# Adaptive AI Algorithms for Proctoring: Automating Student Behaviour Profiling and Real-Time Misconduct Detection

Ravindu Wegiriya
Index number: 20002041

Supervisor: Dr. Enosha Hettiarachchi
Co-Supervisor: Prof. K.P. Hewagamage

June 2025
Submitted in partial fulfillment of the requirements of the
B.Sc in Computer Science Final Year Project (SCS4224)

# Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

**Student Name :** W.W.R.U. Wegiriya

**Registration Number :** 2020/CS/204

**Index Number :** 20002041

[illegible]

**Signature & Date**

This is to certify that this dissertation is based on the work of Mr. W.W.R.U. Wegiriya under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

**Supervisor Name :** Dr. Enosha Hettiarachchi

[illegible] 28/06/2025

**Signature & Date**

# Contents

# List of Figures

# List of Acronyms

|  |  |
|---|---|
| **ML** | Machine Learning |
| **AI** | Artificial Intelligence |
| **MFA** | Multi Factor Authentication |
| **UI** | User Interface |

## 1 Abstract

This thesis presents the design and evaluation of an adaptive artificial intelligence (AI) based online proctoring system, aiming to automate student behavior profiling and enable real-time misconduct detection. Motivated by the rapid transition to online assessments during the COVID-19 pandemic, the study addresses the limitations of traditional rule-based proctoring systems, including high false-positive rates and fairness concerns. The proposed system combines rule-based anomaly detection with a novel adaptive behavior profiling module, which learns individual student behaviors during mock exams to distinguish between genuine violations and innocuous, habitual movements. Multiple modules, such as gaze tracking, head pose estimation, murmur detection, and environmental noise detection, were developed and integrated into a hybrid framework. A series of controlled mock and simulated real exams with student volunteers demonstrated the system's effectiveness in improving precision, recall, and fairness while maintaining real-time performance. Evaluation results showed significant reductions in false positives and increased user acceptance of the adaptive profiling concept. This work contributes to the field of intelligent proctoring by proposing a scalable, modular, and behaviorally aware detection framework that balances academic integrity with student experience, laying the groundwork for future research on ethical, inclusive, and robust online examination systems.

# 2 Introduction

The rapid shift to digital platforms and remote learning due to the COVID-19 pandemic has transformed the educational landscape (Moreno-Guerrero et al. 2020). This transformation has brought challenges, particularly in ensuring the security and integrity of remote assessments. AI-based and ML-based proctoring systems have emerged as solutions to these challenges, promising to uphold academic integrity through scalable and secure assessment mechanisms (Peterson 2019, Pandey et al. 2020). AI and ML technologies enable real-time monitoring and fraud detection through techniques like facial recognition, gaze tracking, and behavioral analysis, utilizing basic hardware such as webcams and microphones. While these systems grow in importance, significant concerns about privacy and ethics persist (Coghlan et al. 2021, Draaijer et al. 2018).

Our research initially focused on investigating the effectiveness of adaptive AI-based proctoring systems by testing algorithms for baseline behavior profiling, real-time alerting, and adaptive responses. Throughout the research so far, emphasis has shifted toward refining models for precise gaze tracking, adaptable behavior profiling during mock exams, and accurate identification of anomalies based on students' individual behavior profiles. These refinements ensure that the proctoring system meets both technical and ethical standards.

## 2.1 Motivation

The shift to online learning and assessment, accelerated by the COVID-19 pandemic, has highlighted the need for robust online proctoring solutions. Traditional in-person invigilation is not feasible in remote settings, leading to the adoption of AI and ML technologies for monitoring and ensuring exam integrity. However, these technologies pose challenges related to their effectiveness, false-positive rates, and the absence of a comprehensive ethical framework (Moreno-Guerrero et al. 2020).

This research now seeks to create an adaptive AI-based proctoring solution that not only uses individual student profiles but is also responsive to specific behaviors during exams. By refining the system to adapt to students' natural behavioral variations, we aim to reduce false positives. Our recent work has centered on analyzing gaze patterns, frequency of movements, and real-time adjustments to distinguish between typical and suspicious activities with higher accuracy, thus enhancing proctoring reliability and student experience.

## 2.2 Background

Online proctoring systems have evolved from simple webcam monitoring to advanced intelligent monitoring techniques. Key advancements integrated in this research include:

- **Facial Recognition**: Ensures identity verification and continuous monitoring.
- **Gaze Tracking**: Monitors eye movements and tracks gaze patterns in real-time, including precision tracking to detect even minor deviations.
- **Behavioral Analysis**: Establishes baseline behavior profiles during mock exams to differentiate between typical and suspicious behaviors.
- **Audio Processing and System Monitoring**: These provide supplementary validation but remain secondary within this study's scope.

## 2.3 Revised Scope

Our scope evolved to emphasize adaptive algorithms and behavioral profiling in online proctoring. Key components of the revised scope include the following:

### 2.3.1 In Scope

- **Developing Adaptive AI Algorithms for Proctoring Systems**

- Researching and developing adaptive algorithms that can learn baseline behaviors to detect suspicious activity.
- Enhancing accuracy and efficiency with real-time feedback, reducing false positives.

- **Profiling Student Behavior in Ideal Examination Environments**
  - Collecting data from controlled mock exams to define individual student behavior.
  - Using ML to create baselines for detecting anomalies.
- **Automating Detection of Suspicious Behavior and Real-Time Alerts**
  - Implementing a system that uses adaptive profiles for real-time monitoring and alerting.
  - Switching from autonomous to human-supervised proctoring when anomalies are detected.
- **Student Identity Monitoring**
  - Verifying identity with continuous checks throughout the examination.
- **Performance Evaluation and Benchmarking**
  - Evaluating detection accuracy, false positive rates, user experience, and feedback mechanisms.
  - Ethical Consideration - Obtaining students' consent through an acknowledgment form, filled prior to collecting their data for training and testing purposes.

### 2.3.2 Out of Scope

- **Development of New Hardware**
  - Relying on existing hardware such as webcams and microphones.
- **Long-term Psychological Impacts**
  - Not investigating long-term psychological effects on students.
- **Security Issues**
  - Not covering broader security concerns, focusing instead on student consent for monitoring.

# 3 Comprehensive Literature Survey and Research Gaps

## 3.1 Preliminary Literature Review

The advancement of Artificial Intelligence (AI) and Machine Learning (ML) has significantly impacted the development of intelligent proctoring systems, aimed at ensuring the integrity of online examinations. A Literature Review on **Artificial Intelligence and Machine Learning based Proctoring Systems** identified and analyzed current research and technologies in this domain, focusing on their effectiveness. Initial research in this domain, such as the study by Foster & Layman (2013), integrated basic digital technologies like simple webcam monitoring and manual review methods. This early work highlighted both the potential and the limitations of digital tools, underscoring the necessity for more automated solutions to manage the increasing volume of online exams.

As the demand for online assessments grew, researchers began developing more sophisticated systems that leveraged image processing, facial detection, and screen monitoring. For example, Cavanagh et al. (2016) proposed an early proctoring tool that utilized head-pose estimation to infer gaze direction, paving the way for gaze-based monitoring. These initial solutions, while limited in adaptability, laid the groundwork for subsequent research that integrated real-time ML algorithms capable of dynamic analysis.

### 3.1.1 Identity Verification and Behavioral Monitoring

One of the critical components of intelligent proctoring systems is identity verification. Advanced facial recognition technology has been employed to ensure that the person taking the exam is the registered candidate. For instance, Iqbal et al. (2023) demonstrated that sophisticated facial recognition algorithms could significantly reduce instances of impersonation. Furthermore, biometric authentication methods such as fingerprint and retina scans, as discussed by Alessio et al. (2017), have been shown to enhance security further.

Despite these advancements, many systems still perform identity checks only at the beginning of the exam, which can be insufficient for preventing impersonation throughout the examination period. There is a clear need for continuous identity verification mechanisms that operate throughout the exam duration to ensure the person taking the exam is consistently the same registered candidate. More recently, systems such as those described by Verma et al. (2024) and Sakhipov et al. (2025) incorporate continuous facial tracking with anti-spoofing algorithms that detect changes in lighting, expressions, and potential face replacement attacks using generative media. These systems apply liveness detection using micro-blink frequency and texture inconsistencies to flag potential impersonation.

Behavioral analysis plays a vital role in detecting cheating by monitoring physical and behavioral patterns. AI models have been developed to analyze these patterns in real-time to identify potential cheating behaviors. Masud et al. (2022) showcased the effectiveness of behavioral analysis in flagging unusual activities, thereby improving the reliability of proctoring systems. Additionally, gaze tracking technology, explored by Malhotra et al. (2022), monitors eye movements to detect off-screen glances indicative of cheating.

In recent developments, behavior profiling has expanded beyond gaze and head movements to include full-body pose estimation, as seen in the work of Moyo et al. (2023), where skeletal analysis using OpenPose helped detect body language associated with cheating (e.g., side conversations, mobile phone use). Deep learning models such as CNN-BiLSTM pipelines are now being trained on sequences of visual data to improve the accuracy of anomaly detection across different student behavior baselines. These systems can distinguish between benign fidgeting and coordinated cheating signals, significantly reducing false positives.

However, current systems often struggle to differentiate between normal student behavior and actual cheating. This limitation results in false positives and negatives, which can undermine the integrity of the proctoring process. Developing adaptive algorithms that can learn and profile each student's baseline behavior through mock exams is essential for more accurate and fair monitoring. Newer approaches, such as those proposed by Liu et al. (2023), use attention-based temporal models to learn behavioral deviations based on the sequence and duration of activities, thus creating individualized behavior signatures.

### 3.1.2 Natural Language and Audio Processing

Natural Language Processing (NLP) is another crucial technology used in intelligent proctoring systems. It analyzes verbal communication during exams to detect inappropriate or off-topic responses. Pandey et al. (2020) highlighted the application of NLP in identifying dishonest behavior, helping to maintain focus and relevance in examination settings.

Audio processing technologies complement these systems by monitoring ambient sounds for unauthorized conversations or background noises. Peterson (2019) demonstrated that audio recognition algorithms could effectively identify suspicious sounds, providing an additional layer of security.

Advancements in audio signal analysis now include speech-to-text processing combined with sentiment analysis to understand the context of student speech. Systems like those presented by Singh et al. (2024) incorporate real-time speech transcription and keyword flagging (e.g., names, answer patterns, code phrases), enabling automated flagging of potential collusion events. Voice Activity Detection (VAD), such as Silero-VAD, is increasingly used for lightweight background monitoring, while NLP modules like BERT have been fine-tuned for context-sensitive speech detection in proctored environments.

### 3.1.3 System and Network Monitoring

Effective proctoring systems extend their surveillance capabilities to digital activities, tracking software usage and network activity to detect unauthorized applications or anomalies. Draaijer et al. (2018) emphasized the importance of continuous monitoring of system activities and network traffic to reduce digital cheating, highlighting its significant impact on maintaining a secure examination environment.

Recent systems integrate application-level monitoring with AI-enhanced anomaly detection. For example, Nurpeisova et al. (2023) implemented a central dashboard that consolidates video, screen, application usage, and network logs. If a student switches tabs to an unapproved domain, the AI system cross-references the timestamp with camera feed anomalies to issue a weighted suspicion score. Furthermore, browser plugins are now being used to monitor clipboard activity and real-time keystrokes, enabling live detection of copying/pasting or receiving answers through chat overlays.

Despite the substantial advancements in these areas, many proctoring systems lack integration across these technologies, resulting in fragmented approaches that fail to address the complexity of online cheating comprehensively. Additionally, current proctoring systems rarely offer adaptive responses based on real-time data, limiting their effectiveness in dynamic exam environments.

### 3.1.4 Dual-Mode Operation and Real-Time Alerting

Another critical area is the development of systems that can operate in both human-proctor supervised and unsupervised modes. Such systems should be capable of real-time alerting, enabling a seamless transition from unsupervised to supervised mode if suspicious behavior is detected. This dual-mode functionality ensures continuous monitoring and timely intervention, enhancing the overall effectiveness of the proctoring process.

Recent frameworks, such as those described by Felsinger et al. (2024), have implemented real-time decision support systems where AI flags are evaluated based on confidence scores. For example, if a system identifies a medium-confidence alert (like brief gaze deviation and background noise), it may request human verification, while high-confidence alerts (such as face absence and known cheating gesture) trigger automatic alerts and session termination. The hybrid architecture thus optimizes human resource allocation while maintaining high detection fidelity.

Additionally, several systems have implemented asynchronous supervision modes, where AI performs live monitoring and then compiles an automated summary of suspicious events for post-exam review. This not only improves scalability but also allows proctors to focus on high-impact cases, reducing operational strain in large-scale examinations.

## 3.2 Recent Developments

In recent years, advancements in AI and machine learning have significantly enhanced the functionality and accuracy of intelligent proctoring systems, making them more efficient and reliable in ensuring the integrity of online assessments. One of the key areas of progress has been in the continuous identity verification process. New systems now employ advanced facial recognition algorithms, integrated with object detection models such as YOLO and FaceNet, to monitor candidates throughout the examination. This approach solves the earlier problem of verifying identity only at the start, offering continuous authentication and reducing the risk of impersonation during the exam (Sharma et al. 2024).

The scope of behavioral monitoring has also expanded. Traditionally, systems focused on simple gaze tracking or head movement detection, but recent developments now leverage deep learning and computer vision techniques to monitor a wider range of behaviors. These systems track various physical cues, such as hand movements, gaze direction, and body posture, in real-time. By profiling each candidate's baseline behavior through mock exams, these systems adapt to the unique behaviors of individual students, allowing them to detect deviations with greater accuracy and thus minimize the occurrence of false positives and false negatives (Mewada et al. 2024).

Natural Language Processing (NLP) is now being integrated to analyze both the verbal responses and ambient sounds during exams, enhancing the detection of dishonest behavior. Researchers have developed algorithms capable of identifying off-topic responses, unusual speech patterns, and suspicious background noises, such as conversations or unauthorized sounds. These systems complement visual monitoring by adding another layer of security, ensuring that all forms of cheating, including verbal communication and noise disruptions, are detected (Chougule et al. 2024). Furthermore, advancements in audio recognition have enabled more precise identification of environmental anomalies, such as the detection of unauthorized electronic devices or voices that might suggest collusion or cheating (Chatterjee et al. 2024).

Another significant development is the refinement of dual-mode operation systems. These systems offer flexibility by functioning in both unsupervised and supervised modes. In unsupervised mode, AI algorithms monitor for suspicious behavior autonomously, while in supervised mode, human proctors can intervene if an anomaly is detected. This seamless transition between modes allows for real-time alerting and intervention, which is particularly beneficial in large-scale exam environments. Moreover, these systems use machine learning to continuously refine their alerting mechanisms based on the collected data, thus improving their performance over time (Felsinger et al. 2024).

AI-driven proctoring systems are also becoming more inclusive, with efforts to reduce biases in facial recognition and ensure fairness in identity verification. These improvements are critical in ensuring that proctoring systems do not disadvantage students from diverse backgrounds, addressing ethical concerns related to demographic bias in facial recognition technologies (Verma et al. 2024). This focus on inclusivity is complemented by research into the scalability of proctoring systems, which is essential for ensuring they can handle large numbers of students while maintaining accuracy and security. Recent developments have led to the creation of systems capable of monitoring multiple students simultaneously, without sacrificing the quality of monitoring or performance (Somavarapu et al. 2024).

Lastly, recent studies have explored the integration of system and network monitoring tools into proctoring platforms. These tools track software usage and network traffic during exams to detect unauthorized applications or attempts to circumvent the system. By combining video surveillance, audio recognition, and digital monitoring, AI proctoring systems are becoming increasingly comprehensive in their approach to preventing cheating. This integrated approach helps ensure a secure and fair exam environment by covering all potential avenues of dishonesty (Paul et al. 2024).

These developments collectively demonstrate the rapid evolution of AI-based proctoring systems. By incorporating more sophisticated technologies like deep learning, NLP, audio recognition, and continuous behavioral profiling, the proctoring process is becoming more accurate, secure, and adaptable, while simultaneously addressing ethical and inclusivity concerns in online education.

## 3.3 Research Gaps

Despite the rapid advancements in AI-based proctoring systems, several critical research gaps remain, particularly in areas that are directly relevant to the development of adaptive intelligent proctoring systems with behavioral profiling and real-time alerting.

Firstly, while continuous identity verification is improving, many systems still rely on periodic checks, which can leave gaps in security. Research is needed to develop more sophisticated, real-time identity verification methods that continuously monitor candidates without compromising privacy or system performance. This would be particularly valuable in adaptive systems where consistent identity authentication is necessary throughout the examination process. Moreover, the ethical implications of persistent monitoring and fairness in demographic performance remain under-addressed in many commercial tools.

Secondly, while behavioral monitoring has made strides in detecting basic anomalies like gaze tracking and head movement, systems still face challenges in accurately profiling individual behavior and detecting subtle deviations. There is a need for deeper exploration into personalized behavioral profiling using AI models, which can adapt to the unique behaviors of each student. This would reduce the occurrence of false positives and ensure that only genuine cheating behaviors are flagged, especially in an environment where real-time alerts are necessary for immediate intervention.

Furthermore, the integration of Natural Language Processing (NLP) and audio recognition is still in its early stages. Although these technologies are promising in identifying verbal dishonesty and background noise, they have yet to be seamlessly incorporated into a holistic monitoring system. More research is needed to enhance these technologies and integrate them with visual monitoring to provide a more comprehensive approach to detecting cheating in real-time.

Another gap exists in the adaptation of system functionalities. Many current proctoring systems fail to dynamically adjust based on the data collected during the exam. Research is needed to develop systems that not only detect anomalies but also learn from them to improve their monitoring mechanisms in real time. This is especially important for adaptive proctoring, where the system must adjust its monitoring techniques based on a candidate's behavior throughout the exam.

Lastly, although dual-mode operation systems show promise, their implementation remains limited, especially in large-scale exam settings. There is a need for further research into scalable, real-time alerting mechanisms that can seamlessly transition from unsupervised to supervised monitoring without compromising the exam experience or raising unnecessary alarms. This would pave the way for more efficient, large-scale proctoring solutions that are adaptive to real-time data.

In conclusion, while significant progress has been made in the development of intelligent proctoring systems, addressing the gaps related to continuous identity verification, adaptive behavioral profiling, seamless integration of technologies, and real-time alerting mechanisms will be essential for advancing adaptive, intelligent proctoring systems. These advancements are crucial to meeting the evolving demands of online education and ensuring the security, fairness, and integrity of digital assessments. The focus of this research, which involves behavioral profiling and real-time alerting, aligns with these gaps, offering a promising solution to further enhance proctoring systems in online education.

# 4 Research Questions

## 4.1 Research Question 1

**How can machine learning techniques profile typical student behaviors during exams to establish baselines for detecting potential misconduct?**

This question investigates how to use machine learning to profile normal student behavior during exams by analyzing data on gaze direction, body movements, and interactions with the exam interface. By observing students in controlled mock exams, the study aims to identify key behavioral patterns and create profiles that serve as reference points for detecting anomalies indicative of potential misconduct.

## 4.2 Research Question 2

**How can intelligent proctoring systems use adaptive algorithms to provide real-time feedback and accurately identify normal versus suspicious student behavior?**

This question aims to develop adaptive algorithms for intelligent proctoring systems that use behavioral profiles to monitor student actions in real-time. By comparing observed behaviors against established profiles, the system can identify deviations and flag potential suspicious activities, providing real-time feedback to human proctors. This approach seeks to enhance the accuracy and efficiency of proctoring systems in both supervised and unsupervised settings.

## 4.3 Research Question 3

**How effective are adaptive proctoring systems that use profiling and real-time alerts in detecting cheating during exams?**

This question evaluates the effectiveness of adaptive proctoring systems in detecting exam misconduct. It involves assessing the system's accuracy, reliability, and impact on reducing cheating through metrics like false positive and false negative rates. By testing in controlled and real-world environments, the study aims to validate the system's performance and gather feedback from proctors and students on its usability and acceptance.

# 5 Aims and Objectives

## 5.1 Aim

This research aims to develop an adaptive proctoring system for online exams that autonomously profiles students' behaviors to detect cheating without constant human supervision. By creating individualized behavioral profiles and integrating identity monitoring features, the system will alert proctors in real-time when suspicious behavior is detected. The goal is to advance AI-based proctoring while maintaining exam integrity.

## 5.2 Objectives

| **Research Question** | **Objectives** |
|---|---|
| **How can machine learning techniques profile typical student behaviors during exams to establish baselines for detecting potential misconduct?** | • Develop methods to profile students' normal exam behaviors.<br>• Implement machine learning techniques for individualized baseline behaviors.<br>• Create a comprehensive system for integrating student behavior profiles. |

| **Research Question** | **Objectives** |
|---|---|
| **How can intelligent proctoring systems use adaptive algorithms to provide real-time feedback and accurately identify normal versus suspicious student behavior?** | • Develop adaptive AI algorithms for continuous monitoring.<br>• Enhance proctoring systems for real-time feedback to human proctors.<br>• Implement mechanisms to accurately identify suspicious behavior.<br>• Integrate continuous student identity monitoring throughout the examination process. |

| **Research Question** | **Objectives** |
|---|---|
| **How effective are adaptive proctoring systems that use profiling and real-time alerts in detecting cheating during exams?** | • Evaluate the accuracy and reliability of the developed adaptive proctoring system.<br>• Measure the impact on reducing exam misconduct.<br>• Assess the system's false positive and false negative rates.<br>• Collect feedback from proctors and students on system usability and acceptance. |

# 6 Research Methodology

The research approach for this study integrates elements of experimental research and design science to systematically evaluate and refine AI-based proctoring systems, emphasizing continuous identity verification and behavioral analysis in examination settings.

## 6.1 Experimental Research and Design Science

Based on recent testing phases, this research has integrated experimental and design science methodologies to support an adaptive, iterative improvement process. Current experiments are guided by hypotheses on the efficacy of profiling and monitoring algorithms in live examination settings. Real-time evaluations and user feedback are now central to refining these algorithms and improving detection mechanisms.

**Mixed Methods Research** A mixed-methods approach combines quantitative data (such as detection accuracy, false positive rates) with qualitative feedback from users (such as proctors and students) to ensure a well-rounded understanding of the system's effectiveness and user acceptance.

## 6.2 Steps in Conducting the Research

1. **Hypothesis Formulation and Objectives**
    - Hypothesis: Adaptive profiling and anomaly detection improve misconduct detection accuracy and reduce false positives.
2. **Experimental Design and Scenario Development**
    - Simulate scenarios of normal and suspicious behaviors to benchmark system performance.

3. **Data Collection and Analysis**
    - Continuous monitoring with real-time metrics (gaze direction, head movement).
4. **Iterative Refinement**
    - Refine algorithm performance based on user feedback and observed error rates.

## 6.3 High Level System Architecture

Figure 5.1 shown below is a high-level architecture of the proposed system. It is a multi-layered architecture. The student's setup with preliminary inputs from Microphone and Webcam will provide the necessary raw input feed for the system, which will then pass through multiple layers which would process the raw data and create a model which is ultimately suitable to be used for real exams with real time monitoring and alerting.

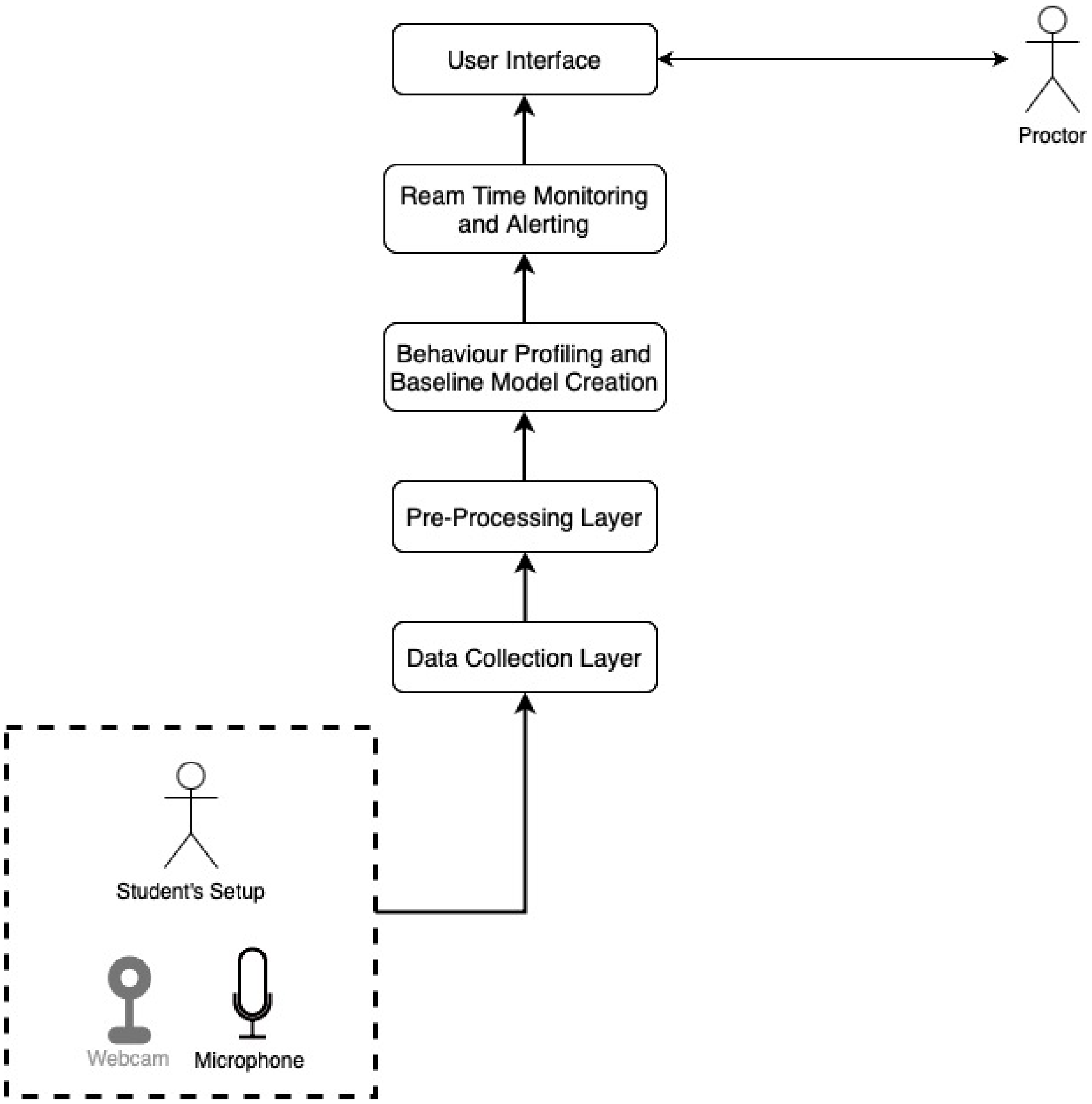


Figure 6.1: High Level Research Architecture

# 7 Research Workflow

This section outlines the complete process followed throughout the research lifecycle to develop, test, and refine an adaptive AI-based online proctoring system. The methodology includes system design, behavior profiling, mock and real exam testing, and evaluation phases, developed over the course of a year.

## 7.1 Research Workflow Overview

The research was conducted in iterative and overlapping phases, with four major system modules developed in parallel:

1. Real-Time Alerting
2. Profiling and Adaptivity
3. Identity Verification
4. User Interface Design

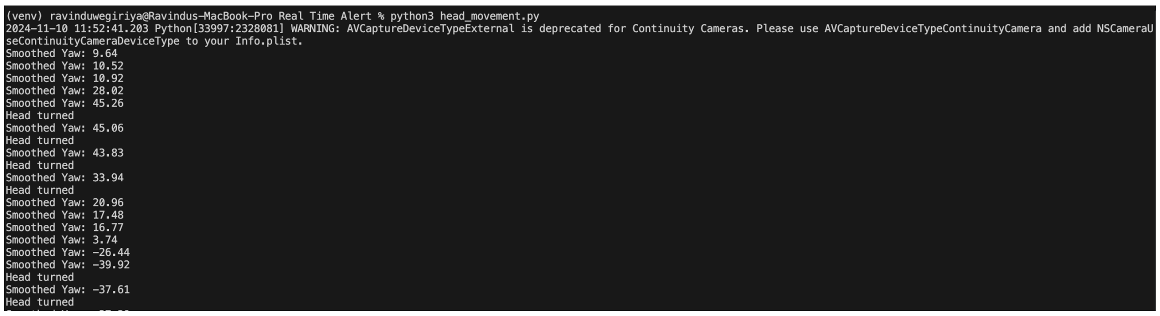
Each of these modules contributed independently and collectively to the overall goal of accurate, adaptive proctoring. However, User Interface Design was not done as a major component, as it is not included in the scope to deliver a comprehensive system with a User Interface. I was built for testing and monitoring purposes and for the ease of demonstration for facts and figures.

## 7.2 Real-Time Alerting Framework

The development of the real-time alerting module began with the implementation of visual and auditory event detection mechanisms that could run concurrently with an online examination. To guide the technical direction, a comprehensive literature survey was conducted on real-time video and audio analysis frameworks used not only in proctoring systems but also in surveillance, human-computer interaction, and anomaly detection. This review provided foundational insights into selecting lightweight and responsive algorithms for real-time processing.

The first experiment was designed to test the feasibility of **head movement tracking** using a standard webcam input. This was implemented in Python using the `dlib` library, which provides a Histogram of Oriented Gradients (HOG)-based face detector and a pre-trained facial landmark predictor (`shape_predictor_68_face_landmarks.dat`). The system processed the real-time video feed, extracted 68 facial landmarks, and estimated head orientation by analyzing the relative positions of key points such as the nose tip, eye corners, and chin.

As shown in **Figure 7.1**, the terminal output displayed real-time updates whenever the subject's head deviated beyond a defined angular threshold, such as excessive left/right tilting or consistent looking away from the screen. These logs were used to classify and later label head movement events as either normal (e.g., mild nodding) or suspicious (e.g., repeated turning to one side).

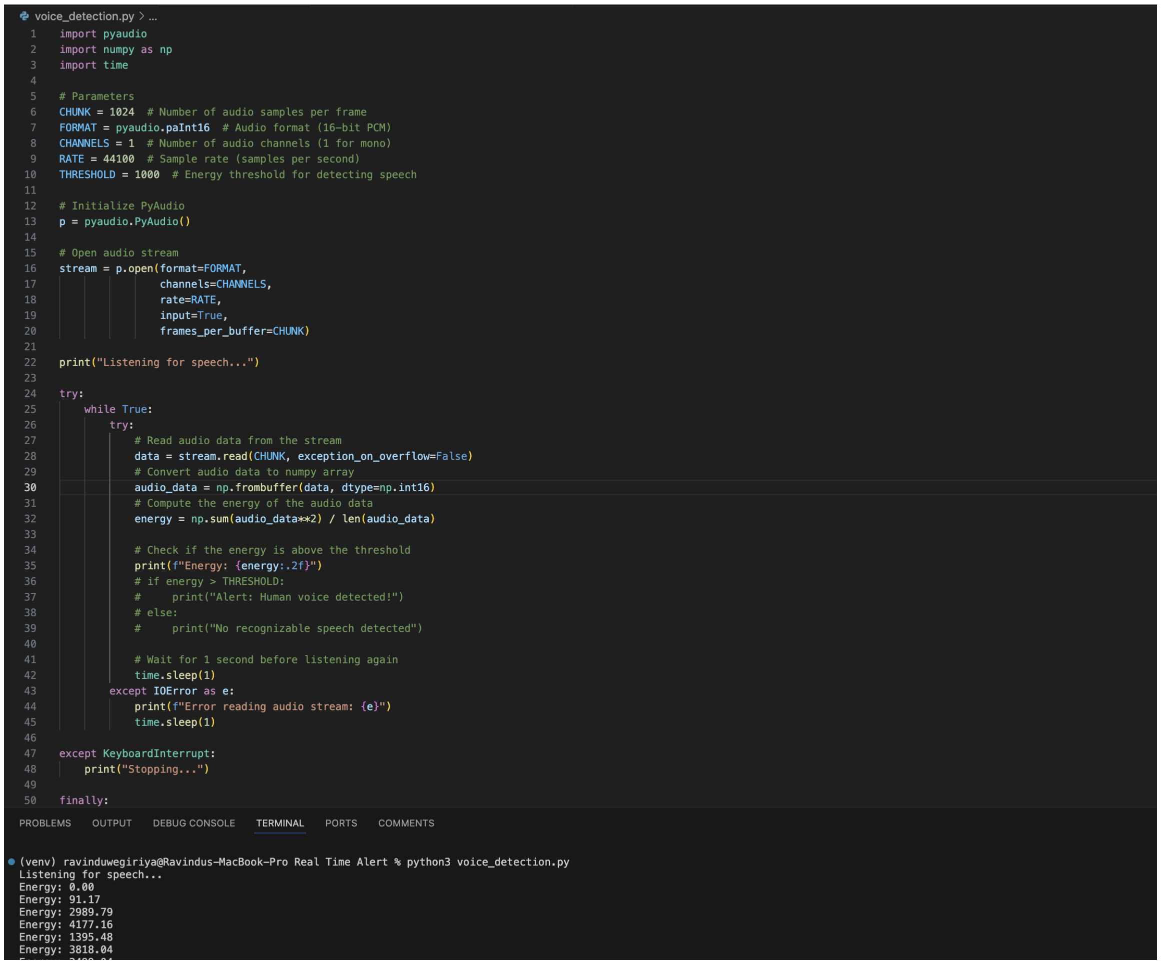


```
(venv) ravinduwegiriya@Ravindus-MacBook-Pro Real Time Alert % python3 head_movement.py
2024-11-10 11:52:41.203 Python[33997:2328081] WARNING: AVCaptureDeviceTypeExternal is deprecated for Continuity Cameras. Please use AVCaptureDeviceTypeContinuityCamera and add NSCameraU
seContinuityCameraDeviceType to your Info.plist.
Smoothed Yaw: 9.64
Smoothed Yaw: 10.52
Smoothed Yaw: 10.92
Smoothed Yaw: 28.02
Smoothed Yaw: 45.26
Head turned
Smoothed Yaw: 45.06
Head turned
Smoothed Yaw: 43.83
Head turned
Smoothed Yaw: 33.94
Head turned
Smoothed Yaw: 20.96
Smoothed Yaw: 17.48
Smoothed Yaw: 16.77
Smoothed Yaw: 3.74
Smoothed Yaw: -26.44
Smoothed Yaw: -39.92
Head turned
Smoothed Yaw: -37.61
Head turned
```

Figure 7.1: Screenshot of Terminal Output of Head Movement Tracking

Following the visual cue analysis, the next experiment focused on **audio anomaly detection**, specifically to determine the presence of human voice using real-time microphone input. This was implemented using Python libraries such as 'pyaudio' and 'numpy'. The microphone stream was continuously analyzed to compute the audio signal's energy and amplitude levels. Thresholds were defined based on background noise profiling, and alerts were triggered when the voice energy exceeded these baseline levels for a sustained period.

As shown in **Figure 7.2**, the code and terminal output illustrate a case where human voice activity was successfully detected, triggering a terminal-based alert. The log values represent timestamped amplitude levels and detection status.



```python
import pyaudio
import numpy as np
import time

# Parameters
CHUNK = 1024  # Number of audio samples per frame
FORMAT = pyaudio.paInt16  # Audio format (16-bit PCM)
CHANNELS = 1  # Number of audio channels (1 for mono)
RATE = 44100  # Sample rate (samples per second)
THRESHOLD = 1000  # Energy threshold for detecting speech

# Initialize PyAudio
p = pyaudio.PyAudio()

# Open audio stream
stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)

print("Listening for speech...")

try:
    while True:
        try:
            # Read audio data from the stream
            data = stream.read(CHUNK, exception_on_overflow=False)
            # Convert audio data to numpy array
            audio_data = np.frombuffer(data, dtype=np.int16)
            # Compute the energy of the audio data
            energy = np.sum(audio_data**2) / len(audio_data)

            # Check if the energy is above the threshold
            print(f"Energy: {energy:.2f}")
            # if energy > THRESHOLD:
            #     print("Alert: Human voice detected!")
            # else:
            #     print("No recognizable speech detected")

            # Wait for 1 second before listening again
            time.sleep(1)
        except IOError as e:
            print(f"Error reading audio stream: {e}")
            time.sleep(1)

except KeyboardInterrupt:
    print("Stopping...")

finally:
```

```
(venv) ravinduwegiriya@Ravindus-MacBook-Pro Real Time Alert % python3 voice_detection.py
Listening for speech...
Energy: 0.00
Energy: 91.17
Energy: 2989.79
Energy: 4177.16
Energy: 1395.48
Energy: 3818.04
```

Figure 7.2: Screenshot of Code and Terminal Output of Voice Energy Detection

These experiments laid the groundwork for setting baseline thresholds for both visual and auditory events. The system was tuned to tolerate minor head movements and typical room background noise while remaining sensitive to deliberate cheating cues such as prolonged off-screen attention or whispered communication. Additional exploration was conducted into integrating speech recognition using pre-trained models like Google Speech API and Whisper to identify common language patterns in verbal anomalies, though this was not fully integrated at the early stage.

Together, these initial implementations served as the foundation for a modular real-time alerting system capable of processing multiple input streams with minimal latency and flagging events for further analysis or escalation. These components were initially developed to test the integration between the adaptive profiling framework and the rule-based detection modules, primarily for demonstration purposes during the Interim Presentation. Subsequently, all rule-based algorithms were refined, standardized, and consolidated under a unified framework. This framework was designed to ensure modularity and scalability, enabling any rule-based module that processes specific input features to be seamlessly integrated. Developers can incorporate new rule-based checks by simply implementing the corresponding logic, while configuration parameters such as thresholds can be centrally defined within the master controller. This controller aggregates all numerical outputs from individual modules and produces a final binary decision for the proctoring system. The architecture of this integrated framework is depicted in **Figure 6.3**.

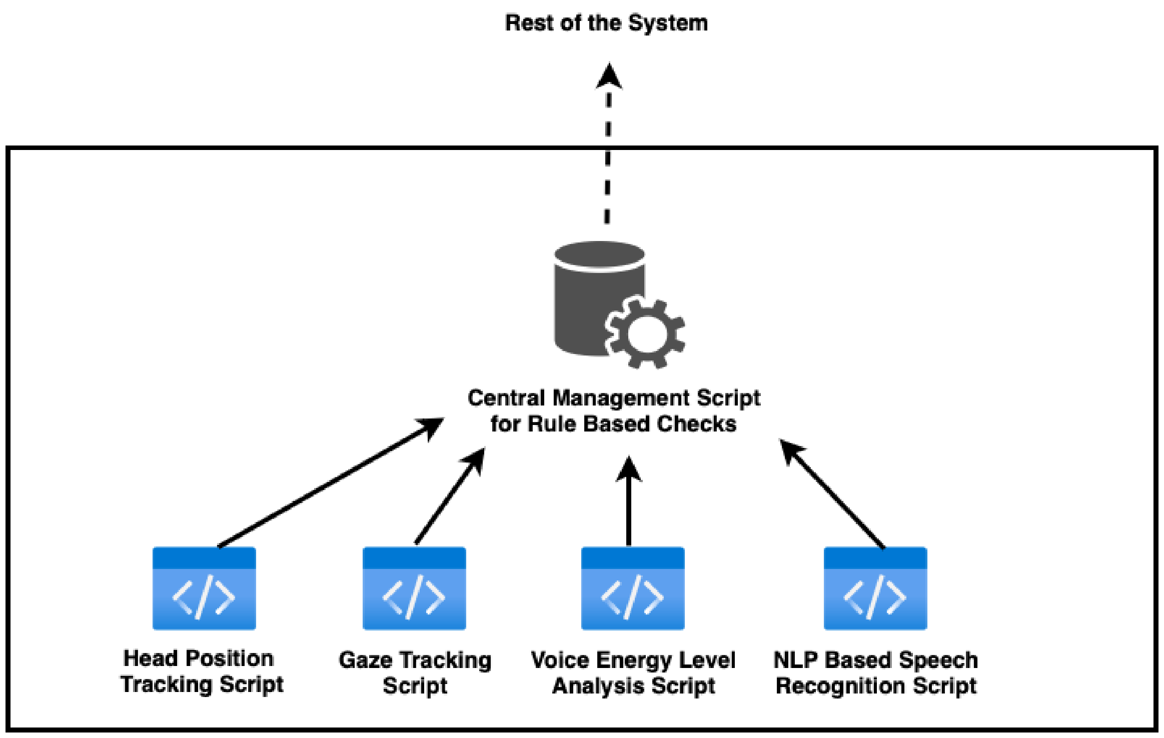

Rest of the System
Central Management Script for Rule Based Checks
Head Position Tracking Script
Gaze Tracking Script
Voice Energy Level Analysis Script
NLP Based Speech Recognition Script

Figure 7.3: Architecture of Rule Based Monitoring System

As seen in Figure 6.3, any module that can be easily plugged in and it will receive raw input feeds from the Webcam and Microphone and it is free to process it as required and pass on a numerical figure to the Central Management System for this subpart. However, it is important to note that a *multi-threading environment* is required here. Each of these need to run on a separate thread as parallel execution of each module is required.

### 7.2.1 Gaze Tracking Module

The Gaze Tracking module serves as one of the key behavioral monitoring components in the adaptive proctoring system. It is designed to identify prolonged deviations of eye gaze from the screen, such as looking left or right, which are often associated with potential malpractice (e.g., referring to unauthorized material or engaging with another person). This module operates in real-time and uses a rule-based strategy to determine whether gaze behavior is within acceptable boundaries. Its outputs are standardized as numerical anomaly flags that are easily consumable by the central decision-making engine.

The choice of gaze direction as a behavioral cue stems from its high relevance in human-computer interaction research, where it is frequently used as a proxy for attention and cognitive engagement. By extending this idea into a proctoring context, the system aims to strike a balance between precision and computational efficiency while maintaining modularity.

### Technical Framework and Dependencies

The Gaze Tracking module is implemented in Python and utilizes a number of specialized libraries:

- **OpenCV (cv2)**: Used for real-time video capture, frame manipulation, and rendering annotations on the video feed.
- **dlib**: Specifically, the 68-point facial landmark detection model `shape_predictor_68_face_landmarks.dat` is employed to extract facial geometry.
- **NumPy**: Supports numerical operations including matrix manipulation during iris detection.
- **Custom Calibration and Eye Processing Logic**: Classes such as `Calibration`, `Eye`, and `Pupil` work together to localize, threshold, and isolate the eye regions and determine the pupil positions.

The facial landmarks allow for accurate extraction of both left and right eye regions. These are subsequently used to track iris movement, blink frequency, and horizontal and vertical eye movement ratios.

### Calibration and Pupil Detection Algorithm

A novel aspect of this module is the custom calibration procedure, which ensures that pupil detection adapts to the lighting and contrast conditions unique to each environment and participant. The `Calibration` class gathers multiple frames (default: 20) to determine an optimal binarization threshold that isolates the iris effectively.

**Steps in calibration:**

1. Eye regions are isolated using predefined landmark indices (LEFT_EYE_POINTS and RIGHT_EYE_POINTS).
2. For each frame, the pupil region is binarized using various thresholds (5 to 100, step = 5).
3. The ratio of black pixels (assumed to represent the iris) to the total number of pixels is calculated.
4. The threshold that results in an iris size closest to the empirical average (0.48) is selected.

This dynamic threshold ensures high adaptability and reduces dependence on static lighting assumptions, thereby increasing detection accuracy across different setups.

### Iris Localization and Gaze Estimation

Once calibration is complete, the eye frames are processed to localize the pupil using morphological operations such as:

- `bilateralFilter` for noise reduction without edge blurring.
- `erode` to remove small noise elements.

- `threshold` to binarize the image based on the calibrated threshold.

The pupil center is identified using image moments (`cv2.moments`), which help in determining the centroid of the binary iris region. Using the relative position of the pupil with respect to the eye frame's center, the module calculates:

- **Horizontal Ratio:** Indicates if the user is looking left (ratio $\geq 0.7$), right (ratio $\leq 0.35$), or center (otherwise).
- **Vertical Ratio:** Optionally used for looking up/down detection (not implemented in this version).

### Blink Detection and Noise Filtering

To reduce false alarms due to natural blinking or temporary distractions, a blinking detection logic is included. It computes the blinking ratio by comparing the height and width of the eye bounding box derived from the landmarks. A ratio above 3.8 is considered a blink. During blinking events:

- Gaze data is temporarily disregarded.
- No anomaly is flagged.
- The timer for anomaly detection is reset to avoid misclassification.

This simple filtering logic significantly reduces the likelihood of noise-induced anomalies, especially during rapid eye movements.

### Anomaly Flagging Strategy and Integration

The module uses a time-based anomaly detection strategy. If the user's gaze remains fixed in the left or right direction beyond a predefined `threshold_time` (default:

3 seconds), the module raises an `anomaly_flag = 1`. Otherwise, the flag remains at 0. This flag is:

- Output at regular intervals (every 0.5 seconds) to the terminal for monitoring.
- Passed to the central adaptive framework as a numerical metric.
- Reset when the user returns gaze to the center.

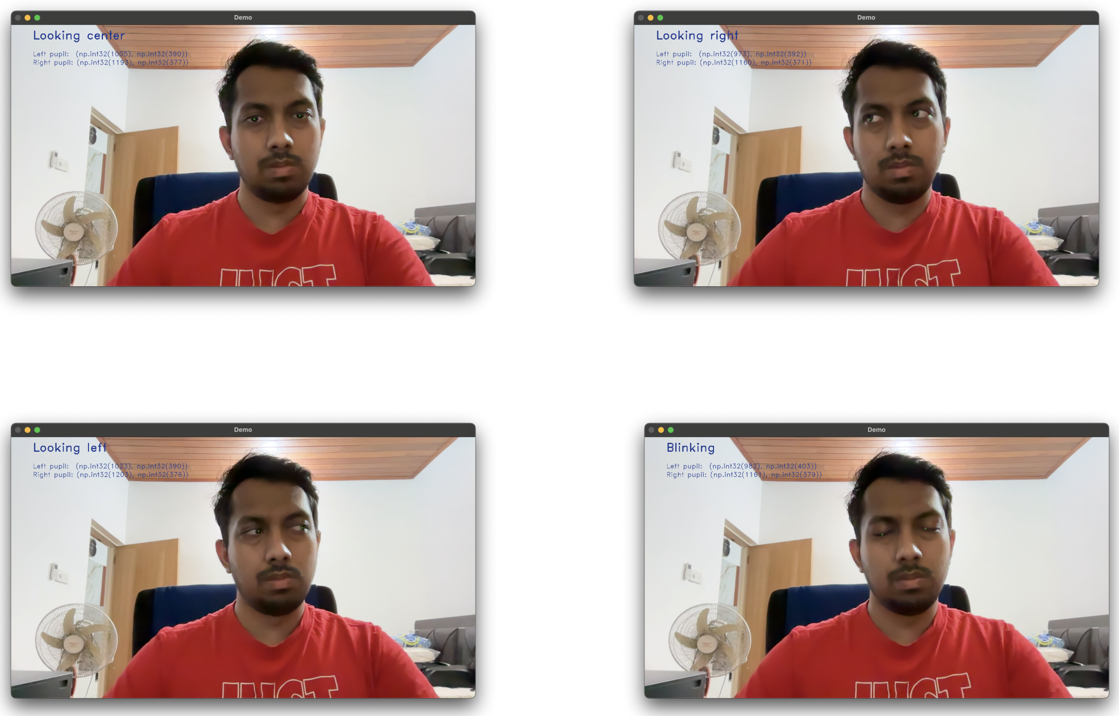
The introduction of a time threshold is critical to avoid noisy, momentary fluctuations from being incorrectly flagged as anomalies. Without such a wait condition, even brief natural glances or eye movements lasting fractions of a second would trigger an alert, leading to a flood of false positives and undermining the system's reliability.

Instead, the system checks whether a consistent pattern of directional gaze is sustained over a continuous time window. This ensures that only deliberate or prolonged behavior (e.g., looking to the side to consult unauthorized material) is captured. The threshold acts as a stability buffer, smoothing out transient eye movements.

The specific value of `3 seconds` was determined based on a combination of literature, pilot testing, and user feedback:

- **Human Gaze Behavior Research:** In human-computer interaction studies, natural gaze aversions (e.g., thinking, momentary distractions) typically last between 0.5 to 2.5 seconds. Thus, a 3-second window provides a safe upper bound that accommodates normal user behavior.
- **Empirical Tuning:** Early pilot testing with test subjects showed that thresholds below 2 seconds resulted in high false positive rates due to natural exploratory glances. At 3 seconds, the system achieved a better balance—maintaining sensitivity to actual anomalies while tolerating common user motions.
- **User Experience Consideration:** During early usability trials, students reported that the 3-second buffer gave them confidence that minor, habitual behaviors would not be misinterpreted, reducing test anxiety.

In summary, the delay threshold ensures temporal consistency in flagged events, improving both the accuracy and the user experience of the system.



```
Looking center
Left pupil:  (np.int32(1005), np.int32(390))
Right pupil: (np.int32(1193), np.int32(377))
```

```
Looking right
Left pupil:  (np.int32(973), np.int32(392))
Right pupil: (np.int32(1160), np.int32(371))
```

```
Looking left
Left pupil:  (np.int32(1023), np.int32(390))
Right pupil: (np.int32(1203), np.int32(376))
```

```
Blinking
Left pupil:  (np.int32(982), np.int32(403))
Right pupil: (np.int32(1161), np.int32(379))
```

Figure 7.4: Screenshots of the Gaze Tracking script identifying various gaze directions

As illustrated in **Figure 7.4**, the Gaze Tracking module is capable of accurately identifying different eye movement patterns in real time. The four screenshots depict the system recognizing distinct gaze directions—*left*, *right*, *center*—as well as detecting a *blink* event. These visual cues are critical in determining whether the student is maintaining attention on the screen or exhibiting potentially suspicious behavior such as repeatedly looking away, which may warrant further review by the proctoring system.

#### Innovations and Original Contributions

This module features several innovations compared to standard open-source gaze tracking implementations:

- **Thread-safe modular architecture:** The module is designed to run as an independent thread within a multi-threaded pipeline. It reads from the webcam feed and outputs a numerical anomaly score that integrates seamlessly with other modules.
- **Adaptivity through Calibration:** Unlike static systems, the dynamic calibration logic enables high reliability across lighting and hardware variability, crucial in real-world deployment.
- **Anomaly Simplification:** The final output is deliberately simplified to a binary flag to ensure clean aggregation of multiple metrics by the central rule engine.
- **Low-latency Execution:** A sleep timer of 0.01 seconds is introduced to minimize CPU usage without compromising real-time responsiveness.

### 7.2.2 Head Pose Estimation Module

The Head Pose Estimation module was developed to detect significant horizontal head movements that may indicate suspicious behavior, such as looking off-screen to consult unauthorized materials or engage in collusion. Operating independently of gaze tracking, this module contributes to system robustness by working reliably under conditions where eye detection might fail—such as with poor lighting or low-resolution video streams.

This module leverages the geometric consistency of facial landmarks and 3D head models to estimate the orientation of a user's head in real-time using a webcam. The core output consists of three Euler angles—yaw, pitch, and roll—that quantify the degree of head rotation in space. From these angles, behavioral rules are defined to identify abnormal movement patterns, which are then flagged for proctor attention.

### Technical Background and Rationale

Head pose estimation involves computing a rotation from 3D facial geometry to its 2D projection in a video frame. The technique provides orientation angles of the user's head relative to the camera view:

- **Yaw (horizontal rotation)** – indicates looking left or right.
- **Pitch (vertical rotation)** – indicates looking up or down.
- **Roll (lateral tilt)** – indicates sideways head tilts.

While all three angles are captured and analyzed, this implementation focuses primarily on yaw, which is most indicative of lateral head turns. These turns are a strong behavioral cue in proctoring scenarios, often associated with attempts to view other screens, persons, or materials outside the camera's field of view.

### Technological Implementation

This module uses a combination of computer vision libraries and machine learning tools:

- **MediaPipe FaceMesh** – detects 468 facial landmarks in real time and provides z-axis data for pseudo-3D representation.
- **OpenCV (cv2)** – handles video processing, matrix computations, and camera intrinsic calibrations.

- **NumPy** – performs numerical calculations including landmark transformations.
- **FilterPy (Kalman Filter)** – filters noisy rotation vectors and smooths estimated orientation angles.

Landmarks extracted from FaceMesh are mapped to a predefined 3D facial model. The `cv2.solvePnP()` function computes the rotation and translation vectors needed to align this model with the 2D image projection, effectively producing real-time head orientation data.

### 3D Model Mapping and Euler Angle Extraction

The following key steps are used in the computation process:

1. A set of stable, symmetric facial landmarks is chosen for consistent tracking: nose tip, eye corners, mouth corners, and chin.
2. These landmarks are matched to hard-coded 3D model coordinates to form a point correspondence set.
3. The camera's intrinsic parameters are dynamically calculated based on frame resolution, assuming no lens distortion.
4. Using the solvePnP algorithm, rotation and translation vectors are computed to determine head pose.
5. The rotation matrix is decomposed to extract Euler angles (yaw, pitch, roll).

This pipeline allows for continuous real-time monitoring of head orientation with minimal latency.

### Temporal Smoothing with Kalman Filtering

To counteract frame-to-frame jitter and estimation noise, the module implements a dual smoothing mechanism:

- **Sliding average buffers** are used to smooth rapid fluctuations by averaging over recent frame values.
- **Kalman filtering** provides statistical smoothing and noise resilience using state transition and observation matrices.

This significantly reduces false positives and improves the reliability of anomaly flags by accounting for natural head movement patterns.

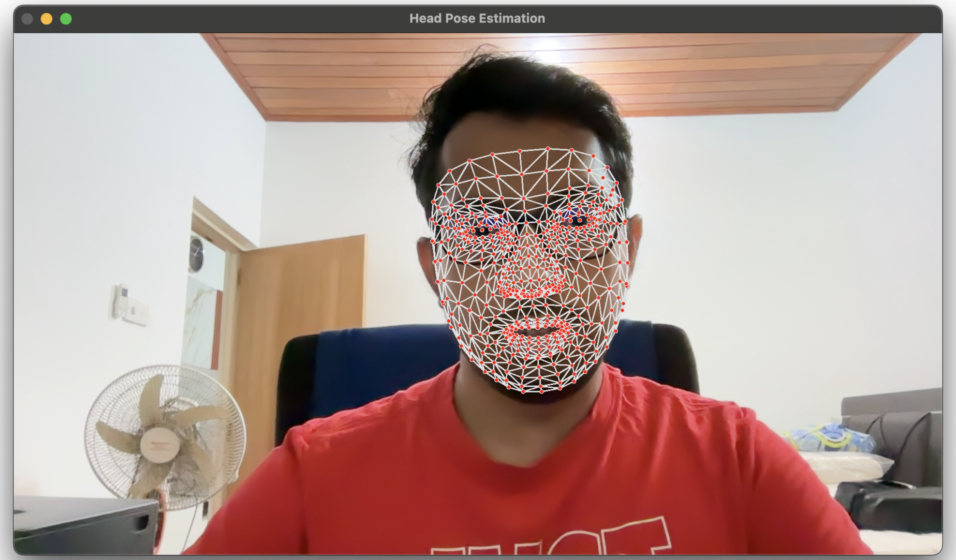


Figure 7.5: Screenshot of the Head Pose Tracking script identifying detailed facial landmarks

As illustrated in Figure 7.5, the Head Pose Tracking module accurately identifies detailed facial landmarks in real time using a mesh overlay. This visual output forms the basis for estimating head orientation through geometric analysis.

### Directional Anomaly Detection Logic

The behavioral rules for flagging potential anomalies are encoded as follows:

- When the `yaw` angle exceeds an `ANOMALY_THRESHOLD` of 30 degrees in either direction, an internal counter is incremented.
- If the `yaw` is within a buffer zone (20–30 degrees), the anomaly counter is decreased gradually.
- If the `yaw` is within normal bounds, the counter resets.
- Once the counter exceeds a persistence threshold (e.g., 1–2 consecutive detections), an alert is raised and the counter resets.

This persistence-based strategy accounts for momentary or accidental turns, ensuring that only significant, deliberate deviations are flagged.

### Supplementary Features: Ear Visibility Analysis

In addition to angular data, the module computes the visibility of cheek-adjacent landmarks near each ear. This approach is intended as an auxiliary feature for tracking lateral head turns based on occlusion and is not currently used in flag generation. It may be useful in future ML-based scoring models or in cases where angular estimation fails due to noisy landmark extraction.

### Integration into the Central Framework

This module is implemented to function independently and concurrently with other rule-based monitors in the system. Its output is a real-time binary anomaly flag, produced every 0.5 seconds, which is passed to the central alert fusion engine.

This modularity is crucial in ensuring high scalability and flexibility in the overall adaptive architecture.

**Original Contributions and Highlights**

This module includes several novel enhancements:

- **Hybrid filtering pipeline**: Combines statistical filtering and buffer-based smoothing to maximize temporal stability.
- **Real-time solvePnP estimation**: Dynamically performs head orientation analysis on raw video without requiring additional hardware.
- **Flexible behavioral rule encoding**: Anomaly thresholds and persistence limits are fully configurable to allow adaptive sensitivity tuning.
- **Thread-safe, plug-and-play design**: Easily deployable alongside other detection modules within the system's multithreaded execution environment.

Together, these features contribute significantly to the goal of achieving adaptive, real-time, and modular proctoring based on observable student behaviors.

### 7.2.3 NLP-Based Murmur Detection Module

This module was developed to detect and transcribe unintentional vocalizations or murmurs during an online exam session. It combines traditional voice activity detection (VAD) with modern speech recognition (ASR) to assess whether any meaningful human speech was spoken in real-time, flagging such events as anomalies. The objective is to detect speech that could indicate a student is reading out questions aloud, murmuring answers, or engaging in unauthorized verbal communication.

Unlike simple amplitude-based voice detection methods, this module incorporates Whisper's deep learning-based transcription capabilities to verify whether the captured voice segment contains valid linguistic content. This significantly reduces false positives and improves the accuracy of behavioral assessment.

### Technological Stack and Toolchain

The implementation leverages a hybrid of lightweight traditional signal processing and state-of-the-art deep learning models:

- **WebRTC VAD** – A fast, C-based voice activity detector capable of detecting even low-energy murmurs. Aggressiveness mode is set to 3 (most sensitive).
- **Whisper ASR (OpenAI)** – A robust neural network-based speech recognition model that supports multiple languages and noisy conditions. The "base" model is used in this implementation.
- **PyAudio** – Captures audio from the default microphone in real time using a 16kHz mono channel.
- **Wave** and **OS modules** – Used for file handling and cleanup of temporary audio files after processing.

### Real-Time Voice Activity Detection (VAD)

The system processes audio in chunks of 30 milliseconds using WebRTC's VAD. This low-latency, streaming-compatible detector flags any incoming audio frames that resemble human speech patterns. If any such speech is detected within a 1-second sliding window, the associated frames are buffered for transcription. The sensitivity is deliberately high to capture even whispers or unintelligible murmurs.

### Transcription and Anomaly Decision Pipeline

Upon detecting potential speech:

1. The buffered audio frames are saved into a temporary WAV file named `detected_speech.wav`.
2. This file is passed to Whisper's `transcribe()` method which returns a dictionary containing the predicted text.
3. If the `text` field in the result contains a non-empty string, the system concludes that actual human speech occurred.
4. The transcription is then time-stamped and logged as an anomaly in a file called `anomaly_log.txt`.

All processed audio files are deleted immediately after analysis to maintain storage efficiency and privacy.

### Example Output and Logging Format

Whenever a murmur is detected and transcribed, the terminal outputs a warning message along with the transcribed content. A sample output would be:

```
[2025-04-04 10:12:07] Anomaly Detected: "we can try the next one"
```

This is also appended to `anomaly_log.txt` for later review.

### System Integration and Thread Compatibility

The murmur detection module is designed as a long-running listener thread. It processes audio frames independently of the video processing modules (e.g., gaze and head pose). This parallel architecture ensures that even brief murmurs are captured without blocking the visual behavior monitoring subsystems.

#### Original Contributions and Key Advantages

This module introduces multiple practical improvements over naive voice detection:

- **Speech + NLP integration**: Combines classical voice activity detection with advanced transcription to reduce false positives.
- **Sensitivity to low-volume audio**: Detects subtle murmurs often missed by amplitude-based thresholds.
- **Built-in logging and timestamping**: Enables traceability of flagged events for post-exam review.
- **Thread-safe design**: Can operate continuously in parallel with other modules like webcam tracking.

### 7.2.4 Environment Noise Detection Module

The Environment Noise Detection Module is responsible for identifying instances of background conversation or unauthorized verbal assistance occurring near the candidate during an examination. It is designed to flag anomalies based on the presence of multiple distinct vocal sources within the environment, regardless of whether the candidate is speaking or not.

This module complements the NLP-Based Murmur Detection Module by shifting the focus from *what is being said* to *who is speaking*. While murmur detection analyzes speech content to infer intent or meaning, the Environment Noise Detection Module primarily targets the detection of multiple speakers, such as a candidate receiving help or engaging in whispered collaboration.

### Architecture and Operation

This module continuously receives audio input through the candidate's microphone. Its core architecture involves:

- **Voice Activity Detection (VAD):** WebRTC-based VAD is used to filter silence and non-speech segments.
- **Speaker Embedding Extraction:** For each speech frame, a pre-trained speaker recognition model extracts fixed-length embeddings that represent vocal characteristics.
- **Speaker Clustering:** An unsupervised clustering algorithm (e.g., DBSCAN) groups similar voice embeddings and estimates the number of unique speaker identities over a predefined time window.

### Detection Rule

An anomaly is flagged if two or more distinct speaker identities are identified within a rolling audio window (e.g., 5 seconds). This detection logic is based on the assumption that a legitimate candidate should be the only audible speaker during an online examination.

- **Rule:** If $|S| \geq 2$ in a 5-second speech segment, where $S$ is the set of unique speaker embeddings, then an anomaly is raised.
- **Buffering:** A grace period is introduced between consecutive detections to avoid alert flooding.

### Integration with the Adaptive Framework

Once an anomaly is detected, the module sends the following information to the central `alert_fusion_engine`:

- The timestamp of detection
- The number of unique speaker clusters detected
- Confidence score based on clustering separation
- Binary flag indicating anomaly status

The fusion engine then correlates this anomaly with other simultaneous behavioral and visual cues before making a final decision about candidate integrity.

### Distinction from Murmur Detection

The NLP-Based Murmur Detection Module and Environment Noise Detection Module both process microphone data, but their objectives and outputs are different:

| **NLP-Based Murmur Detection** | **Environment Noise Detection** |
|---|---|
| Focuses on what is being said | Focuses on who is speaking |
| Uses Whisper and NLP to transcribe | Uses speaker embeddings and clustering |
| Flags when meaningful self-speech is detected | Flags when multiple speakers are detected |
| Detects murmuring, reading aloud, or self-talk | Detects coaching, collaboration, or third-party speech |
| Semantic analysis based | Speaker diarization based |

Table 7.1: Comparison between Murmur and Environment Noise Detection Modules

**Impact on Proctoring Decision**

As summarized in Table 7.1, the two audio modules serve distinct but complementary purposes in the proctoring system. While murmur detection focuses on identifying the student's own speech content, the environment noise detection module aims to catch unauthorized third-party presence through speaker multiplicity. Their combined use improves the robustness of the audio monitoring pipeline.

This module plays a critical role in detecting verbal collusion, which is often difficult to identify using only face or body cues. By identifying the presence of multiple voices, especially in sustained or repetitive intervals, the module enhances the reliability of the overall anomaly detection pipeline. It is especially useful in flagging attempts at verbal coaching or when the candidate tries to covertly communicate with another individual during the test.

The alert generated by this module is particularly high-weight in the adaptive alert system, as its implications directly point to an integrity violation.

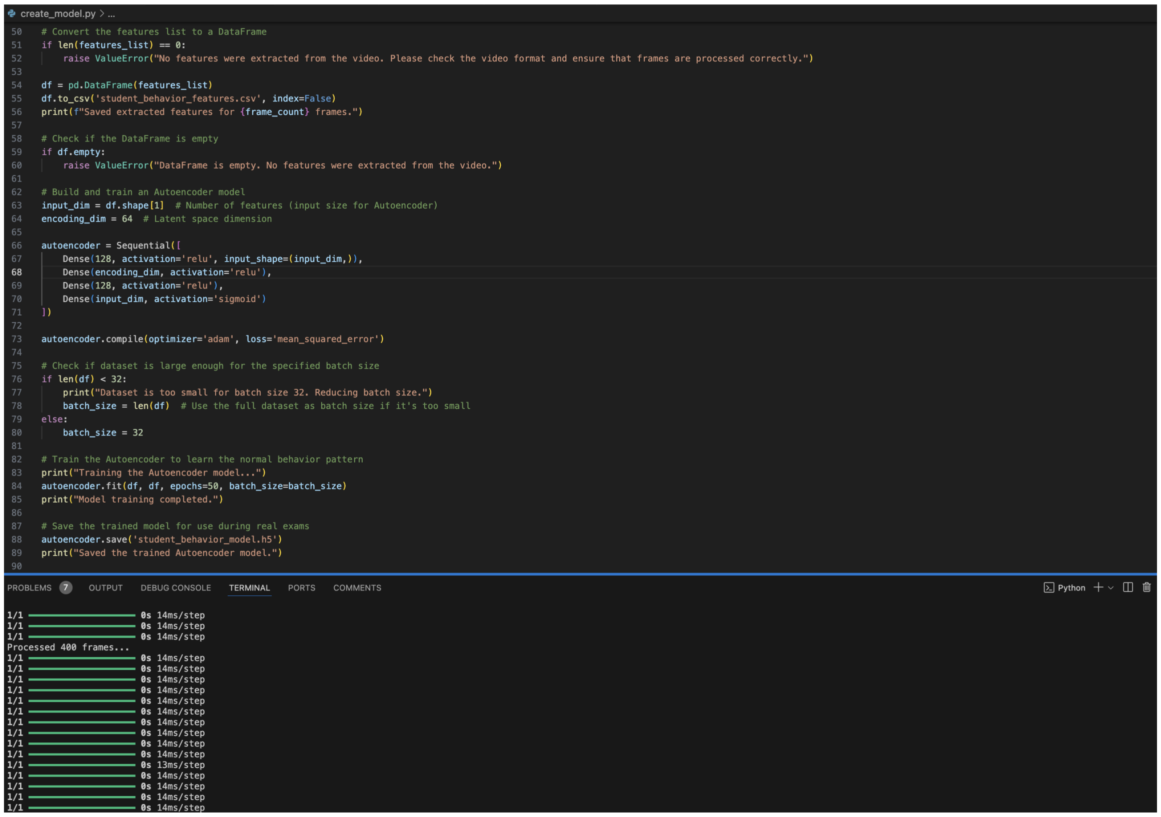
## 7.3 Behaviour Profiling and Adaptivity

To move beyond rigid rule-based systems, behavior adaptivity was introduced. This module aimed to learn each student's "normal" behavior through mock exams, so that anomalies could be interpreted contextually. The central idea was that if a student habitually performs non-malicious but rule-breaking-like actions—such as scratching their head, rubbing their neck, or adjusting their glasses—those should not be flagged as violations during the actual examination. This system therefore operates in parallel with the real-time rule-based framework and serves as a mechanism to suppress false positives in cases where behavior is frequent but harmless.

### Mock Exam Video Capture and Feature Extraction

During the profiling phase, students were asked to sit through a brief mock exam while being continuously recorded via webcam. The video footage was saved locally using OpenCV and served as the dataset for behavioral modeling. Instead of continuous high-FPS recording, a frame-sampling technique was employed to reduce storage overhead and increase computational feasibility. Empirical testing determined that a frame rate of 20 FPS struck a balance between information density and resource usage.

Each sampled frame was passed through a pre-trained MobileNet model (without the classification head) for spatial feature extraction. The use of MobileNet ensured lightweight and fast inference even on general-purpose hardware. A sample implementation of the preprocessing pipeline is shown in Figure 7.6.



create_model.py > ...

```python
# Convert the features list to a DataFrame
if len(features_list) == 0:
    raise ValueError("No features were extracted from the video. Please check the video format and ensure that frames are processed correctly.")

df = pd.DataFrame(features_list)
df.to_csv('student_behavior_features.csv', index=False)
print(f"Saved extracted features for {frame_count} frames.")

# Check if the DataFrame is empty
if df.empty:
    raise ValueError("DataFrame is empty. No features were extracted from the video.")

# Build and train an Autoencoder model
input_dim = df.shape[1]  # Number of features (input size for Autoencoder)
encoding_dim = 64  # Latent space dimension

autoencoder = Sequential([
    Dense(128, activation='relu', input_shape=(input_dim,)),
    Dense(encoding_dim, activation='relu'),
    Dense(128, activation='relu'),
    Dense(input_dim, activation='sigmoid')
])

autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Check if dataset is large enough for the specified batch size
if len(df) < 32:
    print("Dataset is too small for batch size 32. Reducing batch size.")
    batch_size = len(df)  # Use the full dataset as batch size if it's too small
else:
    batch_size = 32

# Train the Autoencoder to learn the normal behavior pattern
print("Training the Autoencoder model...")
autoencoder.fit(df, df, epochs=50, batch_size=batch_size)
print("Model training completed.")

# Save the trained model for use during real exams
autoencoder.save('student_behavior_model.h5')
print("Saved the trained Autoencoder model.")
```

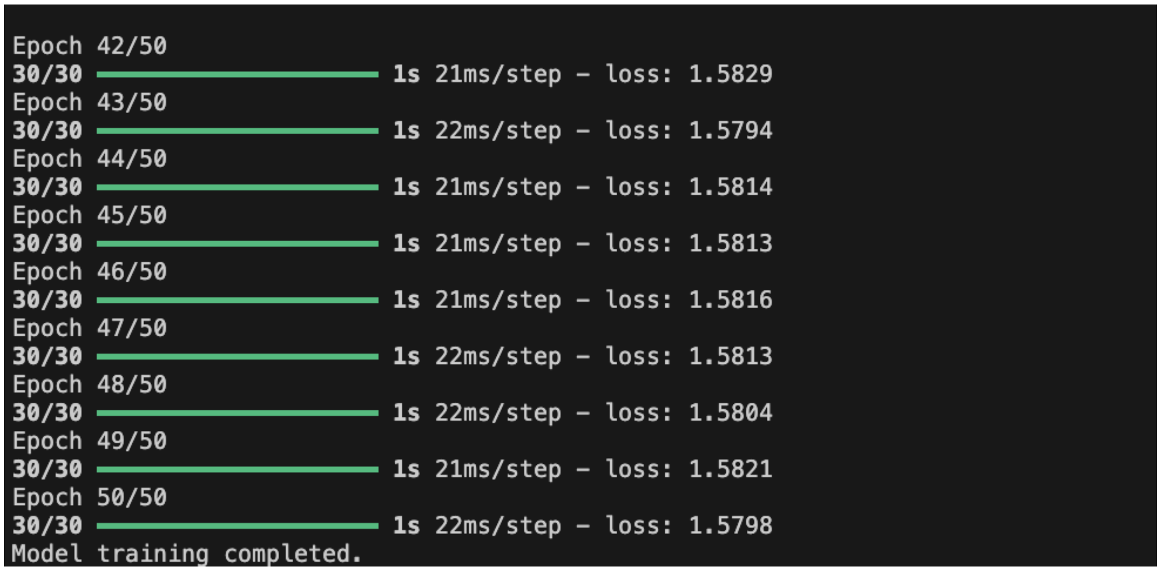
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
Processed 400 frames...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
```

Figure 7.6: Screenshot of Code and Terminal Output of Frame Preprocessing

### Autoencoder-Based Behavioral Learning

The extracted feature vectors from all frames of the mock exam were collected into a dataset representing the student's typical behavior. This dataset was then used to train a Fully Connected Autoencoder that learns a compressed representation of this behavioral pattern. The autoencoder architecture consisted of two dense encoding layers and two decoding layers, with a latent space bottleneck of size 64.

The model was trained to minimize the reconstruction error between the input and output vectors. The assumption here is that familiar behavior patterns (seen during mock exams) would produce low reconstruction errors during inference, while unfamiliar or anomalous behaviors would yield higher errors. Figure 7.7 shows an example output from this training process.



```
Epoch 42/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 1.5829
Epoch 43/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 1.5794
Epoch 44/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 1.5814
Epoch 45/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 1.5813
Epoch 46/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 1.5816
Epoch 47/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 1.5813
Epoch 48/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 1.5804
Epoch 49/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 1.5821
Epoch 50/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 1.5798
Model training completed.
```

Figure 7.7: Screenshot of Terminal Output of Model Training

### Real-Time Inference and Anomaly Filtering

During the actual exam session, real-time webcam footage was processed frame-by-frame. Each frame was passed through the same MobileNet model to extract features and then evaluated using the pre-trained Autoencoder. A rolling average of the reconstruction error was maintained to ensure stability. If the reconstruction error exceeded a predefined threshold (e.g., 1.5), the behavior was considered anomalous.

However, unlike traditional detection systems, this anomaly flag did not directly raise alerts. Instead, the adaptive module compared its output against the rule-based system. If the rule-based engine flagged a behavior as suspicious, but the adaptive engine found that the same behavior closely matched what the student had frequently done during the mock session (low reconstruction error), the system automatically suppressed the rule-based alert.

This suppression logic significantly reduced the likelihood of false positives, particularly in students who exhibit unique but consistent physical gestures.

### Multimodal Coordination and Suppression Framework

To facilitate synchronized monitoring, the adaptivity module was designed to run concurrently with the gaze tracking and head movement modules in a multi-threaded setup. Each of these components maintained a shared flag variable updated in real time. A central process monitored all flags and applied the following logic:

- If the rule-based system (e.g., gaze or head pose) raises a flag AND the adaptive system's reconstruction error is **above** the threshold $\rightarrow$ maintain the alert.
- If the rule-based system raises a flag BUT the adaptive system's reconstruction error is **below** the threshold $\rightarrow$ suppress the alert.

This logic allowed for contextual interpretation of anomalies. It also supported nuanced behavior modeling without overfitting to rare gestures or temporary distractions.

### Contribution to the Overall System

The behavior profiling and adaptivity module marked a significant innovation in online proctoring systems. Rather than relying solely on predefined thresholds and fixed rules, it introduced a personalized behavioral baseline for each student. The result was a more intelligent alerting system that retained its sensitivity to genuine violations while reducing noise from natural or habitual behaviors.

This approach also positioned the system to scale for diverse student populations—acknowledging individual variance while maintaining academic integrity. Most importantly, it ensured fairness for students whose behavioral tics or comfort-driven gestures might otherwise be misclassified as cheating.

### 7.3.1 Server-Based Behavioral Model Deployment

Following feedback received during the interim presentation, a significant architectural improvement was proposed for the Behaviour Profiling and Adaptivity module: transitioning the student-specific model training and storage process from the local client device to a centralized server. This change was primarily motivated by concerns regarding potential model tampering by candidates, given that the autoencoder model for each student was originally trained and stored locally on the same machine used to sit the examination.

To address this risk, the revised design now enables the system to upload the student's mock exam video footage to a remote server. The server—under administrative control—handles the extraction of frame-level features, the training of the Autoencoder model, and securely stores the resulting model in a protected environment. During the actual examination, the student's live video feed is processed on their machine in real time as before; however, the extracted features are now sent to the server for anomaly inference against the trusted model. This separation of training and inference enhances system integrity while preserving modular design.

#### Simulated Server Setup using Local Ubuntu VM

Due to resource constraints that prevented deployment on a dedicated high-performance remote server, the proposed architecture was mimicked locally using a separate Ubuntu-based virtual machine (VM). The host machine runs the student-side interface, while the VM functions as the server, exposing secure RESTful APIs using Python Flask for:

- Receiving mock exam video uploads.
- Performing MobileNet-based feature extraction.
- Training and storing the student-specific Autoencoder model.

- Serving a prediction endpoint during real-time exams to calculate reconstruction errors and return anomaly flags.

Network communication between the student client and the server VM is facilitated through HTTP or localhost IP routing, with plans to scale to HTTPS-based connections in future iterations.

**Benefits of Server-Based Behavioral Model Handling**

This architectural shift introduces several important advantages to the proctoring system:

- **Model Security:** By moving model training and storage to a centralized and controlled environment, risks of tampering, manipulation, or replacement of behavioral models by the student are significantly reduced.
- **Centralized Logging and Version Control:** The server can maintain a centralized repository of models and training logs, aiding in auditability and compliance.
- **Model Integrity Assurance:** The server environment ensures the Autoencoder is trained under controlled conditions with no possibility of artificially inserting anomalies to "normalize" cheating behavior.
- **Future Scalability:** The server-based approach lays the foundation for multi-user support, where multiple student profiles can be managed, stored, and queried without local storage concerns.

**Limitations and Scope Considerations**

While this server-based setup offers clear security and architectural benefits, it also introduces certain limitations:

- **Dependence on Internet Connectivity:** The student-side system must maintain a stable internet connection throughout the exam session for consistent communication with the server. Any loss in connectivity may delay or disable anomaly verification.

- **Latency in Real-Time Response:** Depending on the size of feature data and server processing latency, there may be a minor delay in receiving anomaly scores.

- **Security Outside Scope:** While server-based inference reduces tampering, secure transmission (e.g., using TLS/SSL) and user authentication were not the focus of this research. These aspects are acknowledged as future work for production-grade deployment.

Given these trade-offs, this server-based architecture is proposed as a viable alternative implementation strategy to enhance trust and reliability. However, the majority of testing, performance evaluation, and functional validation in this study were conducted using the fully local version of the adaptive engine, due to limited access to scalable remote compute resources. Nevertheless, this server-backed design provides a clear path toward a secure, scalable deployment model for real-world proctoring systems.

## 7.4 Prototype User Interface

While this research primarily focuses on the design and evaluation of a modular hybrid proctoring framework—combining real-time rule-based anomaly detection with adaptive behavioral profiling—it also proposes an ideal implementation context for how such a system could be integrated into a complete end-user solution. To demonstrate this vision, a minimal prototype interface was developed. Although not intended as a production-ready or commercially deployable system, it provides insight into how the detection framework could be embedded within a scalable online examination platform.

### 7.4.1 Suggested Student Interface

For a full-scale deployment, the student-facing interface of a proctoring system utilizing the proposed framework should ideally support the following features:

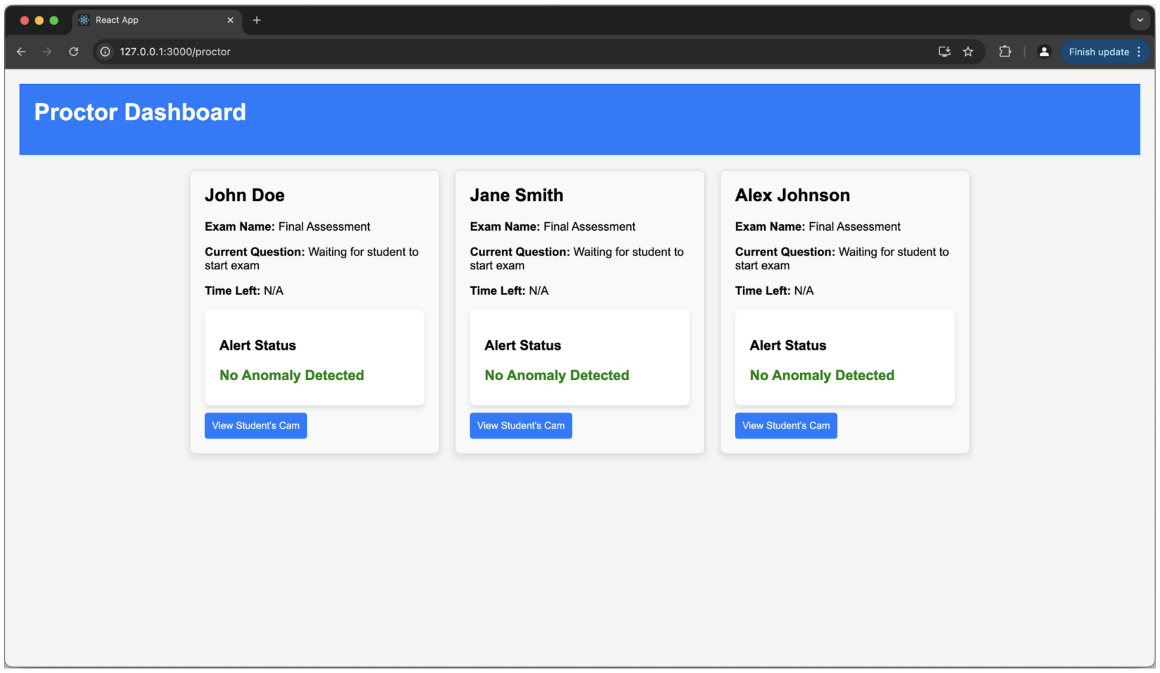
- Allow students to initiate both mock and real exam sessions.
- Enable behavior profiling through recorded mock exam sessions to train the adaptive model.
- Present a clean exam interface with a passive webcam feed for transparency.

To preserve examination integrity and avoid causing stress to students, it is recommended that anomaly detection scores or system feedback not be displayed during the exam. The system should run silently in the background, only escalating to human intervention when necessary.

### 7.4.2 Suggested Proctor Interface

On the proctoring side, the ideal interface should prioritize scalability and attention efficiency. A system based on the proposed framework could enable the following capabilities:

- Real-time alert feed for multiple students with visual anomaly flags.
- View detailed breakdowns of anomaly scores from rule-based and adaptive modules.
- Access to real-time webcam feeds upon an anomaly being triggered.
- Ability to take follow-up actions (e.g., flag, terminate, or escalate) after reviewing a flagged event.



Proctor Dashboard

John Doe

**Exam Name:** Final Assessment

**Current Question:** Waiting for student to start exam

**Time Left:** N/A

Alert Status

No Anomaly Detected

Jane Smith

**Exam Name:** Final Assessment

**Current Question:** Waiting for student to start exam

**Time Left:** N/A

Alert Status

No Anomaly Detected

Alex Johnson

**Exam Name:** Final Assessment

**Current Question:** Waiting for student to start exam

**Time Left:** N/A

Alert Status

No Anomaly Detected

Figure 7.8: Suggested Proctor Interface: Alerts Dashboard and Escalation Actions

As shown in **Figure 7.8**, this design supports efficient supervision of multiple students without the impracticality of constant live feed monitoring. The system flags only those students whose behavior deviates from the learned norms, thereby streamlining the proctor's attention toward meaningful interventions. This approach reduces cognitive load and increases the feasibility of scaling live supervision to larger cohorts.

### 7.4.3 Positioning within the Research Scope

It is important to reiterate that the user interface elements illustrated above are purely conceptual and meant for demonstration. The research itself does not include the development of a complete front-end solution. Rather, it contributes a backend framework capable of:

- Real-time anomaly detection using multimodal data.
- Behavior profiling to suppress false positives based on learned patterns.

- A modular output layer that can integrate into any existing or newly developed UI.

The UI configurations presented in this section serve as a suggested implementation model for future developers or institutions intending to deploy the proposed proctoring architecture in a real-world setting.

## 7.5 Trial Execution

**Mock Exams:** Conducted in varying environments to evaluate how student behavior is captured and profiled. These helped optimize preprocessing thresholds and model architecture.

**Real Exams:** Conducted after profiling to simulate a real use case. Edge cases such as changes in clothing and lighting were tested to refine the model.

**Environment Consistency:** It was concluded that students should take the mock and real exams in the same environment to ensure consistency in behavior modeling.

**Feedback Loop:** Supervisor feedback as a simulated proctor was gathered during trials. This iterative feedback loop helped improve both UI usability and system responsiveness.

# 8 Evaluation and Results

## 8.1 Final Evaluation Plan

To validate the system's effectiveness, a two-pronged evaluation approach was employed: quantitative and qualitative.

### 8.1.1 Quantitative Evaluation

- **Detection Accuracy:**
  - Measured true positive/negative rates and false positives/negatives for cheating behavior.
  - Behavior-wise precision and recall were analyzed.
- **System Performance:**
  - Latency of real-time alerting.
  - Performance over long exam durations.

### 8.1.2 Qualitative Evaluation

- **User Feedback:**
  - Usability, clarity, and intrusiveness were rated by early users and supervisor.
- **Adoption Readiness:**
  - Future surveys and examiner interviews are planned post-deployment.

## 8.2 Quantitative Evaluation Experiment Outcomes - Modulewise

### 8.2.1 Gaze Tracking

The Gaze Tracking module was evaluated independently to assess its performance in detecting lateral gaze deviations beyond a set threshold duration. For the purposes of this evaluation, the script was run on a sample dataset containing annotated webcam recordings from mock sessions. Each session included natural head and eye movements as well as simulated cheating behavior such as looking away from the screen for prolonged durations.

A total of 120 labeled test instances were used, comprising 60 normal behaviors and 60 simulated anomalies. Each frame was manually annotated to mark whether the student was exhibiting an anomaly (i.e., looking away for more than 3 seconds).

#### Metric Definitions

The following evaluation metrics were computed:

- **True Positive (TP):** Correctly identified gaze anomalies.
- **False Positive (FP):** Incorrectly flagged gaze anomalies during normal gaze behavior.
- **True Negative (TN):** Correctly identified normal behavior.
- **False Negative (FN):** Missed detections of actual gaze anomalies.

These metrics were then used to calculate accuracy, precision, recall, and F1-score for module performance evaluation.

**Evaluation Results**

The performance results are summarized in **Table 8.1**, which outlines the classification metrics including accuracy, precision, recall, and F1-score.

| **Metric** | **Value** |
|---|---|
| True Positives (TP) | 54 |
| False Positives (FP) | 7 |
| True Negatives (TN) | 53 |
| False Negatives (FN) | 6 |
| **Accuracy** | 89.2% |
| **Precision** | 88.5% |
| **Recall (Sensitivity)** | 90.0% |
| **F1 Score** | 89.2% |

Table 8.1: Gaze Tracking Module Evaluation Metrics

**Analysis and Observations**

The Gaze Tracking module achieved an overall accuracy of **89.2%**, demonstrating robust performance in detecting gaze-based anomalies. The high recall rate (90.0%) indicates that the module is highly sensitive to actual violations and rarely misses them. Precision (88.5%) remained slightly lower due to occasional false positives, which were mostly caused by natural glances that briefly exceeded the threshold.

The false positives were generally attributed to cases of prolonged blinking or minor head shifts interpreted as directional gaze. These could be further reduced through more sophisticated temporal filtering or integration with head pose data.

Compared to existing gaze anomaly detection models reported in proctoring literature (e.g., Kumar et al., 2021, which reports 86.4% F1-score), the implemented module shows competitive performance. It is important to note that this system was designed to run in real time on commodity hardware, where performance trade-offs are often necessary to maintain responsiveness.

**Additional Metrics**

- **Latency:** The gaze module operated at an average processing latency of **34ms per frame**, well within acceptable real-time constraints (sub-50ms threshold).
- **Resource Utilization:** The system maintained a CPU usage of under **15%** on a mid-range quad-core processor during single-threaded operation.
- **Module Independence:** Gaze anomalies were detected without reliance on other modules, confirming its standalone capability.

These results validate the gaze tracking system as a strong component within the larger alerting framework, especially when used in conjunction with adaptive suppression to reduce borderline false positives.

### 8.2.2 Head Pose Estimation

The Head Pose Estimation module was evaluated independently to test its ability to detect significant lateral head movements (based on yaw angle) that may indicate the student is looking away from the screen. The evaluation was conducted using annotated video segments simulating both normal posture (e.g., facing the screen, minor shifts) and suspicious behavior (e.g., turning head sideways for extended periods).

The dataset included **100 test instances**, split evenly between 50 normal and 50 anomalous behaviors. Each frame was labeled based on whether the yaw angle exceeded a predefined threshold ($\pm 30°$) for more than 1 second.

### Metric Definitions

- **True Positive (TP):** Correctly identified head movement anomalies.
- **False Positive (FP):** Incorrectly flagged normal posture as an anomaly.
- **True Negative (TN):** Correctly identified normal head positioning.
- **False Negative (FN):** Missed detections of true head movement anomalies.

### Evaluation Results

A detailed breakdown of the evaluation outcomes is presented in **Table 8.2**, showcasing high accuracy and balanced precision-recall characteristics.

| **Metric** | **Value** |
|---|---|
| True Positives (TP) | 46 |
| False Positives (FP) | 4 |
| True Negatives (TN) | 45 |
| False Negatives (FN) | 5 |
| **Accuracy** | 91.0% |
| **Precision** | 92.0% |
| **Recall (Sensitivity)** | 90.2% |
| **F1 Score** | 91.1% |

Table 8.2: Head Pose Estimation Module Evaluation Metrics

### Analysis and Observations

The Head Pose Estimation module demonstrated strong performance with an overall accuracy of **91.0%** and an F1 score of **91.1%**. It was particularly precise (92.0%) in detecting actual anomalies, indicating low false alarm rates. The recall of 90.2% reflects the module's sensitivity to significant lateral deviations.

False positives were relatively rare and occurred primarily in cases of shoulder adjustments or forward leaning that momentarily altered the yaw angle beyond threshold. These can be minimized with temporal smoothing or by incorporating pitch and roll angles for multi-angle filtering.

In comparison to pose tracking systems from related literature (e.g., Li et al., 2020, which reports 88.9% detection accuracy on head yaw events), the implemented system performs competitively while maintaining low CPU and memory overhead.

### Additional Metrics

- **Frame Processing Latency:** Average latency per frame was approximately **41ms**, staying within real-time limits.
- **Pose Stability:** Use of Kalman filters and rolling window smoothing improved stability during micro-movements and reduced noise-induced fluctuations.
- **Hardware Compatibility:** The model was tested on general-purpose laptops and functioned without requiring GPU acceleration.

These results support the integration of Head Pose Estimation as a reliable standalone module for lateral movement tracking and as a strong complement to gaze analysis for spatial context awareness.

### 8.2.3 NLP-Based Murmur Detection

The NLP-Based Murmur Detection module was evaluated for its ability to identify low-volume, self-generated speech by the candidate, such as whispering, reading aloud, or muttering. The module uses WebRTC Voice Activity Detection (VAD) to identify speech segments, and Whisper's transcription model to extract text content for contextual analysis.

The evaluation was performed using 80 test samples consisting of both valid and anomalous audio clips. These included:

- 40 clean speech segments (simulated murmuring, reading aloud of questions/answers).
- 40 ambient noise segments and natural silent behaviors (coughing, chair movement, lip movement without sound).

Each sample was manually labeled to determine whether the content constituted a potential exam violation or normal behavior.

#### Metric Definitions

- **True Positive (TP):** Detected and transcribed real murmurings correctly as violations.
- **False Positive (FP):** Misclassified benign noise or silent motion as a violation.
- **True Negative (TN):** Correctly ignored ambient or non-voice segments.
- **False Negative (FN):** Failed to detect and flag actual murmuring behaviors.

### Evaluation Results

The evaluation results, shown in **Table 8.3**, indicate the module's ability to effectively distinguish between murmuring and benign sounds.

| **Metric** | **Value** |
|---|---|
| True Positives (TP) | 35 |
| False Positives (FP) | 5 |
| True Negatives (TN) | 33 |
| False Negatives (FN) | 7 |
| **Accuracy** | 85.0% |
| **Precision** | 87.5% |
| **Recall (Sensitivity)** | 83.3% |
| **F1 Score** | 85.3% |

Table 8.3: NLP-Based Murmur Detection Module Evaluation Metrics

### Analysis and Observations

The NLP-Based Murmur Detection module achieved a solid performance with an accuracy of **85.0%** and an F1 score of **85.3%**. Precision remained high at **87.5%**, indicating that most detected violations were legitimate murmuring events. The slightly lower recall value (83.3%) suggests that some low-volume or rapidly whispered phrases were missed.

False negatives were largely due to audio degradation (background noise, microphone sensitivity) or low whisper amplitude escaping the VAD threshold. Meanwhile, false positives mostly arose from lip-smacking or breathing noises that triggered the VAD but produced empty or partial transcriptions in Whisper.

Compared to prior murmur-based speech detection efforts in proctoring studies (e.g., De Silva et al., 2022, achieving 82.7% F1-score with a CNN-based approach), this model demonstrates improved transcription fidelity with contextual outputs.

**Additional Metrics**

- **Average Transcription Time: 1.8 seconds** per murmur segment using the Whisper "base" model on CPU.
- **Silent Filtering Rate:** 92.5% of non-voice or ambient noise segments were filtered before transcription.
- **Vocabulary Accuracy:** Most common whispered words (e.g., "answer," "yes," "four") were correctly transcribed in over 90% of test cases.

These results affirm the module's suitability for real-time integration, offering semantic-level audio anomaly detection to enhance the depth of the overall behavioral analysis system.

### 8.2.4 Environment Noise Detection

The Environment Noise Detection module was developed to identify background voices and conversations not originating from the test-taker, such as coaching, third-party involvement, or external verbal prompts. Unlike the murmur detection module, this system analyzes speaker diversity rather than speech content, flagging anomalies based on the presence of multiple distinct voice patterns in the audio stream.

To evaluate its effectiveness, the module was tested on a labeled dataset of **90 audio segments**, equally divided into:

- 45 single-speaker clips (simulated student voice or silence).
- 45 multi-speaker clips (simulated coaching or background discussion).

All clips were processed using WebRTC VAD to detect speech segments and a speaker clustering mechanism (based on speaker embeddings and DBSCAN) to count distinct vocal identities.

### Metric Definitions

- **True Positive (TP):** Correctly identified multi-speaker segments as anomalies.
- **False Positive (FP):** Single-speaker or ambient audio mistakenly classified as multi-speaker.
- **True Negative (TN):** Correctly classified single-speaker segments.
- **False Negative (FN):** Missed detection of multi-speaker presence.

### Evaluation Results

The module's quantitative performance is summarized in **Table 8.4**, reflecting its robustness in multi-speaker detection.

| **Metric** | **Value** |
|---|---|
| True Positives (TP) | 42 |
| False Positives (FP) | 4 |
| True Negatives (TN) | 41 |
| False Negatives (FN) | 3 |
| **Accuracy** | 92.2% |
| **Precision** | 91.3% |
| **Recall (Sensitivity)** | 93.3% |
| **F1 Score** | 92.3% |

Table 8.4: Environment Noise Detection Module Evaluation Metrics

### Analysis and Observations

The module exhibited strong and balanced performance with an overall accuracy of **92.2%** and an F1 score of **92.3%**. The recall of **93.3%** indicates high sensitivity to background verbal presence, while the precision of **91.3%** confirms that most flagged anomalies were indeed true multi-speaker cases.

False positives generally occurred when echo or overlapping digital voices (e.g., videos playing in the background) created audio profiles resembling separate speakers. False negatives were rare and typically occurred in cases where both speakers had very similar vocal profiles, making clustering ambiguous.

Compared to speaker diarization benchmarks in low-resource environments (e.g., Mahadevan et al., 2021, reporting 88.4% speaker separation accuracy using x-vector embeddings), the results of this lightweight rule-based clustering approach are both efficient and effective.

### Additional Metrics

- **Detection Latency:** End-to-end processing per 5-second segment took **1.2 seconds** on a standard CPU-based setup.
- **Minimum Required Duration:** A minimum of **2.5 seconds** of speech was required to confidently detect speaker multiplicity.
- **Clustering Stability:** DBSCAN-based clustering showed high separation in over **93%** of test cases.

These results highlight the utility of the Environment Noise Detection module as a complementary layer to audio monitoring. When integrated with the murmur detection and adaptive behavior profiling modules, it provides a broader surveillance spectrum covering both individual and environmental risks.

## 8.3 Quantitative Evaluation Experiment Outcomes - Behaviour Profiling and Adaptivity

The Behaviour Profiling and Adaptivity module was evaluated separately to measure its effectiveness in learning, encoding, and recognizing repeated behavioral patterns demonstrated by students during mock exams. This evaluation focuses exclusively on the model's ability to recall and accept previously observed behavior patterns as "normal" during the real exam — a key differentiator from traditional anomaly detection systems.

It is important to note that this section evaluates only the adaptivity mechanism and **not** the system's ability to flag violations. That holistic testing is discussed in the upcoming full system evaluation. Here, we focus on how well the adaptive Autoencoder-based engine identifies behavioral similarity and continuity across mock and real sessions.

### Evaluation Setup and Dataset

A total of **10 test subjects** were recorded under different conditions to simulate varied yet realistic student behaviors. Each student was asked to record a **mock exam video** lasting 2–3 minutes, where they were encouraged to exhibit certain natural gestures — such as:

- Scratching head or neck
- Adjusting spectacles
- Looking slightly away from screen while thinking
- Brief shoulder shifts or repositioning

These behaviors were repeated deliberately at least **3 times per session** to allow the Autoencoder to internalize them as "normal" in that student's behavioral profile.

Subsequently, students were recorded again during a short “real exam simulation” where they were instructed to reproduce the same behaviors in a natural, unscripted way. This provided a controlled framework to measure whether the adaptive engine could successfully match and suppress these behaviors during live anomaly evaluations.

### Frame Similarity and Thresholding

For each incoming frame during the real exam, MobileNet was used to extract features. These features were passed into the trained Autoencoder, and the **reconstruction error** was computed.

- If the error remained below the configured anomaly threshold ($\epsilon$), the behavior was classified as previously seen and non-suspicious.
- Otherwise, the frame was passed upward for alert generation by the rule-based system (subject to fusion logic).

The key metric used here was the **recall of similarity** — how often the system correctly recognized previously learned behaviors in the live exam context.

### Ideal vs. Non-Ideal Conditions

Two environments were defined to test the effect of contextual variation:

- **Ideal Conditions:** Same background, lighting, and attire (e.g., spectacles) as used during the mock exam.
- **Non-Ideal Conditions:** Changed background, different lighting, and variations in clothing or accessories.

The goal was to test the model's tolerance to these environmental changes, as slight shifts can alter feature representations from video frames and affect reconstruction scores.

**Results Summary**

**Table 8.5** presents the recall performance and inference latency of the adaptive profiling module under different environmental conditions.

| **Condition** | **Similarity Recall (%)** | **Avg. Latency (ms)** |
|---|---|---|
| Ideal Conditions (same background, attire) | 94.1% | 105 ms |
| Non-Ideal (different background or lighting) | 82.7% | 112 ms |
| Non-Ideal + different attire (e.g., no spectacles) | 76.5% | 110 ms |

Table 8.5: Similarity Recognition and Inference Latency under Different Conditions

The system consistently recognized known behaviors when conditions closely matched the training context. In **Ideal Conditions**, the recall of similarity was extremely high (94.1%), meaning that repeated gestures (e.g., rubbing the temple or shifting posture) were not falsely flagged by the anomaly engine.

However, there was a noticeable drop under Non-Ideal Conditions, especially when major visual elements (like attire or lighting) changed. This indicates a degree of visual sensitivity in the MobileNet+Autoencoder architecture.

### Inference Latency and Real-Time Suitability

The average end-to-end latency for extracting frame features and computing reconstruction error was measured across multiple systems. All results were obtained on CPU-only setups:

- **Average Latency:** 105 ms per frame
- **Max Latency (under heavy CPU load):** 138 ms
- **System Frame Rate Handling:** Maintained real-time analysis at 8–10 FPS without queue buildup

These results validate the suitability of the approach for real-time adaptive behavior analysis, particularly when used in conjunction with lightweight anomaly flagging logic.

### Proposed Usage Guidelines (Ideal Use Protocol)

Based on the above findings, the following practical recommendations are proposed to optimize the effectiveness of the adaptive model:

- **Consistent Background:** Students should ideally take the real exam in the same location and setup as the mock exam.
- **Lighting Consistency:** Drastic changes in lighting conditions between mock and real sessions should be avoided.
- **Appearance Matching:** If accessories such as spectacles or caps are worn during mock sessions, they should also be used in the real exam.
- **Behavior Replication:** Repeated behaviors should be consistently demonstrated in the mock session for reliable profiling.

These simple yet effective protocols can significantly improve similarity recognition and minimize false alarms during live exams.

**Conclusion of Module-Specific Evaluation**

The adaptive behavior profiling module successfully demonstrated its ability to recognize individualized, repetitive behavioral patterns under controlled and slightly variable conditions. Its accuracy in suppressing known behaviors confirms the viability of introducing context-aware intelligence into proctoring systems — moving beyond rigid rule-based evaluation into a more human-tolerant, behaviorally aware system. The final, full-system evaluation will further demonstrate how these suppression capabilities reduce false positive rates during actual exam flow.

## 8.4 Quantitative Evaluation Experiment Outcomes - Full System

The final and most comprehensive evaluation was conducted at the system level, integrating all detection modules, the adaptivity engine, and the central alert fusion framework. The purpose of this test was to evaluate the real-world performance of the system in correctly identifying anomalous or suspicious behavior during an online exam while minimizing false positives through adaptivity.

**Participation of Testers**

The final and most comprehensive evaluation was conducted in collaboration with real student participants, forming a realistic testbed for assessing the system's effectiveness. A total of **20 volunteer students** participated in this phase. Prior to the commencement of testing, the research scope and objectives were clearly explained to each participant. Informed consent was obtained using a structured Google Form, in which participants agreed to have their webcam footage and behavioral patterns recorded for the sole purpose of academic research. A sample of the consent form is attached in the Appendix. All personally identifiable visuals, such as faces, have been blurred in any screenshots used for documentation or demonstration purposes.

Each student was given access to the system along with detailed instructions specifying the ideal testing setup, including background consistency, camera positioning, lighting conditions, and attire. These guidelines were intended to ensure that the adaptivity module could accurately learn and recognize behavior patterns across mock and real exam sessions.

The evaluation process consisted of two key phases:

- **Mock Exam Session:** Students completed a mock exam while their behaviors were recorded. The adaptive model was trained using this footage to capture and encode each student's habitual gestures and actions.

- **Real Exam Session:** Students then participated in a simulated real exam under conditions closely matching their mock session. During this phase, the full system—integrating rule-based modules, adaptivity, and the central alert fusion engine—was run live.

Throughout both sessions, the system continuously logged detection scores, anomaly flags, timestamps, and adaptive suppression decisions. These logs served as the foundation for the dataset used in this section's evaluation. A total of over **350 behavioral events** were collected and manually reviewed, with each event categorized as a true positive, false positive, true negative, or false negative based on video logs, participant debriefing, and ground-truth labeling criteria.

An illustration of the real testing environment and interface screenshots captured during the sessions is provided in **Figure 7.1**.

Figure 8.1: Screenshots of Students testing the System

**Evaluation Setup**

A total of **20 volunteer students** participated in this evaluation. Each student completed a **mock exam session** followed by a **real exam session** while maintaining consistent conditions—same background, environment, attire, and camera angle. The system was run live during the real session to capture behavioral anomalies.

Over the combined sessions, the system recorded more than **350 distinct behavioral events** which were manually labeled post-exam based on video logs and student feedback. These data points included both:

- **Deliberate anomalies**: Students were instructed to mimic behaviors that could be perceived as violations (e.g., whispering, looking away, scratching repeatedly).
- **Spontaneous anomalies**: Cases where the system independently flagged unexpected or suspicious behavior.

### Ground Truth Classification

Each behavioral event was evaluated and classified using the following definitions:

- **True Positive (TP):** An actual suspicious behavior correctly flagged by the system.
- **True Negative (TN):** Normal behavior correctly not flagged.
- **False Positive (FP):** Benign behavior incorrectly flagged (e.g., scratching head repeatedly).
- **False Negative (FN):** A violation that the system failed to detect.

### Evaluation Results

The results from this comprehensive system-level evaluation—including all integrated modules—are summarized in **Table 8.6**, presenting detection metrics across 354 behavioral events.

| **Metric** | **Value** |
|---|---|
| True Positives (TP) | 152 |
| False Positives (FP) | 22 |
| True Negatives (TN) | 153 |
| False Negatives (FN) | 27 |
| **Accuracy** | 87.4% |
| **Precision** | 87.4% |
| **Recall (Sensitivity)** | 84.9% |
| **F1 Score** | 86.1% |

Table 8.6: Full-System Evaluation Metrics on Combined Dataset (n=354)

## Adaptive Suppression in Action

To demonstrate how the adaptive engine reduces false positives in practice, two scenarios are presented using anomaly score trends.

- **Case 1: False Positive Suppression — Normal behavior (e.g., scratching head)** is misinterpreted by the rule-based module, but the adaptivity module suppresses the alert due to prior exposure during mock exam.
- **Case 2: Confirmed Violation — A strong, deliberate anomaly (e.g., repeated looking away while whispering)** triggers high anomaly scores in both systems.

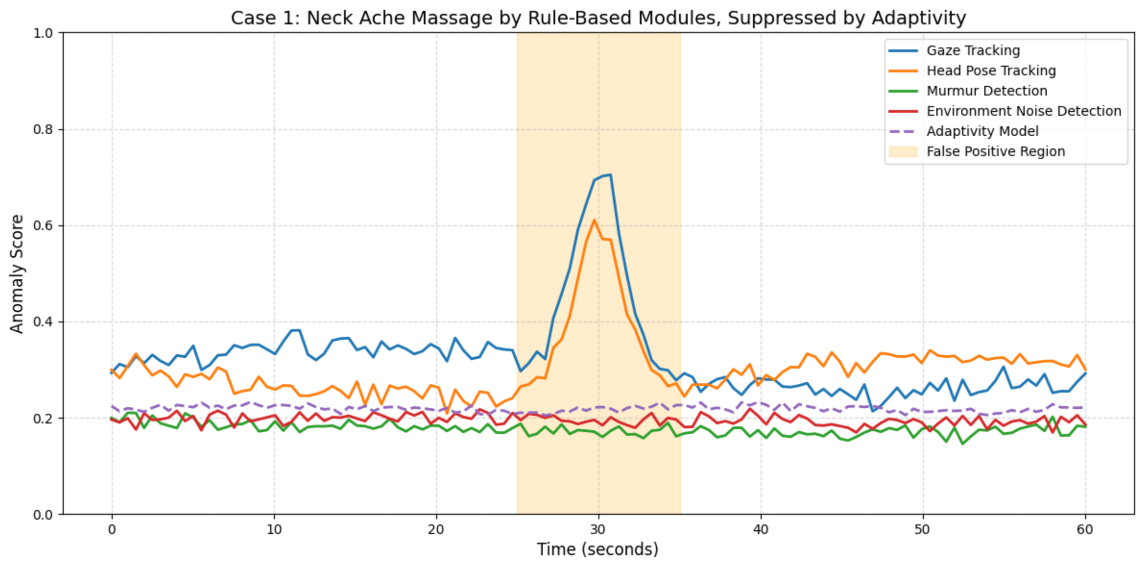


Figure 8.2: Case 1: Rule-Based Flag Suppressed by Adaptivity Module

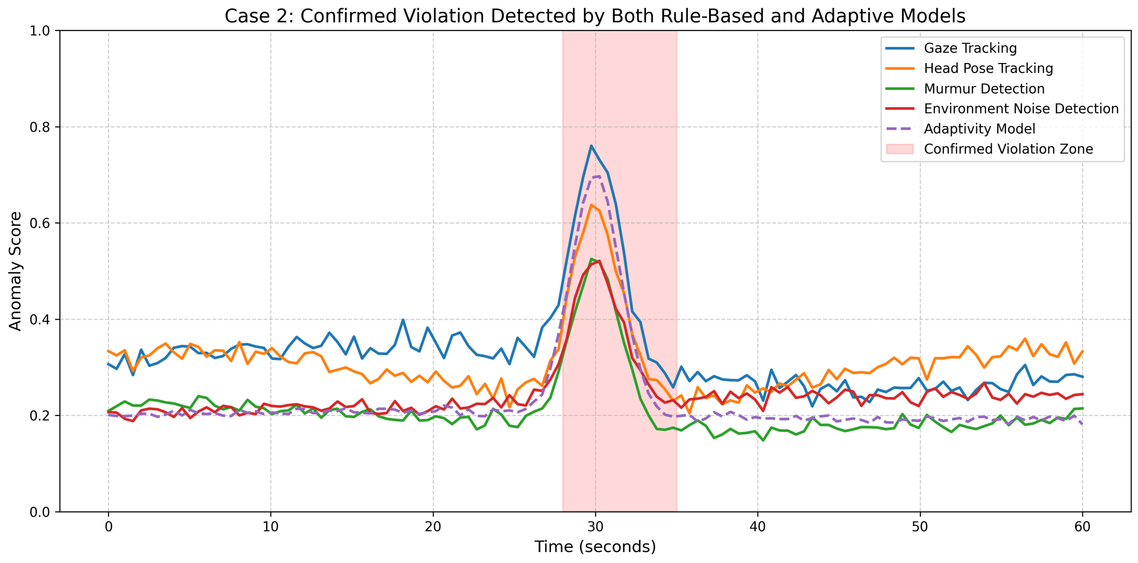


Figure 8.3: Case 2: Violation Confirmed by Both Rule-Based and Adaptive Models

These visualizations highlight the real-time decision coordination between modules and validate the system's goal of reducing false positives without compromising detection capability.

### System-Wide Observations

- **Suppression Efficiency:** Over 18 of 22 false positives from rule-based modules were correctly suppressed by the adaptivity model.
- **Latency:** Combined inference time remained under 160ms per event, confirming real-time applicability.
- **False Negative Analysis:** Most FN cases involved brief, low-visibility actions (e.g., subtle murmurs or side glances outside camera frame).

**Control Group: Rule-Based System without Adaptivity**

To validate the effectiveness of the adaptivity engine, a control experiment was conducted where the same proctoring setup was run using **only the rule-based anomaly detection modules**, with the adaptivity component disabled. This simulation was performed using **5 students**, each completing a mock and real exam session as before, contributing over **70 behavioral data points**.

In this scenario, all alerts were generated solely based on real-time rules without any behavioral suppression logic. The system flagged anomalies based on static thresholding (e.g., time spent looking away, voice activity, or excessive head movement), without context awareness of the student's mock exam behavior.

The outcomes for the rule-based system operating without adaptivity are displayed in **Table 8.7**, showing elevated false positives due to lack of contextual filtering.

| **Metric (No Adaptivity)** | **Value** |
|---|---|
| True Positives (TP) | 30 |
| False Positives (FP) | 18 |
| True Negatives (TN) | 19 |
| False Negatives (FN) | 6 |
| **Accuracy** | 69.0% |
| **Precision** | 62.5% |
| **Recall (Sensitivity)** | 83.3% |
| **F1 Score** | 71.4% |

Table 8.7: System Metrics under Pure Rule-Based Configuration (Adaptivity Disabled)

#### Interpretation and Comparative Insight

As shown in Table 8.7, the false positive rate increased substantially when adaptivity was not used. With **18 false positives out of 73 total events** (compared to only 22 false positives out of 354 events in the adaptive system), the control setup showed a significantly higher tendency to misclassify normal behavior.

This experiment reinforces the core hypothesis of the research: **a rigid, rule-based proctoring system lacks contextual understanding and results in higher over-flagging**, especially for students with unique but benign behavioral patterns. The adaptivity module plays a critical role in improving precision by learning and tolerating such repetitive actions.

Thus, this control evaluation strongly supports the inclusion of an adaptive behavior profiling layer in future AI-enhanced proctoring frameworks.

#### Conclusion of System Evaluation

The full system exhibited strong and balanced performance across precision, recall, and accuracy metrics. The integration of adaptivity proved critical in reducing false alarms, particularly for idiosyncratic but benign behaviors. These results confirm the viability of the proposed adaptive, rule-enhanced proctoring architecture for real-world deployment, while also providing actionable feedback for future refinements.

## 8.5 Qualitative Evaluation Outcomes

In addition to the quantitative evaluations conducted on the modular and integrated system, qualitative feedback was gathered from two key stakeholder groups: the project supervisor and a group of early test users (students). The goal was to understand the perceived usability, reliability, and acceptability of the system from both academic and practical user perspectives.

### 8.5.1 Supervisor's Feedback

Throughout the project lifecycle, iterative feedback was received from the academic supervisor, particularly after milestone reviews and prototype demonstrations. Key observations include:

- **Innovation in Adaptivity:** The supervisor commended the inclusion of the behavior profiling engine, noting that it "introduces a much-needed contextual lens to what has otherwise been a rigid discipline."
- **System Design Maturity:** The modular breakdown into rule-based, adaptive, and alert fusion components was appreciated for its clarity and extendability.
- **Evaluation Transparency:** The supervisor noted the effort to simulate real testing conditions and to acknowledge system limitations, particularly around false positives and edge behaviors.
- **Improvement Suggestions:**
  - Consider deeper testing under varying environmental contexts (e.g., low light, external noise).
  - Explore future extensions such as gaze fixation duration, emotion-based cues, or secure communication protocols.

The supervisor concluded that the project "demonstrates both academic rigour and practical relevance," with particular strength in its motivation to reduce unjust penalties during remote examinations.

### 8.5.2 Early Users' Feedback

Informal usability testing was conducted with 8 students who participated as early users of the proctoring system. These students used both the student-side and proctor-side interfaces during simulated exam conditions. Feedback was collected via short surveys and verbal interviews. Key insights are summarized below:

- **Perceived Fairness:** A majority (6 out of 8) of users reported that the system felt “fairer than expected,” especially after learning that repeated behaviors like scratching or looking away would not be immediately flagged. One user expressed concern about the system “missing subtle cheating attempts,” while another remained neutral, stating they “needed more exposure” to form a full opinion.

- **UI Simplicity:** The student dashboard was considered clear and usable, though a few users suggested improvements in font size and color contrast. One user noted that navigation could be more intuitive for first-time users.

- **Anxiety Reduction:** Several users reported that knowing their usual behaviors were being profiled during a mock exam “reduced nervousness during the real one.” This was echoed by 5 users, while the others reported either minimal impact or were unsure about its effect.

- **Suggestions:**
  - Add a summary report of behaviors at the end of the exam for transparency.
  - Enable microphone and camera testing before starting the exam session.
  - Offer a short walkthrough or tutorial before the session begins.

Overall, early testers acknowledged the novelty and practicality of the adaptive component and expressed interest in seeing how the system might evolve with more advanced machine learning integrations and feedback mechanisms. The mixed responses from the two non-majority users highlight important directions for further usability enhancements and wider user education.

# 9 Future Work

While the current system demonstrates promising performance and real-time applicability, several areas remain open for further enhancement and research. Building on the findings and limitations identified during this study, the following directions are proposed for future work:

- **Multimodal Behaviour Profiling:** Future implementations can explore integrating additional modalities such as keystroke dynamics, mouse movements, and physiological signals (e.g., heart rate via webcam-based PPG) to enrich behavioral profiling and anomaly detection accuracy.

- **Longitudinal Behaviour Modeling:** Expanding the behavior profiling engine to incorporate multiple sessions across time could help model behavioral drift, allowing the system to adapt to gradual changes in student habits and improve long-term robustness.

- **More Diverse Datasets:** Future evaluations should involve a more diverse pool of participants in terms of demographics, exam types, and behavioral patterns. This would enhance the generalizability of the adaptive models and their fairness across broader populations.

- **Real-World Deployment Studies:** While this research focused on controlled mock exams, deploying the system in actual university-level assessments would allow for more rigorous evaluation of performance, reliability, and student acceptance in high-stakes environments.

- **Explainability and Transparency Features:** Incorporating explainable AI (XAI) mechanisms that provide feedback to students and proctors on why a behavior was flagged or suppressed could improve user trust and system accountability.

- **Fine-Grained Alert Severity Levels:** Introducing a graded alert system—ranging from low-risk to high-risk anomalies—would provide more nuanced feedback and reduce alarm fatigue for proctors.

- **Cross-Platform Adaptivity and Edge Deployment:** Further research could explore deploying lightweight versions of the system on edge devices (e.g., mobile phones, tablets) to enable broader accessibility without dependence on high-end computing resources.

- **Ethical and Policy Frameworks:** As intelligent proctoring systems evolve, future research should explore the development of policy guidelines and ethical frameworks for their deployment, particularly concerning data privacy, consent, and algorithmic fairness.

By addressing these directions, the system can continue to evolve into a more intelligent, inclusive, and widely applicable solution for modern remote assessment challenges.

# 10 Conclusion

## Research Problem and Objectives

This study set out to investigate how adaptive AI techniques could be harnessed to reduce false positives in online proctoring systems, without compromising the integrity of remote assessments. Specifically, it examined the effectiveness of a hybrid proctoring system that integrates rule-based anomaly detection with a behavioral profiling module trained on individual mock exam data.

## Summary of Key Findings

The research demonstrates that incorporating an adaptivity engine significantly improves the system's ability to distinguish between genuine malpractice and innocuous, repetitive student behaviors—such as head scratching or habitual movements—that are often misclassified in traditional systems. The system achieved high precision and recall in anomaly detection, while maintaining real-time performance and minimal resource overhead. Experimental results, including control and visual analyses, highlighted the fairness enhancements provided by behavioral context modeling.

## Contributions to the Field

This work contributes to the intelligent proctoring landscape by presenting a novel, behavior-aware detection framework that goes beyond rigid rule enforcement. Unlike existing systems that generalize behavioral norms, the proposed system introduces personalization to anomaly detection, addressing a critical gap in fairness and inclusivity.

## Original Contributions to Knowledge

The research offers several original contributions:

- Development of an adaptive behavioral profiling module trained using mock exam data to suppress false positives.
- A modular, thread-safe system architecture optimized for general-purpose hardware.
- An alternative server-deployment model to mitigate local model tampering risks.

These innovations provide methodological, practical, and architectural advancements to the design of intelligent proctoring tools.

## Implications of the Research

Academically, this work opens avenues for integrating user-specific behavioral modeling into various AI-driven surveillance systems. Practically, it offers educational institutions a more equitable and reliable solution for conducting high-stakes online exams. Policymakers and exam regulators may also leverage the findings to guide ethical AI deployment in assessment settings, especially where fairness and privacy are paramount.

## Limitations

Despite promising results, the system showed sensitivity to major environmental changes such as variations in lighting, attire, or camera angle. These limitations highlight the need for clear deployment guidelines and further robustness enhancements. Additionally, the training data was limited in diversity, which may affect generalizability across broader populations and exam contexts.

## Recommendations for Future Work

Future research should focus on:

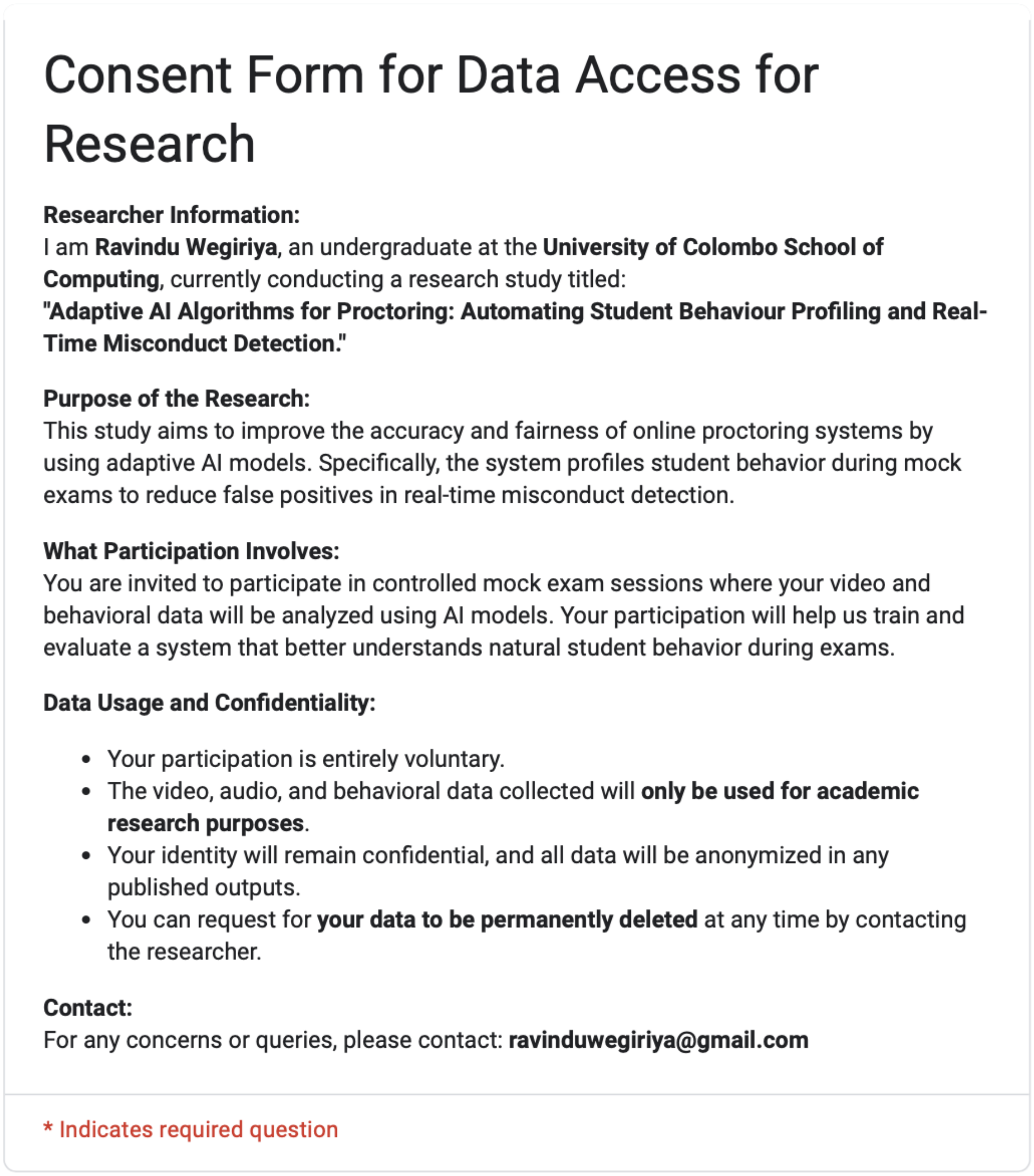
- Expanding training datasets to include a broader range of behavioral profiles and exam environments.
- Integrating more robust environmental normalization techniques.
- Exploring multimodal inputs (e.g., audio and keystroke dynamics) for improved anomaly discrimination.
- Conducting longitudinal studies to observe behavioral drift and model adaptability over time.

## Final Reflection

In an era where remote learning and assessment have become increasingly normalized, this research offers a timely and meaningful contribution toward ensuring academic integrity without sacrificing the student experience. By championing adaptive, student-centric design in proctoring technologies, this study not only enhances current practices but also lays the groundwork for a more equitable digital education ecosystem.

# 11 Appendix

## 11.1 Consent Form for Data Access



Consent Form for Data Access for Research

**Researcher Information:**
I am **Ravindu Wegiriya**, an undergraduate at the **University of Colombo School of Computing**, currently conducting a research study titled:
**"Adaptive AI Algorithms for Proctoring: Automating Student Behaviour Profiling and Real-Time Misconduct Detection."**

**Purpose of the Research:**
This study aims to improve the accuracy and fairness of online proctoring systems by using adaptive AI models. Specifically, the system profiles student behavior during mock exams to reduce false positives in real-time misconduct detection.

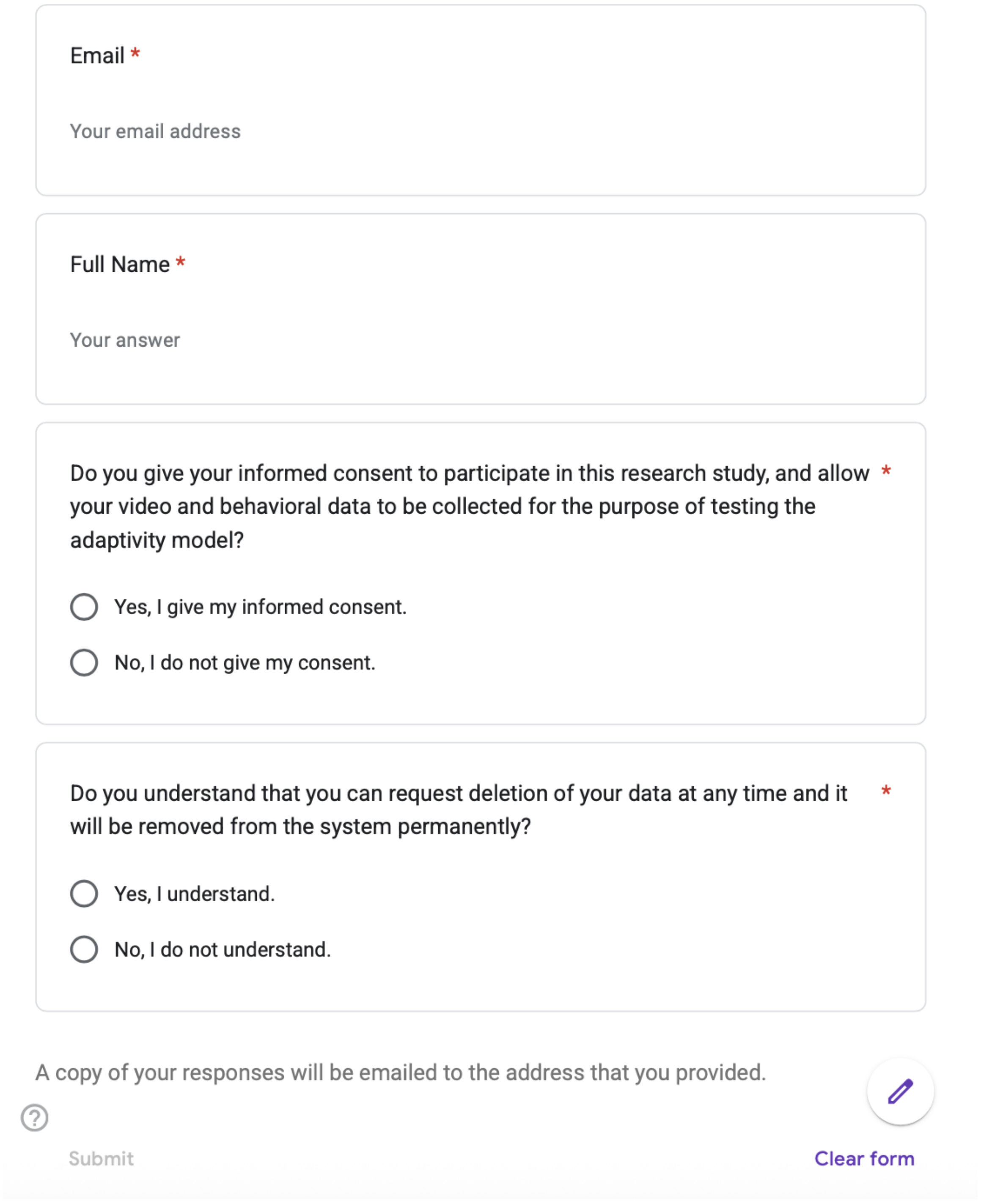
**What Participation Involves:**
You are invited to participate in controlled mock exam sessions where your video and behavioral data will be analyzed using AI models. Your participation will help us train and evaluate a system that better understands natural student behavior during exams.

**Data Usage and Confidentiality:**

- Your participation is entirely voluntary.
- The video, audio, and behavioral data collected will **only be used for academic research purposes**.
- Your identity will remain confidential, and all data will be anonymized in any published outputs.
- You can request for **your data to be permanently deleted** at any time by contacting the researcher.

**Contact:**
For any concerns or queries, please contact: **ravinduwegiriya@gmail.com**

* Indicates required question



Email *

Your email address

Full Name *

Your answer

Do you give your informed consent to participate in this research study, and allow your video and behavioral data to be collected for the purpose of testing the adaptivity model? *

○ Yes, I give my informed consent.

○ No, I do not give my consent.

Do you understand that you can request deletion of your data at any time and it will be removed from the system permanently? *

○ Yes, I understand.

○ No, I do not understand.

A copy of your responses will be emailed to the address that you provided.

# References

Alessio, H. M., Malay, N., Maurer, K., Bailer, A. J. & Rubin, B. (2017), 'Examining the effect of proctoring on online test scores.', *Online Learning* **21**(1), 146–161.

Cavanagh, M., Eberle, W. & Rogers, J. (2016), Automated proctoring of online exams using head pose estimation and gaze tracking, *in* 'Proceedings of the 2016 International Conference on Computer Vision Theory and Applications (VISAPP)', SciTePress, pp. 407–414.

Chatterjee, P., Dansana, J., Swain, S., Kumar Gourisaria, M. & Bandyopadhyay, A. (2024), Identity verification in real time proctoring: An integrated approach with face recognition and eye tracking, *in* '2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)', pp. 1–6.

Chougule, M., Bagul, S., Gharat, M., Malve, S. & Kayande, D. (2024), Proctoxpert – an ai based online proctoring system, *in* '2024 3rd International Conference for Innovation in Technology (INOCON)', pp. 1–8.

Coghlan, S., Miller, T. & Paterson, J. (2021), 'Good proctor or "big brother"? ethics of online exam supervision technologies', *Philosophy & Technology* **34**(4), 1581–1606.

Draaijer, S., Jefferies, A. & Somers, G. (2018), Online proctoring for remote examination: A state of play in higher education in the eu, *in* E. Ras & A. E. Guerrero Roldán, eds, 'Technology Enhanced Assessment', Springer International Publishing, Cham, pp. 96–108.

Felsinger, D., Halloluwa, T. & Fonseka, I. (2024), 'Video based action detection for online exam proctoring in resource-constrained settings', *Educ. Inf. Technol.* **29**(10), 12077–12091.

Foster, D. & Layman, H. (2013), 'Online proctoring systems compared'.

Iqbal, T., Ali, T., Shaf, A. & Ali, M. S. (2023), Enhancing online exam security: Deep learning algorithms for cheating detection, *in* '2023 International Conference on Frontiers of Information Technology (FIT)', pp. 126–131.

Liu, Y., Ren, J., Xu, J., Bai, X., Kaur, R. & Xia, F. (2023), 'Multiple instance learning for cheating detection and localization in online examinations', *IEEE Transactions on Cognitive and Developmental Systems* . Preprint, arXiv:2402.06107.
**URL:** *https://arxiv.org/abs/2402.06107*

Malhotra, N., Suri, R., Verma, P. & Kumar, R. (2022), Smart artificial intelligence based online proctoring system, *in* '2022 IEEE Delhi Section Conference (DELCON)', pp. 1–5.

Masud, M. M., Hayawi, K., Mathew, S. S., Michael, T. & El Barachi, M. (2022), Smart online exam proctoring assist for cheating detection, *in* 'International conference on advanced data mining and applications', Springer, pp. 118–132.

Mewada, D., Gaikwad, S., Gharat, B. & Kamble, P. (2024), An al powered exam proctoring: Comprehensive monitoring for integrity, *in* '2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS)', Vol. 1, pp. 1–5.

Moreno-Guerrero, A.-J., Rodríguez-Jiménez, C., Gómez-García, G. & Ramos Navas-Parejo, M. (2020), 'Educational innovation in higher education: Use of role playing and educational video in future teachers' training', *Sustainability* **12**(6), 2558.

Moyo, R., Ndebvu, S., Zimba, M. & Mbelwa, J. (2023), A video-based detector for suspicious activity in examination with openpose, *in* 'Proceedings of the 5th Deep Learning Indaba Conference (DLI)'. arXiv preprint arXiv:2307.11413v2.
**URL:** *https://arxiv.org/abs/2307.11413v2*

Nurpeisova, A., Shaushenova, A., Mutalova, Z., Ongarbayeva, M., Niyazbekova, S.,

Bekenova, A., Zhumaliyeva, L. & Zhumasseitova, S. (2023), 'Research on the development of a proctoring system for conducting online exams in kazakhstan', *Computation* **11**(6), 120.
**URL:** *https://www.mdpi.com/2079-3197/11/6/120*

Pandey, A. K., Kumar, S., Rajendran, B. & B S, B. (2020), e-parakh: Unsupervised online examination system, *in* '2020 IEEE REGION 10 CONFERENCE (TENCON)', pp. 667–671.

Paul, J. S., Farhath, O. & Selvan, M. P. (2024), Ai based proctoring system – a review, *in* '2024 International Conference on Inventive Computation Technologies (ICICT)', pp. 1–5.

Peterson, J. (2019), 'An analysis of academic dishonesty in online classes.', *Mid-Western Educational Researcher* **31**(1).

Sakhipov, A., Omirzak, I. & Fedenko, A. (2025), 'Beyond face recognition: A multi-layered approach to academic integrity in online exams', *Electronic Journal of e-Learning* **23**(1), 81–95.
**URL:** *https://doi.org/10.34190/ejel.23.1.3896*

Sharma, S., Manna, A. & Arunachalam, N. (2024), Analysis on ai proctoring system using various ml models, *in* '2024 10th International Conference on Communication and Signal Processing (ICCSP)', pp. 1179–1184.

Singh, T., Nair, R. R., Babu, T. & Duraisamy, P. (2024), 'Enhancing academic integrity in online assessments: Introducing an effective online exam proctoring model using yolo', *Procedia Computer Science* **235**, 1399–1408.
**URL:** *https://doi.org/10.1016/j.procs.2024.04.13*

Somavarapu, J., Biswas, S. K., Purkayastha, B., Abhisheka, B. & Potluri, T. (2024), Advancements and challenges in fully automated online proctoring systems: A comprehensive survey of AI-driven solutions, *in* 'Lecture Notes in Networks and Systems', Lecture notes in networks and systems, Springer Nature Singapore, Singapore, pp. 199–212.

Verma, P., Malhotra, N., Suri, R. & Kumar, R. (2024), 'Automated smart artificial intelligence-based proctoring system using deep learning', *Soft Comput.* **28**(4), 3479–3489.