International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

AI-Based Intelligent Online Exam Proctoring System

E.Drakshavalli[1] ,Vjs Pranathi[2], B.Akshitha[3], A.Harshitha[4]

[1]Assistant Professor; Bhoj Reddy Engineering College For Women Department Of Computer Science And Engineering (Ai&Ml), Hyderabad, India.

[2,3,4]B.Tech Students; Bhoj Reddy Engineering College For Women Department Of Computer Science And Engineering (Ai&Ml), Hyderabad, India

pranathivjs@gmail.com

Abstract

The rapid growth of online education has increased the use of online examinations. However, maintaining fairness and preventing malpractice during online exams is a major challenge because students are not physically monitored by invigilators. Traditional methods such as manual monitoring through video calls are time-consuming and difficult to manage when a large number of students are involved.

This project presents an AI-Based Intelligent Online Exam Proctoring System that helps automate the monitoring process using computer vision techniques. The system uses a webcam to observe the student during the examination and detect suspicious activities such as absence of face, presence of multiple faces, or unusual movements. Based on these observations, a cheating score is calculated to identify potential malpractice.

The results of the monitoring process are stored in a database, and a web-based dashboard allows the proctor to view and analyze the exam records. This system reduces human effort, improves monitoring efficiency, and provides a scalable solution for maintaining academic integrity in online examinations.

1. Introduction

Online examinations have become increasingly popular with the rapid growth of digital education. However, ensuring fairness and preventing malpractice during online exams remains a significant challenge. Traditional online examination systems depend heavily on manual monitoring by human proctors, which is often inefficient and prone to errors. To overcome these limitations, this project proposes an AI-based intelligent online exam proctoring system that automatically monitors students using computer vision techniques and identifies suspicious behavior during examinations.

2.Scope

The scope of the proposed system involves real-time monitoring of students through webcams to maintain exam integrity. The system is designed to automatically detect suspicious activities such as the absence of a face, the presence of multiple faces, and frequent looking away from the screen. It generates a cheating score based on observed behavior and stores exam results in a database for later review by proctors. Additionally, the system reduces the need for manual supervision and can be extended in the future to include advanced features such as eye tracking, object detection, and detailed analytics.

ExistingSystem

In the current online examination systems, proctors manually monitor students through live video feeds. The effectiveness of such monitoring largely depends on the attentiveness and efficiency of human proctors. These systems lack automated mechanisms to detect cheating, making them difficult to scale for large numbers of students. Furthermore, they do not provide structured reports or recorded evidence for post-exam analysis.

ProblemsinExistingSystem

The existing system faces several limitations, including a high dependency on human proctors, making the process time-consuming and labor-intensive. It is not suitable for large-scale examinations due to limited scalability. The system is also prone to human error and lacks automated methods for detecting cheating behavior. Moreover, it does not maintain structured evidence or detailed exam logs for future reference.

ProposedSystem

The proposed system leverages artificial intelligence and computer vision techniques to automatically monitor students during online examinations. It uses a pre-trained face detection model to ensure the presence of the student and applies rule-based behavior analysis to identify suspicious activities such as no face detected, multiple faces detected, and looking away from the camera. Based on these observations, the system calculates a cheating score and classifies the exam session as normal, suspicious, or cheating. All results are securely stored in a database for proctor review.

AdvantagesofProposedSystem

The proposed system offers several advantages, including automated cheating detection, which significantly reduces human effort. It enables real-time monitoring and is scalable for large-scale examinations. The system ensures consistent and unbiased evaluation while securely storing exam results for future reference and analysis.

3.Requirement Analysis

Functional Requirements

The proposed system is structured into three primary modules, namely the Student Module, Proctor Module, and Admin Module, each designed to

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

perform specific functionalities within the online examination framework. The Student Module facilitates secure participation in online examinations while enabling real-time monitoring through artificial intelligence techniques. It ensures proper authentication and seamless submission of examination responses. The Proctor Module provides capabilities for live monitoring of candidates, detection and review of suspicious activities, and access to detailed reports generated by the system. The Admin Module is responsible for overall system management, including user administration, exam scheduling, configuration of system parameters, and supervision of analytical reports to ensure smooth operation.

Non-Functional Requirements

In addition to functional capabilities, the system is designed to satisfy several non-functional requirements essential for robust performance. The system must deliver high performance by enabling real-time monitoring with minimal latency, preferably within two seconds. Scalability is a critical requirement, allowing the system to support a large number of concurrent users without degradation in performance. Reliability is ensured by maintaining uninterrupted operation throughout the examination process. The system also emphasizes security and privacy by safeguarding user data and adhering to relevant data protection policies. Usability is addressed by providing an intuitive and user-friendly interface that enhances user experience. Furthermore, the system is designed to be maintainable, enabling easy updates and future enhancements. Platform independence is also considered to ensure compatibility across various operating systems.

Computational Resource Requirements

Hardware Requirements

The implementation of the proposed system requires a computing environment equipped with a processor equivalent to Intel i5 or higher, a minimum of 8 GB RAM, and a webcam to support real-time video monitoring. Additionally, at least 100 GB of storage capacity is recommended to accommodate system data, examination records, and logs.

Software Requirements

The system is developed on a platform such as Windows 10 and utilizes Python version 3.8 or higher as the primary programming language for application development and model integration. The user interface is designed using frontend technologies including HTML and CSS. The backend framework is implemented using Flask to manage server-side operations. SQLite is employed as the database management system for storing examination data and user information. Key libraries such as OpenCV and Dlib are integrated to support computer vision and facial analysis functionalities. Development and testing of the system can be carried out using integrated development environments such as Visual Studio Code or PyCharm.

ARCHITECTURE

SOFTWARE ARCHITECTURE

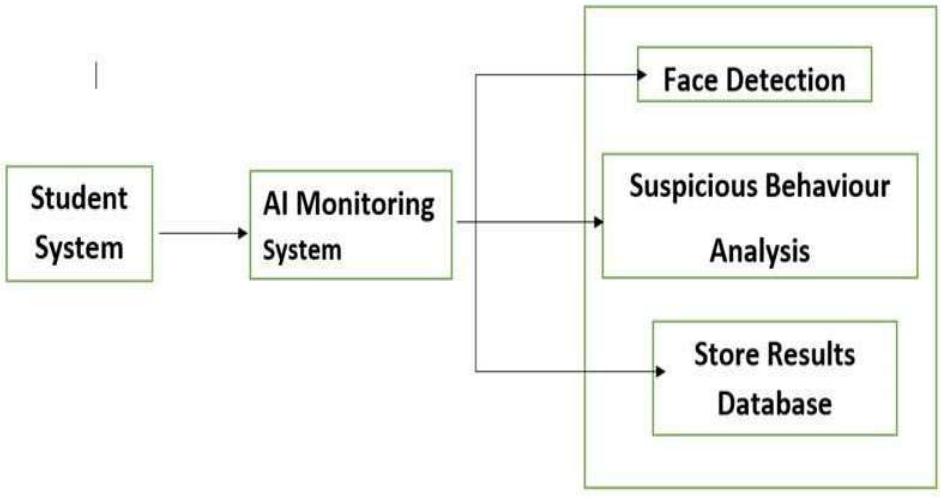

Figure-1

TECHNICAL ARCHITECTURE

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

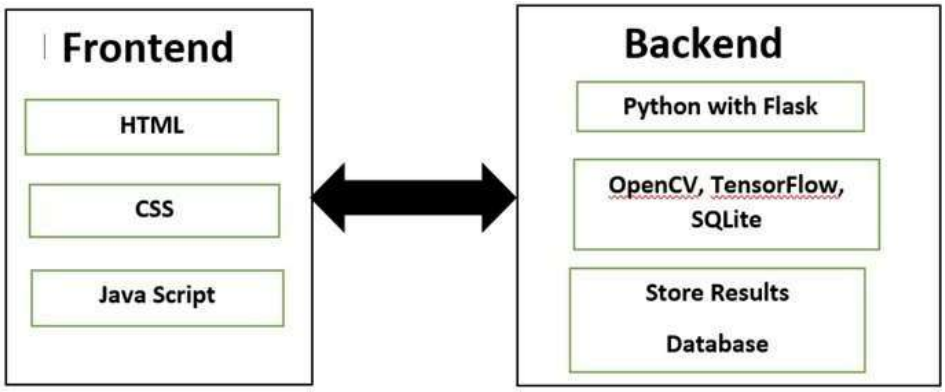

Figure-2

In the fig, computers, software or networks, the overall design of a computing system and the logical and physical interrelationships between its components. A system architecture is the conceptual model that defines the structure, behaviour. and more views of a system. An architecture description isa formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A system architecture can consist of system components and the sub-systems developed that will work together to implement the overall system. There have been efforts to formalize languages to deseribe system architecture, collectively these are called architecture description languages(ADLS).

4. IMPLEMENTATION

Technologies and Libraries Used

The proposed system is primarily developed using Python, which serves as the core programming language for implementing face detection logic, calculating cheating scores, and handling backend operations. The computer vision tasks are performed using OpenCV, an open-source library that enables real-time webcam access, face detection, and image processing. The backend of the web application is built using Flask, a lightweight framework that facilitates server-side development and communication between system components.

The frontend interface is designed using standard web technologies such as HTML and CSS, which provide structure and styling to the proctor dashboard. For data storage, SQLite is used as a lightweight database to maintain records of student activity, cheating scores, and timestamps. Version control and project management are handled using Git and GitHub.

Additional libraries such as NumPy are used for numerical computations and matrix operations on image data, while imutils simplifies image processing tasks within OpenCV. The system also employs pre-trained deep learning models, including *res10_300x300_ssd_iter_140000.caffemodel* and *deploy.prototxt*, which are integrated using the OpenCV DNN module for efficient face detection.

Data Preprocessing

The preprocessing stage begins with capturing real-time video frames from the student's webcam using OpenCV. Each frame is extracted from the video stream and processed individually. To ensure consistent input for the detection model, frames are resized to a fixed resolution, thereby reducing computational complexity and improving processing speed.

Subsequently, each frame is converted into a blob format using the OpenCV DNN module. This process normalizes pixel values and prepares the image for input into the neural network. The processed blob is then passed to the pre-trained face detection model to identify facial regions within the frame.

Once faces are detected, the system extracts bounding box coordinates corresponding to each detected face. Using these coordinates, the center position of the face is computed. This information is essential for analyzing head movement and determining whether the student is looking away from the screen. Finally, the calculated face position is compared with the frame center using predefined thresholds. Any deviation beyond the threshold is considered suspicious and contributes to the cheating score.

Model Implementation

The AI-based online exam proctoring system employs computer vision techniques to monitor student behavior during examinations. Instead of training a custom model, the system utilizes a pre-trained deep learning model based on the SSD framework with a ResNet backbone, available through the OpenCV DNN module. This model detects faces in real time from webcam input.

The detected face coordinates are used to calculate the center of the face, which is then compared with

the center of the video frame. Based on predefined thresholds, the system determines whether the student is attentive or exhibiting suspicious behavior. If irregularities are detected, the cheating score is incremented accordingly.

The system also evaluates specific conditions such as absence of a face, presence of multiple faces, and excessive face movement. These scenarios are treated as indicators of potential malpractice and are incorporated into the scoring mechanism. This rule-based approach enables real-time monitoring without the need for complex model training.

Web Application Frontend

The frontend of the system provides a user-friendly interface for proctors to monitor examination activities. Developed using HTML and CSS, the interface ensures simplicity and responsiveness. It includes a secure login page for authentication, followed by a dashboard that displays monitoring results.

The dashboard presents key information such as student identification, cheating scores, behavioral status (e.g., normal, no face detected, or multiple faces detected), and timestamps of detected events. This structured presentation allows the proctor to quickly assess student behavior and identify suspicious activities during the examination.

Flask Backend and Routes

The backend is implemented using Flask, which acts as the communication bridge between the AI monitoring module and the frontend dashboard. It handles data processing, routing, and database interactions.

Several routes are defined within the application. The root route ("/") displays the login page, while the "/login" route manages authentication. The "/dashboard" route presents monitoring results, including cheating scores and timestamps. Additionally, the "/results" route retrieves stored data from the database and sends it to the frontend for display. This structured routing mechanism ensures smooth data flow and real-time updates.

System Deployment

The system can be deployed in multiple environments depending on the scale and requirements. In local deployment, the application runs on a single machine where both the AI module and the web dashboard operate through a local server. For broader accessibility, the backend can be deployed on a remote server, enabling proctors to monitor students from different locations.

Cloud deployment is also feasible using platforms such as Amazon Web Services, Google Cloud Platform, or Heroku, which support scalable monitoring. The system integrates SQLite for storing examination data, including student IDs, cheating scores, verdicts, and timestamps.

Summary of Component Integration

The system integrates multiple components, including the webcam-based monitoring module, preprocessing unit, face detection model, scoring logic, database storage, backend server, and frontend dashboard. These components work together to provide a complete proctoring solution capable of detecting and reporting suspicious behavior in real time.

Pseudocode

The system operation begins by initializing the webcam and loading the face detection model. During the examination, video frames are continuously captured and analyzed. If no face is detected, the cheating score is incremented. If multiple faces are detected, a higher penalty is applied. Otherwise, the system evaluates the position of the face to determine whether the student is looking away from the screen, incrementing the score if necessary.

5. TESTING

Overview

The AI-based online exam proctoring system was evaluated through multiple levels of testing to ensure its accuracy, reliability, and efficiency in real-time monitoring scenarios. The testing process focused on several critical aspects, including face detection accuracy, cheating score computation, real-time video processing performance, functionality of the web dashboard, and proper storage and retrieval of monitoring data from the database. These evaluations ensured that the system can effectively detect suspicious behavior and present the results clearly on the proctor interface.

Test Objectives

The primary objective of the testing phase was to validate the overall performance and correctness of the system. This included ensuring accurate face detection through webcam input, identifying suspicious activities such as head movement away from the screen, and verifying that the cheating score is computed correctly based on predefined rules. Additionally, testing ensured that monitoring results are properly stored in the database and accurately displayed on the proctor dashboard for real-time observation.

Stages of Testing

The system was tested in multiple stages to ensure reliability at both individual component and system levels.

During unit testing, each module was tested independently to confirm its proper functionality. Components such as the face detection module implemented using OpenCV, the cheating score calculation logic, database operations, and dashboard rendering were evaluated separately. For instance, when a student moved their face away from the center of the screen, the system was expected to update the status to indicate suspicious behavior and increment the cheating score accordingly.

Integration testing was then conducted to verify the interaction between different modules. This

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

included testing the integration of face detection with cheating detection logic, connecting the monitoring module with database storage, and ensuring smooth communication between the Flask backend and the frontend dashboard. This stage ensured seamless data flow throughout the system.

Functional testing was performed to validate that all system features operate according to the specified requirements. Each function was tested under different conditions to confirm correct behavior and expected outputs.

Finally, system testing was carried out to evaluate the complete application as a whole. This involved monitoring student activity via webcam, detecting head movements, assessing system performance during continuous video processing, and verifying the accurate display of results on the dashboard.

Types of Testing

Both white box and black box testing methodologies were applied to ensure system robustness.

White box testing focused on verifying the internal logic and implementation details of the system. This included validating the correctness of the cheating detection algorithm, face position thresholds, and scoring mechanisms to ensure accurate identification of suspicious behavior.

In contrast, black box testing evaluated the system based solely on input and output behavior without considering internal code structure. Various real-world scenarios were tested, including cases where no face was detected, multiple faces appeared in the frame, and the student looked away from the screen. In all cases, the system successfully identified these conditions and updated the cheating score accordingly.

Test Cases

A comprehensive set of test cases was designed to validate different operational scenarios of the system. These cases ensured that the application responds correctly to various inputs and accurately reflects the corresponding outputs in the proctor dashboard.

SCREENSHOTS

Figure-3

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

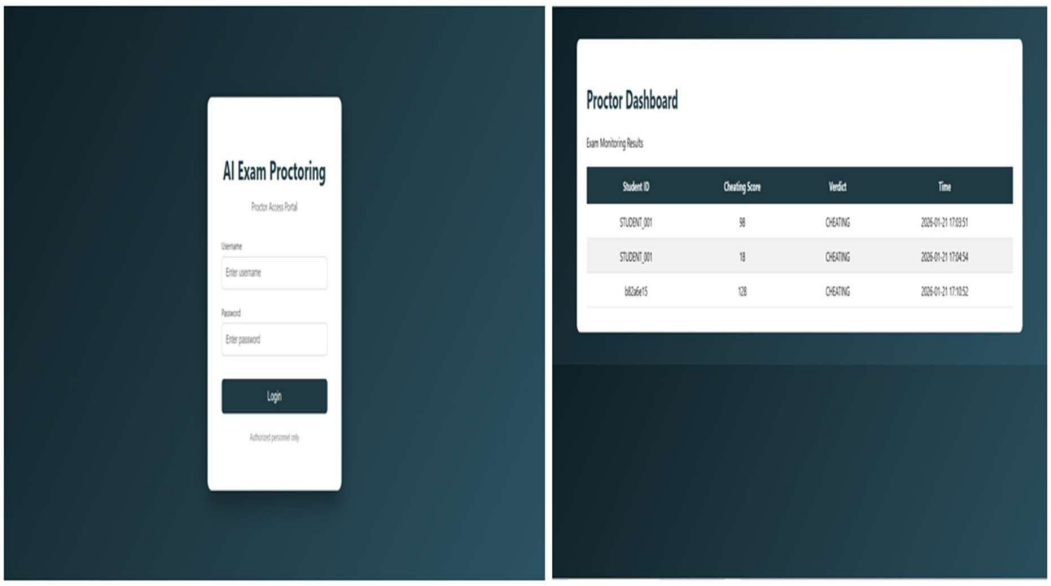

Figure-4

Conclusion And Future Scope

The AI-Based Intelligent Online Exam Proctoring System provides an effective solution to maintain academic integrity in online examinations. By automating the detection of suspicious behavior and storing exam results for proctor review, the system reduces human effort and improves reliability.

The project demonstrates the practical application of artificial intelligence in education and can be further enhanced with advanced features such as eye tracking and object detection.

Future Scope:

The future scope of the AI-based proctoring system includes enhancing monitoring accuracy by integrating advanced deep learning models for eye movement tracking, head pose estimation, and emotion detection. Audio monitoring can be added to detect suspicious conversations or background voices during exams. Object detection techniques can be implemented to identify unauthorized materials such as mobile phones or books. The system can also be improved by incorporating browser activity and screen monitoring to prevent tab switching or external searches. Deploying the platform on cloud infrastructure would allow better scalability and remote access. Real-time alerts can be introduced to notify proctors instantly when suspicious behavior is detected. Additionally, facial recognition can be integrated for continuous identity verification, and advanced analytics dashboards can be developed to provide detailed behavioral and performance reports for institutions.

REFERENCES

[1] Paul Viola and Michael Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[2] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly Media, 2008.

[3] OpenCV, “Open Source Computer Vision Library.” [Online]. Available: https://opencv.org