Departamentul Automatică şi Informatică Industrială

Facultatea Automatică şi Calculatoare

Universitatea Nationala de Stiinta si Tehnologie

POLITEHNICA din Bucureşti

Splaiul Independenţei 313, 060042, Bucureşti, România

Sala ED 412, Tel. 021/402.92.69

www.aii.pub.ro, email: secretariat.aii@upb.ro

Sesiunea de Comunicări Ştiinţifice Studenţeşti

Ediţia 2025

Sistem de Monitorizare Anti-Plagiat pentru Examene

Autor: Valentin PLETEA-MARINESCU, Facultatea de Automatică și Calculatoare, anul 3, Grupa 332AB

Adresă email: pletea.valentin2003@gmail.com

Îndrumător ştiinţific: Conf. dr. ing. Ștefan Alexandru MOCANU

Cuprins

1 Introducere

1.1 Contextul și relevanța temei

Plagiatul reprezintă una dintre problemele grave ale sistemului de învățământ, iar, de când cu apariția și dezvoltarea inteligenței artificiale, lucrurile s-au accentuat. Alegerea acestei teme constă în oportunitatea de a oferi șanse egale tuturor participanților, în cadrul unui examen, iar acest sistem de anti-plagiat, propus în acest proiect, reprezintă o soluție inovatoare, prin combinarea tehnicilor moderne de procesare a imaginilor și inteligență artificială[9], care își propune să monitorizeze comportamentul candidaților și să semnaleze posibile tentative de fraudă.

Utilizarea telefoanelor mobile ca instrument de înșelătorie în sălile de examinare a crescut considerabil în ultimii ani, creând o povară suplimentară supraveghetorilor în asigurarea integrității examenelor[7]. Mai mult, studii recente în domeniul integrității academice arată că plagiatul este „o abatere etică care afectează calitatea, lizibilitatea și credibilitatea" rezultatelor academice, fiind esențială dezvoltarea de soluții pentru combaterea practicilor inacceptabile[14].

Conform studiului realizat de Gabriela Pelican[13], plagiatul afectează absolut toate nivelurile educaționale, iar, în cazul României, lucrurile nu merg foarte bine la capitolul acesta, autoarea menționând că "România prezintă cea mai ridicată rată a plagiatelor descoperite pe tărâmul Uniunii, și anume 26.1% din totalitatea operelor verificate, valoare aproape dublă în comparație cu media europeană, precum și aceea că incidența cazurilor de plagiat este cu mult mai ridicată în statele din regiunea estică a Europei, țări ale căror niveluri de trai și de educație sunt considerabil inferioare raportat la țările vestice."

De asemenea, un alt studiu[3], publicat de cei de la Science, ne relatează că România este țara cu cel mai mare număr de articole științifice eliminate din cuprinsul publicațiilor de specialitate ca urmare a nerespectării normelor de conduită în cercetarea științifică, prin raportare la totalul fondurilor alocate pentru cercetare și de asemenea, ocupă locul secund în topul țărilor cu cel mai mare număr de articole retrase raportat la totalul articolelor publicate, în urma descoperirii plagierii.

1.2 Obiectivele proiectului

Acest proiect își propune să ofere un sistem inovator pentru monitorizarea examenelor, cu următoarele obiective specifice:

1. **Detecția privirii candidatului:** Semnalarea și înregistrarea situațiilor în care candidatul privește în alte direcții precum stânga, dreapta sau jos, decât în propriul ecran al monitorului sau în propria foaie, în cazul unui examen scris.
2. **Identificarea obiectelor neautorizate:** Detectarea dispozitivelor precum ceasuri inteligente sau telefoane, care pot fi folosite pentru a obține informații, ceea ce ar duce la fraudarea respectivului examen.
3. **Înregistrare și arhivare:** Realizarea unei înregistrări video în care vor fi capturate și arhivate sesiunile de examen pentru o analiză ulterioară, evidențiind tentativele probabile de fraudă.
4. **Generarea rapoartelor:** Crearea rapoartelor detaliate în formate HTML, CSV și JSON, contribuind astfel la o analiză mai clară și mai sigură pentru evaluatori.
5. **Interfață intuitivă:** Dezvoltarea unei interfețe grafice intuitive, care să permită supraveghetorilor utilizarea tuturor funcționalităților menționate anterior.

1.3 Avantajele abordării propuse

În general, multe soluții de combatere a plagiatului se concentrează mai mult pe monitorizarea ecranului candidatului, dar acest sistem este centrat pe comportamentul fizic al acestuia. Consider că, prin urmărirea direcției privirii și detectarea obiectelor neautorizate, sistemul poate identifica indicii ale comportamentului fraudulos, oferind dovezi concrete care pot fi analizate ulterior. Acest lucru poate contribui semnificativ la menținerea integrității academice, fără a mai fi nevoie să se creeze acea atmosferă de supraveghere excesivă.

Este normal ca respectivul sistem să ofere, pe lângă rezultatele remarcabile, și falsuri pozitive, din cauza unor factori precum iluminarea sălii de examen sau mișcări naturale ale capului, fără intenția de a privi în altă direcție. Astfel, prin intermediul funcționalităților de înregistrare video și semnalare în timp real a comportamentelor posibil frauduloase, poate să existe, după

terminarea respectivului examen, o discuție între profesor și student pe baza onestității sale și de a ajunge la rezultatul final.

Tehnologia avansează pe zi ce trece, iar diversele tentative de plagiat se diversifică automat. Așadar, este absolut firesc ca și sistemele care încearcă, prin diverse mijloace, să combată fraudarea examenelor, să avanseze și ele.

2 Analiza domeniului și soluții existente

2.1 Soluții similare existente pe piață

Au apărut pe piață, de-a lungul timpului, diferite soluții de monitorizare și combatere a fraudei în cadrul examenelor. În urma cercetării, am identificat trei sisteme principale care propun soluții similare cu acest proiect: ProctorU[21], Proctorio[20] și Respondus Lockdown Browser[24].

Recent, sisteme moderne de supraveghere online precum Honorlock au implementat tehnologii avansate care detectează utilizarea telefoanelor mobile „printr-o combinație de instrumente de supraveghere” și pot determina când participanții încearcă să folosească dispozitive mobile neautorizate pentru a accesa conținut în timpul examenelor[26]. De asemenea, cercetările recente în domeniul eye-tracking au propus „o soluție nouă de detectare a înșelăciunii în examene online prin utilizarea tehnologiei de urmărire a privirii”[11], demonstrând eficacitatea acestei abordări în detectarea comportamentelor suspecte.

2.1.1 ProctorU

ProctorU este una dintre cele mai utilizate platforme în supravegherea examenelor online. Se bazează atât pe monitorizarea ecranului candidatului, cât și pe comportamentul acestuia.

Puncte forte	Limitări
• Verificarea identității candidatului prin legitimație sau act de identitate • Supravegherea umană în timp real • Monitorizarea completă a activității pe calculator • Detectarea comportamentului suspect prin AI	• Costuri ridicate (15-25 USD per candidat) • Necesită conexiune stabilă la internet • Mai puțin adecvat pentru examenele fizice • Probleme potențiale de confidențialitate

Tabela 1: Analiza comparativă a sistemului ProctorU

Platforma înregistrează fiecare secundă, se asigură că nu există alte tab-uri deschise, cu excepția aceluia în care se susține respectivul examen, nu permite anumite combinații de taste din tastatură pentru funcții precum print, copy sau paste și, prin intermediul unei camere web, platforma poate supraveghea candidatul eficient.

2.1.2 Proctorio

Un alt sistem similar este Proctorio, asemănător cu ProctorU, dar care este complet automatizat, întrucât se bazează doar pe tehnologii de AI și învățare automată pentru detectarea posibilelor tentative de fraudă, fără a mai avea nevoie de asistență umană.

Puncte forte	Limitări
• Instalare și configurare simplă prin extensie browser • Funcționalități de blocare a navigării web • Algoritmi care reduc necesitatea intervenției umane • Costuri mai reduse decât ProctorU	• Rată ridicată de falsuri pozitive • Dificultate în detectarea utilizării dispozitivelor secundare • Funcționează doar pe Google Chrome • Probleme de confidențialitate

Tabela 2: Analiza comparativă a sistemului Proctorio

Monitorizarea se bazează pe recunoaștere facială, analizând comportamentul prin intermediul mișcării capului sau a ochilor. Sistemul se ocupă și de monitorizarea ecranului, observă dacă

există alte tab-uri deschise și dacă sunt pornite în fundal aplicații care ar putea facilita fraudarea examenului.

2.1.3 Respondus Lockdown Browser

Respondus Lockdown Browser reprezintă o abordare diferită, întrucât se concentrează pe crearea unui mediu securizat pentru examen, prin blocarea accesului candidatului la alte aplicații și resurse.

Conform documentației sale oficiale[24], această soluție se integrează cu sisteme de management al învățământului (LMS), precum Canvas, Blackboard, Moodle, Schoology și Brightspace.

Puncte forte	**Limitări**
• Împiedicarea accesului la alte programe și resurse • Integrare bună cu platforme educaționale (LMS) • Costuri mai reduse comparativ cu alte soluții • Ușor de implementat la nivel instituțional	• Incapacitatea de a detecta dispozitive secundare • Nu monitorizează comportamentul fizic al candidatului • Funcționalitate restrânsă pe dispozitivele mobile • Incompatibilitate cu programe specifice (IDE-uri, etc.)

Tabela 3: Analiza comparativă a sistemului Respondus Lockdown Browser

2.2 Aspecte multimedia relevante

2.2.1 Protocoale și transmisie de date

În cadrul aspectelor legate de multimedia, întâlnim protocoale care asigură transmisia datelor sigură și optimizată:

- **WebRTC** pentru comunicația audio-video în timp real. Acest protocol este utilizat pentru a transmite fluxul video capturat de la webcam către sistemul de procesare, asigurând o latență minimă și o calitate optimă a transmisiei[1]. În proiectul nostru, WebRTC este folosit pentru a permite monitorizarea în timp real a candidaților, oferind o conexiune

stabilă și eficientă.

- **TLS/SSL** pentru criptarea întregului flux de date. Acest strat de securitate este esențial pentru a proteja datele transmise între client și server, prevenind accesul neautorizat sau interceptarea informațiilor sensibile[5]. În contextul proiectului nostru, TLS/SSL este utilizat pentru a securiza atât fluxul video, cât și datele generate, cum ar fi rapoartele de încălcări.

  Cu toate ca versiunea actuală a proiectului nostru se bazează pe procesare locală fără componente de comunicație în rețea, implementarea acestor tehnologii ar reprezenta o direcție valoroasă de dezvoltare viitoare, permițând monitorizarea la distanță a candidaților.

2.2.2 Procesare video

În ceea ce privește procesarea video, fluxul capturat de la webcam este procesat în timp real pentru a detecta comportamente suspicioase. Pentru salvarea înregistrărilor, sistemul nostru utilizează formate standard precum MP4, care incorporează codecul H.264, asigurând o eficiență bună în utilizarea spațiului de stocare[2].

Cercetările recente în domeniul sistemelor de supraveghere a examenelor au demonstrat eficiența detecției faciale și a urmăririi ochilor pentru identificarea comportamentelor suspecte[10]. Implementarea noastră se bazează pe aceste cercetări, integrând biblioteci precum `dlib` și `OpenCV` pentru a detecta poziția feței și direcția privirii, oferind astfel o soluție robustă pentru identificarea comportamentelor suspicioase.

Acest sistem se concentrează în principal pe analiza vizuală, detectând atât direcția privirii candidatului cât și prezența obiectelor neautorizate precum telefoanele mobile și ceasurile inteligente, elemente care reprezintă cele mai frecvente mijloace de fraudare în examene.

2.2.3 Watermarking

În soluțiile de tip Lockdown Browser se aplică watermark-uri vizibile sau invizibile asupra conținutului afișat pe ecran, cu scopul de a preveni capturarea neautorizată a informațiilor de examen. În proiectul nostru, watermark-urile sunt folosite pentru a marca fluxul video înregistrat cu informații precum data și ora reală, dar și valorile H și V, care indică in ce limite se afla direcția privirii candidatului. Acest tip de marcaj contribuie la asigurarea autenticității și

integrității datelor înregistrate.

2.2.4 Sincronizarea evenimentelor

Pentru a asigura o corelare precisă între cadrele video și evenimentele detectate, proiectul nostru utilizează marcaje temporare. Acestea sunt generate în momentul capturării fiecărui cadru video și sunt utilizate pentru a lega alertele generate de sistem cu momentul exact al încălcării. Această abordare permite consultarea ulterioară a înregistrărilor video și verificarea eficientă a rapoartelor generate, oferind astfel dovezi concrete pentru evaluarea comportamentului candidatului în timpul examenului.

2.2.5 Optimizări pentru resurse limitate

Pentru a permite rularea eficientă a sistemului pe dispozitive cu resurse imitate, in acest sistem au fost implementate mai multe optimizări:

- Procesarea unui cadru din 30 pentru detectarea obiectelor, reducând astfel consumul de resurse.
- Utilizarea unui buffer circular pentru gestionarea eficientă a cadrelor video.
- Integrarea unui sistem de cache pentru alertele generate, evitând duplicarea mesajelor de alertă.

2.3 Avantajele soluției propuse față de cele existente

Sistemul de monitorizare anti-plagiat dezvoltat în cadrul acestui proiect oferă numeroase beneficii comparative față de alternativele disponibile în prezent pe piață:

1. **Monitorizare completă a comportamentului fizic** - În timp ce majoritatea soluțiilor comerciale (precum Proctorio sau Respondus) se limitează la supravegherea activității de pe ecranul calculatorului, sistemul nostru observă mișcările și gesturile candidatului. Această abordare permite detectarea unor tactici de fraudare care altfel ar trece neobservate, cum ar fi consultarea notițelor fizice sau comunicarea cu alte persoane prin gesturi. Sistemul

analizează direcția privirii, orientarea capului și alte indicii fizice care pot semnala intenții frauduloase.

2. **Identificarea precisă a dispozitivelor electronice neautorizate** - Prin implementarea algoritmilor YOLOv8 și a tehnicilor moderne de recunoaștere a obiectelor, aplicația noastră poate identifica cu acuratețe ridicată telefoane mobile, ceasuri inteligente și alte dispozitive folosite frecvent pentru înșelăciune. Această capabilitate depășește semnificativ funcționalitățile oferite de Respondus Lockdown Browser, care nu poate detecta utilizarea dispozitivelor secundare, și chiar și ale ProctorU, care se bazează în mare măsură pe supraveghetori umani pentru această sarcină.

3. **Sistem avansat de raportare și documentare** - Aplicația generează automat rapoarte detaliate în formate versatile (HTML pentru vizualizare ușoară, CSV pentru analiză în Excel și JSON pentru integrare cu alte sisteme). Aceste rapoarte includ capturi de ecran cu momentele suspicioase, marcaje temporale precise și descrieri clare ale incidentelor, oferind evaluatorilor toate informațiile necesare pentru a lua decizii documentate. Sistemul păstrează astfel un istoric complet al sesiunii de examinare, permițând verificarea ulterioară și reducând disputele legate de posibilele sancțiuni.

4. **Independență tehnologică și flexibilitate în implementare** - Acest sistem funcționează ca aplicație independentă, fără a fi constrâns de limitările specifice browserelor web sau platformelor online. Această arhitectură autonomă elimină dependențele externe și simplifică procesul de instalare și configurare la nivel instituțional. Natura sa independentă permite utilizarea pe o gamă largă de dispozitive și sisteme de operare, oferind instituțiilor educaționale libertatea de a implementa soluția fără a fi nevoite să adopte tehnologii sau servicii adiționale. Flexibilitatea sistemului facilitează adaptarea rapidă la cerințele specifice ale diferitelor tipuri de examene și contexte academice.

5. **Eficiență economică și scalabilitate** - Implementarea locală a sistemului oferă avantaje semnificative din perspectiva costurilor pe termen lung. În timp ce multe soluții comerciale existente operează pe baza unui model de tarifare per student per examen, care poate deveni prohibitiv pentru instituțiile cu număr mare de studenți sau examene frecvente, aceasta propunere de sistem anit-plagiat necesită doar investiția inițială în infrastructura hardware existentă. Acest model scalabil permite instituțiilor educaționale să monitorizeze un număr nelimitat de examene fără costuri suplimentare, reducând costul efectiv per

student cu fiecare sesiune organizată, iar această abordare face tehnologia de monitorizare anti-plagiat accesibilă inclusiv pentru instituțiile cu resurse financiare limitate.

6. **Confidențialitate sporită a datelor studenților** - Prin procesarea locală a informațiilor, acest prototip elimină riscurile asociate cu transmiterea datelor către servere externe. Acest aspect este deosebit de important în contextul reglementărilor stricte privind protecția datelor personale, oferind instituțiilor control complet asupra informațiilor sensibile colectate în timpul examenelor.

Combinația acestor avantaje face acest prototip de sistem anti-plagiat să reprezinte o soluție inovatoare și eficientă pentru asigurarea integrității academice, adaptată nevoilor actuale ale instituțiilor de învățământ.

3 Arhitectura sistemului anti-plagiat

Sistemul anti-plagiat dezvoltat reprezintă o soluție software complexă destinată monitorizării candidaților în timpul examenelor, cu scopul de a detecta și preveni comportamentele frauduloase. Arhitectura sistemului a fost concepută conform principiilor moderne de inginerie software, punând accent pe modularitate, extensibilitate și separarea clară a responsabilităților.

Baza dezvoltării sistemului este reprezenta de programarea orientată pe obiecte, care a permis structurarea aplicației în componente interconectate, fiecare având un rol bine definit în procesul de monitorizare. Această abordare facilitează nu doar dezvoltarea și testarea independentă a modulelor, ci și extinderea ulterioară a funcționalităților prin adăugarea de noi componente.

Sistemul prelucrează imaginile video pe măsură ce sunt captate, urmărind încotro se uită candidații și identificând obiecte nepermise cum ar fi telefoanele mobile și ceasurile inteligente. Această urmărire se bazează pe programe informatice care pot „vedea” și „înțelege” imaginile, organizate într-o structură pe niveluri care ține separate funcțiile simple de cele mai avansate.

Pentru a ilustra structura și funcționarea sistemului, am elaborat trei diagrame UML complementare: diagrama de clase, diagrama secvențială si diagrama de activitate. Aceste reprezentări oferă perspective diferite asupra arhitecturii, de la organizarea statică a claselor până la fluxurile dinamice de control și date.

3.1 Diagrama de clase

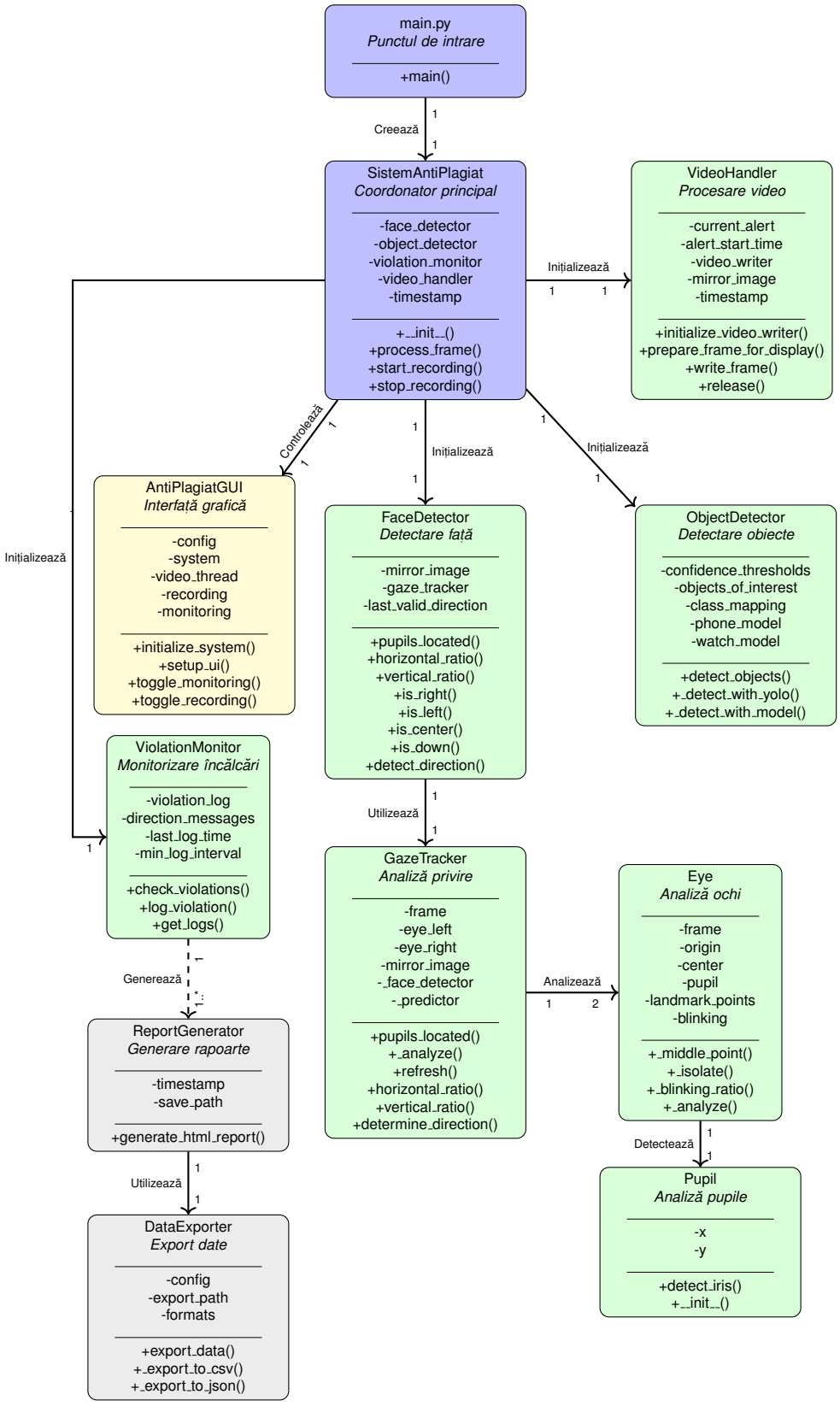


Figura 1: Diagrama de clase a sistemului Anti-Plagiat cu cardinalitate relațională

Diagrama de clase evidențiază organizarea ierarhică a modulelor sistemului anti-plagiat și relațiile dintre acestea. Punctul de intrare `main.py` creează instanța clasei `SistemAntiPlagiat`, care ocupă poziția centrală în arhitectură, coordonând toate celelalte componente. Săgeata de la `main.py` către `SistemAntiPlagiat` reprezintă această inițializare fundamentală de la care pornește întregul sistem.

`SistemAntiPlagiat` inițializează patru componente esențiale, reprezentate prin săgețile divergente: `AntiPlagiatGUI` pentru interfața grafică, `VideoHandler` pentru procesarea video, `FaceDetector` pentru analiza facială și `ObjectDetector` pentru identificarea obiectelor. Această structură reflectă metodele din constructorul clasei `SistemAntiPlagiat`, unde sunt create instanțele acestor obiecte.

Relația dintre `FaceDetector` și `GazeTracker` este una de creare, unde detectorul facial inițializează și utilizează modulul de urmărire a privirii. Această structură tip Facade simplifică interfața cu subsistemul complex de analiză a privirii. Observăm apoi cum `GazeTracker` utilizează `Eye` pentru izolarea regiunii oculare, care la rândul său folosește `Pupil` pentru localizarea precisă a pupilei. Această succesiune de delegări reflectă nivelul crescând de specializare a analizei.

`ViolationMonitor`, inițializat de `SistemAntiPlagiat`, agregă informațiile despre direcția privirii și obiectele detectate pentru a identifica comportamente suspecte. Când detectează încălcări, acestea sunt transmise către `ReportGenerator` care utilizează `DataExporter` pentru a salva rapoartele în diverse formate.

Această organizare reflectă principiul separării responsabilităților, fiecare clasă având un rol specific în analiza și procesarea datelor video pentru identificarea tentativelor de plagiat.

3.2 Diagrama secvențială

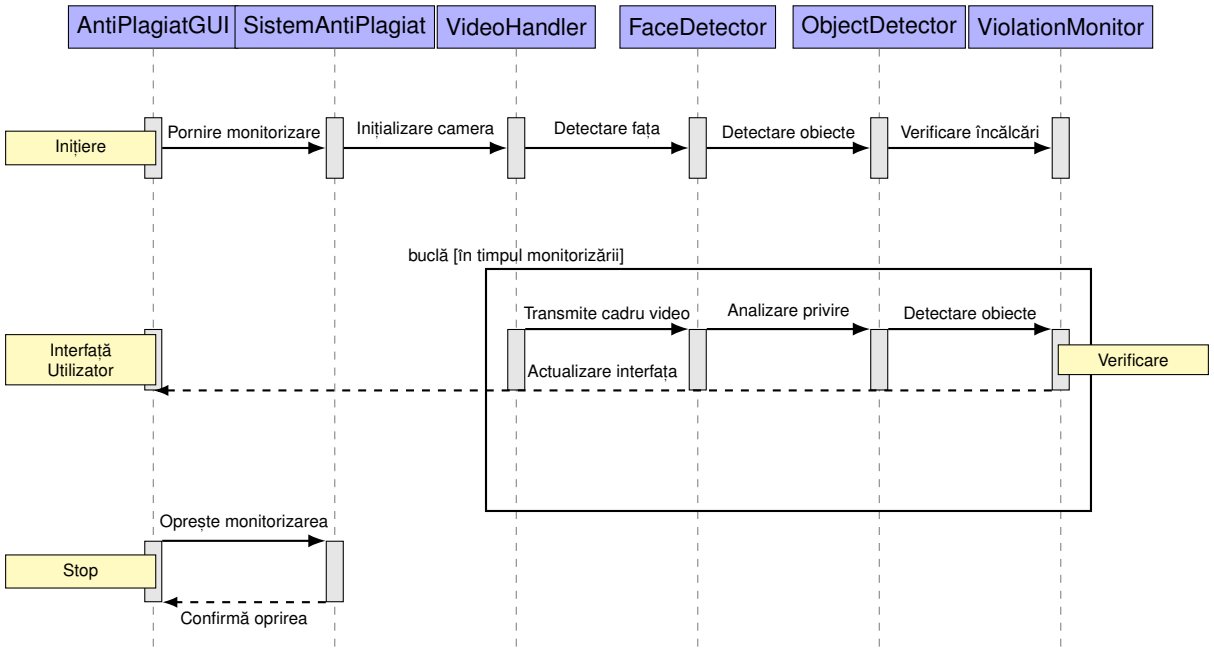


Figura 2: Diagrama secvențială a sistemului Anti-Plagiat

Diagrama secvențială ilustrează interacțiunea temporală dintre componentele principale ale sistemului anti-plagiat. Secvența începe când utilizatorul inițiază monitorizarea prin interfața `AntiPlagiatGUI`, care transmite comanda către `SistemAntiPlagiat`. Acesta, la rândul său, inițializează camera prin `VideoHandler` și pornește procesele de detecție facială, detectare obiecte și monitorizare a încălcărilor.

Bucla principală de procesare, evidențiată în diagramă, arată cum `VideoHandler` capturează cadre video care sunt apoi analizate de `FaceDetector` pentru poziția feței și direcția privirii, urmată de `ObjectDetector` pentru identificarea obiectelor interzise. `ViolationMonitor` primește rezultatele ambelor analize și determină dacă există încălcări, iar interfața grafică este actualizată corespunzător cu noile informații.

Secvența se încheie cu oprirea monitorizării, când utilizatorul trimite comanda de stop prin interfață, iar sistemul confirmă încheierea procesului de monitorizare. Această succesiune de mesaje și activări ilustrează fluxul complet de control și date în sistem, de la inițiere până la încheiere, evidențiind modul în care toate componentele colaborează pentru a realiza monitorizarea în timp real a candidatului.

3.3 Diagrama de activitate

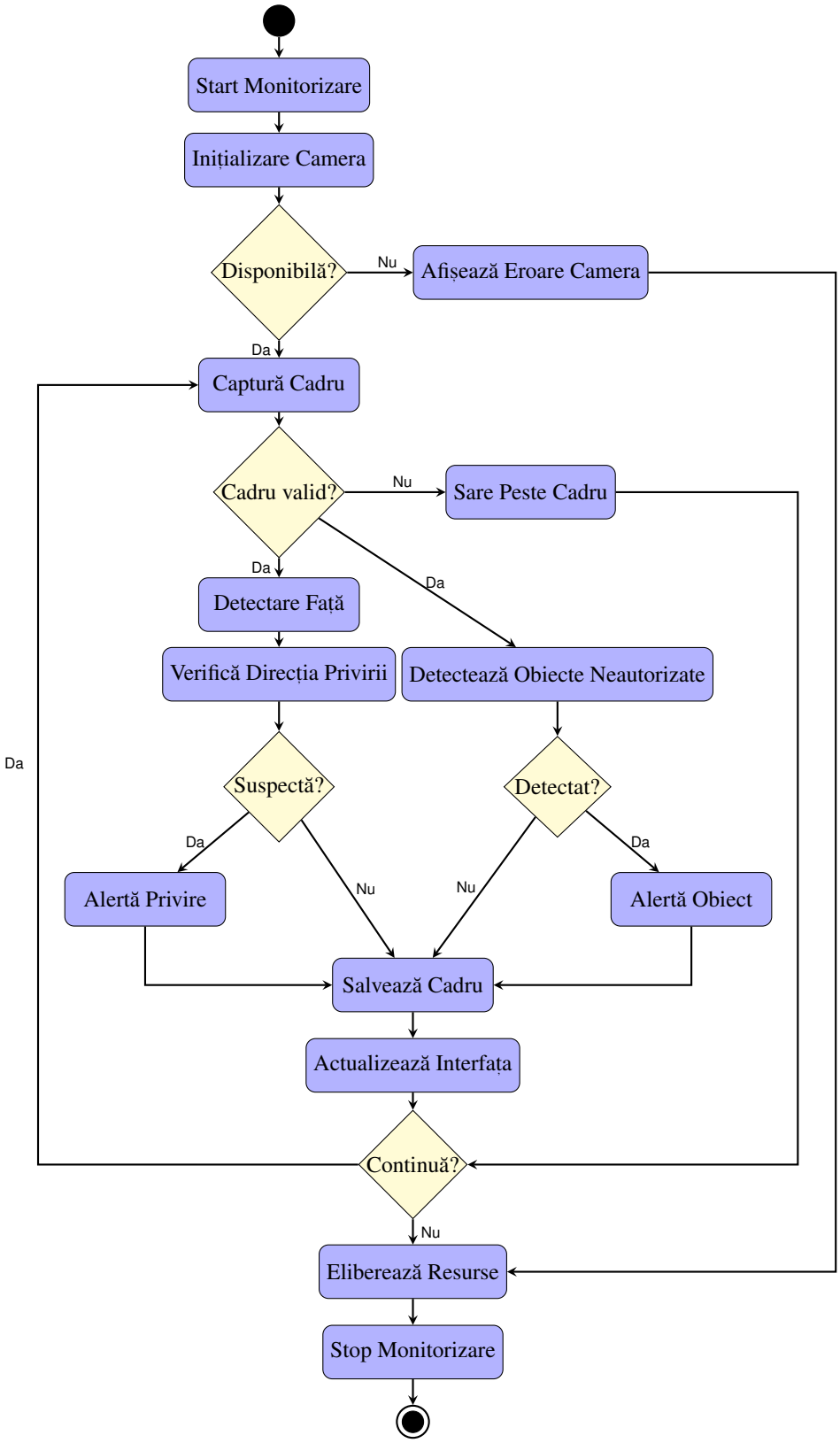


Figura 3: Diagrama de activitate a sistemului Anti-Plagiat

Diagrama de activitate prezintă fluxul operațional al sistemului, evidențiind pașii procesării și punctele de decizie. Din starea inițială `Start Monitorizare`, activată când utilizatorul apasă butonul corespunzător, sistemul trece la `Inițializare Camera`, unde configurează parametrii de captură și verifică disponibilitatea camerei web. Această tranziție corespunde apelului metodei `toggle_monitoring()` din clasa principală `AntiPlagiatGUI`.

După inițializare, sistemul verifică dacă respectiva cameră este disponibilă prin punctul de decizie `Disponibilă?`. În cazul unei erori, sistemul afișează un mesaj de eroare prin procesul `Afișează Eroare Camera` și avansează direct la eliberarea resurselor. Această verificare este implementată în metoda `VideoProcessingThread.run()`, unde se testează conexiunea la camera web.

Dacă aceasta este funcțională, procesul continuă cu `Captură Cadru`, unde se preiau cadre de la camera web. Sistemul validează apoi fiecare cadru capturat prin decizia `Cadru valid?`, care verifică dacă imaginea a fost obținută corect. Cadrele invalide sunt gestionate prin procesul `Sare Peste Cadru`, care le ignoră și continuă fluxul. Această verificare este implementată prin condiția `if ret and frame is not None` din modulul de procesare video.

După validarea cadrului, analiza se ramifică în două fluxuri paralele independente:

1. În primul flux, sistemul execută `Detectare Față`, unde `FaceDetector` localizează fața candidatului în imagine, urmată de `Verifică Direcția Privirii`, care analizează poziția reală a pupilelor pentru a determina în ce direcție privește candidatul, utilizând metoda `face_detector.detect_direction()`. La punctul de decizie `Suspectă?`, se evaluează dacă direcția privirii indică un comportament suspect (privire laterală sau în jos).

2. În paralel, în al doilea flux, `Detectează Obiecte Neautorizate` folosește modelele YOLOv8 pentru a scana imaginea și identifica telefoane sau ceasuri inteligente prin metoda `object_detector.detect_objects()`. Decizia `Detectat?` verifică prezența acestor obiecte interzise.

Aceste două fluxuri operează complet independent, procesând același cadru valid dar analizând aspecte diferite - unul concentrându-se pe comportamentul candidatului și altul pe mediul înconjurător. Această paralelizare reflectă arhitectura modulară a sistemului, unde `FaceDetector` și `ObjectDetector` funcționează ca module separate.

Fiecare flux poate genera alerte specifice când detectează potențiale încălcări: `Alertă Privire`

pentru priviri suspecte și `Alertă Obiect` pentru obiecte neautorizate. Acestea sunt implementate prin apelul metodei `violation_monitor.log_violation()`, care înregistrează detaliile fiecărei încălcări.

Fluxurile converg apoi la activitatea `Salvează Cadru`, unde sistemul stochează imaginea procesată, urmată de `Actualizează Interfața`, care reîmprospătează display-ul cu datele analizate. Această actualizare este realizată prin semnalul `frame_ready` din PyQt5, care declanșează actualizarea interfeței grafice.

La punctul de decizie `Continuă?`, sistemul evaluează dacă monitorizarea trebuie să continue. În caz afirmativ, procesul revine la `Captură Cadru` pentru o nouă iterație. În caz negativ, sistemul trece la `Eliberează Resurse`, unde se închid conexiunile la cameră și se eliberează memoria, urmată de `Stop Monitorizare`, când procesul se încheie oficial.

Această arhitectură robustă, cu procesare paralelă, permite sistemului să analizeze simultan diferite aspecte ale situației de examinare, crescând eficiența și precizia în detectarea tentativelor de fraudă, chiar și când una dintre metodele de analiză (detecția feței sau a obiectelor) ar putea întâmpina dificultăți temporare.

4 Implementarea propriu-zisă

În cadrul implementării sistemului de anti-plagiat, a fost necesară utilizarea mai multor biblioteci specializate, fiecare având un rol diferit în funcționarea sa.

4.1 OpenCV

Am utilizat biblioteca OpenCV (`Open Source Computer Vision Library`), care a avut un rol principal în prelucrarea și procesarea imaginilor[12]. Cu ajutorul ei, am captat fluxul video de la camera web, prin intermediul funcției `VideoCapture`, care joacă rolul de interfață primară cu lumea fizică, capturând imaginile candidatului care sunt apoi analizate pentru a detecta comportamente suspecte.

Această bibliotecă mi-a oferit funcția `cvtColor`, cu care am reușit să transform imaginile color în tonuri de gri, ceea ce a dus la o eficiență mult mai mare pe partea de detectare facială, deoarece reduce complexitatea computațională, făcând referire doar la o singură valoare pentru

fiecare pixel, în loc de trei, și îmbunătățește performanțele algoritmilor de analiză a imaginii, care adesea funcționează mai bine cu imagini în tonuri de gri.

În cadrul detecției precise al pupilelor, am aplicat un filtru gaussian, cu scopul de a reduce zgomotul din imagini, apelând funcția `cv2.GaussianBlur`, oferită de respectiva bibliotecă, care este un filtru esențial în procesarea imaginilor. Parametrii acestei funcții sunt imaginea sursă, fiind regiunea izolată a ochiului, dimensiunea kernel-ului, în cazul nostru fiind de 7x7, deoarece s-a constatat, după mai multe teste, că oferă cele mai bune rezultate pentru detectarea pupilelor și este suficient de mare pentru a acoperi variațiile minore de intensitate din jurul pupilei, în comparație cu kernel-uri mai mici de 3x3 sau 5x5, care păstrează prea multe detalii fine și zgomot sau kernel-uri prea mari de 9x9 sau 11x11, care estompează prea mult imaginea, ducând la pierderea marginii pupilei și deviația standard a distribuției Gaussiene, acest parametru fiind setat la valoarea 0, însemnând că este calculată automat, folosind formula:

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

Pentru un kernel de dimensiune 7×7, această formulă devine:

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

Funcția `cv2.minMaxLoc` din OpenCV este un instrument esențial pentru detecția pupilelor. Această funcție identifică valorile minime și maxime dintr-o imagine, împreună cu coordonatele acestora. Ea retunează patru valori: valoarea minimă, valoarea maximă, coordonatele valorii minime și coordonatele valorii maxime. În cazul nostru, valoarea de interes este reprezentată de coordonatele valorii minime, deoarece reprezintă cel mai întunecat punct din regiunea ochiului, iar pupila este tipic cea mai întunecată parte a ochiului, deci această abordare eficientă permite localizarea rapidă a centrului pupilei. Această metodă funcționează deoarece, înainte de a aplica `minMaxLoc`, imaginea este pre-procesată de acel filtru gaussian, menționat mai sus, care reduce zgomotul și uniformizează valorile, făcând detectarea mai robustă la variații minore în iluminare sau textura irisului.

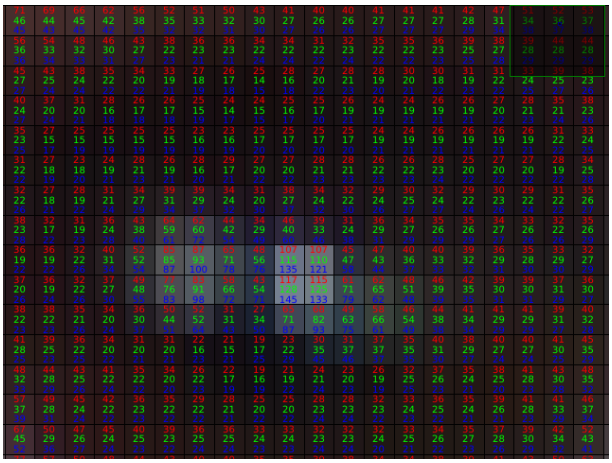


Figura 4: Valori RGB ale pixelilor din zona oculară

Imaginea de mai sus reprezintă o captură de ecran a valorilor RGB ale pixelilor din propria mea zona oculară, iar aceste rezultate au fost obtinute cu ajutorul rulării fisierului de testare `test_camera.py`.

Tot cu ajutorul lui OpenCV, am reușit să desenez elemente vizuale sugestive, cum ar fi dreptunghiuri în zona obiectelor detectate, folosind funcția `cv2.rectangle()`, și să afișez text pentru alerte, folosind `cv2.putText()`, iar salvarea și comprimarea înregistrărilor a fost realizată folosind `VideoWriter`.

De asemenea, am reușit să creez o mască, pentru a izola obiectul de interes, precum ochiul, utilizând funcții precum `fillPoly` și `bitwise_not`. Funcția `cv2.fillPoly` umple o zonă poligonală într-o imagine cu o culoare specifică, iar, în contextul acestui sistem, este folosită pentru a crea o mască pentru izolarea regiunii ochilor. Primește ca parametrii imaginea pe care se desenează, un array de puncte care definec poligonul, în cazul nostru fiind vorba de conturul ochiului, și culoarea de umplere, spre exemplu (0, 0, 0), care în RGB înseamnă culoare negru. Astfel, rezultatul este o mască în care regiunea ochiului este colorată în negru, iar restul imaginii rămâne alb.

Funcția `cv2.bitwise_not` inversează valorile de bit ale fiecărui pixel din imagine, fiind echivalentă cu operația de negare bitwise ($\sim$). Am folosit-o pentru a inversa masca creată, astfel încât regiunea ochiului să fie izolată din restul imaginii. Această funcție primește ca parametrii imaginea sursă și o mască opțională care determină pe ce regiune se aplică operația. Astfel, aceste două funcții lucrează împreună pentru a crea o "tăietură" precisă a regiunii ochiului, din întreaga imagine, pentru a analiza foarte precis pupila, fără interferențe din alte părți ale

imaginii.

Funcția `cv2.circle` din OpenCV desenează un cerc pe o imagine, astfel eu mă folosesc de aceasta, pentru evidențierea vizuală a pupilelor detectate. Marchez poziția pupilelor detectate pe imagine, le evidențiez utilizând culori diferite pentru a indica starea, precum verde pentru privire centrată sau roșu pentru privire suspectă, și ofer în același timp și un feedback vizual pentru supraveghetor, în ceea ce privește direcția privirii candidatului. Ca și parametrii, funcția primește imaginea pe care se desenează respectivul cerc, coordonatele centrului cercului, în cazul nostru al centrului pupilei, raza cercului, culoarea și grosimea liniei, dar, pentru grosime, am utilizat valoarea -1, deoarece am vrut un cerc plin.

Așadar, biblioteca OpenCV este responsabilă pentru tot ce înseamnă "a vedea" în acest sistem.

4.2 Dlib

Biblioteca Dlib a fost utilizată pentru detectarea feței și extragerea reperelor faciale[17]. Aceasta a fost esențială pentru analiza privirii și determinarea direcției acesteia.

Funcția `dlib.get_frontal_face_detector` a fost utilizată pentru a inițializa detectorul de fețe bazat pe HOG (Histogram of Oriented Gradients), care s-a dovedit eficient chiar și în condiții de iluminare variabilă. Acest algoritm funcționează prin împărțirea imaginii în celule mici și calcularea gradientului de intensitate pentru fiecare celulă. Gradientul este apoi normalizat pentru a reduce efectele variațiilor de iluminare, iar rezultatul este un descriptor robust care poate fi utilizat pentru a detecta fețele în imagine. Detectorul returnează coordonatele dreptunghiurilor care încadrează fețele detectate, iar aceste coordonate sunt utilizate pentru a izola regiunea feței.

Predictorul de repere faciale a fost încărcat cu ajutorul funcției `dlib.shape_predictor`, care a permis identificarea celor 68 de puncte de pe față. Acest predictor utilizează un model preantrenat bazat pe regresie, care estimează pozițiile punctelor faciale pe baza caracteristicilor extrase din imagine. Dintre cele 68 de puncte, punctele 36-47 au fost utilizate pentru a izola regiunea ochilor. Aceste puncte sunt distribuite astfel încât să încadreze conturul ochilor, permițând izolarea precisă a regiunii de interes.

Funcția `shape.part` a fost utilizată pentru a accesa coordonatele fiecărui punct facial, iar aceste coordonate au fost folosite pentru a calcula raporturile orizontale și verticale ale privirii.

De exemplu, raportul orizontal este calculat ca raportul dintre distanța dintre pupile și lățimea ochilor, iar raportul vertical este calculat ca raportul dintre înălțimea ochilor și lățimea acestora. Aceste raporturi sunt utilizate pentru a determina direcția privirii, cum ar fi privirea spre stânga, dreapta, sus sau jos.

De asemenea, aceste puncte au fost utilizate pentru a determina orientarea capului, prin calcularea distanțelor dintre punctele cheie, cum ar fi ochii, nasul și gura. Orientarea capului este estimată prin compararea acestor distanțe cu valorile de referință, iar rezultatul este utilizat pentru a detecta mișcările capului care pot indica comportamente suspecte.

4.3 PyTorch și YOLOv8

Pentru detectarea obiectelor interzise, precum telefoanele mobile și ceasurile inteligente, am utilizat biblioteca PyTorch[19] și modelul YOLOv8 dezvoltat de Ultralytics[16].

Alegerea algoritmului YOLO pentru acest proiect este fundamentată pe cercetări academice recente care subliniază că „YOLO este adoptat în diverse aplicații în principal datorită inferențelor sale mai rapide”[15], făcându-l ideal pentru aplicații în timp real precum acest prototip. Acest algoritm „utilizează o rețea neuronală convoluțională profundă simplă pentru a detecta obiecte în imagine”[27], ceea ce îl face potrivit pentru detecția în timp real a obiectelor precum telefoanele sau ceasurile inteligente, chiar și în condiții de iluminare variabilă.

Funcția `torch.load` a fost utilizată pentru a încărca modelele YOLOv8 preantrenate. Aceste modele sunt bazate pe arhitectura YOLO (You Only Look Once), care este optimizată pentru detectarea rapidă și precisă a obiectelor în imagini[4]. YOLOv8 utilizează o rețea neuronală convoluțională care împarte imaginea în grile și prezice simultan clasele obiectelor și coordonatele acestora. Această abordare permite detectarea în timp real, chiar și pe dispozitive cu resurse limitate.

Metoda `model.predict` a fost folosită pentru a detecta obiectele în cadrul video. Modelul a fost configurat să proceseze doar un cadru din 20 pentru a economisi resurse și să detecteze doar categoriile de interes. Această optimizare reduce semnificativ consumul de procesor și memorie, permițând rularea eficientă a aplicației pe hardware obișnuit.

Pentru a reduce falsurile pozitive, am utilizat două modele YOLOv8 separate: unul pentru detectarea telefoanelor și altul pentru ceasuri inteligente. Această abordare a permis clasificarea

precisă a obiectelor detectate, chiar și în condiții de iluminare slabă sau când obiectele erau parțial vizibile. De exemplu, modelul pentru telefoane a fost antrenat să recunoască formele și dimensiunile tipice ale telefoanelor mobile, iar modelul pentru ceasuri inteligente a fost antrenat să detecteze dispozitivele purtabile pe încheietura mâinii.

Funcția `model.detect` a fost utilizată pentru a genera coordonatele obiectelor detectate, iar aceste coordonate au fost utilizate pentru a desena dreptunghiuri în jurul obiectelor și pentru a genera alerte în timp real. De asemenea, coordonatele au fost utilizate pentru a calcula distanțele dintre obiectele detectate și alte elemente din imagine, cum ar fi fața candidatului, pentru a determina dacă obiectele sunt utilizate în mod activ.

4.4 NumPy

Biblioteca NumPy a fost utilizată pentru manipularea matricelor și a datelor numerice[8].

Funcția `np.zeros` a fost folosită pentru a crea măști pentru regiunile de interes din imagine, iar `np.full` a fost utilizată pentru a inițializa matrici cu valori constante. Aceste funcții sunt esențiale pentru preprocesarea imaginilor, deoarece permit izolarea regiunilor de interes și aplicarea de operații specifice doar asupra acestor regiuni[2].

Funcțiile `np.min` și `np.max` au fost utilizate pentru a determina limitele regiunilor de decupat, iar operațiile vectoriale au permis calcularea rapidă a coordonatelor pupilelor și a raporturilor orizontale și verticale. De exemplu, coordonatele pupilelor sunt calculate ca punctele cu intensitatea minimă din regiunea ochilor, iar raporturile sunt calculate ca raporturi între distanțele dintre aceste coordonate și dimensiunile regiunii.

4.5 Datetime

Biblioteca `datetime`[22] din Python a fost utilizată pentru gestionarea timpului și a marcajelor temporare în cadrul proiectului. Aceasta a fost esențială pentru sincronizarea evenimentelor, generarea rapoartelor și înregistrarea timpului de monitorizare.

Funcția `datetime.now()` a fost folosită pentru a obține timpul curent în momentul generării unui eveniment, cum ar fi detectarea unei nereguli si salvarea unui raport. Marcajele temporare create sunt utilizate pentru a lega evenimentele detectate de momentele exacte din fluxul video.

De exemplu, în cadrul clasei `ViolationMonitor`, funcția `datetime.now().strftime()` a fost utilizată pentru a formata marcajele temporare într-un format ușor de citit, cum ar fi `YYYY-MM-DD HH:MM:SS`. Acestea sunt incluse în rapoartele generate și sunt utilizate pentru a analiza comportamentul candidatului în timp.

Funcția `timedelta` a fost utilizată pentru a calcula diferențele de timp între evenimente. De exemplu, timpul scurs de la începutul înregistrării este calculat utilizând `datetime.now()` și timpul de start al înregistrării. Acest lucru este afișat în interfața grafică pentru a oferi utilizatorului informații în timp real despre durata monitorizării.

De asemenea, biblioteca `datetime` a fost utilizată pentru a genera nume unice pentru fișierele de înregistrare și rapoarte, utilizând funcția `strftime()` pentru a include marcaje temporare în numele fișierelor.

Biblioteca `datetime` a fost astfel un instrument esențial pentru gestionarea timpului în toate aspectele proiectului, de la monitorizare în timp real până la generarea rapoartelor detaliate.

4.6 Logging

Biblioteca `logging`[23] din Python a fost utilizată pentru gestionarea mesajelor de jurnalizare (*logging*) în cadrul proiectului. Aceasta a fost esențială pentru monitorizarea funcționării aplicației, diagnosticarea problemelor și păstrarea unui istoric al evenimentelor importante.

Funcția `logging.basicConfig()` a fost utilizată pentru a configura formatul și nivelul de jurnalizare. În cadrul proiectului, mesajele de jurnalizare sunt salvate atât într-un fișier (`sistem_anti_plagiat.log`), cât și afișate în consolă. Acest lucru permite o monitorizare eficientă a aplicației în timp real, precum și o analiză ulterioară a problemelor.

De exemplu, în cadrul clasei `SistemAntiPlagiat`, biblioteca `logging` a fost utilizată pentru a înregistra evenimente importante, cum ar fi pornirea înregistrării video, detectarea încălcărilor sau erorile apărute în timpul procesării.

Biblioteca `logging` a fost utilizată și pentru a înregistra excepțiile apărute în timpul rulării aplicației, utilizând funcția `logger.exception()`, care include automat detaliile despre excepție în mesajul de jurnalizare.

Această abordare a permis identificarea rapidă a problemelor și îmbunătățirea stabilității aplicației.

Biblioteca `logging` a fost astfel un instrument esențial pentru gestionarea mesajelor de jurnalizare în toate modulele proiectului.

4.7 PyQt5

Pentru dezvoltarea interfeței grafice, am utilizat biblioteca PyQt5[25]. Aceasta a permis crearea unei interfețe intuitive și funcționale, utilizând următoarele componente principale:

- **QPushButton:** Am folosit acest widget pentru a crea butoane interactive, cum ar fi cele pentru pornirea monitorizării, înregistrării sau exportarea rapoartelor. Aceste butoane sunt esențiale pentru a permite utilizatorului să interacționeze cu aplicația într-un mod simplu și direct. De exemplu, butonul de export al raportului oferă o funcționalitate clară și accesibilă pentru utilizator.
- **QTextEdit:** Acest widget este utilizat pentru afișarea în timp real a rapoartelor generate. Este o alegere potrivită deoarece permite afișarea unui text formatat și poate fi setat ca ”read-only”, astfel încât utilizatorul să nu poată modifica conținutul raportului.
- **QTimer:** Timer-ul este folosit pentru actualizarea periodică a timpului de înregistrare. Aceasta este o soluție eficientă pentru a menține interfața sincronizată cu procesele din fundal, cum ar fi monitorizarea în timp real.
- **QLabel:** Am utilizat QLabel pentru afișarea informațiilor importante, cum ar fi numărul de încălcări detectate sau timpul de înregistrare. QLabel este ideal pentru afișarea textului static sau dinamic, fiind ușor de actualizat în funcție de starea aplicației.
- **QCheckBox:** Checkbox-ul pentru activarea sau dezactivarea modului de oglindire a imaginii este o alegere intuitivă. Acesta permite utilizatorului să personalizeze modul în care este afișat fluxul video, ceea ce poate fi util în funcție de preferințele sau nevoile acestuia.
- **QGroupBox și Layout-uri (QVBoxLayout, QHBoxLayout):** Acestea sunt folosite pentru organizarea logică a elementelor din interfață. De exemplu, gruparea statisticilor sau a controalelor într-un QGroupBox ajută la crearea unei interfețe mai clare și mai structurate. Layout-urile verticale și orizontale sunt esențiale pentru aranjarea componentelor într-un mod responsiv și estetic.

- **QMainWindow:** Am utilizat această clasă ca fereastră principală a aplicației. Este o alegere standard pentru aplicațiile complexe, deoarece oferă suport pentru meniuri, bare de instrumente și alte elemente avansate.
- **QMessageBox:** Pentru afișarea mesajelor de confirmare sau eroare, QMessageBox este o soluție simplă și eficientă. Acesta permite comunicarea clară cu utilizatorul în momente critice, cum ar fi confirmarea exportului unui raport sau notificarea unei erori.

Interfața a fost proiectată pentru a fi intuitivă și ușor de utilizat, oferind acces rapid la toate funcționalitățile sistemului.

4.8 Modularizarea si structura proiectului

Proiectul a fost organizat modular, fiecare fișier Python având un rol bine definit. Această abordare facilitează întreținerea codului, testarea și extinderea funcționalităților. Mai jos sunt detaliate principalele module și responsabilitățile acestora:

- **main.py:** Acesta este punctul de intrare al aplicației. Inițializează componentele principale, cum ar fi detectorul de fețe, detectorul de obiecte, monitorul de încălcări și handler-ul video. De asemenea, gestionează procesarea cadrelor și exportul rapoartelor.
- **gui_app.py:** Conține implementarea interfeței grafice utilizând biblioteca PyQt5. Include funcționalități precum pornirea monitorizării, înregistrarea video, afișarea alertelor în timp real și exportul rapoartelor.
- **test_camera.py:** Un script auxiliar pentru testarea funcționalității camerei web. Verifică dacă camera poate fi accesată și dacă fluxul video este disponibil.
- **modules/face_detector.py:** Acest modul implementează detectarea feței și analiza direcției privirii utilizând biblioteca Dlib. Este responsabil pentru identificarea feței și determinarea direcției privirii candidatului.
- **modules/object_detector.py:** Conține implementarea pentru detectarea obiectelor interzise, cum ar fi telefoanele mobile și ceasurile inteligente, utilizând modelul YOLOv8. Este optimizat pentru procesarea eficientă a cadrelor.

- **modules/video_handler.py:** Gestionează procesarea și afișarea cadrelor video. Include funcționalități pentru oglindirea imaginii, afișarea alertelor și salvarea înregistrărilor video.
- **modules/violation_monitor.py:** Monitorizează încălcările detectate, cum ar fi privirea suspectă sau utilizarea obiectelor interzise. Înregistrează alertele și le sincronizează cu marcaje temporare.
- **modules/report_generator.py:** Este responsabil pentru generarea rapoartelor în formate HTML, CSV și JSON. Aceste rapoarte includ detalii despre încălcările detectate și statistici generale.
- **modules/data_exporter.py:** Gestionează exportul datelor în formatele specificate (CSV, JSON). Este utilizat pentru a salva informațiile despre încălcări într-un mod structurat.
- **modules/gaze_tracking/gaze_tracker.py:** Acest fișier implementează logica principală pentru urmărirea privirii. Utilizează reperele faciale pentru a calcula raporturile orizontale și verticale ale privirii și pentru a determina direcția acesteia.
- **modules/gaze_tracking/eye.py:** Acest modul izolează regiunea ochilor din imagine și detectează pupilele. De asemenea, calculează raportul de clipire pentru a identifica dacă ochiul este închis.
- **modules/gaze_tracking/pupil.py:** Acest fișier implementează logica pentru detectarea și analiza pupilelor, utilizând coordonatele și dimensiunile acestora pentru a determina starea ochilor.
- **requirements.txt:** Acest fișier conține lista tuturor bibliotecilor și pachetelor necesare pentru rularea aplicației. Este utilizat pentru instalarea rapidă a dependențelor.
- **config.json:** Fișierul de configurare al aplicației, care stochează setările personalizabile, cum ar fi pragurile de detecție sau opțiunile de salvare a rapoartelor.
- **.gitignore:** Fișierul care specifică ce fișiere sau directoare ar trebui ignorate de Git, cum ar fi fișierele temporare sau directoarele care se generează la rularea programului.
- **.devcontainer/:** Directorul care conține configurația pentru mediul de dezvoltare în container, care include fișiere Docker.

Această structură modulară permite o separare clară a responsabilităților, ceea ce face proiectul mai ușor de întreținut și extins. Fiecare modul poate fi testat și îmbunătățit independent, fără a afecta alte părți ale aplicației.

5 Prezentarea modului de utilizare, interacțiunea cu utilizatorul si configurarea

Această secțiune descrie modul de utilizare al aplicației, interacțiunea cu utilizatorul și configurarea acesteia. De asemenea, sunt incluse capturi de ecran pentru a ilustra funcționalitățile principale.

5.1 Mod de utilizare

Pentru a utiliza aplicația, urmați pașii de mai jos:

1. **Lansarea aplicației:** Deschideți aplicația rulând fișierul `gui_app.py`.
2. **Configurarea inițială:** Ajustați parametrii de configurare, cum ar fi limitele privirii (*left limit*, *right limit*, *down limit*), din interfața grafică sau din fișierul `config.json`. Tot in fisierul `config.json` se poate modifica si pragul de incredere pentru detectarea obiectelor interzise, dar nu este recomandat, deoarece este setat la o valoare care asigura o detectie cat mai precisa a obiectelor interzise, datorita numeroaselor teste efectuate.
3. **Pornirea monitorizării:** Apăsați butonul **Start Monitorizare** pentru a începe capturarea video și analiza comportamentului.
4. **Pornirea înregistrării:** Apăsați butonul **Start Înregistrare** pentru a salva fluxul video și alertele generate.
5. **Vizualizarea alertelor:** Monitorizați alertele generate în timp real în secțiunea dedicată din interfață.
6. **Exportarea rapoartelor:** După finalizarea sesiunii, utilizați butonul **Export Raport** pentru a salva rapoartele în format HTML, CSV sau JSON.

5.2 Interacțiune cu utilizatorul

Interfața grafică a aplicației este intuitivă și include următoarele componente:

- **Butoane de control:** Permite utilizatorului să pornească/oprească monitorizarea și să exporte rapoarte.
- **Secțiunea de alerte:** Afișează în timp real alertele generate de sistem.
- **Setări personalizabile:** Oferă opțiuni pentru ajustarea parametrilor limitei privirii și activarea/dezactivarea modului de oglindire a imaginii.
- **Oprirea monitorizării:** Apăsați butonul **Stop Monitorizare** pentru a opri capturarea video și analiza comportamentului.
- **Oprirea înregistrării:** Apăsați butonul **Stop Înregistrare** pentru a salva fluxul video și alertele generate într-un fișier.
- **Captură instantanee:** Apăsați butonul **Captură Instantanee** pentru a salva o imagine a fluxului video curent.

Se va observa pe parcursul utilizării aplicației că, în funcție de direcția privirii, se va schimba culoarea cercului desenat pe pupilele detectate. De exemplu, dacă privirea este centrată, cercul va fi verde, iar dacă privirea este spre stânga, dreapta sau jos, cercul va fi roșu.

5.3 Configurare

Aplicația poate fi configurată prin editarea fișierului `config.json`. Exemple de parametri configurabili:

- `"left_limit": 0.7` - Limita stângă pentru direcția privirii.
- `"right_limit": 0.3` - Limita dreaptă pentru direcția privirii.
- `"down_limit": 0.6` - Limita inferioară pentru direcția privirii.

5.4 Capturi de ecran

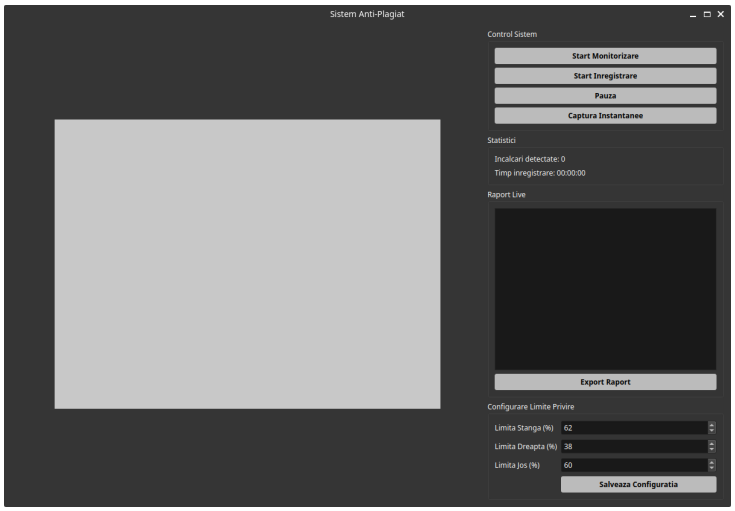


Figura 5: Interfața principală a aplicației

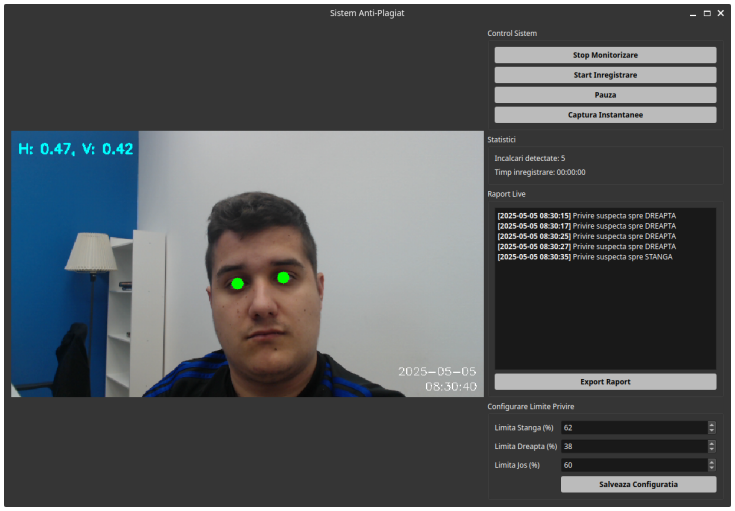


Figura 6: Pornirea monitorizării și înregistrării

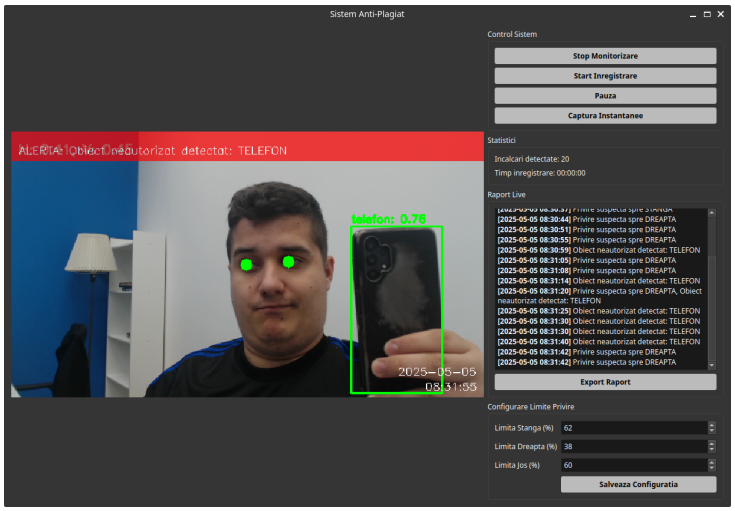


Figura 7: Detectare telefon

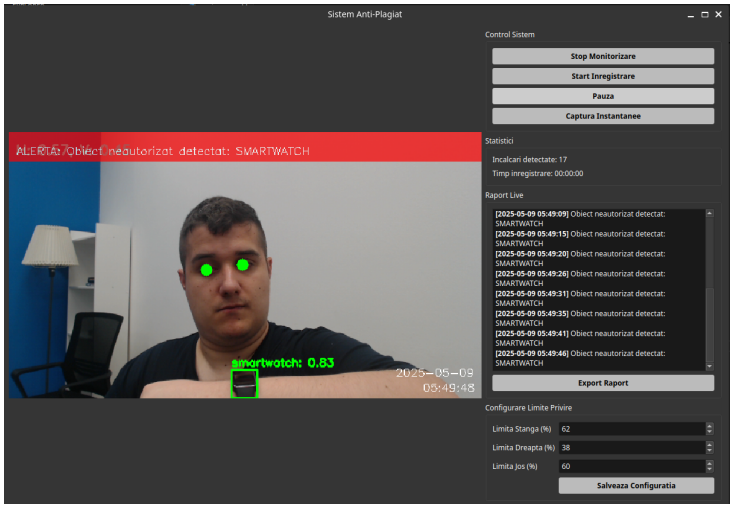


Figura 8: Detectare ceas inteligent

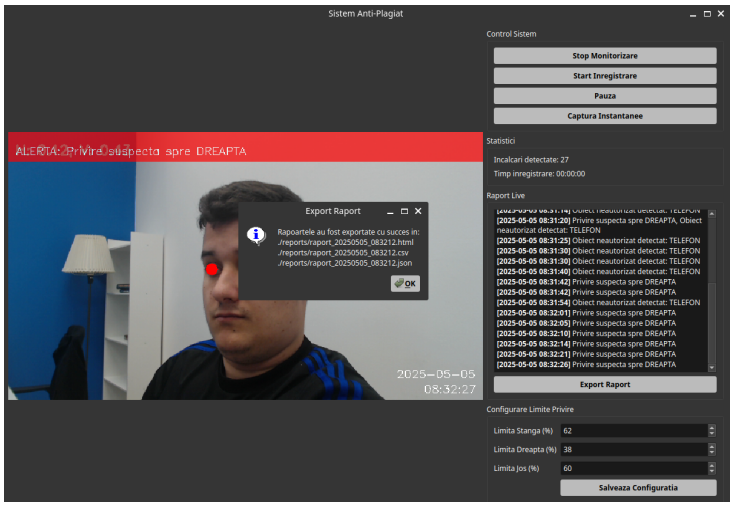


Figura 9: Exportarea rapoartelor

6 Contribuția personală

În cadrul acestui proiect, am adus mai multe contribuții semnificative, care includ:

6.1 Utilizarea a două modele YOLOv8 pentru detectarea obiectelor interzise

Am utilizat două modele YOLOv8 separate, unul specializat pentru detectarea telefoanelor mobile și altul pentru detectarea ceasurilor inteligente. Această abordare a fost aleasă pentru a lucra eficient și pentru a elimina falsurile pozitive. În loc să reantrenăm un model unic pentru ambele categorii, am aplicat clasificări stricte bazate pe dimensiunile *x-y* ale obiectelor detectate. Precizez că fiecare model YOLOv8 are un prag de încredere diferit, iar aceaste valoari au fost stabilite în urma anumitor teste efectuate si se găsesc în fișierul principal de configurație `config.json`. Astfel, `confidence_threshold` este setat la 0.55 pentru telefoane și 0.4 pentru ceasuri inteligente. Aceste valori sunt esențiale pentru a asigura o detecție precisă și pentru a minimiza riscul de falsuri pozitive.

6.2 Implementarea calculelor de algebră liniară pentru analiza privirii

Am dezvoltat și integrat calcule de algebră liniară pentru a determina direcția privirii utilizatorului. Aceste calcule includ:

- Determinarea coordonatelor pupilelor utilizând vectori și puncte de referință.
- Calcularea raporturilor orizontale și verticale (*horizontal ratio* și *vertical ratio*) pentru a estima direcția privirii.
- Filtrarea și ajustarea valorilor pentru a elimina zgomotul și variațiile neașteptate.

Raportul orizontal este esențial pentru a determina dacă respectivul candidat privește spre stânga sau spre dreapta. Acest algoritm calculează raportul bazat pe pozițiile pupilelor și include o componentă de ajustare pentru orientarea capului.

Algorithm 1 Calculul raportului orizontal (axa X - stanga/dreapta)

procedure CALCULRAPORTORIZONTAL
  if $pupile\_gasite = false$ **then**
    return 0.5 ▷ Pupilele nu sunt detectate $\implies$ returnam valoare neutra
  end if
  $poz\_pupila\_stanga \leftarrow \frac{pupila\_stanga.x}{(centru\_ochi\_stanga.x \cdot 2 - 10)}$ ▷ Normalizare pozitie pe X
  $poz\_pupila\_dreapta \leftarrow \frac{pupila\_dreapta.x}{(centru\_ochi\_dreapta.x \cdot 2 - 10)}$
  $raport\_pupile \leftarrow \frac{poz\_pupila\_stanga + poz\_pupila\_dreapta}{2}$ ▷ Media pozitiilor normalizate
  if $origine\_stanga \neq null$ **si** $origine\_dreapta \neq null$ **then**
    $centru\_abs\_stanga \leftarrow origine\_stanga.x + centru\_ochi\_stanga.x$
    $centru\_abs\_dreapta \leftarrow origine\_dreapta.x + centru\_ochi\_dreapta.x$
    $distanta\_ochi \leftarrow centru\_abs\_dreapta - centru\_abs\_stanga$
    if $cadru \neq null$ **then**
      $latime\_cadru \leftarrow cadru.shape[1]$ ▷ Latimea reala a imaginii
    else
      $latime\_cadru \leftarrow 640$ ▷ Valoare implicita daca nu exista frame
    end if
    $factor\_pozitie \leftarrow \frac{distanta\_ochi}{latime\_cadru \cdot 0.3}$ ▷ Evaluam pozitia relativa a capului
    $raport\_ajustat \leftarrow raport\_pupile$ ▷ Initial, raportul nu este modificat
    if $factor\_pozitie < 0.8$ **then**
      $raport\_ajustat \leftarrow \max(0.6, raport\_pupile)$ ▷ Capul e intors spre stanga
    else if $factor\_pozitie > 1.2$ **then**
      $raport\_ajustat \leftarrow \min(0.4, raport\_pupile)$ ▷ Capul e intors spre dreapta
    end if
    return $raport\_ajustat$
  end if
  return $raport\_pupile$ ▷ Nu avem date despre cap $\implies$ returnam direct media
end procedure

Raportul vertical este utilizat pentru a determina dacă privirea candidatului este orientată în jos. Acest algoritm calculează raportul bazat pe pozițiile verticale ale pupilelor și include o componentă de ajustare pentru înclinarea capului.

Algorithm 2 Calculul raportului vertical (axa Y - sus/jos)

procedure CALCULRAPORTVERTICAL
  if $pupile\_gasite = false$ **then**
    return 0.5 ▷ Nu putem estima directia $\implies$ valoare implicita
  end if
  $poz\_pupila\_stanga \leftarrow \frac{pupila\_stanga.y}{(centru\_ochi\_stanga.y \cdot 2 - 10)}$ ▷ Normalizare pozitie pe Y
  $poz\_pupila\_dreapta \leftarrow \frac{pupila\_dreapta.y}{(centru\_ochi\_dreapta.y \cdot 2 - 10)}$
  $raport\_pupile \leftarrow \frac{poz\_pupila\_stanga + poz\_pupila\_dreapta}{2}$ ▷ Media pozitiilor normalizate
  if $origine\_stanga \neq null$ **si** $origine\_dreapta \neq null$ **then**
    $centru\_ochi\_y \leftarrow \frac{origine\_stanga.y + centru\_ochi\_stanga.y + origine\_dreapta.y + centru\_ochi\_dreapta.y}{2}$
    $poz\_gura \leftarrow origine\_stanga.y + centru\_ochi\_stanga.y + 50$ ▷ Estimare gura
    $inaltime\_fata \leftarrow |centru\_ochi\_y - poz\_gura|$ ▷ Estimare inaltime fata
    if $inaltime\_fata > 0$ **then**
      $factor\_pozitie \leftarrow \frac{|origine\_stanga.y - centru\_ochi\_y|}{inaltime\_fata}$ ▷ Inclinare cap (sus/jos)
    else
      $factor\_pozitie \leftarrow 0.5$ ▷ Valoare implicita cand calculul nu este valid
    end if
    $raport\_ajustat \leftarrow raport\_pupile$
    if $factor\_pozitie > 0.6$ **then**
      $raport\_ajustat \leftarrow \min(0.4, raport\_pupile)$ ▷ Capul este aplecat
    else if $factor\_pozitie < 0.4$ **then**
      $raport\_ajustat \leftarrow \max(0.6, raport\_pupile)$ ▷ Capul este ridicat
    end if
    return $raport\_ajustat$
  end if
  return $raport\_pupile$ ▷ Date incomplete $\implies$ returnam media initiala
end procedure

Acești algoritmi lucrează împreună pentru a crea un sistem robust de detectare a direcției privirii. Rapoartele calculate sunt comparate cu praguri predefinite pentru a determina dacă candidatul privește în centru, stânga, dreapta sau jos, iar rezultatele acestei analize sunt utilizate pentru a genera alertele corespunzătoare.

Această implementare oferă mai multe avantaje semnificative:

1. **Robustețe la variații de iluminare** - Prin utilizarea metodei punctului minim pentru detectarea pupilelor, algoritmul este mai puțin sensibil la schimbările de iluminare.

2. **Compensarea orientării capului** - Ajustările implementate pentru factorul de poziție a ochilor și factorul de înclinare a capului permit distingerea între o privire suspectă și o mișcare naturală a capului.

3. **Normalizarea coordonatelor** - Convertirea coordonatelor absolute în rapoarte normalizate

face algoritmul adaptabil la diferite rezoluții de cameră și distanțe față de ecran.

4. **Filtrare bazată pe praguri optimizate** - Pragurile utilizate au fost determinate empiric pentru a oferi cel mai bun echilibru între rata de detecție și rata falsurilor pozitive.

Acești algoritmi au fost testați în diverse condiții de iluminare, demonstrând o acuratețe ridicată în detectarea direcției privirii, contribuind astfel la eficiența sistemului anti-plagiat.

6.3 Optimizarea procesării video în timp real

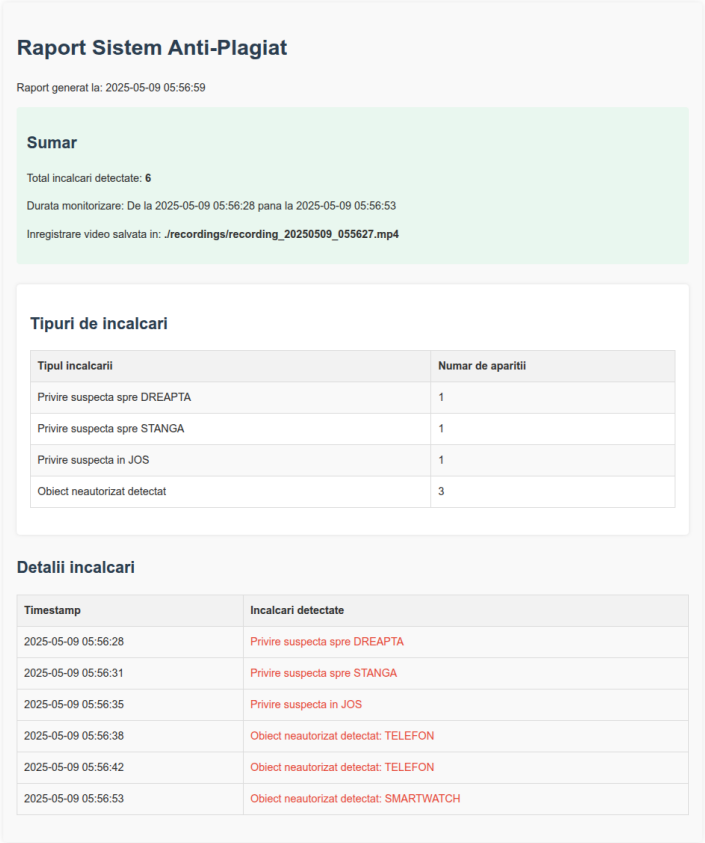
Am implementat optimizări pentru procesarea video, avand in vedere ca mintea umana nu poate procesa instant o anumita informație care poate fi utilizata in scopul de a frauda respectivul examen, incluzând:

- Procesarea unui cadru din 30 pentru detectarea obiectelor, reducând astfel consumul de resurse.
- Utilizarea unui buffer circular pentru gestionarea eficientă a cadrelor video.
- Integrarea unui sistem de cache pentru alertele generate, evitând duplicarea mesajelor de alertă.

6.4 Generarea rapoartelor detaliate

Am dezvoltat un sistem de generare automată a rapoartelor în formate HTML, CSV și JSON. Aceste rapoarte includ:

- Timpul exact al fiecărei încălcări detectate.
- Tipul obiectului detectat și locația acestuia în cadrul video.
- Rezumatul general al sesiunii de monitorizare.



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

Figura 10: Exemplu raport generat în format HTML

6.5 Integrarea interfeței grafice intuitive

Am dezvoltat o interfață grafică intuitivă utilizând PyQt5, care permite utilizatorilor să:

- Pornească și oprească monitorizarea în timp real.
- Vizualizeze alertele generate în timp real.
- Exporte rapoarte detaliate cu un singur clic.

6.6 Editarea parametrilor limitei privirii în timp real

Am implementat o funcționalitate care permite utilizatorului să editeze parametrii limitei privirii (*left limit*, *right limit*, *down limit*) în timp real, fără a fi necesară repornirea aplicației. Această funcționalitate este utilă în scenarii particulare, cum ar fi diferențele dintre susținerea unui examen pe calculator și pe foaie.

De exemplu, în cazul unui examen pe foaie, candidatul este nevoit să se aplece mai mult pentru a scrie, ceea ce poate reduce rata falsurilor pozitive prin ajustarea limitei inferioare (*down limit*). Această flexibilitate oferă supraveghetorilor posibilitatea de a adapta rapid sistemul la contextul specific al examenului.

Această funcționalitate contribuie la reducerea falsurilor pozitive și la creșterea acurateței sistemului în cadrul anumitor scenarii frecvent intalnite.

7 Dificultăți întâmpinate

În realizarea acestui proiect, am întâmpinat mai multe dificultăți, in cadrul carora am utilizat anumite constrangeri si solutii mai putin conventionale, pentru a avea rezultate cat mai aproape de cele dorite:

1. **Integrarea modelelor YOLOv8:** Configurarea și optimizarea modelelor YOLOv8 pentru detectarea obiectelor interzise a fost o provocare, mai ales în ceea ce privește reducerea falsurilor pozitive. A fost necesar să ajustez pragurile de detecție și să folosesc modele separate pentru categorii diferite de obiecte.

2. **Performanța pe hardware limitat:** Procesarea video în timp real a fost dificilă pe dispozitive cu resurse hardware limitate. Am implementat optimizări, cum ar fi procesarea unui cadru din 30 și utilizarea unui buffer circular, pentru a reduce consumul de resurse.

3. **Detectarea privirii în condiții de iluminare variabilă:** Detectarea precisă a direcției privirii a fost afectată de variațiile de iluminare. A fost necesar să aplic filtre și să ajusteze parametrii algoritmilor pentru a îmbunătăți robustețea.

4. **Sincronizarea datelor multimedia:** Corelarea precisă între fluxul video, audio și evenimentele detectate a fost o sarcină complexă. Am utilizat marcaje temporare sincronizate pentru a lega alertele generate de momentele exacte din fluxul video.

5. **Complexitatea interfeței grafice:** Dezvoltarea unei interfețe grafice intuitive și funcționale a fost o provocare, mai ales în ceea ce privește organizarea logică a componentelor și sincronizarea acestora cu procesele din fundal.

6. **Gestionarea falsurilor pozitive:** În timpul testării, am observat o rată ridicată de falsuri pozitive, mai ales în detectarea obiectelor. A fost necesar să ajustez algoritmii și să implementez filtre suplimentare pentru a îmbunătăți acuratețea.

7. **Documentarea și modularizarea codului:** Structurarea proiectului într-un mod modular și documentarea clară a fiecărui modul au necesitat timp suplimentar, dar au fost esențiale pentru întreținerea și extinderea ulterioară a aplicației.

8. **Testarea și validarea:** Testarea aplicației în scenarii reale a fost dificilă, mai ales în simularea condițiilor variate de utilizare, cum ar fi diferite unghiuri ale camerei, iluminare slabă sau zgomot de fundal.

9. **Gestionarea dependențelor:** Instalarea și configurarea corectă a tuturor bibliotecilor și pachetelor necesare au fost uneori problematice, mai ales din cauza incompatibilităților între versiuni.

8 Concluzii

Proiectul *Sistem de Monitorizare Anti-Plagiat pentru Examene* a reușit să îndeplinească obiectivele propuse, oferind o soluție inovatoare și eficientă pentru monitorizarea comportamentului candidaților în timpul examenelor.

8.1 Îndeplinirea obiectivelor

- **Detecția privirii candidatului:** Sistemul utilizează tehnologii avansate de procesare a imaginilor pentru a analiza direcția privirii și a semnala abaterile de la comportamentul normal. Această funcționalitate a fost implementată cu succes, oferind o detecție precisă și în timp real.

- **Identificarea obiectelor neautorizate:** Prin integrarea modelelor YOLOv8, aplicația detectează dispozitive precum telefoanele mobile și ceasurile inteligente, contribuind la prevenirea tentativelor de fraudă.

- **Înregistrare și arhivare:** Sistemul permite înregistrarea sesiunilor de examen, oferind posibilitatea de a analiza ulterior comportamentul candidaților și de a arhiva dovezile.

- **Generarea rapoartelor:** Aplicația generează rapoarte detaliate în formate HTML, CSV și JSON, facilitând analiza și documentarea incidentelor.
- **Interfață intuitivă:** Interfața grafică dezvoltată cu PyQt5 este ușor de utilizat și oferă acces rapid la toate funcționalitățile sistemului.

8.2 Utilitate și actualitate

Plagiatul și tentativele de fraudă reprezintă probleme majore în sistemele educaționale moderne. Aplicația propusă oferă o soluție actuală și relevantă, utilizând tehnologii de ultimă generație pentru a combate aceste probleme. Prin focalizarea pe comportamentul fizic al candidaților, sistemul aduce un avantaj semnificativ față de soluțiile existente, care se concentrează în principal pe monitorizarea activității pe ecran.

8.3 Performanță și optimizări

Sistemul a fost optimizat pentru a funcționa eficient chiar și pe hardware limitat, utilizând tehnici precum procesarea unui cadru din 30 și reducerea consumului de resurse. Testele efectuate au demonstrat o rată ridicată de acuratețe în detectarea comportamentelor suspecte și a obiectelor interzise, cu o rată scăzută de falsuri pozitive.

8.4 Impact și relevanță

Impactul aplicației este semnificativ, contribuind la menținerea integrității academice și la crearea unui mediu de examen echitabil. Prin utilizarea tehnologiilor de inteligență artificială și procesare a imaginilor, sistemul poate fi extins și adaptat pentru diverse contexte educaționale și profesionale.

8.5 Avantaje față de alte soluții similare

Comparativ cu alte soluții existente, aplicația noastră oferă următoarele avantaje:

- **Focalizare pe comportamentul fizic:** Spre deosebire de soluțiile care monitorizează doar activitatea pe ecran, acest sistem analizează direcția privirii și detectează obiectele neautorizate.

- **Costuri reduse:** Implementarea locală reduce semnificativ costurile per candidat, comparativ cu soluțiile comerciale precum ProctorU.
- **Independență de platformă:** Aplicația funcționează independent de browser sau alte platforme, oferind o flexibilitate sporită.
- **Rapoarte detaliate:** Generarea automată a rapoartelor în multiple formate facilitează analiza ulterioară și documentarea incidentelor.

8.6 Compatibilitate multiplatformă

Aplicația noastră rulează atât pe Windows, cât și pe Linux (Ubuntu), oferind flexibilitate utilizatorilor în funcție de sistemul de operare preferat.

Pe Windows, aplicația beneficiază de suport pentru GPU prin CUDA, ceea ce permite utilizarea plăcii video NVIDIA pentru accelerarea procesării video și a detecției obiectelor. Acest lucru este posibil datorită modului în care driverele NVIDIA sunt stocate și gestionate pe Windows.

Pe Linux, deși aplicația funcționează corect, integrarea cu GPU prin CUDA nu a fost încă implementată complet. Această limitare se datorează diferențelor în modul de gestionare a driverelor NVIDIA pe Linux comparativ cu Windows. În prezent, procesarea pe Linux se realizează exclusiv pe CPU, dar optimizările implementate asigură o performanță acceptabilă chiar și în aceste condiții.

Această compatibilitate multiplatformă face ca aplicația să fie accesibilă unui spectru larg de utilizatori, indiferent de sistemul de operare utilizat.

8.7 Îmbunătățiri posibile

Deși aplicația actuală oferă funcționalități avansate pentru monitorizarea anti-plagiat, există câteva aspecte care pot fi îmbunătățite sau extinse pentru a crește eficiența și utilitatea acesteia:

1. **Integrarea cu platforme LMS:** Adăugarea suportului pentru integrarea directă cu platforme de management al învățământului (LMS), precum Moodle, Blackboard sau Canvas, pentru a facilita gestionarea examenelor și a rapoartelor.

2. **Suport multi-camera:** Extinderea aplicației pentru a permite utilizarea mai multor camere simultan, oferind o perspectivă mai completă asupra comportamentului candidatului.

3. **Monitorizare audio avansată:** Dezvoltarea unui modul de procesare audio mai sofisticat, care să detecteze conversațiile suspecte sau utilizarea dispozitivelor audio neautorizate.

Aceste îmbunătățiri ar putea crește semnificativ valoarea aplicației și ar extinde aria de utilizare a acesteia în diverse contexte educaționale și profesionale.

Cercetările recente în domeniul ceasurilor inteligente au propus „o categorizare a utilizării ceasurilor inteligente în sectorul de sănătate în 3 domenii funcționale cheie: monitorizare, îndrumarea și predicție”[18], concepte ce ar putea fi adaptate pentru monitorizarea comportamentului candidaților. De asemenea, studii au demonstrat eficiența „cadrelor bazate pe ceasuri inteligente pentru evaluarea și monitorizarea mobilității în timp real”[6], sugerând posibilitatea integrării acestor tehnologii în viitoarele versiuni ale acestui sistemu pentru a îmbunătăți acuratețea detectării comportamentelor suspecte. [3]

În concluzie, proiectul reprezintă o contribuție semnificativă la domeniul monitorizării anti-plagiat, oferind o soluție modernă, eficientă și adaptabilă la nevoile actuale ale sistemelor educaționale.

Bibliografie

[1] Branislav Sredojev, Dragan Samardzija și Dragan Posarac, „WebRTC technology overview and signaling solution design and implementation”, în *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 și 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 și Jia You, „What a massive database of retracted papers reveals about science publishing’s ’death penalty’”, în *Science* 362.6413 (2018), Ultima accesare: 6 mai 2025, pp. 390–393, DOI: 10.1126/science.aav8384.

[4] Joseph Redmon și Ali Farhadi, „YOLOv3: An Incremental Improvement”, în *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”, în *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”, în *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”, în *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”, în *Nature* 585.7825 (2020), Ultima accesare: 6 mai 2025, pp. 357–362, DOI: 10.1038/s41586-020-2649-2.

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

[10] Yasmine Alem, Bachir Boudraa și Adel Merabet, „A Novel Deep Learning-based Online Proctoring System using Face Recognition, Eye Blinking, and Object Detection Techniques”, în *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”, în *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 și R. Rajesh, „Face Detection with OpenCV and Deep Learning: A Comprehensive Review”, în *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”, în *Revista Română de Studii Juridice* 5.2 (2021), Ultima accesare: 6 mai 2025, pp. 43–59.

[14] Olena Zimba și Armen Yuri Gasparyan, „Plagiarism detection and prevention: a primer for researchers”, în *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 și Hong-Yuan Mark Liao, „Object detection using YOLO: challenges, architectural successors, datasets and applications”, în *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”, în *IEEE Access* 11 (2023), Ultima accesare: 6 mai 2025, pp. 45983–45997, DOI: 10.1109/ACCESS.2023.3267452.

[18] Rabee Moshawrab, Konstantin Aal și Volker Wulf, „The Value of Smartwatches in the Health Care Sector for Monitoring, Nudging, and Predicting: Viewpoint on 25 Years of Research”, în *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", în *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", în *Computer Vision Quarterly* 7.2 (2023), Ultima accesare: 6 mai 2025, pp. 23–35.