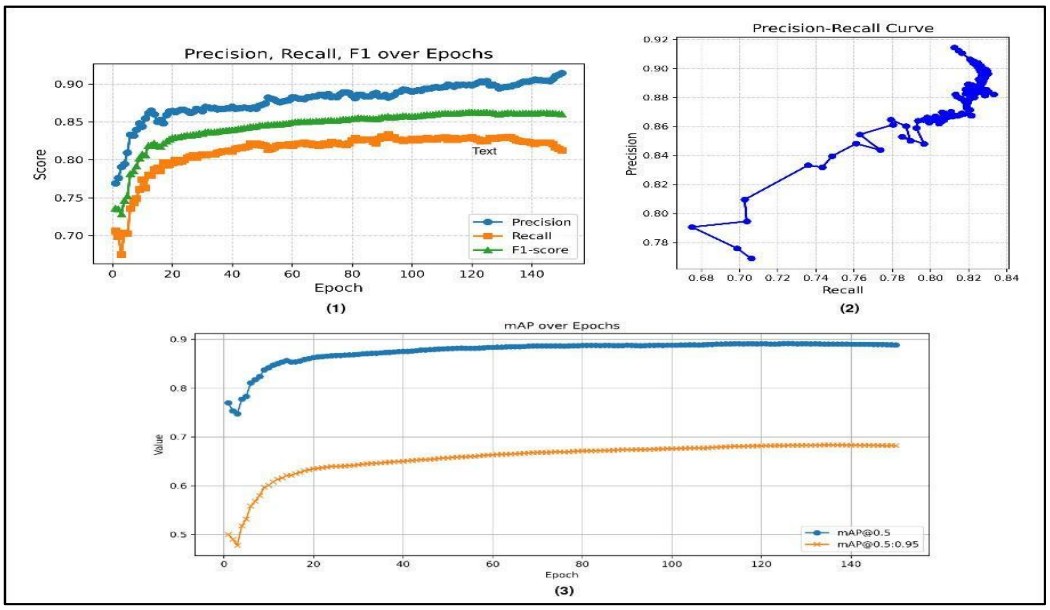

Figure 9. Variation of Precision, Recall, F1-Score, and Mean Average Precision During Validation

From Table 3, the proposed final model is appropriate for use in real-time proctoring tasks. Its precision value of 0.914 indicates high accuracy, meaning the majority of the detected faces, phones, and books are indeed accurate, ensuring limited false alerts during exams. Furthermore, its high recall of 0.813 ensures that a vast majority of the objects are detected, aiming to capture illegal devices during exams. The F1-score of 0.860 ensures a sound balance between accuracy and scope. Based on its bounding box accuracy, the mAP@0.5 of 0.889 implies that performance on a reasonable overlap threshold was good, while its mAP@0.5:0.95 of 0.682 implies that its performance on a more rigid localization threshold was still sound.

Table 3. Performance Metrics Table

Metric	Value
Precision	0.914
Recall	0.813
F1-score	0.860
mAP@0.5	0.889
mAP@0.5:0.95	0.682

4.2 Confusion Matrix Analysis

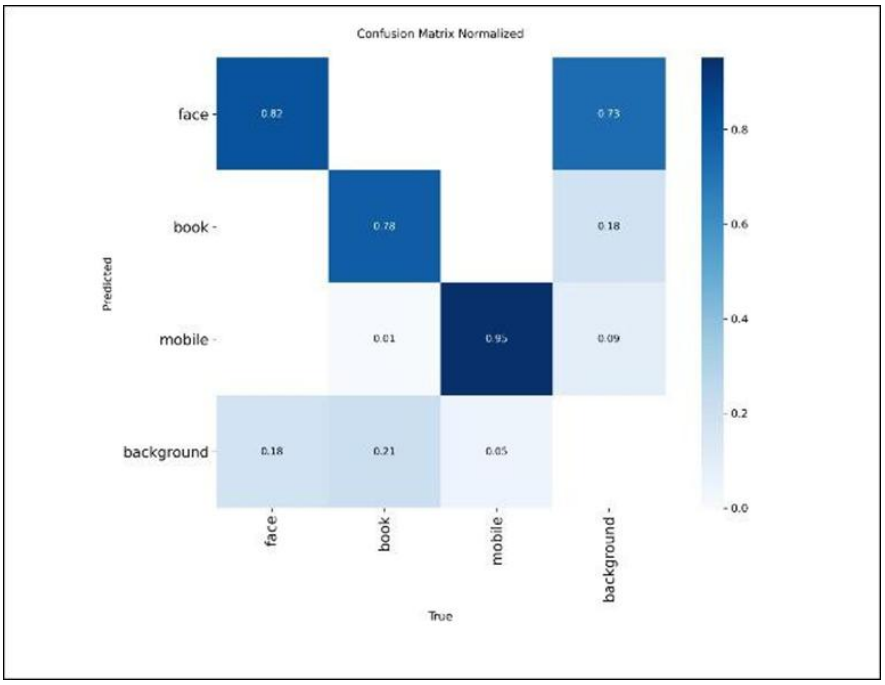

Figure 10. Normalised Confusion Matrix

The normalized confusion matrix shows that the model recognizes the three monitored classes reasonably well. About 82% of face instances, 78% of book instances, and 95% of mobile-phone instances are classified correctly. Most of the remaining errors for these classes occur when the model predicts background instead, meaning that some objects are simply missed. The opposite problem appears for the background class: many background regions are wrongly labeled as objects, mainly as faces (approximately 0.73), and to a lesser extent as books (approximately 0.18) and mobiles (approximately 0.09). Overall, the model performs

well in detecting real faces, phones, and books, but it still generates several false alarms in background areas, especially for faces, which can occasionally lead to unnecessary flags in the proctoring system. Figure 10 shows the normalized confusion matrix of the YOLOv8s proctoring model. The system correctly detects faces, books, and mobile phones most of the time, while some mistakes occur with background areas, especially faces, which can occasionally trigger false alerts.

4.3 Face, Gaze, and Prohibited Object Detection Performance

The face-detection part of the system worked consistently under normal exam conditions. When a single student was in front of the camera, it detected the face and updated the status as well as the head and eye direction if the student looked away. If the student moved out of view, it correctly showed “No Face Detected.” When another person entered the frame, the system reported more than one face (faceCount = 2), issued a warning, and saved each of these events with a timestamp for later review.

The gaze tracker could follow the gaze of the test taker accurately throughout the test. When the student was taking the test by looking at the screen, this was marked as normal gaze, and small natural blinks were eliminated to avoid false warnings. When the eyes remained off the computer screen past a certain threshold, the system generated a warning and marked the time and image frame. In Figure 11, the eye and head movement is indicated on the dashboard and is dependent on real-time tracking of facial points. The gaze estimation module is inactive if no face is detected on the video image. Thus, no gaze estimation decision is made until a valid face is detected again. Figure 11 shows a proctoring interface that analyzes faces and eye points in real time to identify gazing, blinking, looking away, on-screen focus, fast eye movement, rotated head, presence of multiple faces and partial faces, absence of a face and marks any violations as needed. The system’s restricted device check will identify a phone or book and report it immediately to the admin. When a phone or book is viewed on the screen, the dashboard displays a message such as "Phone Detected" or "Book Detected". It also continues to show the face and eye status with a warning message. The system will record the incident in the log including the exact moment and image frame each time it occurs. This provides a clear record of restricted devices detected during the exam.

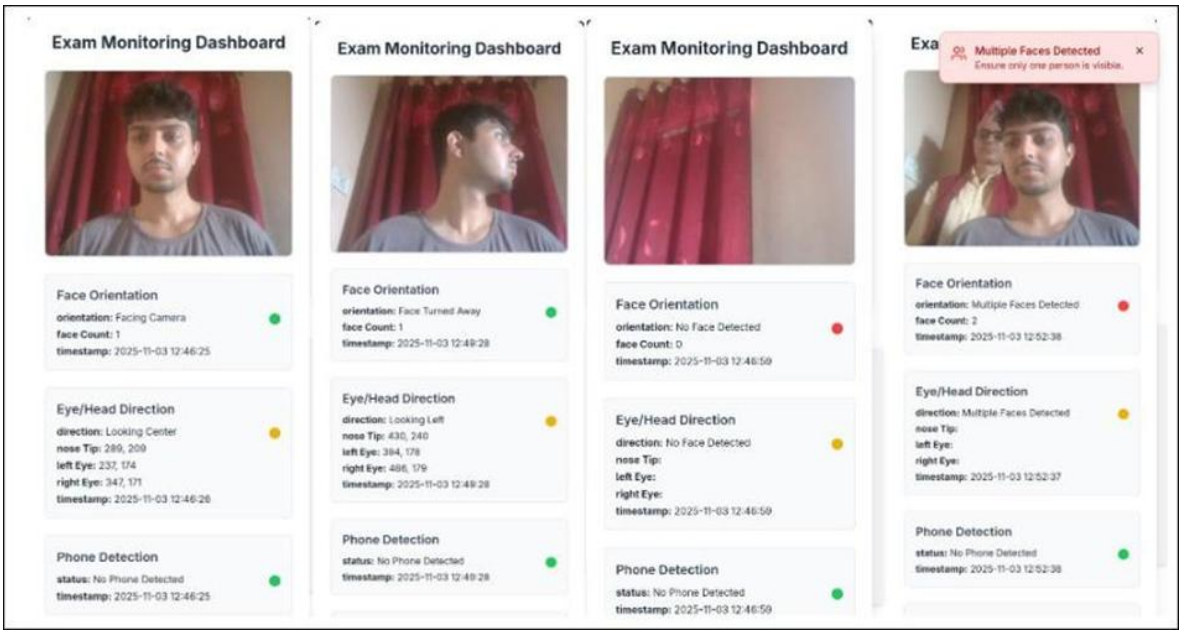

Figure 11. Face and Gaze Detection in Proctoring Interface

Figure 12 shows that the proctoring interface recognizes the restricted objects, including books and mobile devices, in real-time. The system displays a warning pop-up window when an item is identified, highlighting the image as a problem and saving the date for future reference.

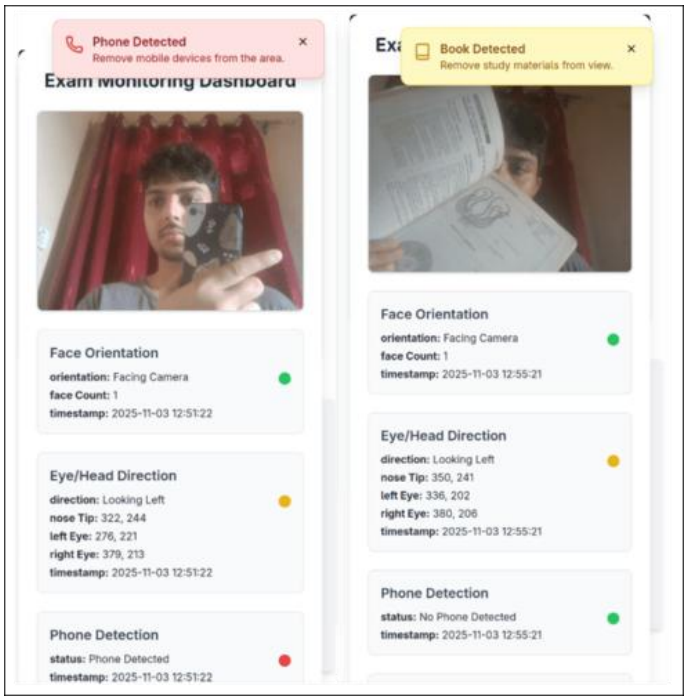

Figure 12. Prohibited Object Detection Results in the Proctoring Interface

4.4 Log Records and Evidence Capture

The system maintains a structured log entry for every proctoring event. When any of the face, eye, object-detection, or web-based modules raise an alert, the backend saves the user ID, exam or session ID, exact time, type of violation, confidence value, and a pointer to the related image frame or screen grab. All this information is stored in a database and can be exported if needed for review or appeals. In testing, every alert shown on the dashboard has a matching log entry, providing a complete, time-ordered account of the exam.

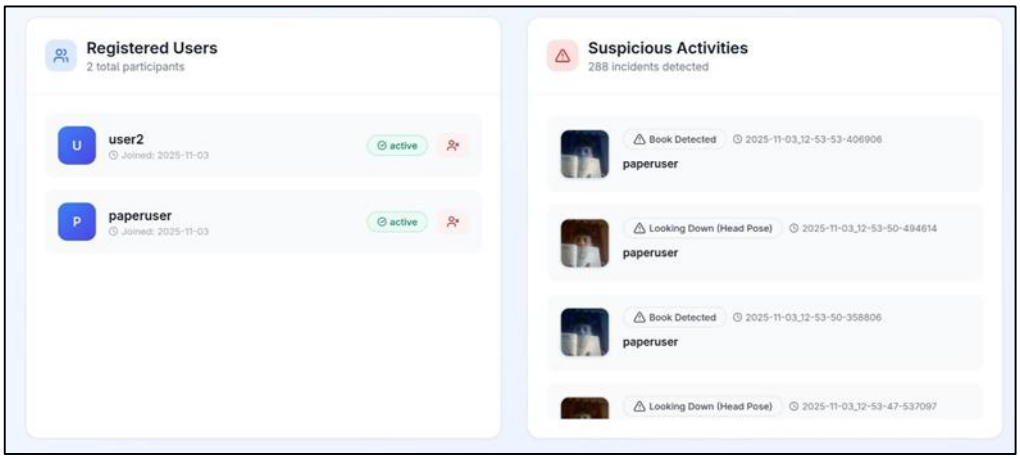

Figure 13. Suspicious Events Log

Figure 13 shows the admin panel view of the suspicious event log, including all the alerts that occurred during the exam. Such events include gaze violations, the absence or co-occurrence of faces, as well as the occurrence of restricted objects, alongside the precise exam time and the video frame.

4.5 Overall Performance and Discussion

The proposed proctoring system shows strong performance for real-time exam monitoring. The YOLOv8s-based detector consistently recognizes faces, mobile phones, and books, while the gaze-estimation and web-based components capture off-screen viewing and tab-switch behavior. Most relevant violations are detected and each event is logged with a timestamp and linked to visual evidence to support later review. The main drawback is the presence of some false positives in cluttered or visually complex scenes, suggesting that improved background handling and filtering would further increase the system's robustness.

5. Future Scope

In the future, the system will improve reliability and avoid false warnings by training under distributed situations and verifying the detections using multiple video frames. The system can also be enhanced to detect other restricted objects and improve the eye based tracking. Furthermore, future work will focus on real-time implementation, improved privacy protection and expanded testing in exam situations.

6. Conclusion

The proposed system will create an automated online proctoring system that includes live video monitoring and a web-based activity tracking process. It uses YOLOv8 facial recognition to identify books, mobile devices and faces. This method also used facial landmarks from Dlib to determine eye and head position. This system will detect web-based activity includes the use of several tabs during the evaluation period. Every incident will be stored in a system with supporting visual evidence. The system scored well in terms of detection accuracy and its dashboard performed well in typical examination circumstances such as single face verification matching, absence of face/multiple faces, eye movement, and limited item detection. Overall, the findings show that the developed scheme is effective and reliable for automating tests and generating reports.

References

[1] Dilini, Nimesha, Asara Senaratne, Tharindu Yasarathna, Nalin Warnajith, and Leelanga Seneviratne. "Cheating detection in browser-based online exams through eye gaze tracking." In 2021 6th International Conference on Information Technology Research (ICITR), pp. 1-8. IEEE, 2021.

[2] Patil, None Akshay Vinod, None Payal Atul Chavan, None Shruti Rangnath Jadhav, None Atharva Umesh Phodkar, None Mayuresh Bhagwat Gulame, and None Aarti Paresh Pimpalkar. “Online Exam Proctoring Application using AI.” International Journal of Science and Research Archive 15, no. 2 (May 28, 2025): 1228–34. https://doi.org/10.30574/ijsra.2025.15.2.1440.

[3] Motwani, Sahil, Chirag Nagpal, Manav Motwani, Nikhil Nagdev, and Anjali Yeole. "AI-based proctoring system for online tests." In Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST2021). 2021.

[4] Kaddoura, Sanaa, and Abdu Gumaei. "Towards effective and efficient online exam systems using deep learning-based cheating detection approach." Intelligent Systems with Applications 16 (2022): 200153.

[5] Ong, Seng Zi, Tee Connie, and Michael Kah Ong Goh. "Cheating detection for online examination using clustering based approach." JOIV: International Journal on Informatics Visualization 7, no. 3-2 (2023): 2075-2085.

[6] Sridhar, Arjun, and J. S. Rajshekhar. "AI-integrated Proctoring system for online exams." Journal of Artificial Intelligence and Capsule Networks 4, no. 2 (2022): 139-148.

[7] Gaikwad, Vaibhav Ratan. "A Novel Unsupervised AI/ML based proctored system." PhD diss., Dublin, National College of Ireland, 2022.

[8] Naik, Chinmaya Nilakantha, Adarsh S Shetty, Vismita Kuppayya Naik, Rakshith CP, and IJARCCE. “Dlib and YOLO Based Online Proctoring System.” International Journal of Advanced Research in Computer and Communication Engineering. Vol. 11, April 2022. DOI: 10.17148/IJARCCE.2022.11586.

[9] Singh, Tripty, Rekha R. Nair, Tina Babu, and Prakash Duraisamy. "Enhancing academic integrity in online assessments: Introducing an effective online exam proctoring model using yolo." Procedia Computer Science 235 (2024): 1399-1408.

[10] Yulita, Intan Nurma, Fauzan Akmal Hariz, Ino Suryana, and Anton Satria Prabuwono. "Educational innovation faced with COVID-19: deep learning for online exam cheating detection." Education Sciences 13, no. 2 (2023): 194.

[11] Abbas, Muhanad Abdul Elah, and Saad Hameed. "A systematic review of deep learning based online exam proctoring systems for abnormal student behaviour detection." International Journal of Scientific Research in Science, Engineering and Technology 9, no. 4 (2022): 192-209.

[12] Zuo, Yan, Soo See Chai, and Kok Luong Goh. "Cheating detection in examinations using improved yolov8 with attention mechanism." *Journal of Computer Science* 20, no. 12 (2024): 1668-1680.

[13] Hylton, Kenrie, Yair Levy, and Laurie P. Dringus. "Utilizing webcam-based proctoring to deter misconduct in online exams." Computers & Education 92 (2016): 53-63.

[14] Garg, Kavish, Kunal Verma, Kunal Patidar, and Nitesh Tejra. "Convolutional neural network based virtual exam controller." In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 895-899. IEEE, 2020.

[15] Ahmad, Istiak. "A novel deep learning-based online proctoring system using face recognition, eye blinking, and object detection techniques." International Journal of Advanced Computer Science and Applications (2021).

[16] Erdem, Bahaddin, and Murat Karabatak. "Cheating detection in online exams using deep learning and machine learning." Applied Sciences 15, no. 1 (2025): 400.

[17] Hunh. "AiFaceAttribute 2.0 Dataset." Roboflow Universe (Roboflow), July 2025. https://universe.roboflow.com/hunh/aifaceattribute-2.0-o3oih.

[18] Tusker AI. "Mobile Phone Detection Dataset." Roboflow Universe (Roboflow), August 2023. https://universe.roboflow.com/tusker-ai/mobile-phone-detection-2vads.

[19] Darmawan, Fransiscus Xaverius Surya. "Book Detection With YOLO Dataset." Roboflow Universe (Roboflow), May 2025. 1872 https://universe.roboflow.com/fransiscus-xaverius-surya-darmawan-qaqpt/book-detection-with-yolo.

[20] DJ2. "Books Dataset." Roboflow Universe (Roboflow), March 2025. https://universe.roboflow.com/dj2/books-qqqmx.

[21] Esha. "BookData Dataset." Roboflow Universe (Roboflow), March 2025. https://universe.roboflow.com/esha-imlgz/bookdata.

[22] Hidayatullah, Priyanto, Nurjannah Syakrani, Muhammad Rizqi Sholahuddin, Trisna Gelar, and Refdinal Tubagus. "YOLOv8 to YOLO11: A Comprehensive Architecture

In-depth Comparative Review." *arXiv (Cornell University)*, January 23, 2025. https://doi.org/10.48550/arxiv.2501.13400.

[23] Rosebrock, Adrian. "Facial Landmarks With Dlib, OpenCV, and Python - PyImageSearch." PyImageSearch, July 3, 2021. https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.