Research Article / Review Article

FUPRE JOURNAL 9(4):67-85(2025)
DOI: /10.60787/FJSIR.v9i4.67-85

FUPRE Journal

of

Scientific and Industrial Research

ISSN: 2579-1184(Print) ISSN: 2578-1129 (Online)

http://fupre.edu.ng/journal

Development of an Automated Eye Gaze Tracking System for Detecting Malpractice in Online Assessment

***IYAMAH, B. E.*[1,*], *OJIE, D. V.* [2], *OKPAKO, E. A.* [3], *OKONMAH, R. O.*[4], *CHIEMEKE, S. C.* [5]**

[1,2,4]*Department of Software Engineering, University of Delta, Agbor, Delta State.*

[3]*Department of Cyber Security, University of Delta, Agbor Delta State*

[5]*Department of Computer Science, University of Benin, Benin City. Edo State.*

ARTICLE INFO

Received: 30/08/2025
Accepted: 05/11/2025

Keywords

Academic Integrity, Computer vision, Eye Gaze Tracking, FaceMesh, Mediapipe, Online proctoring, OpenCV

ABSTRACT

This study presents the development of an automated eye gaze monitoring system designed to detect possible malpractice during online examinations. It addresses the increasing reliance on digital learning and the challenge of maintaining academic integrity in remote environments where traditional supervision is limited. The system uses computer vision techniques with Python, incorporating libraries like OpenCV and MediaPipe to track eye gaze direction in real-time. A rule-based method flags suspicious behaviors such as prolonged off-screen glances or rapid eye movements that may suggest cheating. The system includes a user-friendly web interface developed with HTML, CSS, and JavaScript, backed by a FastAPI server for efficient processing. It was tested on 10 students across academic levels (100 to 400), achieving an average accuracy of over 85% in gaze and malpractice detection, as detailed in Chapter 4. With low computational demand and a privacy-focused approach that avoids facial recognition or audio monitoring, the system offers a scalable solution for institutions and supports future enhancements such as machine learning integration or LMS compatibility.

1. INTRODUCTION

The rapid rise of online learning, accelerated by the COVID-19 pandemic, has reshaped the educational landscape globally. Institutions shifted from in-person to remote instruction to comply with public health restrictions, leading to a surge in the use of digital platforms and learning management systems (Indi *et al.*, 2021; Sander, 2022). Tools such as the Delta State Learning Management System now enable seamless delivery of virtual lectures, assignments, and evaluations. However, while these systems offer convenience, they also introduce significant challenges, particularly in ensuring academic integrity during online assessments where traditional forms of physical supervision are absent.

One of the most pressing concerns in remote examinations is the increased potential for academic dishonesty, including cheating and collusion (Newton *and* Draper, 2023). Current online proctoring solutions such as browser lockdowns, webcam-based monitoring, and video reviews have proven inadequate due

*Corresponding author, e-mail: boyce.iyamah@unidel.edu.ng, deborah.ojie@unidel.edu.ng, okpako.ejaita@unidel.edu.ng, C, s.chiemeke@uniben.edu
DIO:/10.60787/FJSIR.v9i4.67-85
©Scientific Information, Documentation and Publishing Office at FUPRE Journal

to high costs, inefficiencies, privacy concerns, and their reliance on human invigilators or specialized hardware (Hussein *et al.*, 2020; Nafiul *et al.*, 2023). Moreover, these tools often fail to detect subtle cues like eye movements, which may indicate malpractice even when a candidate's posture appears normal on camera. The manual review of video feeds is also time-consuming and susceptible to human bias.

This study aims to develop a cost-effective and privacy-conscious eye gaze tracking system for online examination proctoring. The system is designed to monitor a student's eye movement in real time to detect suspicious behavior such as prolonged off-screen gazing or rapid eye shifts. By utilizing computer vision techniques (e.g., OpenCV, MediaPipe), the system identifies behavioral patterns indicative of possible malpractice without capturing full video feeds or requiring high-end hardware. Specifically, it seeks to: (i) track students' eye movement during assessments, (ii) identify gaze patterns associated with malpractice, and (iii) generate real-time alerts when such behaviors are detected.

The significance of this research lies in its ability to strengthen academic integrity in remote assessments while respecting students' privacy and reducing reliance on human supervision. By automating the detection process and focusing exclusively on eye gaze, the system avoids invasive practices such as facial recognition or audio monitoring. Furthermore, it offers a scalable, low-resource solution that can be integrated into existing learning management systems, making it especially suitable for institutions with limited technical infrastructure, such as the University of Agbor. This study contributes toward building fairer, more reliable online assessment environments for the future.

2. REVIEW OF RELEVANT LITERATURE

The expanding landscape of digital education has intensified the imperative to uphold academic integrity in online assessments. Eye gaze tracking has emerged as a leading technique for detecting dishonest behavior during remote testing. Basic implementations utilizing tools such as OpenCV, MediaPipe, and YOLO continue to serve as foundational building blocks, while deep learning models like L2CS-Net, iTracker, and multiple-instance learning have significantly improved detection accuracy (Shih *et al.*, 2024; Singh *et al.*, 2024; Yaqub *et al.*, 2022).

These systems monitor behavioral cues such as off-screen glances, gaze regression anomalies, and other irregular visual patterns potentially indicative of cheating. While some rely solely on gaze inputs, multimodal systems augment this by integrating head posture, lip movement, or audio cues for enhanced detection robustness.

The field is rapidly transitioning from retrospective post-exam video review toward proactive real-time monitoring and alert systems. Although earlier efforts such as Dilini *et al.* (2021) and Shih *et al.* (2024) focused on asynchronous video analysis, more recent approaches (Liu *et al.*, 2024; Shahid *et al.*, 2023) demonstrate that real-time systems offer timely detection with high accuracy.

Building on this trend, Lamba and Sharma (2024) introduced a real-time multimodal cheating detection framework that combines gaze tracking, face detection, head pose estimation, and audio analysis in a CNN–BiLSTM architecture to comprehensively detect suspicious behavior. Complementarily, Shih *et al.* (2024) offered an AI-assisted gaze detection system that improves asynchronous proctoring efficiency by enabling rapid navigation to suspicious video segments via a gaze scatter-plot interface. Additionally, Bühler (2024)

presented a multimodal machine learning framework capable of detecting attention lapses such as mind-wandering using webcam-based monitoring, contributing a scalable perspective toward behavioral detection in learning environments.

Despite these technological advances, persistent challenges include elevated false positive rates due to natural behaviors, calibration sensitivity, and hardware constraints. Multimodal systems may inadvertently misinterpret benign movements as violations.

Ethical concerns continue to weigh heavily on such technologies. Bukvic and Zhao (2023) emphasize the need for strict data protection protocols, as many existing systems lack privacy safeguards. Against this backdrop, there remains a clear and urgent demand for solutions that are lightweight, scalable, and privacy-sensitive, offering real-time monitoring without compromising integrity or user trust.

3. METHODOLOGY

This section presents the systematic approach to developing the eye gaze tracking system. It includes the methods and procedures followed to achieve project objectives. The design architecture follows a description of the analysis of the existing system, along with materials and technologies used in developing the project. The chapter also seeks to contain data collection techniques.

3.1 Analysis of Existing Systems

Prior research has explored various methods for detecting cheating in online exams, including multi-modal systems and machine learning models. However, many face challenges such as high computational demands, privacy concerns, overreliance on head posture, and limited scalability.

i. Shih *et al.* (2024) combined gaze, lip, and audio analysis for detection, but their approach is complex, raises privacy concerns, and risks false positives. The proposed system improves on this by focusing solely on gaze, reducing overhead and enhancing privacy.
ii. Singh *et al.* (2024) used gaze and head posture tracking with YOLO, but their reliance on head movement is misleading and resource-heavy. In contrast, the proposed system uses lightweight FaceMesh, improving accuracy and efficiency.
iii. Yaqub *et al.* (2022) applied multiple instances learning with gaze and blink data, requiring large datasets and expert tuning. The proposed rule-based approach avoids these dependencies, offering broader applicability.
iv. Dilini *et al.* (2021) relied on post-exam video reviews, which are labor-intensive and reactive. The proposed solution offers real-time detection, enabling timely intervention and reducing resource demands.

Overall, the proposed FastAPI-based gaze tracking system addresses common drawbacks in prior work by prioritizing real-time efficiency, user privacy, and ease of integration into existing educational platforms.

3.2 System Design (Materials and Method)

3.2.1 System Architecture

This eye gaze tracking system is designed as a client-server architecture as illustrated in Fig 3.1, where the front-end (web-based exam interface) communicates with a FastAPI-powered back-end. The application is modular and comprises the following key components:

i. Eye Gaze Detection Module (Tracking module): Captures video frames via the user's webcam,

processes the frames using MediaPipe FaceMesh, and extracts eye landmarks for gaze estimation.

ii. Analysis Engine: Analyzes gaze patterns to determine whether the user is looking at the screen, left, right, or down.

iii. Flagging Logic: Applies rule-based logic to count and log suspicious gaze behaviors.

iv. Web Interface: Simple user interface built with HTML, CSS, and JavaScript to allow users to start the exam and run the system passively in the background.

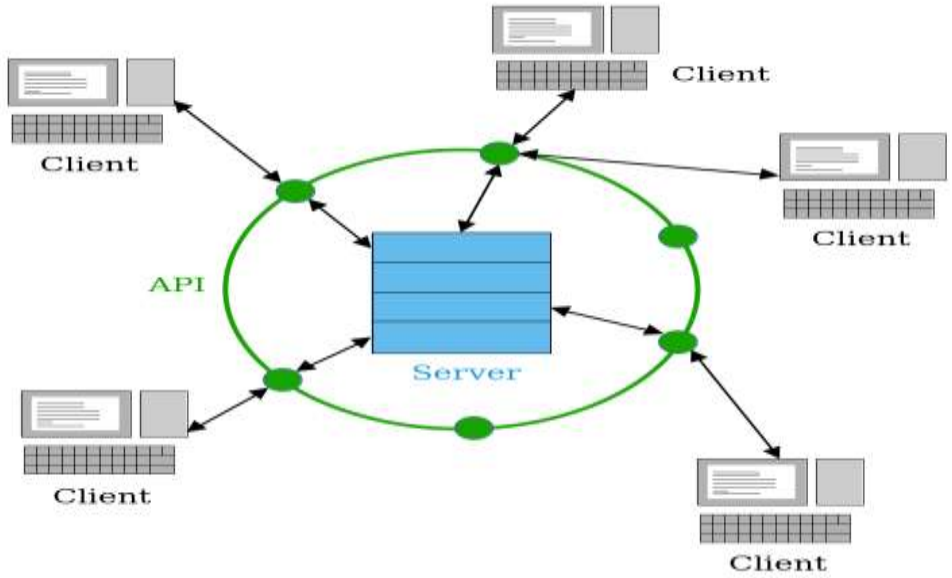


Fig 3.1 Client-Server Architecture

3.2.2 System Workflow

Fig 3.2 shows the diagrammatic representation of the system work flow as detailed below

1. User Access and Initialization:
   i. The user accesses the system via a local URL.
   ii. A homepage with a "Start Sample Test" button is displayed.
   iii. Upon clicking the button, the system initializes with a session ID and loads test questions.
2. Webcam Access:
   i. The system requests access to the webcam.
   ii. If granted, the test begins.
3. Real-Time Monitoring:
   i. The webcam captures video frames in real-time.
   ii. These frames are sent to a FastAPI backend.
   iii. The backend preprocesses the frames and sends them to a gaze tracking module.
   iv. Gaze direction is estimated using FaceMesh.
   v. Gaze direction is analyzed for anomalies (i.e., suspicious behavior).
4. Anomaly Logging:
   i. If suspicious gaze behavior is detected, the system logs the timestamp, session ID, and type of anomaly.
   ii. System captures image of examinee immediately the suspicious behavior is flagged.
5. Test Completion:

i. The system checks if the test is submitted or if time has expired.

ii. If so, it ends the session, saves the logs, displays a results page, and terminates the webcam feed.

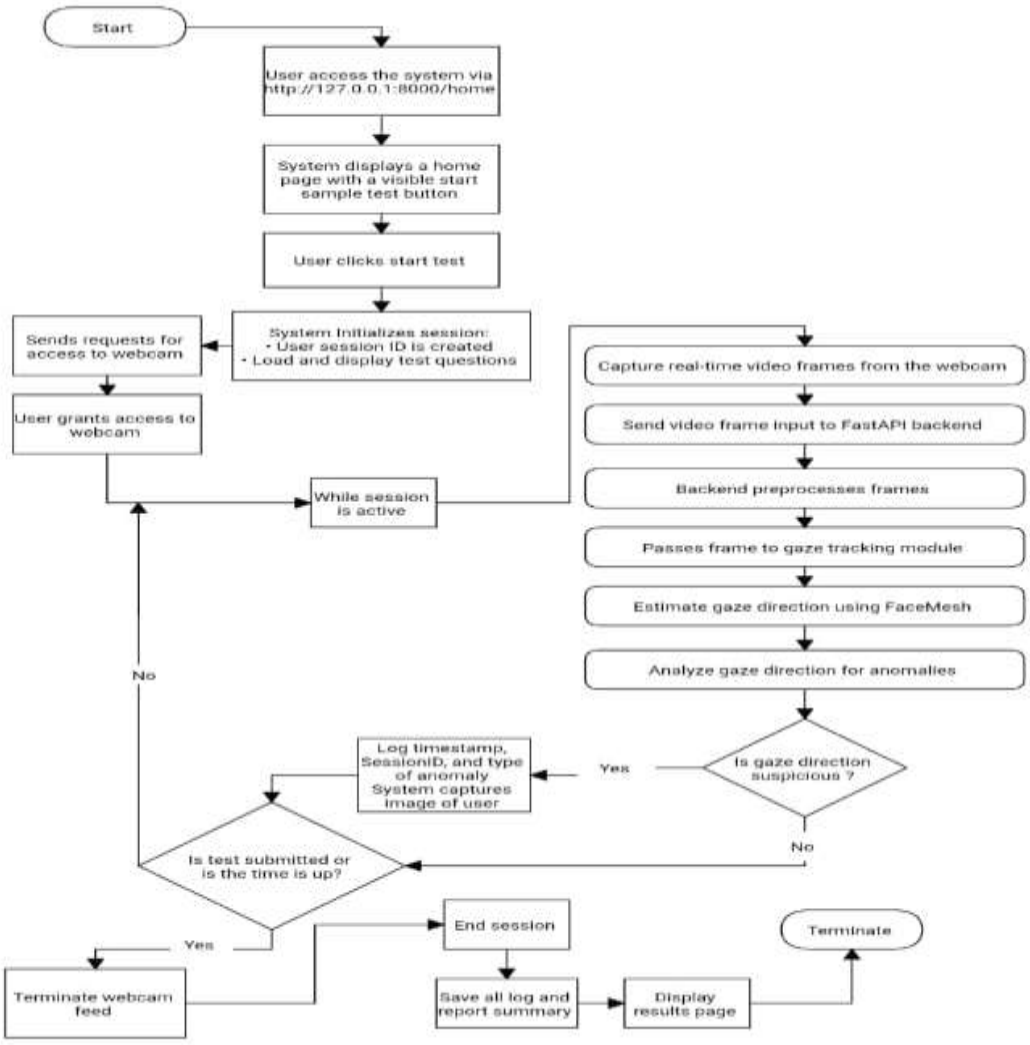


Fig 3.2 Work flow diagram of Eye gaze tracking system.

3.2.3 System Algorithms

a. Eye Gaze Tracking Algorithm

This algorithm, as illustrated Figure 3.3 captures real-time video input, extracts facial landmarks, and estimates the user's gaze direction using the FaceMesh model.

Input: Live webcam video stream

Output: Gaze direction (left, right, center,)

1. Start webcam

2. For each frame captured:

a. Convert frame to RGB

b. Detect facial landmarks using FaceMesh

c. Identify eye landmarks (e.g., left and right iris)

d. Compute eye aspect ratio or vector direction

e. Determine gaze direction based on landmark displacement

3. Return estimated gaze direction

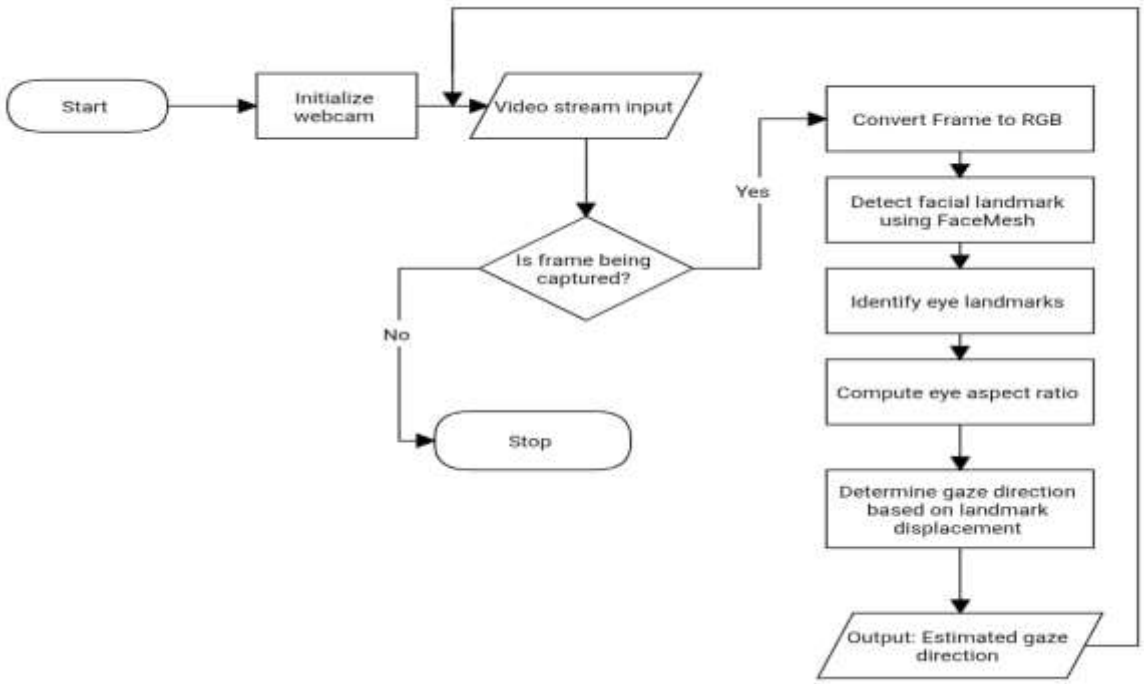


Figure 3.3 Flow chart of eye gaze tracking algorithm

b. Gaze-Based Malpractice Detection

This algorithm, as illustrated in Fig 3.4 monitors the estimated gaze directions in real-time and flags abnormal behaviors based on predefined thresholds or patterns.

Input: Stream of gaze directions

Output: Log of suspicious events

1. Initialize session ID and log file

2. While the test is ongoing:

a. Receive gaze direction from tracking module

b. Track duration and frequency of off-screen gazes

c. If a suspicious pattern is detected:

I. Log timestamp, session ID, and type of anomaly

ii. Flag user for review

3. upon test submission or timeout,

a. End session

b. Save log and summary report

3.2.4 Component Diagram

This component diagram as shown in Figure 3.5, explains the interaction between the main components of this system: the browser (frontend) and the app server (backend).

i. The browser handles the user interface, displaying test pages, running JavaScript, and accessing the webcam.
ii. The app server, running with Uvicorn, processes incoming data.

It handles eye-gaze tracking, detects anomalies, logs session data, and applies malpractice detection logic.

Communication occurs through:

i. HTTP for page loads and initial requests.
ii. WebSocket for real-time, asynchronous video data exchange.

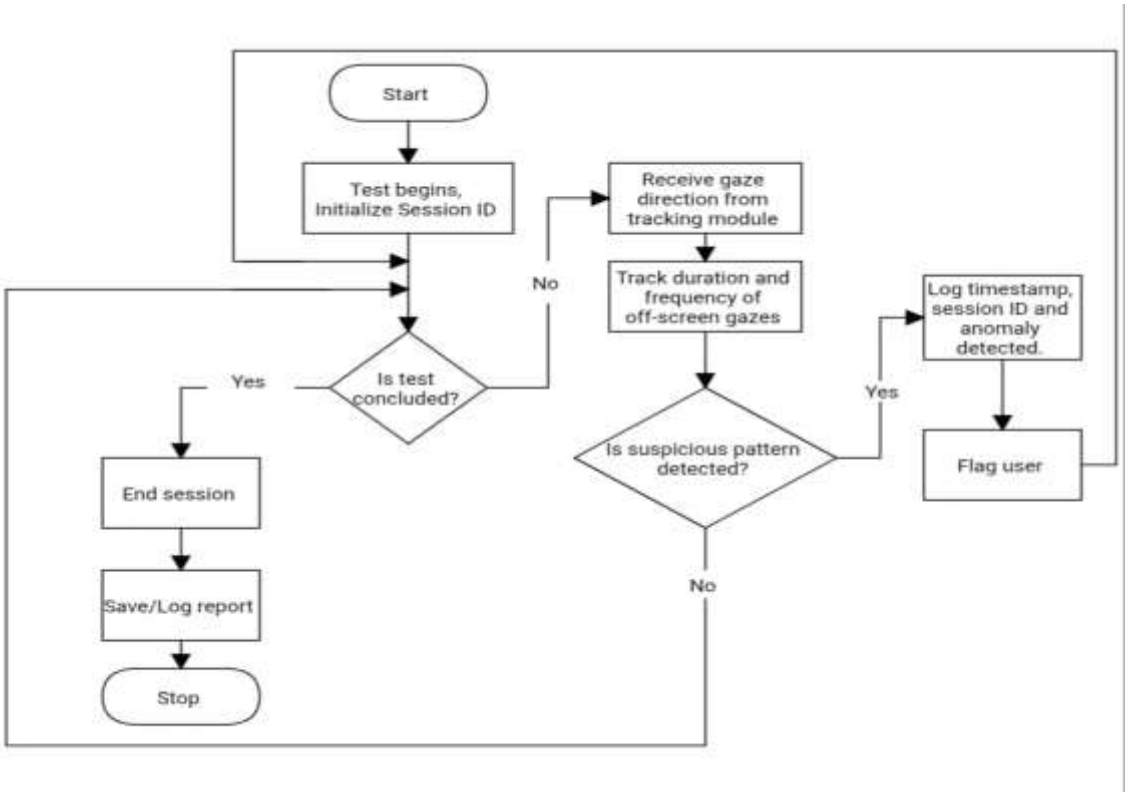


Figure 3.4 Flow chart of gaze anomaly detection algorithm

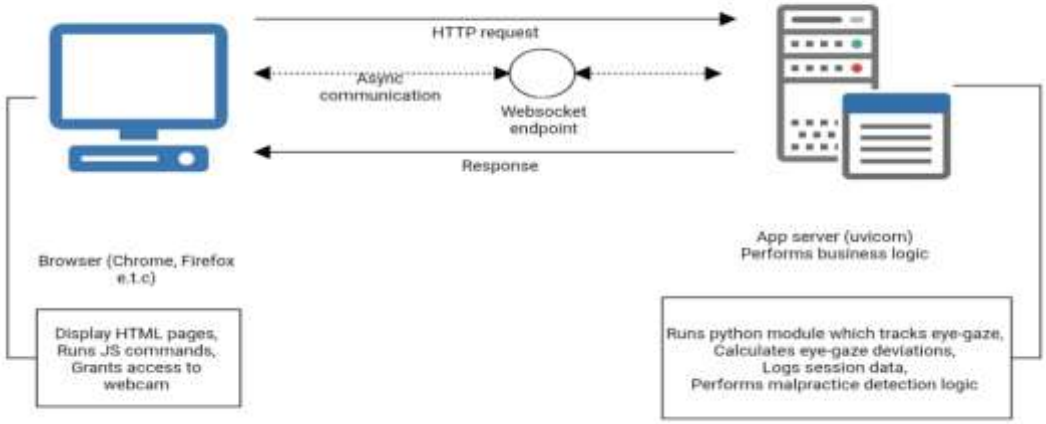


Figure 3.5 Hybrid view of Component diagram of this system

3.2.5 Sequence Diagram

The sequence diagram depicted in Figure 3.6, establishes how different parts of a web-based gaze tracking system interact during a test and data flow among components.

i. User starts and ends the test.
ii. Browser handles user input, connects to the server, and displays results.
iii. The webcam streams video through a WebSocket to the backend.
iv. FastAPI backend sends video to a tracker module.
v. The tracker analyzes eye gaze and sends results back.
vi. Logs are stored, and the user sees the results.

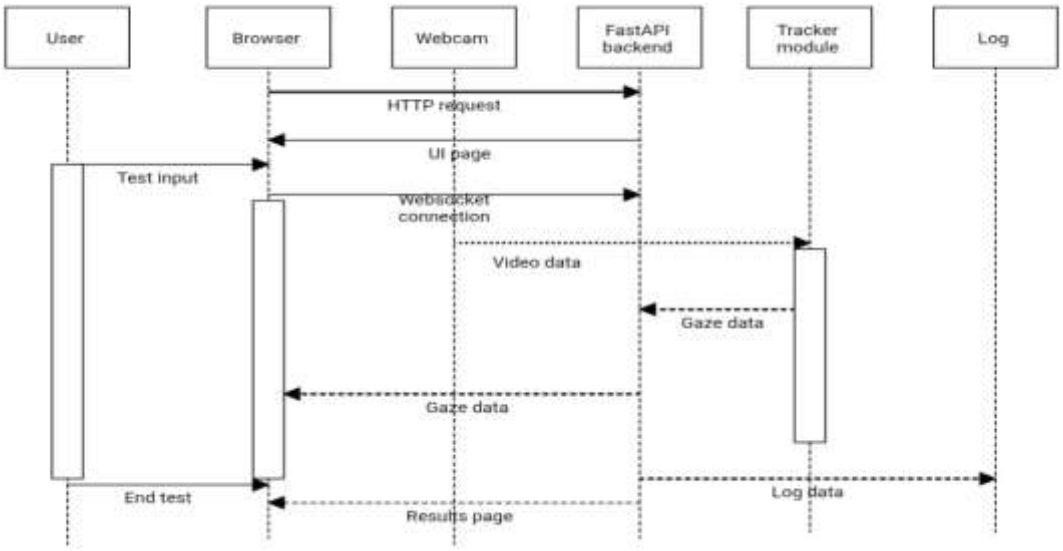


Figure 3.6 - Sequence diagram showing dataflow and object interaction

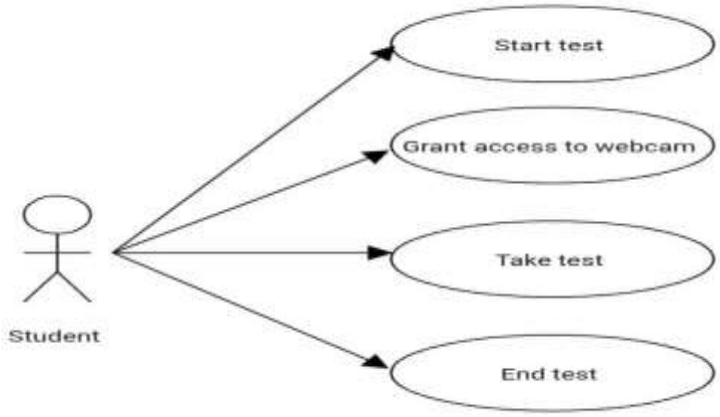


Figure 3.7 Student use case diagram

3.2.6 Use Case Diagram

Figure 3.7 illustrates the basic interactions a student has with the system. The stick figure labeled "Student" represents the user (Actor) of the system. The ovals represent different actions or "use cases" the student can perform. The arrows indicate that the "Student" initiates these actions:

i. Start test: The student can begin a test.
ii. Grant access to webcam: The student needs to allow the system to use their webcam.
iii. Take test: The student can proceed to answer the questions in the test.
iv. End test: The student can finish and submit the test.

3.3 Data Collection/ Dataset

3.3.1 Data Collection

A. System data collection

The data collected by the system for internal processes included:

Data Collection

The system collects data exclusively through the webcam, with student permission, and ensures no video is stored to protect privacy. It captures:

1. Video Frame Data: Serves as the primary input, used in real-time to extract relevant features like camera distance and iris position.
2. Eye Landmark Position: Extracted using FaceMesh to accurately identify eye and iris locations for tracking.
3. Gaze Direction: Determined by analyzing iris movement relative to the screen's center across video frames.

B. Performance Evaluation Data Collection:

The primary data collection tool employed in this study was a prototype eye gaze tracking system developed using FastAPI, OpenCV, and MediaPipe's FaceMesh technology. This tools monitor users' eye movement patterns, detect gaze direction, and assess attention span during a simulated online assessment session.

Tools for performance evaluation data collection:

1. Eye gaze tracking prototype

Tools for analyzing performance evaluation data:

1. Python: A high level programming language that can be used for data processing and analysis.

2. Microsoft excel: Used to view the data set and perform calculations on them

3. Mathplotlib: A python library that assist in generating charts from a data set

4. Pandas: A python library for performing data analysis

3.3.2 Data Set

For sample collection, a total of 10 students across different academic levels from the University of Delta, Agbor, participated in a 10-minute in-person test. Each participant's gaze movements and behavioral cues were monitored using the prototype system. Participants signed a consent form before the evaluation.

3.4 Performance Metrics

The following metrics were used to evaluate the system:

a) Gaze Detection Accuracy: This is used to measures how accurately the system identifies the student's gaze direction. This is the percentage match between the

system's predicted gaze direction and the actual screen region the user was focused on.

Formula used:

Gaze Detection Accuracy = (Total Gaze Predictions/Correct Gaze Predictions) ×100

b) Malpractice Detection Accuracy: Measures the system's ability to correctly flag cheating-like behaviors.

Formula used:

Malpractice Detection Accuracy = (Total Malpractice Events/Correct Malpractice Flags) ×100

4. IMPLEMENTATION

4.1 System Requirement

The eye gaze tracking system was designed to be lightweight and effective for real-time monitoring of students during online assessments. Hardware and software requirements were selected to ensure compatibility with typical institutional setups, balancing performance with accessibility.

4.1.1 Hardware Requirements.

i. Processor: Intel Core i5 or higher
ii. RAM: Minimum 4GB
iii. Webcam: Integrated or external HD webcam
iv. Storage drive (1 Gb and above)
v. Operating System: Windows 10+

4.1.2 Software Requirements.

i. Python 3.9+
ii. FastAPI Web Framework.
iii. OpenCV Library.
iv. Mediapipe Library.
v. Uvicorn Web Server.
vi. HTML/CSS/JavaScript.
vii. SQLite.
viii. Modern Web Browser (Google Chrome, Mozilla Firefox, or Microsoft Edge or any web browser to interact with the UI).

4.3 Results and Discussion

4.3.1 - Frontend

i. Homepage UI - This is the first page the user encounters in the system as shown in Figure 4.1. This page was designed to simulate the start of an online exam. It's simplicity in design makes it user friendly and less complex to navigate.



Fig 4.1 Home page UI

ii. Test Simulation interface

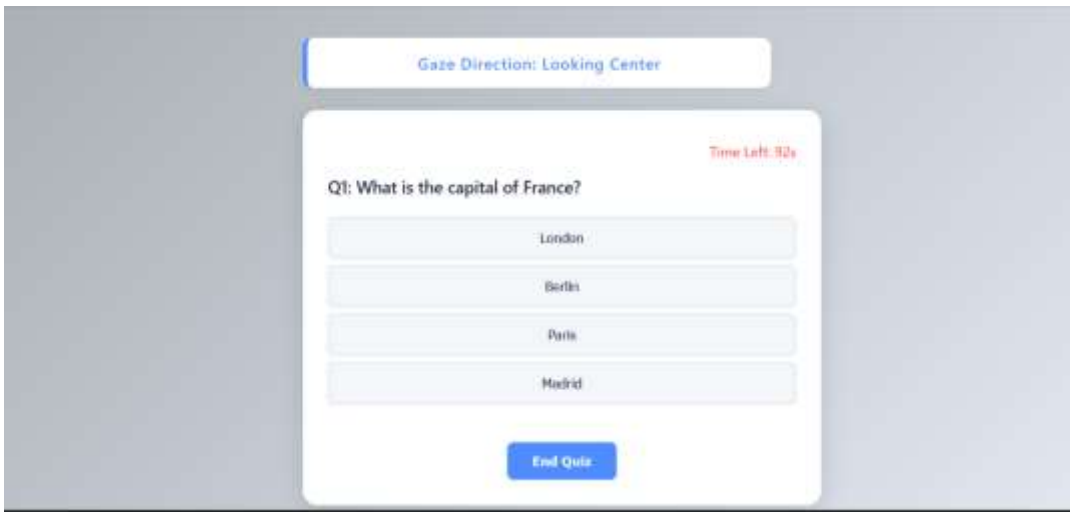


Fig 4.2 Test simulation UI

This page was designed to simulate an ongoing online assessment. This page is displayed after the user click the start quiz button in the HomeUI page. As seen in fig 4.2 this action initializes the timer and the sample questions are displayed. Camera access is granted and the system begins to track the student's eye movement while the test is ongoing.

The interface also shows the gaze direction as detected by the system in real-time. This was integrated in the UI for testing and observation purpose. This feature will not be present in real-life applications. As seen in Figure 4.2, Figure 4.3 *and* Figure 4.4, the direction the user faces is flagged on the test screen.

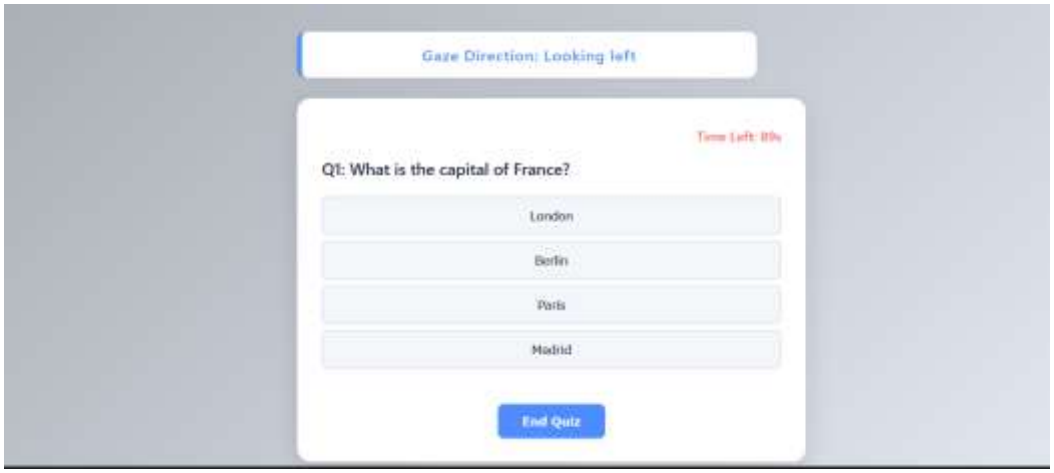


Figure 4.3Gaze Direction block changes to "Looking Left"

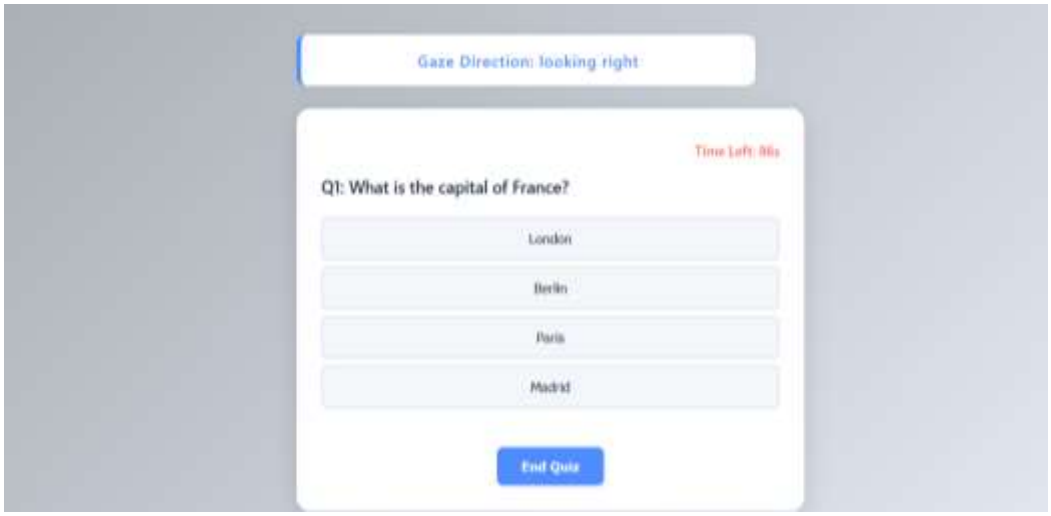


Figure 4.4 Gaze Direction block changes to "Looking Right"

When the system can no longer detect the users face, the system updates the gaze direction to "No face detected" as seen in figure 4.5.

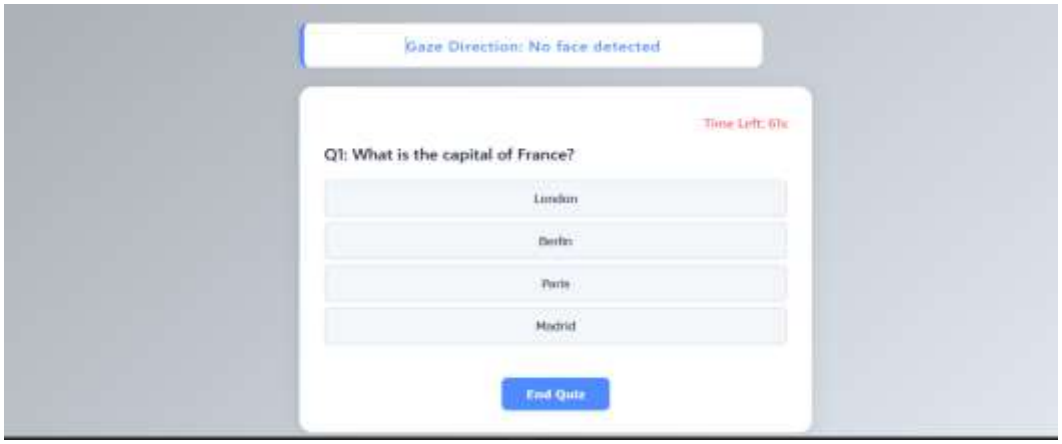


Figure 4.5 Gaze direction changes to "No face detected"

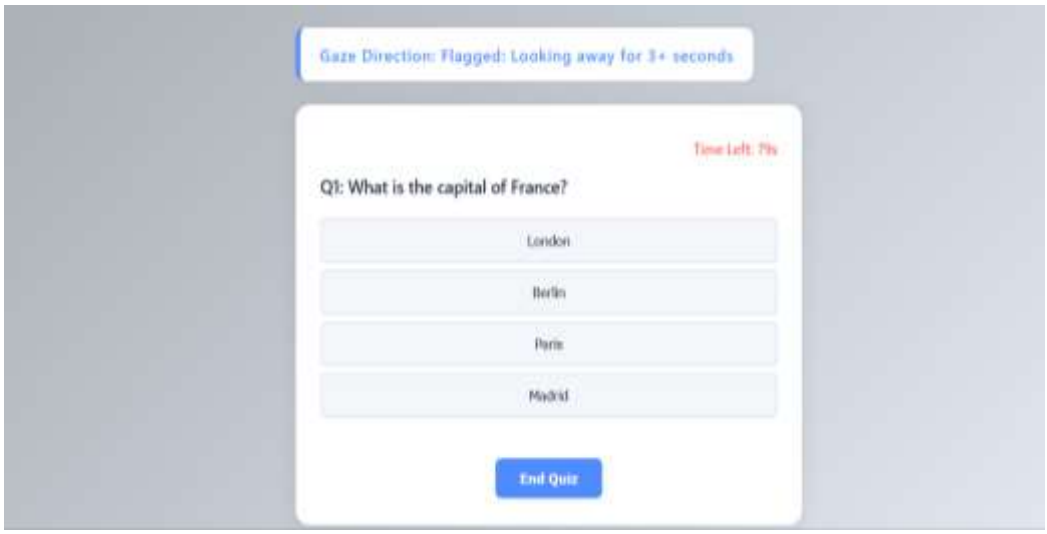


Figure 4.6 Gaze Direction changes to "Flagged: Looking away for 3+ seconds"

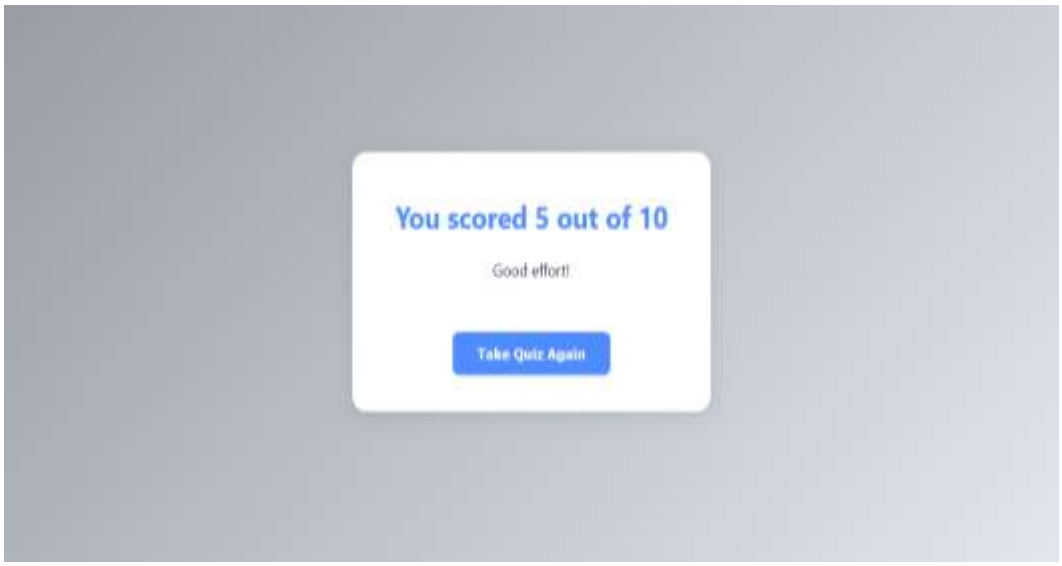


Figure 4.7 Result UI Displayed after test completion.

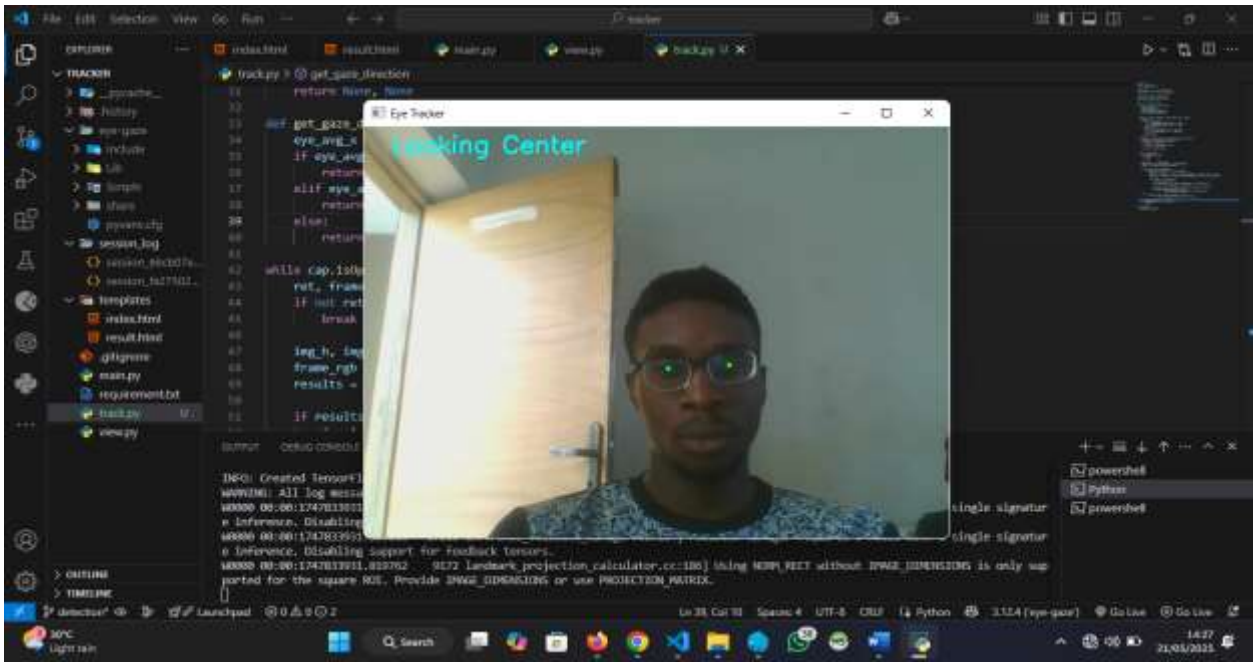


Fig 4.8 Eye tracking system User gaze when user is Looking at the screen

When user looks away from the system or no face was detected for more than 3 seconds. The system flags the user by displaying “Flagged: Looking away for 3+ seconds” as seen in figure 4.6

4.3.2 Back End Perspective

This section shows how the live video feed collected by the system, and the eye gaze tracking mechanism operating in real-time. Figure 4.8 display the live video feed used for processing by the system. The green dots on the eye shows the part of the eye tracked by the system.

Fig 4.9 shows how the system identifies gaze direction change and responds to it. In the above image, the system responds looking right based on the direction of the gaze deviation.

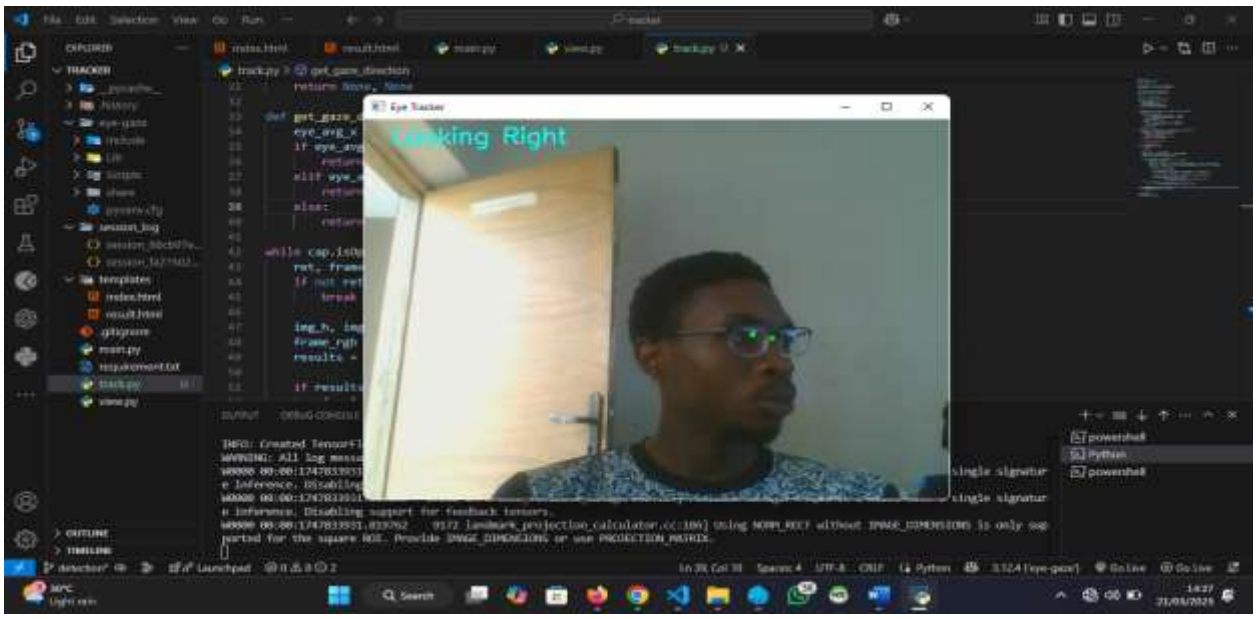


Fig 4.9 Eye tracking system User gaze when user is looking right

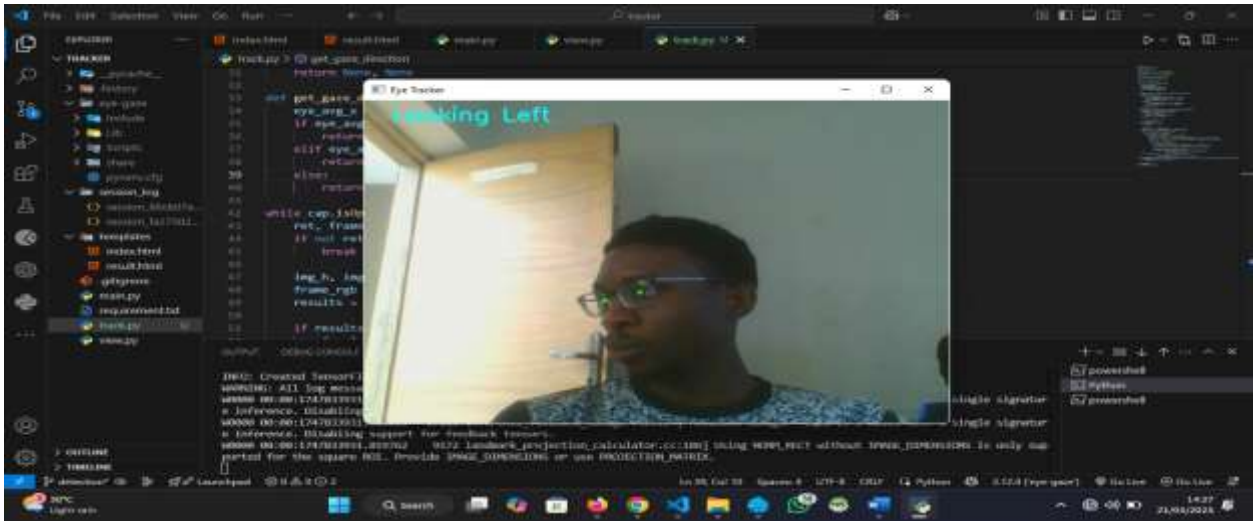


Fig 4.10 Eye tracking system User gaze when user is Looking left

Fig 4.10 shows how the system identifies gaze direction change and responds to it. In the above image, the system responds Looking left based on the direction of the gaze deviation.

Fig 4.11 shows the how the system responds when the users face is no longer detected. This mechanism is well suited for situations where the user leaves he or her sit or blocks he's or her camera to avoid being seen while engaging in malpractice.

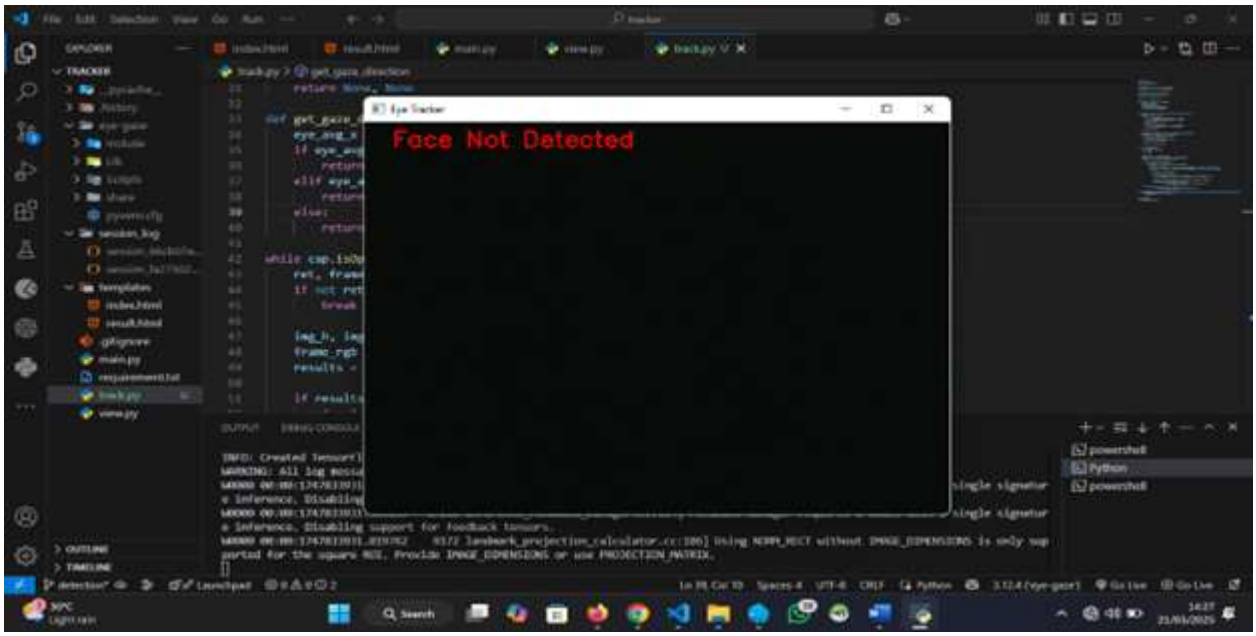


Fig 4.11 Eye tracking system response when No face is detected

ID	Level	Correct Gaze	Total Gaze	Correct Malpractice	Total Malpractice	Gaze Accuracy (%)	Malpractice Accuracy(%)	Overall Accuracy(%)
P1	100	8	10	4	5	80	80	80
P2	100	8	10	4	5	80	80	80
P3	200	9	10	5	5	90	100	95
P4	200	9	10	5	5	90	100	95
P5	300	10	10	5	5	100	100	100
P6	300	9	10	5	5	90	100	95
P7	400	8	10	4	5	80	80	80
P8	400	8	10	4	5	80	80	80
P9	300	9	10	4	5	90	80	85
P10	400	9	10	5	5	90	100	95
Formulas and Explanations:								
Gaze Accuracy = (Correct Gaze / Total Gaze) * 100								
Malpractice Accuracy = (Correct Malpractice / Total Malpractice) * 100								
Overall Accuracy = (Gaze Accuracy + Malpractice Accuracy) / 2								

Figure 4.12 shows the bar chart representation of the gaze detection accuracy result

4.2 Results

The Table 4.1 shows the raw data set collected from each test figures and were then used to calculate the performance of the system, using the performance metrics as stated earlier. The table below shows each the result of each of the calculations done.

Gaze Detection Accuracy =
(Total Gaze Predictions/Correct Gaze Pred

ictions)×100, this formular above was used to calculate the Gaze Detection Accuracy, while

Malpractice Detection Accuracy = (Total Malpractice Events/Correct Malpractice Flags) × 100, was used to calculate the malpractice Accuracy.

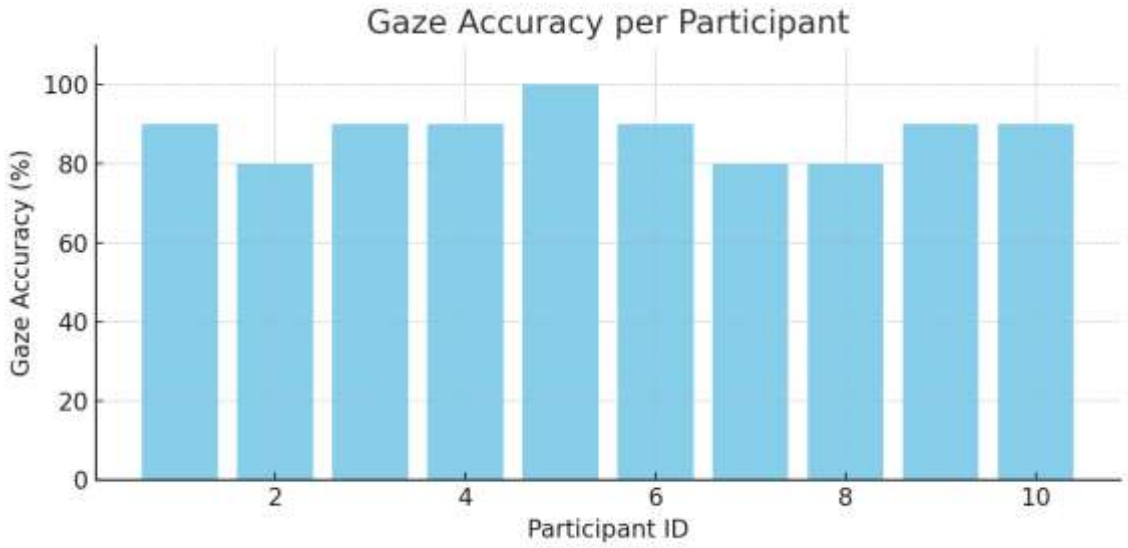


Figure 4.12 Gaze Detection Accuracy

Figure 4.13 shows the bar chart representation of the Malpractice detection accuracy results for better comprehension.

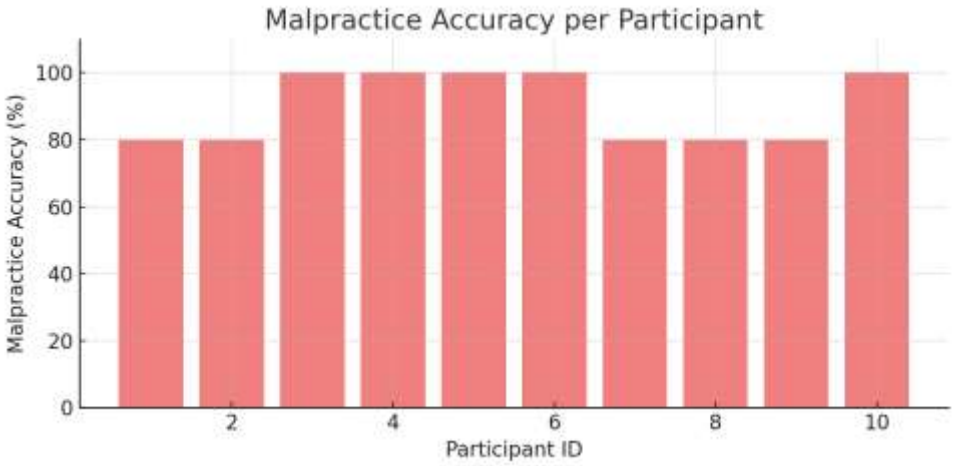


Fig 4.13 Malpractice Detection Accuracy

4.3.4 Discussion

As shown in Tables 4.1, the eye gaze tracking system exhibited strong performance in both gaze accuracy and malpractice detection, making it suitable for online exam monitoring. Most participants achieved 80–100% gaze accuracy, enabled by efficient tools like OpenCV and MediaPipe's FaceMesh. Minor accuracy drops were linked to environmental issues such as poor lighting or webcam angles, though all results remained within acceptable limits.

The system also effectively detected deliberate off-screen gazes, with around 50% of participants reaching 100% detection accuracy. Overall system accuracy—combining gaze tracking and malpractice detection—ranged from 80% to 100%, confirming its reliability for institutional use and potential for scalable deployment in virtual learning environments.

5. CONCLUSION

This study developed an automated eye-gaze tracking system to detect potential malpractice in online exams. Built with FastAPI, OpenCV, and MediaPipe, the system enables lightweight, platform-independent real-time gaze tracking and anomaly detection. The working prototype shows that rule-based gaze detection can effectively monitor student behavior during assessments. The project lays a groundwork for future enhancements and testing, providing a practical framework for real-world academic settings.

5.1 Recommendation

To enhance the system's effectiveness and usability, the following recommendations are proposed:

1. Real-World Testing: Validate performance and refine detection logic by testing with a controlled group of students.
2. User Interface Enhancement: Develop an intuitive front-end for ease of use by examiners and students.
3. Platform Integration: Enable compatibility with online exam platforms and LMSs to support broader adoption.
4. Expanded Detection: Incorporate behaviors like head movement and tab switching to strengthen cheating detection.
5. Logging *and* Reporting: Implement gaze activity logging and automated report generation for examiner support to make the system more useful to examiners.

5.2 Contribution to Knowledge

This study presents a practical, gaze-based framework for detecting malpractice in online exams, emphasizing simplicity and accessibility over complex machine learning or high-resource infrastructure. By utilizing lightweight computer vision tools, the system offers a scalable solution for institutions with limited resources. While large-scale user testing remains pending, the design provides a reproducible foundation for future research and development, reinforcing the role of affordable technologies in supporting the integrity of remote education.

References

Bühler, C. (2024). Multimodal machine learning for attention assessment in online and in-class environments. *arXiv preprint* arXiv:2407.05803.

Bukvic, D., *and* Zhao, X. (2023). Ethical and privacy challenges in remote

proctoring systems: A data protection perspective. *Journal of Academic Integrity and Technology*, 5(2), 45–62.

Daffin, L. W., *and* Jones, A. A. (2018). Comparing student performance on proctored and non-proctored exams in online psychology courses. Online Learning, 22(1), 131–145.

Dendir, S., *and* Maxwell, R. S. (2020). Cheating in online courses: Evidence from online proctoring. Computers in Human Behavior Reports, 2, 100033.

Dilini, N., Senaratne, A., Yasarathna, T., Warnajith, N., *and* Seneviratne, L. (2021). Cheating detection in browser-based online exams through eye gaze tracking. 2021 6th International Conference on Information Technology Research (ICITR).

Dilini, W., Perera, H., *and* Fernando, M. (2021). Automated cheating detection in online examinations using asynchronous video analysis. *International Journal of Emerging Technologies in Learning*, 16(15), 45–60.

Fask, A., Englander, F., *and* Wang, Z. (2014). Do online exams facilitate cheating? An experiment designed to separate possible cheating from the effect of the online test taking environment. Journal of Academic Ethics, 12(2), 101–112.

Hylton, K., Levy, Y., *and* Dringus, L. P. (2016). Utilizing webcam-based proctoring to deter misconduct in online exams. Computers *and* Education, 92–93, 53–63.

Klette, R. (2014). Concise computer vision: An introduction into theory and algorithms. Springer London.

Lamba, P., *and* Sharma, S. (2024). Real-time AI-powered multimodal e-cheating detection system for online exams. *Journal of Engineering Science*, 15(1), 55–70.

Lim, S., Walber, T., Schaefer, C., *and* Riehl, L. (2022). Webcam Eye Tracking: Study Conduction and Acceptance of Remote Tests with Gaze Analysis [Preprint]. arXiv. https://arxiv.org/abs/2207.14380.

Liu, J., Chen, Y., *and* Huang, W. (2024). Real-time gaze-based cheating detection in online assessments using deep learning. *Computers and Education: Artificial Intelligence*, 6, 100225.

Liu, Y., Ren, J., Xu, J., Bai, X., Kaur, R., *and* Xia, F. (2024). Multiple Instance Learning for Cheating Detection and Localization in Online Examinations [Preprint]. arXiv. https://arxiv.org/abs/2402.06107.

Mediapipe solutions guide | google ai edge. (n.d.). Google AI for Developers. Retrieved April 17, 2025, from.

Minaee, S., Luo, P., Lin, Z., *and* Bowyer, K. (2021). Going deeper into face detection: A survey.

Newton, P., *and* Draper, M. (2023, November 19). Integrity of undergraduate degrees undermined by remote exams. The Times.

OEB Insights. (2021, November 5). The development of online proctoring. https://oeb.global/oeb-insights/the-development-of-online-proctoring/ .

Real time Gaze Tracking in Remote Proctoring: A Study of Appearance-based Gaze Estimation. (n.d.).

Shahid, M., Khan, R., *and* Iqbal, M. (2023). Enhancing academic integrity through real-time gaze tracking and multimodal analysis. *Educational Technology Research and Development*, 71(4), 1823–1845.

Shih, H., Lin, T., *and* Wang, C. (2024). AI-assisted gaze detection for efficient asynchronous online exam proctoring. *arXiv preprint* arXiv:2409.16923.

Shih, Y.-S., Zhao, Z., Niu, C., Iberg, B., Sharpnack, J., *and* Baig, M. B. (2024). AI-assisted Gaze Detection for Proctoring Online Exams [Preprint]. arXiv. https://arxiv.org/abs/2409.16923.

Singh, R., Kaur, G., *and* Verma, S. (2024). Eye-tracking based cheating detection using L2CS-Net and deep learning models. *IEEE Access*, 12, 102345–102356.

Singh, T., Nair, R. R., Babu, T., *and* Duraisamy, P. (2024). Enhancing academic integrity in online assessments: Introducing an effective online exam proctoring model using yolo. Procedia Computer Science, 235, 1399–1408.

Yaqub, A., Khalid, H., *and* Ahmed, S. (2022). Intelligent proctoring systems: A gaze-based approach. *International Journal of Advanced Computer Science and Applications*, 13(4), 155–164.

Yaqub, W., Mohanty, M., *and* Suleiman, B. (2023). Proctoring online exam using eye tracking: Proceedings of the 20th International Conference on Security and Cryptography, 738–745.