Department of Automatic Control and Industrial Informatics
Faculty of Automatic Control and Computers
National University of Science and Technology POLITEHNICA Bucharest
Splaiul Independenţei 313, 060042, Bucharest, Romania
Room ED 412, Tel. 021/402.92.69
www.aii.pub.ro, email: secretariat.aii@upb.ro

Student Scientific Communications Session

2025 Edition

Anti-Plagiarism Monitoring System for Exams

Author: Valentin PLETEA-MARINESCU, Faculty of Automatic Control and Computers, 3rd year, Group 332AB

Email address: pletea.valentin2003@gmail.com

Scientific advisor: Assoc. Prof. Dr. Eng. Ștefan Alexandru MOCANU

Contents

1 Introduction

1.1 Context and relevance of the topic

Plagiarism represents one of the serious problems of the educational system, and since the emergence and development of artificial intelligence, things have intensified. The choice of this topic consists in the opportunity to offer equal chances to all participants in an exam, and this anti-plagiarism system, proposed in this project, represents an innovative solution, through the combination of modern image processing techniques and artificial intelligence[9], which aims to monitor the behavior of candidates and signal possible fraud attempts.

The use of mobile phones as an instrument of cheating in examination halls has increased considerably in recent years, creating an additional burden on supervisors in ensuring exam integrity[7]. Moreover, recent studies in the field of academic integrity show that plagiarism is ”an ethical deviation that affects the quality, legibility and credibility” of academic results, being essential the development of solutions to combat unacceptable practices[14].

According to the study conducted by Gabriela Pelican[13], plagiarism affects absolutely all educational levels, and in the case of Romania, things are not going very well in this regard, the author mentioning that ”Romania presents the highest rate of plagiarism discovered in the Union territory, namely 26.1% of all verified works, a value almost double compared to the European average, as well as the fact that the incidence of plagiarism cases is much higher in the countries of the eastern region of Europe, countries whose living and education levels are considerably lower compared to western countries.”

Also, another study[3], published by Science, tells us that Romania is the country with the highest number of scientific articles removed from the content of specialized publications as a result of non-compliance with conduct standards in scientific research, by reporting to the total funds allocated for research and also, it ranks second in the ranking of countries with the highest number of withdrawn articles relative to the total articles published, following the discovery of plagiarism.

1.2 Project objectives

This project aims to provide an innovative system for exam monitoring, with the following specific objectives:

1. **Candidate gaze detection:** Signaling and recording situations where the candidate looks in other directions such as left, right or down, rather than at their own monitor screen or at their own sheet, in the case of a written exam.
2. **Identification of unauthorized objects:** Detection of devices such as smart watches or phones, which can be used to obtain information, which would lead to fraud of the respective exam.
3. **Recording and archiving:** Creating a video recording in which exam sessions will be captured and archived for subsequent analysis, highlighting probable fraud attempts.
4. **Report generation:** Creating detailed reports in HTML, CSV and JSON formats, thus contributing to a clearer and safer analysis for evaluators.
5. **Intuitive interface:** Development of an intuitive graphical interface that allows supervisors to use all the functionalities mentioned above.

1.3 Advantages of the proposed approach

In general, many anti-plagiarism solutions focus more on monitoring the candidate's screen, but this system is centered on their physical behavior. I believe that by tracking the direction of gaze and detecting unauthorized objects, the system can identify clues of fraudulent behavior, providing concrete evidence that can be analyzed later. This can contribute significantly to maintaining academic integrity, without the need to create that atmosphere of excessive surveillance.

It is normal for the respective system to offer, in addition to remarkable results, false positives as well, due to factors such as the illumination of the exam room or natural head movements, without the intention of looking in another direction. Thus, through the video recording and real-time signaling functionalities of possibly fraudulent behaviors, there can be, after the completion of the respective exam, a discussion between teacher and student based on their honesty to reach the final result.

Technology advances day by day, and various plagiarism attempts automatically diversify. Therefore, it is absolutely natural that the systems that try, through various means, to combat exam fraud, also advance.

2 Domain analysis and existing solutions

2.1 Similar existing solutions on the market

Various monitoring and anti-fraud solutions have appeared on the market over time for exams. Following research, I identified three main systems that propose solutions similar to this project: ProctorU[21], Proctorio[20] and Respondus Lockdown Browser[24].

Recently, modern online surveillance systems like Honorlock have implemented advanced technologies that detect mobile phone use ”through a combination of surveillance tools” and can determine when participants attempt to use unauthorized mobile devices to access content during exams[26]. Also, recent research in the field of eye-tracking has proposed ”a new solution for detecting cheating in online exams using gaze tracking technology”[11], demonstrating the effectiveness of this approach in detecting suspicious behavior.

2.1.1 ProctorU

ProctorU is one of the most used platforms in online exam supervision. It is based both on monitoring the candidate’s screen and their behavior.

Strengths	**Limitations**
• Candidate identity verification through ID or identity document • Real-time human supervision • Complete monitoring of computer activity • Suspicious behavior detection through AI	• High costs (15-25 USD per candidate) • Requires stable internet connection • Less suitable for physical exams • Potential privacy issues

Table 1: Comparative analysis of the ProctorU system

The platform records every second, ensures that no other tabs are open, except the one in which the respective exam is taken, does not allow certain keyboard combinations for functions like print, copy or paste and, through a webcam, the platform can efficiently supervise the candidate.

2.1.2 Proctorio

Another similar system is Proctorio, similar to ProctorU, but which is completely automated, as it relies only on AI and machine learning technologies for detecting possible fraud attempts, without the need for human assistance.

Strengths	**Limitations**
• Simple installation and configuration through browser extension • Web browsing blocking functionalities • Algorithms that reduce the need for human intervention • Lower costs than ProctorU	• High rate of false positives • Difficulty in detecting secondary device use • Works only on Google Chrome • Privacy issues

Table 2: Comparative analysis of the Proctorio system

Monitoring is based on facial recognition, analyzing behavior through head or eye movement. The system also handles screen monitoring, observes if other tabs are open and if background applications are running that could facilitate exam fraud.

2.1.3 Respondus Lockdown Browser

Respondus Lockdown Browser represents a different approach, as it focuses on creating a secure exam environment, by blocking the candidate's access to other applications and resources.

According to its official documentation[24], this solution integrates with learning management systems (LMS), such as Canvas, Blackboard, Moodle, Schoology and Brightspace.

Strengths	Limitations
• Prevents access to other programs and resources • Good integration with educational platforms (LMS) • Lower costs compared to other solutions • Easy to implement at institutional level	• Inability to detect secondary devices • Does not monitor candidate's physical behavior • Limited functionality on mobile devices • Incompatibility with specific programs (IDEs, etc.)

Table 3: Comparative analysis of the Respondus Lockdown Browser system

2.2 Relevant multimedia aspects

2.2.1 Protocols and data transmission

In terms of multimedia-related aspects, we encounter protocols that ensure secure and optimized data transmission:

- **WebRTC** for real-time audio-video communication. This protocol is used to transmit the video stream captured from the webcam to the processing system, ensuring minimal latency and optimal transmission quality[1]. In our project, WebRTC is used to enable real-time candidate monitoring, providing a stable and efficient connection.

- **TLS/SSL** for encrypting the entire data stream. This security layer is essential to protect data transmitted between client and server, preventing unauthorized access or interception of sensitive information[5]. In the context of our project, TLS/SSL is used to secure both the video stream and the generated data, such as violation reports.

  Although the current version of our project is based on local processing without network communication components, implementing these technologies would represent a valuable direction for future development, allowing remote monitoring of candidates.

2.2.2 Video processing

Regarding video processing, the stream captured from the webcam is processed in real time to detect suspicious behavior. For saving recordings, our system uses standard formats like MP4, which incorporates the H.264 codec, ensuring good efficiency in storage space usage[2].

Recent research in the field of exam surveillance systems has demonstrated the effectiveness of facial detection and eye tracking for identifying suspicious behavior[10]. Our implementation is based on these studies, integrating libraries like `dlib` and `OpenCV` to detect face position and gaze direction, thus providing a robust solution for identifying suspicious behavior.

This system focuses primarily on visual analysis, detecting both the candidate's gaze direction and the presence of unauthorized objects such as mobile phones and smart watches, elements that represent the most frequent means of exam fraud.

2.2.3 Watermarking

In Lockdown Browser type solutions, visible or invisible watermarks are applied over the content displayed on the screen, with the purpose of preventing unauthorized capture of exam information. In our project, watermarks are used to mark the recorded video stream with information such as the real date and time, but also the H and V values, which indicate the limits within which the candidate's gaze direction is found. This type of marking contributes to ensuring the authenticity and integrity of recorded data.

2.2.4 Event synchronization

To ensure precise correlation between video frames and detected events, our project uses temporal markings. These are generated at the moment of capturing each video frame and are used to link the alerts generated by the system with the exact moment of the violation. This approach allows subsequent consultation of video recordings and efficient verification of generated reports, thus providing concrete evidence for evaluating the candidate's behavior during the exam.

2.2.5 Optimizations for limited resources

To enable efficient system operation on devices with limited resources, several optimizations have been implemented in this system:

- Processing one frame out of 30 for object detection, thus reducing resource consumption.
- Using a circular buffer for efficient video frame management.
- Integrating a cache system for generated alerts, avoiding duplication of alert messages.

2.3 Advantages of the proposed solution compared to existing ones

The anti-plagiarism monitoring system developed within this project offers numerous comparative benefits compared to alternatives currently available on the market:

1. **Complete physical behavior monitoring** - While most commercial solutions (such as Proctorio or Respondus) are limited to supervising activity on the computer screen, our system observes the candidate's movements and gestures. This approach allows detection of fraud tactics that would otherwise go unnoticed, such as consulting physical notes or communicating with other people through gestures. The system analyzes gaze direction, head orientation and other physical clues that may signal fraudulent intentions.
2. **Precise identification of unauthorized electronic devices** - Through implementation of YOLOv8 algorithms and modern object recognition techniques, our application can identify with high accuracy mobile phones, smart watches and other devices frequently used for cheating. This capability significantly exceeds the functionalities offered by Respondus Lockdown Browser, which cannot detect secondary device use, and even those of ProctorU, which relies heavily on human supervisors for this task.
3. **Advanced reporting and documentation system** - The application automatically generates detailed reports in versatile formats (HTML for easy viewing, CSV for Excel analysis and JSON for integration with other systems). These reports include screenshots of suspicious moments, precise time stamps and clear descriptions of incidents, providing evaluators with all necessary information to make documented decisions. The system thus

maintains a complete history of the examination session, allowing subsequent verification and reducing disputes related to possible sanctions.

4. **Technological independence and implementation flexibility** - This system functions as an independent application, without being constrained by the specific limitations of web browsers or online platforms. This autonomous architecture eliminates external dependencies and simplifies the installation and configuration process at the institutional level. Its independent nature allows use on a wide range of devices and operating systems, offering educational institutions the freedom to implement the solution without being forced to adopt additional technologies or services. System flexibility facilitates rapid adaptation to the specific requirements of different types of exams and academic contexts.

5. **Economic efficiency and scalability** - Local system implementation offers significant advantages from a long-term cost perspective. While many existing commercial solutions operate on a per-student per-exam pricing model, which can become prohibitive for institutions with large numbers of students or frequent exams, this proposed anti-plagiarism system requires only the initial investment in existing hardware infrastructure. This scalable model allows educational institutions to monitor an unlimited number of exams without additional costs, reducing the effective cost per student with each organized session, and this approach makes anti-plagiarism monitoring technology accessible even to institutions with limited financial resources.

6. **Enhanced student data confidentiality** - Through local information processing, this prototype eliminates the risks associated with transmitting data to external servers. This aspect is particularly important in the context of strict regulations regarding personal data protection, offering institutions complete control over sensitive information collected during exams.

The combination of these advantages makes this anti-plagiarism system prototype represent an innovative and efficient solution for ensuring academic integrity, adapted to the current needs of educational institutions.

3 Anti-plagiarism system architecture

The developed anti-plagiarism system represents a complex software solution designed for monitoring candidates during exams, with the purpose of detecting and preventing fraudulent behavior. The system architecture was conceived according to modern software engineering principles, emphasizing modularity, extensibility and clear separation of responsibilities.

The basis of system development is represented by object-oriented programming, which allowed structuring the application into interconnected components, each having a well-defined role in the monitoring process. This approach facilitates not only the independent development and testing of modules, but also the subsequent extension of functionalities through the addition of new components.

The system processes video images as they are captured, tracking where candidates are looking and identifying unauthorized objects such as mobile phones and smart watches. This tracking is based on computer programs that can ”see” and ”understand” images, organized in a layered structure that keeps simple functions separate from more advanced ones.

To illustrate the system structure and operation, I have developed three complementary UML diagrams: the class diagram, the sequence diagram and the activity diagram. These representations offer different perspectives on the architecture, from the static organization of classes to dynamic control and data flows.

3.1 Class diagram

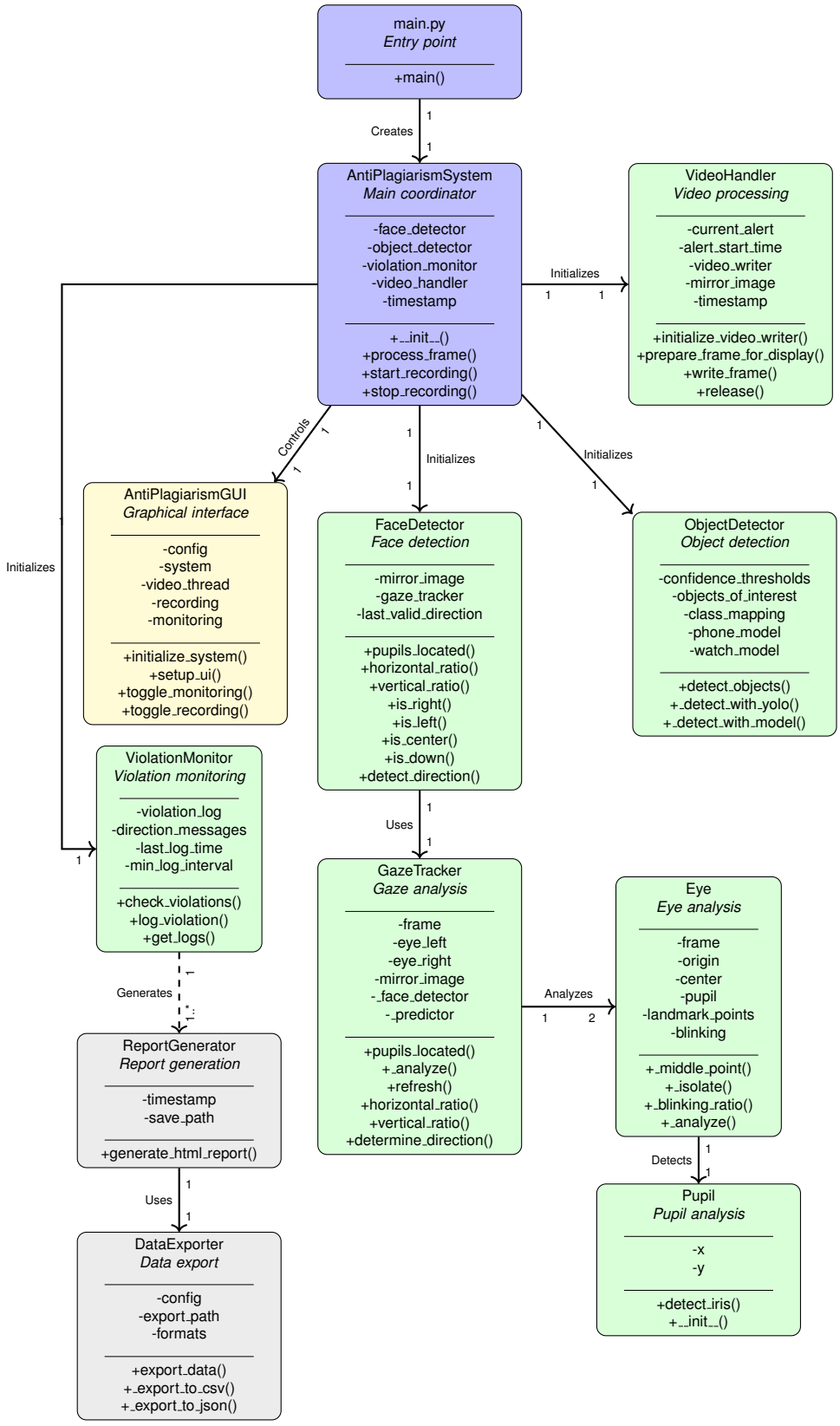


Figure 1: Class diagram of the Anti-Plagiarism system with relational cardinality

The class diagram highlights the hierarchical organization of anti-plagiarism system modules and the relationships between them. The entry point `main.py` creates an instance of the `AntiPlagiarismSystem` class, which occupies the central position in the architecture, coordinating all other components. The arrow from `main.py` to `AntiPlagiarismSystem` represents this fundamental initialization from which the entire system starts.

`AntiPlagiarismSystem` initializes four essential components, represented by divergent arrows: `AntiPlagiarismGUI` for the graphical interface, `VideoHandler` for video processing, `FaceDetector` for facial analysis and `ObjectDetector` for object identification. This structure reflects the methods from the `AntiPlagiarismSystem` class constructor, where instances of these objects are created.

The relationship between `FaceDetector` and `GazeTracker` is one of creation, where the facial detector initializes and uses the gaze tracking module. This Facade-type structure simplifies the interface with the complex gaze analysis subsystem. We then observe how `GazeTracker` uses `Eye` for ocular region isolation, which in turn uses `Pupil` for precise pupil localization. This succession of delegations reflects the increasing level of analysis specialization.

`ViolationMonitor`, initialized by `AntiPlagiarismSystem`, aggregates information about gaze direction and detected objects to identify suspicious behavior. When it detects violations, these are transmitted to `ReportGenerator` which uses `DataExporter` to save reports in various formats.

This organization reflects the principle of separation of responsibilities, each class having a specific role in the analysis and processing of video data for identifying plagiarism attempts.

3.2 Sequence diagram

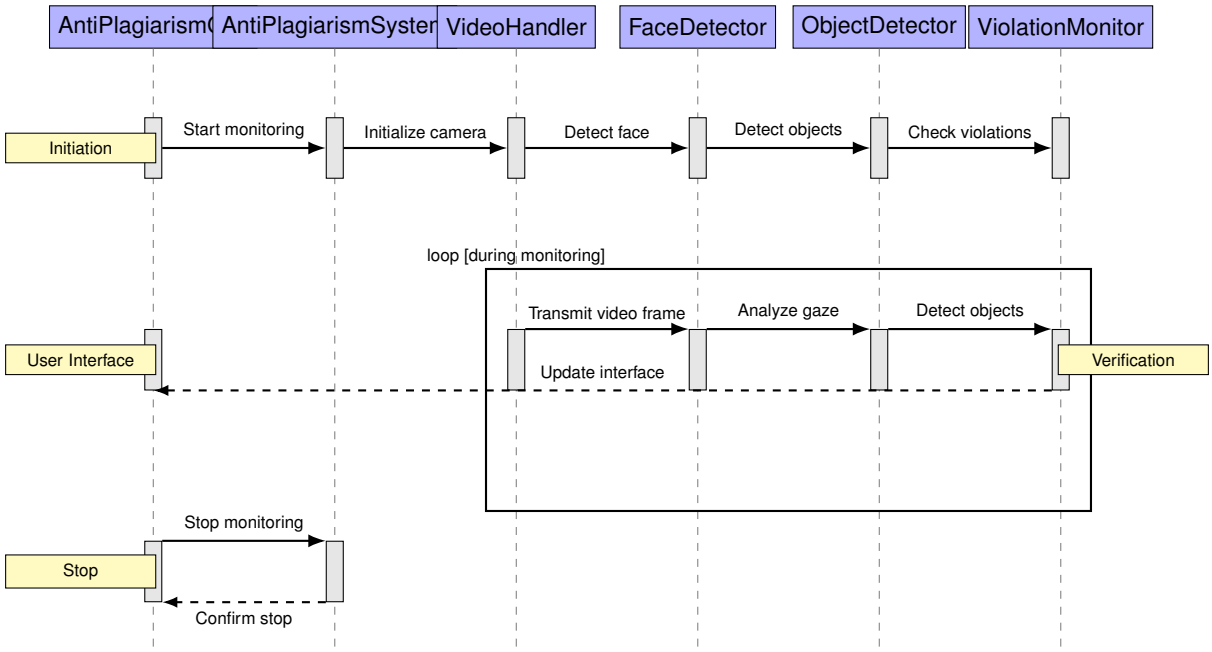


Figure 2: Sequence diagram of the Anti-Plagiarism system

The sequence diagram illustrates the temporal interaction between the main components of the anti-plagiarism system. The sequence begins when the user initiates monitoring through the `AntiPlagiarismGUI` interface, which transmits the command to `AntiPlagiarismSystem`. This, in turn, initializes the camera through `VideoHandler` and starts the facial detection, object detection and violation monitoring processes.

The main processing loop, highlighted in the diagram, shows how `VideoHandler` captures video frames which are then analyzed by `FaceDetector` for face position and gaze direction, followed by `ObjectDetector` for identifying forbidden objects. `ViolationMonitor` receives the results of both analyses and determines if there are violations, and the graphical interface is updated accordingly with the new information.

The sequence ends with stopping monitoring, when the user sends the stop command through the interface, and the system confirms the end of the monitoring process. This succession of messages and activations illustrates the complete flow of control and data in the system, from initiation to conclusion, highlighting how all components collaborate to achieve real-time candidate monitoring.

3.3 Activity diagram

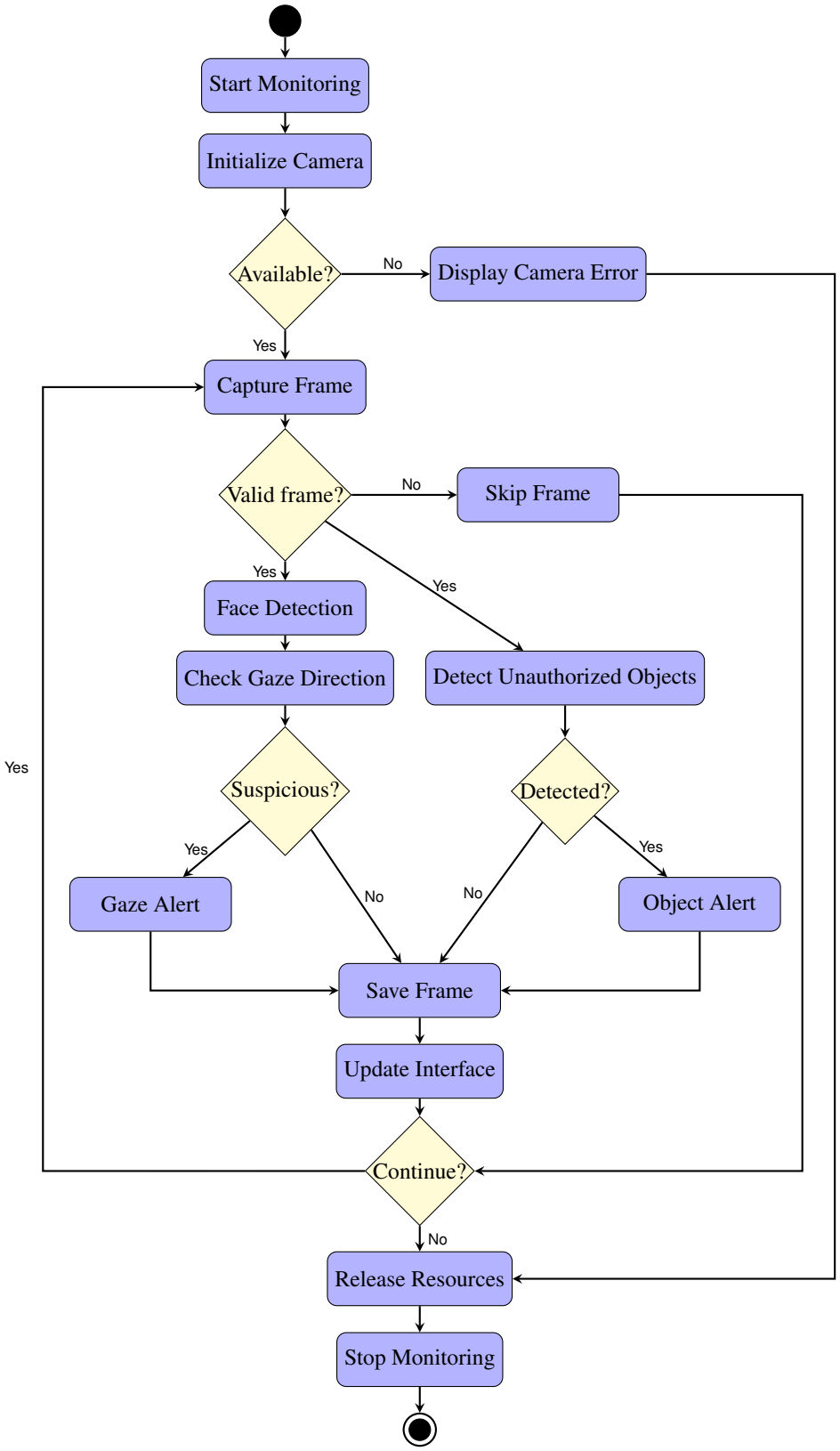


Figure 3: Activity diagram of the Anti-Plagiarism system

The activity diagram presents the operational flow of the system, highlighting processing steps and decision points. From the initial state `StartMonitoring`, activated when the user presses the corresponding button, the system moves to `InitializeCamera`, where it configures capture parameters and checks webcam availability. This transition corresponds to calling the `toggle_monitoring()` method from the main `AntiPlagiarismGUI` class.

After initialization, the system checks if the respective camera is available through the `Available?` decision point. In case of an error, the system displays an error message through the `DisplayCameraError` process and advances directly to resource release. This verification is implemented in the `VideoProcessingThread.run()` method, where the webcam connection is tested.

If it is functional, the process continues with `CaptureFrame`, where frames are captured from the webcam. The system then validates each captured frame through the `Validframe?` decision, which checks if the image was obtained correctly. Invalid frames are handled through the `SkipFrame` process, which ignores them and continues the flow. This verification is implemented through the `ifretandframeisnotNone` condition from the video processing module.

After frame validation, the analysis branches into two independent parallel flows:

1. In the first flow, the system executes `FaceDetection`, where `FaceDetector` locates the candidate's face in the image, followed by `CheckGazeDirection`, which analyzes the actual position of pupils to determine which direction the candidate is looking, using the `face_detector.detect_direction()` method. At the `Suspicious?` decision point, it is evaluated whether the gaze direction indicates suspicious behavior (lateral or downward gaze).

2. In parallel, in the second flow, `DetectUnauthorizedObjects` uses YOLOv8 models to scan the image and identify phones or smart watches through the `object_detector.detect_objects()` method. The `Detected?` decision checks the presence of these forbidden objects.

These two flows operate completely independently, processing the same valid frame but analyzing different aspects - one focusing on candidate behavior and the other on the surrounding environment. This parallelization reflects the modular architecture of the system, where `FaceDetector` and `ObjectDetector` function as separate modules.

Each flow can generate specific alerts when detecting potential violations: `GazeAlert` for suspicious gazes and `ObjectAlert` for unauthorized objects. These are implemented through calling the `violation_monitor.log_violation()` method, which records the details of each violation.

The flows then converge at the `SaveFrame` activity, where the system stores the processed image, followed by `UpdateInterface`, which refreshes the display with the analyzed data. This update is performed through the `frame_ready` signal from PyQt5, which triggers the graphical interface update.

At the `Continue?` decision point, the system evaluates whether monitoring should continue. If affirmative, the process returns to `CaptureFrame` for a new iteration. If negative, the system moves to `ReleaseResources`, where camera connections are closed and memory is freed, followed by `StopMonitoring`, when the process officially ends.

This robust architecture, with parallel processing, allows the system to simultaneously analyze different aspects of the examination situation, increasing efficiency and precision in detecting fraud attempts, even when one of the analysis methods (face or object detection) might encounter temporary difficulties.

4 Actual Implementation

In implementing the anti-plagiarism system, it was necessary to use several specialized libraries, each having a different role in its operation.

4.1 OpenCV

I used the OpenCV library (`OpenSourceComputerVisionLibrary`), which had a main role in image processing and analysis[12]. With its help, I captured the video stream from the webcam, through the `VideoCapture` function, which acts as the primary interface with the physical world, capturing images of the candidate which are then analyzed to detect suspicious behavior.

This library provided me with the `cvtColor` function, with which I managed to transform color images into grayscale, which led to much higher efficiency on the facial detection side,

because it reduces computational complexity, referring only to a single value for each pixel, instead of three, and improves the performance of image analysis algorithms, which often work better with grayscale images.

In the context of precise pupil detection, I applied a Gaussian filter, with the purpose of reducing noise in images, calling the `cv2.GaussianBlur` function, offered by the respective library, which is an essential filter in image processing. The parameters of this function are the source image, being the isolated region of the eye, the kernel size, in our case being 7x7, because it was found, after several tests, that it offers the best results for pupil detection and is large enough to cover minor intensity variations around the pupil, compared to smaller kernels of 3x3 or 5x5, which preserve too many fine details and noise or kernels too large of 9x9 or 11x11, which blur the image too much, leading to loss of the pupil edge and the standard deviation of the Gaussian distribution, this parameter being set to the value 0, meaning it is calculated automatically, using the formula:

$$\sigma = 0.3 \cdot \left(\frac{\texttt{kernel\_size} - 1}{2} - 1 \right) + 0.8 \tag{1}$$

For a kernel of size 7×7, this formula becomes:

$$\begin{aligned} \sigma &= 0.3 \cdot \left(\frac{7-1}{2} - 1 \right) + 0.8 \\ &= 0.3 \cdot (3-1) + 0.8 \\ &= 0.3 \cdot 2 + 0.8 = 1.4 \end{aligned}$$

The `cv2.minMaxLoc` function from OpenCV is an essential tool for pupil detection. This function identifies minimum and maximum values in an image, along with their coordinates. It returns four values: the minimum value, the maximum value, the coordinates of the minimum value and the coordinates of the maximum value. In our case, the value of interest is represented by the coordinates of the minimum value, because it represents the darkest point in the eye region, and the pupil is typically the darkest part of the eye, so this efficient approach allows rapid localization of the pupil center. This method works because, before applying `minMaxLoc`, the image is pre-processed by that Gaussian filter, mentioned above, which reduces noise and uniformizes values, making detection more robust to minor variations in illumination or iris texture.

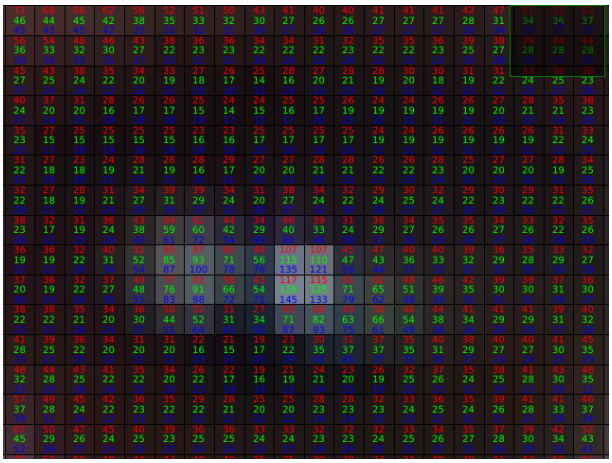


Figure 4: RGB values of pixels in the ocular area

The image above represents a screenshot of the RGB values of pixels in my own ocular area, and these results were obtained with the help of running the test file `test_camera.py`.

Also with the help of OpenCV, I managed to draw suggestive visual elements, such as rectangles in the area of detected objects, using the `cv2.rectangle()` function, and to display text for alerts, using `cv2.putText()`, and the saving and compression of recordings was performed using `VideoWriter`.

Also, I managed to create a mask, to isolate the object of interest, like the eye, using functions like `fillPoly` and `bitwise_not`. The `cv2.fillPoly` function fills a polygonal area in an image with a specific color, and, in the context of this system, it is used to create a mask for isolating the eye region. It receives as parameters the image on which it draws, an array of points that define the polygon, in our case being the eye contour, and the fill color, for example (0, 0, 0), which in RGB means black color. Thus, the result is a mask in which the eye region is colored black, and the rest of the image remains white.

The `cv2.bitwise_not` function inverts the bit values of each pixel in the image, being equivalent to the bitwise negation operation ($\sim$). I used it to invert the created mask, so that the eye region is isolated from the rest of the image. This function receives as parameters the source image and an optional mask that determines on what region the operation is applied. Thus, these two functions work together to create a precise ”cut” of the eye region, from the entire image, to analyze the pupil very precisely, without interference from other parts of the image.

The `cv2.circle` function from OpenCV draws a circle on an image, thus I use it for visual

highlighting of detected pupils. I mark the position of detected pupils on the image, highlight them using different colors to indicate the state, such as green for centered gaze or red for suspicious gaze, and provide at the same time visual feedback for the supervisor, regarding the candidate's gaze direction. As parameters, the function receives the image on which the respective circle is drawn, the coordinates of the circle center, in our case the pupil center, the circle radius, the color and line thickness, but, for thickness, I used the value -1, because I wanted a filled circle.

Therefore, the OpenCV library is responsible for everything that means "seeing" in this system.

4.2 Dlib

The Dlib library was used for face detection and facial landmark extraction[17]. This was essential for gaze analysis and determining its direction.

The `dlib.get_frontal_face_detector` function was used to initialize the HOG (Histogram of Oriented Gradients) based face detector, which proved efficient even in variable lighting conditions. This algorithm works by dividing the image into small cells and calculating the intensity gradient for each cell. The gradient is then normalized to reduce the effects of illumination variations, and the result is a robust descriptor that can be used to detect faces in the image. The detector returns the coordinates of rectangles that frame the detected faces, and these coordinates are used to isolate the face region.

The facial landmark predictor was loaded using the `dlib.shape_predictor` function, which allowed identification of the 68 points on the face. This predictor uses a pre-trained regression-based model, which estimates facial point positions based on features extracted from the image. Among the 68 points, points 36-47 were used to isolate the eye region. These points are distributed to frame the eye contour, allowing precise isolation of the region of interest.

The `shape.part` function was used to access the coordinates of each facial point, and these coordinates were used to calculate horizontal and vertical gaze ratios. For example, the horizontal ratio is calculated as the ratio between the distance between pupils and the width of the eyes, and the vertical ratio is calculated as the ratio between the height of the eyes and their width. These ratios are used to determine gaze direction, such as looking left, right, up or down.

Also, these points were used to determine head orientation, by calculating distances between

key points, such as eyes, nose and mouth. Head orientation is estimated by comparing these distances with reference values, and the result is used to detect head movements that may indicate suspicious behavior.

4.3 PyTorch and YOLOv8

For detecting forbidden objects, such as mobile phones and smart watches, I used the PyTorch library[19] and the YOLOv8 model developed by Ultralytics[16].

The choice of the YOLO algorithm for this project is based on recent academic research that emphasizes that "YOLO is adopted in various applications mainly due to its faster inference"[15], making it ideal for real-time applications like this prototype. This algorithm "uses a simple deep convolutional neural network to detect objects in images"[27], making it suitable for real-time detection of objects like phones or smart watches, even in variable lighting conditions.

The `torch.load` function was used to load pre-trained YOLOv8 models. These models are based on the YOLO (You Only Look Once) architecture, which is optimized for fast and precise object detection in images[4]. YOLOv8 uses a convolutional neural network that divides the image into grids and simultaneously predicts object classes and their coordinates. This approach enables real-time detection, even on devices with limited resources.

The `model.predict` method was used to detect objects in the video frame. The model was configured to process only one frame out of 20 to save resources and to detect only categories of interest. This optimization significantly reduces processor and memory consumption, allowing efficient application running on ordinary hardware.

To reduce false positives, I used two separate YOLOv8 models: one for phone detection and another for smart watches. This approach allowed precise classification of detected objects, even in low lighting conditions or when objects were partially visible. For example, the phone model was trained to recognize typical shapes and dimensions of mobile phones, and the smart watch model was trained to detect wearable devices on the wrist.

The `model.detect` function was used to generate coordinates of detected objects, and these coordinates were used to draw rectangles around objects and to generate real-time alerts. Also, coordinates were used to calculate distances between detected objects and other elements in the image, such as the candidate's face, to determine if objects are actively used.

4.4 NumPy

The NumPy library was used for matrix manipulation and numerical data[8].

The `np.zeros` function was used to create masks for regions of interest in the image, and `np.full` was used to initialize matrices with constant values. These functions are essential for image preprocessing, because they allow isolation of regions of interest and application of specific operations only on these regions[2].

The `np.min` and `np.max` functions were used to determine the limits of regions to crop, and vectorial operations allowed rapid calculation of pupil coordinates and horizontal and vertical ratios. For example, pupil coordinates are calculated as points with minimum intensity in the eye region, and ratios are calculated as ratios between distances between these coordinates and region dimensions.

4.5 Datetime

The `datetime`[22] library from Python was used for time management and temporal markings in the project. This was essential for event synchronization, report generation and monitoring time recording.

The `datetime.now()` function was used to obtain the current time at the moment of generating an event, such as detecting an irregularity and saving a report. The created temporal markings are used to link detected events to exact moments in the video stream.

For example, in the `ViolationMonitor` class, the `datetime.now().strftime()` function was used to format temporal markings in an easy-to-read format, such as `YYYY-MM-DDHH:MM:SS`. These are included in generated reports and are used to analyze candidate behavior over time.

The `timedelta` function was used to calculate time differences between events. For example, the time elapsed since the beginning of recording is calculated using `datetime.now()` and the recording start time. This is displayed in the graphical interface to provide the user with real-time information about monitoring duration.

Also, the `datetime` library was used to generate unique names for recording files and reports, using the `strftime()` function to include temporal markings in file names.

The `datetime` library was thus an essential tool for time management in all aspects of the project, from real-time monitoring to detailed report generation.

4.6 Logging

The `logging`[23] library from Python was used for managing logging messages in the project. This was essential for monitoring application operation, diagnosing problems and maintaining a history of important events.

The `logging.basicConfig()` function was used to configure the format and logging level. In the project, logging messages are saved both in a file (`anti_plagiarism_system.log`) and displayed in the console. This allows efficient real-time application monitoring, as well as subsequent problem analysis.

For example, in the `AntiPlagiarismSystem` class, the `logging` library was used to record important events, such as starting video recording, detecting violations or errors that occurred during processing.

The `logging` library was also used to record exceptions that occurred during application execution, using the `logger.exception()` function, which automatically includes exception details in the logging message.

This approach allowed rapid problem identification and application stability improvement. The `logging` library was thus an essential tool for managing logging messages in all project modules.

4.7 PyQt5

For graphical interface development, I used the PyQt5 library[25]. This allowed creating an intuitive and functional interface, using the following main components:

- **QPushButton:** I used this widget to create interactive buttons, such as those for starting monitoring, recording or exporting reports. These buttons are essential to allow the user to interact with the application in a simple and direct way. For example, the report export button offers clear and accessible functionality for the user.

- **QTextEdit:** This widget is used for real-time display of generated reports. It is a suitable choice because it allows displaying formatted text and can be set as ”read-only”, so that the user cannot modify the report content.
- **QTimer:** The timer is used for periodic updating of recording time. This is an efficient solution to keep the interface synchronized with background processes, such as real-time monitoring.
- **QLabel:** I used QLabel for displaying important information, such as the number of detected violations or recording time. QLabel is ideal for displaying static or dynamic text, being easy to update depending on application state.
- **QCheckBox:** The checkbox for enabling or disabling image mirroring mode is an intuitive choice. This allows the user to customize how the video stream is displayed, which can be useful depending on their preferences or needs.
- **QGroupBox and Layouts (QVBoxLayout, QHBoxLayout):** These are used for logical organization of interface elements. For example, grouping statistics or controls in a QGroupBox helps create a clearer and more structured interface. Vertical and horizontal layouts are essential for arranging components in a responsive and aesthetic way.
- **QMainWindow:** I used this class as the main window of the application. It is a standard choice for complex applications, because it offers support for menus, toolbars and other advanced elements.
- **QMessageBox:** For displaying confirmation or error messages, QMessageBox is a simple and efficient solution. This allows clear communication with the user at critical moments, such as confirming report export or notifying of an error.

The interface was designed to be intuitive and easy to use, offering quick access to all system functionalities.

4.8 Modularization and project structure

The project was organized modularly, each Python file having a well-defined role. This approach facilitates code maintenance, testing and functionality extension. Below are detailed the main modules and their responsibilities:

- **main.py:** This is the application entry point. It initializes main components, such as face detector, object detector, violation monitor and video handler. It also manages frame processing and report export.
- **gui_app.py:** Contains the graphical interface implementation using the PyQt5 library. Includes functionalities such as starting monitoring, video recording, real-time alert display and report export.
- **test_camera.py:** An auxiliary script for testing webcam functionality. Checks if the camera can be accessed and if the video stream is available.
- **modules/face_detector.py:** This module implements face detection and gaze direction analysis using the Dlib library. It is responsible for face identification and determining the candidate's gaze direction.
- **modules/object_detector.py:** Contains the implementation for detecting forbidden objects, such as mobile phones and smart watches, using the YOLOv8 model. It is optimized for efficient frame processing.
- **modules/video_handler.py:** Manages video frame processing and display. Includes functionalities for image mirroring, alert display and saving video recordings.
- **modules/violation_monitor.py:** Monitors detected violations, such as suspicious gaze or use of forbidden objects. Records alerts and synchronizes them with temporal markings.
- **modules/report_generator.py:** Is responsible for generating reports in HTML, CSV and JSON formats. These reports include details about detected violations and general statistics.
- **modules/data_exporter.py:** Manages data export in specified formats (CSV, JSON). It is used to save violation information in a structured way.
- **modules/gaze_tracking/gaze_tracker.py:** This file implements the main logic for gaze tracking. Uses facial landmarks to calculate horizontal and vertical gaze ratios and to determine its direction.
- **modules/gaze_tracking/eye.py:** This module isolates the eye region from the image and detects pupils. It also calculates the blinking ratio to identify if the eye is closed.

- **modules/gaze_tracking/pupil.py:** This file implements the logic for pupil detection and analysis, using their coordinates and dimensions to determine eye state.
- **requirements.txt:** This file contains the list of all libraries and packages needed to run the application. It is used for rapid dependency installation.
- **config.json:** The application configuration file, which stores customizable settings, such as detection thresholds or report saving options.
- **.gitignore:** The file that specifies what files or directories should be ignored by Git, such as temporary files or directories generated when running the program.
- **.devcontainer/:** The directory that contains configuration for the container development environment, which includes Docker files.

This modular structure allows clear separation of responsibilities, making the project easier to maintain and extend. Each module can be tested and improved independently, without affecting other parts of the application.

5 Presentation of usage mode, user interaction and configuration

This section describes the application usage mode, user interaction and its configuration. Also, screenshots are included to illustrate main functionalities.

5.1 Usage mode

To use the application, follow the steps below:

1. **Application launch:** Open the application by running the `gui_app.py` file.
2. **Initial configuration:** Adjust configuration parameters, such as gaze limits (*left limit*, *right limit*, *down limit*), from the graphical interface or from the `config.json` file. Also in the `config.json` file you can modify the confidence threshold for detecting forbidden objects, but it is not recommended, because it is set to a value that ensures the most accurate detection of forbidden objects, due to numerous tests performed.

3. **Starting monitoring:** Press the **Start Monitoring** button to begin video capture and behavior analysis.

4. **Starting recording:** Press the **Start Recording** button to save the video stream and generated alerts.

5. **Alert viewing:** Monitor alerts generated in real time in the dedicated section of the interface.

6. **Report export:** After session completion, use the **Export Report** button to save reports in HTML, CSV or JSON format.

5.2 User interaction

The application's graphical interface is intuitive and includes the following components:

- **Control buttons:** Allows the user to start/stop monitoring and export reports.
- **Alert section:** Displays real-time alerts generated by the system.
- **Customizable settings:** Offers options for adjusting gaze limit parameters and enabling/disabling image mirroring mode.
- **Stop monitoring:** Press the **Stop Monitoring** button to stop video capture and behavior analysis.
- **Stop recording:** Press the **Stop Recording** button to save the video stream and generated alerts in a file.
- **Instant capture:** Press the **Instant Capture** button to save an image of the current video stream.

It will be observed during application use that, depending on gaze direction, the color of the circle drawn on detected pupils will change. For example, if the gaze is centered, the circle will be green, and if the gaze is left, right or down, the circle will be red.

5.3 Configuration

The application can be configured by editing the `config.json` file. Examples of configurable parameters:

- `"left_limit":0.7` - Left limit for gaze direction.
- `"right_limit":0.3` - Right limit for gaze direction.
- `"down_limit":0.6` - Lower limit for gaze direction.

5.4 Screenshots

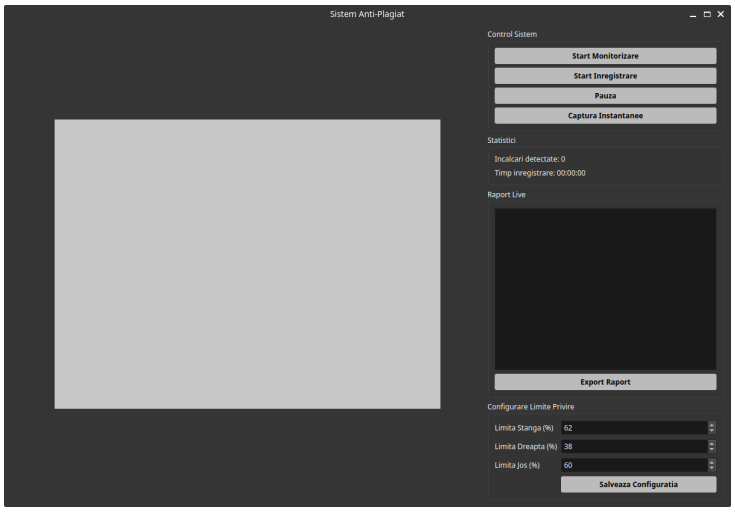


Figure 5: Main application interface

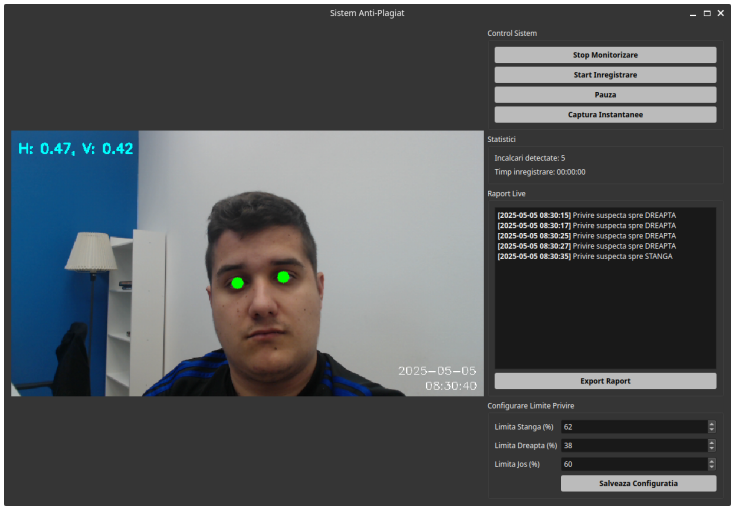


Figure 6: Starting monitoring and recording

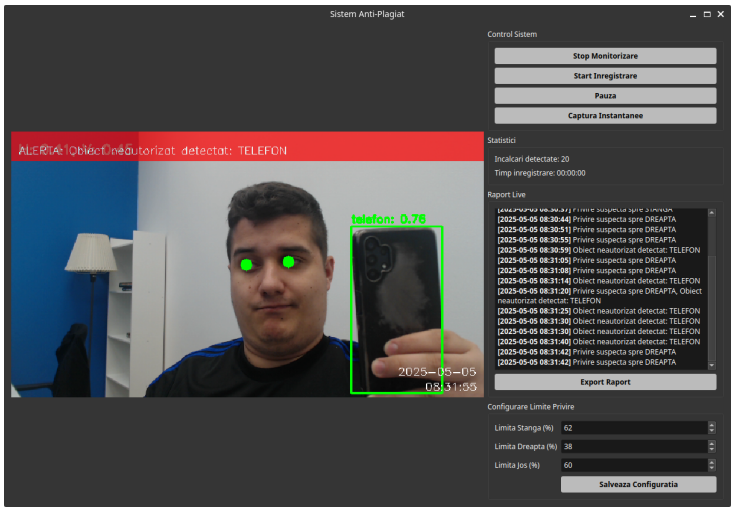


Figure 7: Phone detection

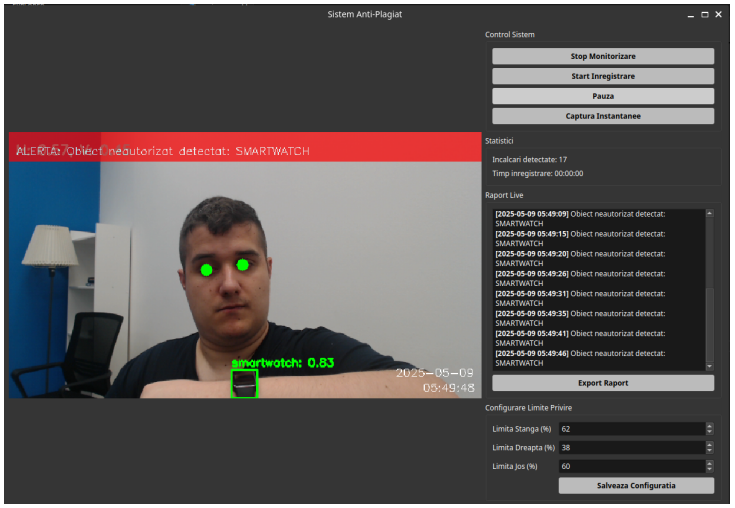


Figure 8: Smart watch detection

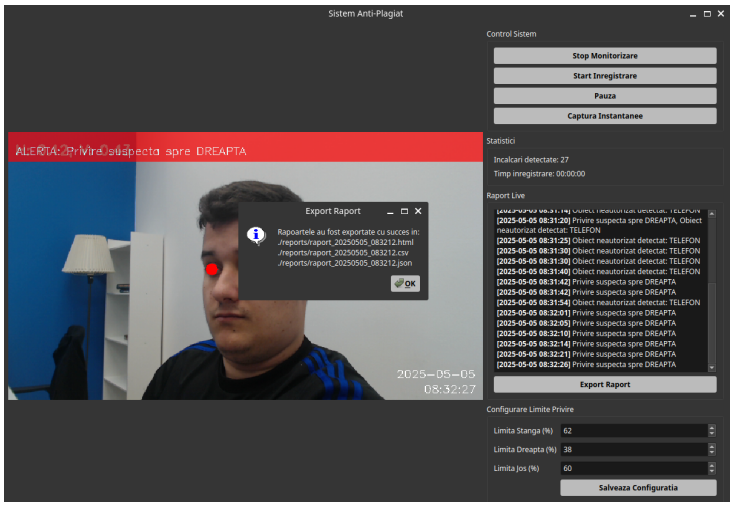


Figure 9: Report export

6 Personal contribution

In this project, I brought several significant contributions, which include:

6.1 Using two YOLOv8 models for detecting forbidden objects

I used two separate YOLOv8 models, one specialized for detecting mobile phones and another for detecting smart watches. This approach was chosen to work efficiently and to eliminate false positives. Instead of retraining a single model for both categories, I applied strict classifications based on the *x*-*y* dimensions of detected objects. I specify that each YOLOv8 model has a different confidence threshold, and these values were established following certain tests performed and are found in the main configuration file `config.json`. Thus, `confidence_threshold` is set to 0.55 for phones and 0.4 for smart watches. These values are essential to ensure precise detection and to minimize the risk of false positives.

6.2 Implementation of linear algebra calculations for gaze analysis

I developed and integrated linear algebra calculations to determine the user's gaze direction. These calculations include:

- Determining pupil coordinates using vectors and reference points.
- Calculating horizontal and vertical ratios (*horizontal ratio* and *vertical ratio*) to estimate gaze direction.
- Filtering and adjusting values to eliminate noise and unexpected variations.

The horizontal ratio is essential for determining if the respective candidate is looking left or right. This algorithm calculates the ratio based on pupil positions and includes an adjustment component for head orientation.

Algorithm 1 Horizontal ratio calculation (X axis - left/right)

procedure CALCULATEHORIZONTALRATIO
  if $pupils\_found = false$ **then**
    return 0.5 ▷ Pupils not detected $\Longrightarrow$ return neutral value
  end if
  $left\_pupil\_pos \leftarrow \frac{left\_pupil.x}{(left\_eye\_center.x \cdot 2 - 10)}$ ▷ Normalize X position
  $right\_pupil\_pos \leftarrow \frac{right\_pupil.x}{(right\_eye\_center.x \cdot 2 - 10)}$
  $pupil\_ratio \leftarrow \frac{left\_pupil\_pos + right\_pupil\_pos}{2}$ ▷ Average of normalized positions
  if $left\_origin \neq null$ **and** $right\_origin \neq null$ **then**
    $left\_abs\_center \leftarrow left\_origin.x + left\_eye\_center.x$
    $right\_abs\_center \leftarrow right\_origin.x + right\_eye\_center.x$
    $eye\_distance \leftarrow right\_abs\_center - left\_abs\_center$
    if $frame \neq null$ **then**
      $frame\_width \leftarrow frame.shape[1]$ ▷ Actual image width
    else
      $frame\_width \leftarrow 640$ ▷ Default value if no frame exists
    end if
    $position\_factor \leftarrow \frac{eye\_distance}{frame\_width \cdot 0.3}$ ▷ Evaluate relative head position
    $adjusted\_ratio \leftarrow pupil\_ratio$ ▷ Initially, ratio is not modified
    if $position\_factor < 0.8$ **then**
      $adjusted\_ratio \leftarrow \max(0.6, pupil\_ratio)$ ▷ Head turned left
    else if $position\_factor > 1.2$ **then**
      $adjusted\_ratio \leftarrow \min(0.4, pupil\_ratio)$ ▷ Head turned right
    end if
    return $adjusted\_ratio$
  end if
  return $pupil\_ratio$ ▷ No head data $\Longrightarrow$ return direct average
end procedure

The vertical ratio is used to determine if the candidate's gaze is oriented downward. This algorithm calculates the ratio based on vertical pupil positions and includes an adjustment component for head inclination.

Algorithm 2 Vertical ratio calculation (Y axis - up/down)

procedure CALCULATEVERTICALRATIO
  if $pupils\_found = false$ **then**
    return 0.5 ▷ Cannot estimate direction $\implies$ default value
  end if
  $left\_pupil\_pos \leftarrow \frac{left\_pupil.y}{(left\_eye\_center.y \cdot 2 - 10)}$ ▷ Normalize Y position
  $right\_pupil\_pos \leftarrow \frac{right\_pupil.y}{(right\_eye\_center.y \cdot 2 - 10)}$
  $pupil\_ratio \leftarrow \frac{left\_pupil\_pos + right\_pupil\_pos}{2}$ ▷ Average of normalized positions
  if $left\_origin \neq null$ **and** $right\_origin \neq null$ **then**
    $eye\_center\_y \leftarrow \frac{left\_origin.y + left\_eye\_center.y + right\_origin.y + right\_eye\_center.y}{2}$
    $mouth\_pos \leftarrow left\_origin.y + left\_eye\_center.y + 50$ ▷ Mouth estimation
    $face\_height \leftarrow |eye\_center\_y - mouth\_pos|$ ▷ Face height estimation
    if $face\_height > 0$ **then**
      $position\_factor \leftarrow \frac{|left\_origin.y - eye\_center\_y|}{face\_height}$ ▷ Head inclination (up/down)
    else
      $position\_factor \leftarrow 0.5$ ▷ Default value when calculation is not valid
    end if
    $adjusted\_ratio \leftarrow pupil\_ratio$
    if $position\_factor > 0.6$ **then**
      $adjusted\_ratio \leftarrow \min(0.4, pupil\_ratio)$ ▷ Head is bent
    else if $position\_factor < 0.4$ **then**
      $adjusted\_ratio \leftarrow \max(0.6, pupil\_ratio)$ ▷ Head is raised
    end if
    return $adjusted\_ratio$
  end if
  return $pupil\_ratio$ ▷ Incomplete data $\implies$ return initial average
end procedure

These algorithms work together to create a robust gaze direction detection system. The calculated ratios are compared with predefined thresholds to determine if the candidate is looking center, left, right or down, and the results of this analysis are used to generate corresponding alerts.

This implementation offers several significant advantages:

1. **Robustness to illumination variations** - By using the minimum point method for pupil detection, the algorithm is less sensitive to lighting changes.

2. **Head orientation compensation** - The adjustments implemented for the eye position factor and head inclination factor allow distinguishing between a suspicious gaze and natural head movement.

3. **Coordinate normalization** - Converting absolute coordinates to normalized ratios makes the algorithm adaptable to different camera resolutions and distances from the screen.

4. **Optimized threshold-based filtering** - The thresholds used were determined empirically to offer the best balance between detection rate and false positive rate.

These algorithms have been tested in various lighting conditions, demonstrating high accuracy in gaze direction detection, thus contributing to the efficiency of the anti-plagiarism system.

6.3 Real-time video processing optimization

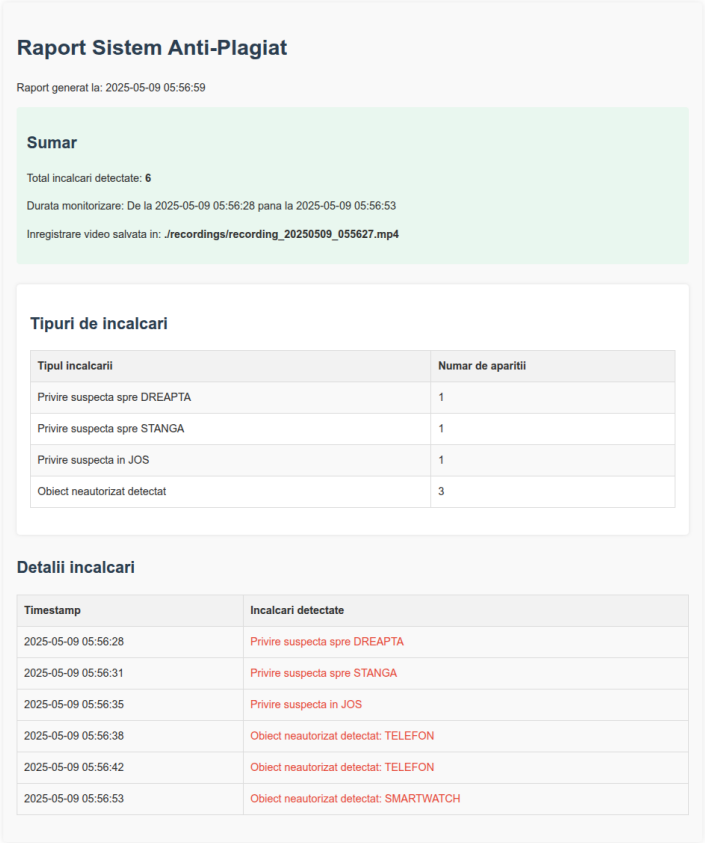
I implemented video processing optimizations, considering that the human mind cannot instantly process certain information that can be used for exam fraud purposes, including:

- Processing one frame out of 30 for object detection, thus reducing resource consumption.
- Using a circular buffer for efficient video frame management.
- Integrating a cache system for generated alerts, avoiding duplication of alert messages.

6.4 Detailed report generation

I developed an automatic report generation system in HTML, CSV and JSON formats. These reports include:

- The exact time of each detected violation.
- The type of detected object and its location in the video frame.
- General summary of the monitoring session.



Raport Sistem Anti-Plagiat

Raport generat la: 2025-05-09 05:56:59

Sumar

Total incalcari detectate: **6**

Durata monitorizare: De la 2025-05-09 05:56:28 pana la 2025-05-09 05:56:53

Inregistrare video salvata in: **./recordings/recording_20250509_055627.mp4**

Tipuri de incalcari

Tipul incalcarii	Numar de aparitii
Privire suspecta spre DREAPTA	1
Privire suspecta spre STANGA	1
Privire suspecta in JOS	1
Obiect neautorizat detectat	3

Detalii incalcari

Timestamp	Incalcari detectate
2025-05-09 05:56:28	Privire suspecta spre DREAPTA
2025-05-09 05:56:31	Privire suspecta spre STANGA
2025-05-09 05:56:35	Privire suspecta in JOS
2025-05-09 05:56:38	Obiect neautorizat detectat: TELEFON
2025-05-09 05:56:42	Obiect neautorizat detectat: TELEFON
2025-05-09 05:56:53	Obiect neautorizat detectat: SMARTWATCH

Figure 10: Example report generated in HTML format

6.5 Intuitive graphical interface integration

I developed an intuitive graphical interface using PyQt5, which allows users to:

- Start and stop real-time monitoring.
- View alerts generated in real time.
- Export detailed reports with a single click.

6.6 Real-time gaze limit parameter editing

I implemented a functionality that allows the user to edit gaze limit parameters (*left limit*, *right limit*, *down limit*) in real time, without requiring application restart. This functionality is useful in particular scenarios, such as differences between taking an exam on computer and on paper.

For example, in the case of a paper exam, the candidate is forced to bend more to write, which

can reduce the false positive rate by adjusting the lower limit (*down limit*). This flexibility offers supervisors the possibility to quickly adapt the system to the specific context of the exam.

This functionality contributes to reducing false positives and increasing system accuracy in certain frequently encountered scenarios.

7 Difficulties encountered

In realizing this project, I encountered several difficulties, in which I used certain constraints and less conventional solutions, to have results as close as possible to those desired:

1. **YOLOv8 model integration:** Configuring and optimizing YOLOv8 models for detecting forbidden objects was a challenge, especially regarding reducing false positives. It was necessary to adjust detection thresholds and use separate models for different object categories.

2. **Performance on limited hardware:** Real-time video processing was difficult on devices with limited hardware resources. I implemented optimizations, such as processing one frame out of 30 and using a circular buffer, to reduce resource consumption.

3. **Gaze detection in variable lighting conditions:** Precise gaze direction detection was affected by lighting variations. It was necessary to apply filters and adjust algorithm parameters to improve robustness.

4. **Multimedia data synchronization:** Precise correlation between video, audio stream and detected events was a complex task. I used synchronized temporal markings to link alerts generated by exact moments in the video stream.

5. **Graphical interface complexity:** Developing an intuitive and functional graphical interface was a challenge, especially regarding logical organization of components and their synchronization with background processes.

6. **False positive management:** During testing, I observed a high rate of false positives, especially in object detection. It was necessary to adjust algorithms and implement additional filters to improve accuracy.

7. **Code documentation and modularization:** Structuring the project in a modular way and clear documentation of each module required additional time, but were essential for maintenance and subsequent application extension.

8. **Testing and validation:** Testing the application in real scenarios was difficult, especially in simulating varied usage conditions, such as different camera angles, poor lighting or background noise.

9. **Dependency management:** Correct installation and configuration of all necessary libraries and packages were sometimes problematic, especially due to incompatibilities between versions.

8 Conclusions

The *Anti-Plagiarism Monitoring System for Exams* project successfully achieved the proposed objectives, offering an innovative and efficient solution for monitoring candidate behavior during exams.

8.1 Objective fulfillment

- **Candidate gaze detection:** The system uses advanced image processing technologies to analyze gaze direction and signal deviations from normal behavior. This functionality was successfully implemented, offering precise and real-time detection.

- **Identification of unauthorized objects:** Through integration of YOLOv8 models, the application detects devices such as mobile phones and smart watches, contributing to preventing fraud attempts.

- **Recording and archiving:** The system allows recording of exam sessions, offering the possibility to subsequently analyze candidate behavior and archive evidence.

- **Report generation:** The application generates detailed reports in HTML, CSV and JSON formats, facilitating analysis and incident documentation.

- **Intuitive interface:** The graphical interface developed with PyQt5 is easy to use and offers quick access to all system functionalities.

8.2 Utility and relevance

Plagiarism and fraud attempts represent major problems in modern educational systems. The proposed application offers a current and relevant solution, using cutting-edge technologies to combat these problems. By focusing on candidates' physical behavior, the system brings a significant advantage compared to existing solutions, which focus mainly on screen activity monitoring.

8.3 Performance and optimizations

The system was optimized to function efficiently even on limited hardware, using techniques such as processing one frame out of 30 and reducing resource consumption. Conducted tests demonstrated a high accuracy rate in detecting suspicious behavior and forbidden objects, with a low false positive rate.

8.4 Impact and relevance

The application's impact is significant, contributing to maintaining academic integrity and creating a fair exam environment. Through the use of artificial intelligence and image processing technologies, the system can be extended and adapted for various educational and professional contexts.

8.5 Advantages compared to other similar solutions

Compared to other existing solutions, our application offers the following advantages:

- **Focus on physical behavior:** Unlike solutions that monitor only screen activity, this system analyzes gaze direction and detects unauthorized objects.
- **Reduced costs:** Local implementation significantly reduces costs per candidate, compared to commercial solutions like ProctorU.
- **Platform independence:** The application functions independently of browser or other platforms, offering increased flexibility.

- **Detailed reports:** Automatic report generation in multiple formats facilitates subsequent analysis and incident documentation.

8.6 Multi-platform compatibility

Our application runs on both Windows and Linux (Ubuntu), offering users flexibility depending on their preferred operating system.

On Windows, the application benefits from GPU support through CUDA, which allows using the NVIDIA graphics card for accelerating video processing and object detection. This is possible due to how NVIDIA drivers are stored and managed on Windows.

On Linux, although the application functions correctly, GPU integration through CUDA has not yet been fully implemented. This limitation is due to differences in NVIDIA driver management on Linux compared to Windows. Currently, processing on Linux is performed exclusively on CPU, but implemented optimizations ensure acceptable performance even in these conditions.

This multi-platform compatibility makes the application accessible to a wide spectrum of users, regardless of the operating system used.

8.7 Possible improvements

Although the current application offers advanced functionalities for anti-plagiarism monitoring, there are several aspects that can be improved or extended to increase its efficiency and utility:

1. **Integration with LMS platforms:** Adding support for direct integration with learning management systems (LMS), such as Moodle, Blackboard or Canvas, to facilitate exam and report management.

2. **Multi-camera support:** Extending the application to allow simultaneous use of multiple cameras, offering a more complete perspective on candidate behavior.

3. **Advanced audio monitoring:** Developing a more sophisticated audio processing module, which detects suspicious conversations or use of unauthorized audio devices.

These improvements could significantly increase the application's value and would extend its usage area in various educational and professional contexts.

Recent research in the field of smart watches has proposed ”a categorization of smart watch use in the health sector in 3 key functional domains: monitoring, guidance and prediction”[18], concepts that could be adapted for monitoring candidate behavior. Also, studies have demonstrated the efficiency of ”smart watch-based frameworks for real-time mobility assessment and monitoring”[6], suggesting the possibility of integrating these technologies in future versions of this system to improve accuracy in detecting suspicious behavior. [3]

In conclusion, the project represents a significant contribution to the anti-plagiarism monitoring field, offering a modern, efficient and adaptable solution to current educational system needs.

Bibliography

[1] Branislav Sredojev, Dragan Samardzija, and Dragan Posarac. "WebRTC technology overview and signaling solution design and implementation". In: *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (2015). Ultima accesare: 6 mai 2025, pp. 1006–1011. DOI: 10.1109/MIPRO.2015.7160422.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Ultima accesare: 6 mai 2025. Cambridge, MA: MIT Press, 2016. ISBN: 978-0262035613. URL: https://www.deeplearningbook.org/.

[3] Jeffrey Brainard and Jia You. "What a massive database of retracted papers reveals about science publishing's 'death penalty'". In: *Science* 362.6413 (2018). Ultima accesare: 6 mai 2025, pp. 390–393. DOI: 10.1126/science.aav8384.

[4] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767* (2018). Ultima accesare: 6 mai 2025.

[5] Tianyi Yang et al. "TLS/SSL: A Comprehensive Review of Public Key Infrastructure, Cipher Suites, and SSL/TLS/DTLS". In: *Journal of Network and Computer Applications* 108 (2018). Ultima accesare: 6 mai 2025, pp. 1–16. DOI: 10.1016/j.jnca.2018.01.012.

[6] Matin Kheirkhahan et al. "A smartwatch-based framework for real-time and online assessment and mobility monitoring". In: *Journal of Biomedical Informatics* 89 (2019). Ultima accesare: 6 mai 2025, pp. 29–40. DOI: 10.1016/j.jbi.2018.11.003.

[7] Najla Aiman Nazari et al. "Detection of Active Mobile Phone in Exam Hall". In: *International Journal of Recent Technology and Engineering* 8.4 (2019). Ultima accesare: 6 mai 2025, pp. 8432–8437. DOI: 10.35940/ijrte.D8894.118419.

[8] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020). Ultima accesare: 6 mai 2025, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[9] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th ed. Ultima accesare: 6 mai 2025. Hoboken, NJ: Pearson, 2020. ISBN: 978-0134610993.

[10] Yasmine Alem, Bachir Boudraa, and Adel Merabet. “A Novel Deep Learning-based Online Proctoring System using Face Recognition, Eye Blinking, and Object Detection Techniques”. In: *International Journal of Educational Technology in Higher Education* 18.1 (2021). Ultima accesare: 6 mai 2025, pp. 1–19. DOI: 10.1186/s41239-021-00278-7.

[11] Nimesha Dilini et al. “Cheating Detection in Browser-based Online Exams through Eye Gaze Tracking”. In: *6th International Conference on Information Technology Research (ICITR)* (2021). Ultima accesare: 6 mai 2025, pp. 1–6. DOI: 10.1109/ICITR54349.2021.9657277.

[12] Mohammed Hasan, Himanshu Kashyap, and R. Rajesh. “Face Detection with OpenCV and Deep Learning: A Comprehensive Review”. In: *International Journal of Computer Vision and Image Processing* 11.2 (2021). Ultima accesare: 6 mai 2025, pp. 51–67. DOI: 10.4018/IJCVIP.2021040103.

[13] Gabriela Pelican. “Plagiatul - o plagă academică endemică”. In: *Revista Română de Studii Juridice* 5.2 (2021). Ultima accesare: 6 mai 2025, pp. 43–59.

[14] Olena Zimba and Armen Yuri Gasparyan. “Plagiarism detection and prevention: a primer for researchers”. In: *Rheumatology International* 41.11 (2021). Ultima accesare: 6 mai 2025, pp. 2045–2055. DOI: 10.1007/s00296-021-04976-3.

[15] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. “Object detection using YOLO: challenges, architectural successors, datasets and applications”. In: *Multimedia Tools and Applications* 81.4 (2022). Ultima accesare: 6 mai 2025, pp. 28963–28989. DOI: 10.1007/s11042-022-13644-y.

[16] Glenn Jocher et al. *Ultralytics YOLOv8 Documentation*. https://github.com/ultralytics/ultralytics. Ultima accesare: 6 mai 2025. 2023.

[17] El-Sayed M. El-Kenawy et al. “Drowsiness Detection using Dlib Facial Landmarks in Surveillance Systems”. In: *IEEE Access* 11 (2023). Ultima accesare: 6 mai 2025, pp. 45983–45997. DOI: 10.1109/ACCESS.2023.3267452.

[18] Rabee Moshawrab, Konstantin Aal, and Volker Wulf. “The Value of Smartwatches in the Health Care Sector for Monitoring, Nudging, and Predicting: Viewpoint on 25 Years of Research”. In: *JMIR mHealth and uHealth* 11 (2023). Ultima accesare: 6 mai 2025, e45413. DOI: 10.2196/45413.

[19] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. https://pytorch.org/. Ultima accesare: 6 mai 2025. 2023.

[20] Proctorio Inc. *Proctorio: Remote Proctoring Solutions*. https://proctorio.com/. Ultima accesare: 6 mai 2025. 2023.

[21] ProctorU Inc. *ProctorU: Online Proctoring Services*. https://www.proctoru.com/. Ultima accesare: 6 mai 2025. 2023.

[22] Python Software Foundation. *Python datetime Module Documentation*. https://docs.python.org/3/library/datetime.html. Ultima accesare: 6 mai 2025. 2023.

[23] Python Software Foundation. *Python logging Module Documentation*. https://docs.python.org/3/library/logging.html. Ultima accesare: 6 mai 2025. 2023.

[24] Respondus Inc. *Respondus LockDown Browser Documentation*. https://web.respondus.com/lockdownbrowser/. Ultima accesare: 6 mai 2025. 2023.

[25] Riverbank Computing Limited. *PyQt5 Documentation*. https://www.riverbankcomputing.com/software/pyqt/intro. Ultima accesare: 6 mai 2025. 2023.

[26] Honorlock Research Team. "Detecting Cell Phones With Online Proctoring". In: *Honorlock Digital Proctoring Research Reports* 3.2 (2023). Ultima accesare: 6 mai 2025, pp. 1–8. URL: https://honorlock.com/research/proctoring-cell-phone-detection.

[27] V7 Labs Research Team. "YOLO Algorithm for Object Detection Explained". In: *Computer Vision Quarterly* 7.2 (2023). Ultima accesare: 6 mai 2025, pp. 23–35.