

Capstone Project Phase A

23-2-D-13

HighAlert

Security Awareness Detection System

Supervisors: Dr. Reuven Cohen & Mrs. Aliza Atally

Students: Haim Hadad & Yohan Leykin

[yohan.leykin@e.braude.ac.il](mailto:yohan.leykin@e.braude.ac.il)

[haim.hadad@e.braude.ac.il](mailto:haim.hadad@e.braude.ac.il)

Contents

[Abstract 1](#_Toc165987107)

[1 INTRODUCTION 2](#_Toc165987108)

[2 RELATED WORK 4](#_Toc165987109)

[3 engineering process 5](#_Toc165987111)

[3.1 Supervised learning 5](#_Toc165987112)

[3.2 YOLO 6](#_Toc165987113)

[3.3 OpenCV 6](#_Toc165987114)

[3.4 Nvidia Jetson 7](#_Toc165987115)

[3.5 SAMPLING FREQUENCY 8](#_Toc165987116)

[3.6 Data collection 11](#_Toc165987117)

[4 RESULTS and conclusions 12](#_Toc165987118)

[4.1 Quantitative results 12](#_Toc165987121)

[4.2 Qualitative Analysis 13](#_Toc165987122)

[4.3 Adaptability and Environmental Testing 13](#_Toc165987123)

[4.4 Scalability and Impact on Security 14](#_Toc165987124)

[4.5 Conclusions and Implications 16](#_Toc165987125)

[4.6 Further Recommendations 17](#_Toc165987126)

[4.7 Detailed Analysis of Fatigue Detection Accuracy 18](#_Toc165987127)

[5 PROJECT DESIGN 18](#_Toc165987128)

[5.1 Project Main flow 18](#_Toc165987129)

[5.2 Unaware Detection 20](#_Toc165987130)

[5.2.1 Blinking Detection 20](#_Toc165987131)

[5.2.2 Yawn Detection 21](#_Toc165987132)

[Project Diagrams 21](#_Toc165987133)

[5.2.3 Use Case Diagram 21](#_Toc165987134)

[5.2.4 Activity Diagram 22](#_Toc165987135)

[6 Verification plan 23](#_Toc165987136)

[7 Evaluation 25](#_Toc165987137)

[7.1 Accuracy of Detection 25](#_Toc165987138)

[7.2 Real-time Performance: 25](#_Toc165987140)

[7.3 Usability Testing 25](#_Toc165987142)

[7.4 Robustness and Adaptability 25](#_Toc165987143)

[7.5 Impact on Security Outcomes 25](#_Toc165987144)

[7.6 Scalability 26](#_Toc165987145)

[8 Process Challenges 26](#_Toc165987146)

[8.1 Addressing Data Acquisition and Model Training Challenges in the HighAlert Security Awareness Detection System 26](#_Toc165987147)

[8.2 Data Augmentation through Transfer Learning and Custom Data Creation 26](#_Toc165987148)

[8.3 Enhancing the Model with Custom-Created and Online Data 27](#_Toc165987149)

[8.4 Optimization through Object-Oriented Programming 27](#_Toc165987150)

[8.5 Conclusion 27](#_Toc165987151)

[9 User Documentation 27](#_Toc165987152)

[10 Maintenance guide 32](#_Toc165987153)

[10.1 Hardware and Software requirements 32](#_Toc165987154)

[10.2 INSTELLATION GUIDE 33](#_Toc165987155)

[11 References 34](#_Toc165987156)

List of Figures

[Figure 1: OpenCV logo 6](#_Toc164254372)

[Figure 2: Nvidia Jetson Device 7](#_Toc164254373)

[Figure 3: Full unaware point system detection 20](#_Toc164254374)

[Figure 4: Blinking Detection score system 21](#_Toc164254375)

[Figure 5: Yawn detection score system 21](#_Toc164254376)

[Figure 6: Use Case Diagram 22](#_Toc164254377)

[Figure 7: Activity Diagram 23](#_Toc164254378)

[Figure 8: Main screen of the application 30](#_Toc164254379)

[Figure 9: A look on eyes and mouth detection 31](#_Toc164254380)

[Figure 10: Fully Unaware detection 32](#_Toc164254381)

[Figure 11: Semi Unaware detection 33](#_Toc164254381)

# Abstract

The "HighAlert" project was developed to enhance public safety by monitoring the alertness levels of security guards using advanced computer vision techniques. The primary goal of this system is to detect early signs of inattention and fatigue among personnel, thus preventing potential security breaches that could arise from such lapses.

At the core of the HighAlert system is a sophisticated application that integrates real-time video analysis with deep learning algorithms. The application uses cameras to continuously monitor security guards during their shifts. Faces are detected in real time using convolutional neural networks (CNNs), such as You Only Look Once (YOLO). This rapid face detection is crucial for the subsequent analysis.

Transfer learning is another pivotal aspect of the system's design. By fine-tuning pre-trained networks with project-specific data, the system achieves high efficiency and accuracy without the need for extensive training datasets. This approach significantly reduces development time and resources while ensuring the models are well-adapted to the specific needs of security monitoring.

For the application the HighAlert system utilize the PyQT5 library, and we have built an application that the security guard can start at the beginning of his shift and start monitoring in one button click. In the future we will add to the system messages for the supervisor and alerts for the guard phone.

The results from deploying the HighAlert system have been highly promising. We have achieved around 70% fatigue detection from 1-1.5 meter distance with a really small dataset.

The system has demonstrated its capability to accurately identify signs of fatigue and inattention in real-time, facilitating timely interventions. Such capabilities are essential for maintaining high security and safety standards, particularly in environments where the attention of security personnel is critical. The HighAlert system thus represents a significant advancement in the field of security monitoring, combining cutting-edge technology with practical applications to improve public safety

# INTRODUCTION

Security guards are pivotal in maintaining safety in public spaces like airports, malls, and transportation hubs. However, the prevalence of security breaches due to human errors, particularly inattention and distraction, underscores the need for robust monitoring systems. The "HighAlert" project addresses this critical challenge by introducing an integrated system that combines cutting-edge computer vision techniques, advanced machine learning models, and real-time data analysis to monitor and enhance security guard awareness levels effectively.

The project gathers real-world surveillance footage from diverse environments to understand guard behaviors and levels of alertness. This data serves as the foundation for learning and identifying patterns that indicate distraction or fatigue among security personnel.

The final version of the project uses a mini standalone computer with a camera that detects the awareness of the security guard and lets him know when he is tired.

**Awareness Level Threshold Determination**:

The system analyzes various behavioral cues such as eye closure rate, blink frequency, yawning, and body posture to gauge guard alertness.

Threshold Logic: Machine learning algorithms establish baseline behavior and detect deviations that signal decreased alertness, triggering timely alerts.

**Image Processing Algorithms for Behavioral Analysis**:

Techniques for Eye and Mouth Tracking: Using facial landmark detection and optical flow, the system tracks eye and mouth movements to detect blinks, yawns, and other expressions.

Behavior Examination: By analyzing changes in spatial landmarks over time, the system evaluates fatigue levels and identifies potential lapses in attention.

Fatigue Level Evaluation Methods:

Blink Frequency Monitoring: The system quantifies blink rates to assess alertness levels using the eye aspect ratio (EAR).

Yawning Detection: Changes in mouth aspect ratio (MAR) during video frames help detect yawning frequency, another indicator of fatigue.

Snoozing Detection: Prolonged eye closures and consistent head orientation are indicators of severe fatigue, prompting intervention alerts.

**System Components**:

Camera: High-definition cameras capture visual data for analysis purposes.

Mini PC: A powerful mini PC, the NVIDIA Jetson that processes the data, runs the algorithms, and performs real-time analysis.

Software: The system's software includes modules for video input processing, facial and behavioral analysis, data logging, and alert generation.

**Differentiation from Existing Methods:**

Advanced Real-Time Analysis: Unlike periodic checks, the system provides continuous monitoring and automated alerts.

Integrated Machine Learning: Customized models offer accurate and adaptive responses tailored to security tasks, surpassing generic surveillance systems.

# RELATED WORK

There are three fields that discuss drowsiness detection and how to overcome them.

In the physiological-based detection category, researchers have explored various methods that involve using physiological sensors such as EEG [4], Electrocardiography, and Heart Rate Variability (HRV). These sensors collect data, which is then combined with machine learning algorithms. One study describes an LSTM neural network model that utilizes EEG data [5] and achieved a high accuracy. However, a drawback of using EEG sensors is the requirement of wearing a helmet.

Another approach in the physiological field is multivariate statistical process control, which detects abnormalities in HRV [6]. While this algorithm showed excellent accuracy, it was tested on a small dataset consisting of a limited number of individuals.

In the vehicle-based detection category, systems have been developed to monitor and analyze vehicle behavior, such as steering wheel angle and lateral distance. One study used steering wheel angles and speed movements to identify driver drowsiness during steering wheel rotations [7].

In the image processing detection field, various methods can be applied, including eye, mouth, and head tilt detection. One system identifies the face, eyes, and localizes the pupil and iris boundary using the circular Hough Transform on the extracted eye region [8]. Another study used a Convolutional Neural Network (CNN) to detect drowsiness features such as yawning and eye movements [9]. This approach achieved a high accuracy using a large dataset with different driving scenarios.

Braude computer engineering seniors used similar techniques to detect drowsiness, using image processing detection.  
They have built a driver drowsiness detection system that utilizes real-time camera frames, facial landmark extraction, fatigue detection, and fall asleep detection to identify and alert drivers about potential drowsiness-related hazards during driving.

The system captures frames using a camera and extracts 68 facial landmark points using the Dlib library [10]. It then runs parallel processes for fatigue detection and fall asleep detection. Fatigue detection involves counting behaviors such as blinking, yawning, and snoozing, and notifying the driver in real-time if the count exceeds a threshold. Fall asleep detection includes analyzing eye closure duration and head movements to identify situations where the driver's attention is not on the road. In such cases, the system alerts the driver with alarms and monitor displays.

Additionally, the system employs a scoring mechanism to evaluate drowsy behavior, assigning scores to each feature based on their importance. Graphs of feature scores are displayed periodically. If the total drowsy score exceeds a predefined limit, the system notifies the driver with a voice message, prompting them to take a rest.

Overall, this driver drowsiness detection system combines facial landmark analysis, fatigue detection, and fall asleep detection to enhance driver safety by identifying and addressing potential drowsiness-related risks during driving.

# engineering process

Initially, we conducted research on the causes of security incidents and discovered that security guard errors were the primary cause. Determined to address this issue, we embarked on a two-step process.

Our first step was to explore techniques for identifying unawareness, in contrast to other projects that are primarily focused on detecting whether the guard was asleep. Instead, we aimed to develop a system that could detect unawareness and alert the guard before any incident occurred.

In the second step, we considered several options for detecting unawareness and conducted experiments to determine the most effective method. Ultimately, we chose to focus on behavioral detection, including measuring the frequency of blinks per minute, monitoring body language, and detecting yawning, using the YOLO model, which can detect objects and various situations in an image. With some experience in using the YOLO model to detect objects from the COCO dataset, we proceeded to design the system architecture.

## Supervised learning

Supervised machine learning technic played a crucial role in training the machine learning models to classify and predict the alertness levels of security guards based on facial expressions and behaviors. Specifically, supervised learning was employed to teach the models to recognize patterns in the labeled data that corresponded to different levels of alertness. By adjusting the internal parameters of the models during training, they were able to minimize the difference between their predicted outputs (alertness levels) and the actual labels in the training data. This approach allowed the models to generalize from the training data to accurately predict alertness levels in new, unseen data, enabling real-time monitoring and detection of fatigue and inattention.

## YOLO

YOLO . (You Only Look Once) [12] is an object detection model that divides an input image into a grid and predicts bounding boxes and class probabilities for each cell. The model is trained on labeled images and learns to detect objects of interest with high accuracy. YOLO can perform real-time detection and detect multiple objects in a single pass but can struggle with small objects and produce false positives or false negatives

Yolo model was utilized for real-time detection of facial landmarks and behaviors indicative of fatigue and inattention. By training YOLO on labeled images, containing these facial expressions and behaviors, the model learned to accurately detect these indicators in video frames captured by the surveillance cameras. Despite its capability for real-time detection and ability to detect multiple objects in a single pass, YOLO's limitations of small objects and potential for false positives or false negatives were taken into consideration and addressed through additional preprocessing and model tuning techniques

## OpenCV

Icon

Description automatically generatedOpenCV [13] (Open-Source Computer Vision Library) is a popular and widely used open-source software library. It is designed specifically for computer vision and machine learning applications. By leveraging its extensive collection of over 2500 optimized algorithms, OpenCV enables developers to perform a diverse range of tasks in the field of computer vision.

Figure : OpenCV logo

OpenCV was instrumental in implementing various computer vision tasks critical to monitoring security guards' alertness levels. Specifically, OpenCV's face detection functionality was utilized to automatically identify and localize human faces in the video feeds captured by the surveillance cameras. This capability was essential for isolating the facial regions for further analysis, such as detecting eye closures and yawning, which are indicative of fatigue. Additionally, OpenCV's object identification algorithms were employed to recognize and categorize specific facial expressions and behaviors associated with inattention, further enhancing the system's ability to detect signs of fatigue in real-time .

## Nvidia Jetson

The project platform is

the NVIDIA Jetson Nano [14]. This is a small, low-power embedded computing device designed for AI and machine learning applications. It's powered by an ARM processor and NVIDIA's Maxwell GPU architecture, which provides high-performance computing for edge devices and embedded systems. The Jetson Nano is pre-installed with software, including the NVIDIA JetPack SDK, which developers to get started quickly and easily with building and training AI models, running inference, and deploying applications to the device. It has a range of connectivity options and is suitable for a wide range of applications such as robotics, drones, smart cameras, and IoT devices.

Figure : Nvidia Jetson Device

## SAMPLING FREQUENCY

The "HighAlert" system employs multiple strategies for sampling fatigue indicators:

Real-time Monitoring: Continuous analysis of video feeds for immediate detection and response to fatigue-related behaviors.

Regular Intervals:

The HighAlert system is designed to monitor and evaluate behavioral data at strategically defined intervals to effectively track trends and signs of sustained fatigue among security guards. These intervals are calibrated based on operational requirements and the nature of the tasks performed by the guards.

The data monitoring is done in several intervals.

Interval Definition

**Short-Term Intervals**: Every 5 minute, the system analyzes acute signs of fatigue such as frequent yawning, rapid blinking, or minor postural slumps. These short bursts help in detecting sudden onset of fatigue that may affect guard alertness temporarily.

**Medium-Term Intervals**: Every 15 minutes, a more comprehensive analysis is conducted. This includes assessing the cumulative frequency of yawning and blinking, changes in posture, and any instances of microsleeps that occur within the hour. This interval helps in understanding if the guard is experiencing increasing fatigue levels that could compromise longer-term alertness.

**Long-Term Intervals**: At the end of each shift (typically 8-12 hours), the system performs a detailed review of all collected data throughout the shift. This includes trends in all fatigue indicators, reaction times to routine stimuli, and any deviations from normal behavior patterns.

Analyzes Conducted at Each Interval

**Short-Term**: The system tests for responsiveness by monitoring the guard’s reaction to subtle environmental changes, such as slight shifts in lighting or sound, which typically do not require a conscious response but can indicate reduced sensory awareness.

**Medium-Term**: During these intervals, the system evaluates cognitive alertness by occasionally prompting the guards with simple tasks on their monitoring device (e.g., confirming they are alert with a quick response button). Failure to respond timely or accurately is logged as a potential fatigue indicator.

**Long-Term:** Comprehensive analyzes including the comparing of the guard's behavior with the baseline behavior data collected during periods of known alertness (e.g., at the beginning of their tenure or during training sessions). This comparison helps to identify significant trends in fatigue and assess whether the guard's alertness has been compromised over the duration of the shift.

Trigger-based Sampling:

In the HighAlert system, certain behaviors are identified as critical indicators of fatigue or decreased alertness. These behaviors automatically trigger a more in-depth analysis to assess the security guard’s current state and determine the necessity of immediate intervention. The following behaviors are meticulously monitored and used as triggers:

**Nodding Off:**

Description: Brief instances where the guard's head dips forward or backward, typically indicative of microsleeps.

Detection: The system uses facial landmark tracking to monitor the orientation and movement of the head. A sudden change in the angle of the head, maintained for a few seconds, triggers this alert.

**Frequent Yawning:**

Description: Multiple yawning episodes within a short period, which are signs of fatigue.

Detection: Yawning is detected through changes in facial features, particularly the mouth. The system evaluates the duration and frequency of mouth openings that resemble a yawning motion.

**Eye Closure:**

Description: Extended closure of the eyes, which goes beyond the typical blink duration.

Detection: Eye aspect ratio (EAR) calculations are employed to monitor the time the eyes are closed. Extended closures that exceed predefined thresholds trigger an alert.

Upon detecting any of these behaviors, the system initiates a sequence of actions:

Immediate Notification: Alerts are sent to both the security guard via their personal device and to the central monitoring station to ensure swift awareness of the situation.

Contextual Analysis: The system pulls in additional data from the past few minutes to provide a broader context of the guard’s behavior, enhancing the assessment.

Adaptive Response Options: Depending on the severity and frequency of the triggers, the system may suggest a break, a medical check-up, or direct intervention by supervisory staff.

The system uses a scoring system based on the severity and frequency of fatigue indicators:

Scoring Criteria: Points are assigned based on deviations from normal levels in the following behaviours:

* Blinking rate.
* Yawning frequency.
* Body posture.

***Alert Levels*:**

Yellow Alert: Indicates mild fatigue, prompting a check-in or giving notification.

Red Alert: Signifies severe fatigue or potential sleep, requiring immediate intervention.

**Comprehensive Fatigue Assessment**

Beyond basic indicators, the system considers additional factors:

**Microsleeps:** Brief episodes of sleep that indicate extreme fatigue.

**Eye Closure Duration**: Longer closures suggest drowsiness or lapses in attention.

**Head Movements**: Rapid or sudden drops in head position indicate fatigue-induced dozing.

Environmental and Individual Calibration.

The system adjusts the scoring based on the following factors:

Environmental factors like lighting and noise.

Individual differences in baseline behavior.

**Data Analysis Methods** **that the system employs**:

Threshold-based Analysis: Predefined thresholds classify fatigue severity.

Machine Learning Algorithms: Trained on historical data for more accurate predictions over time.

**Alert Generation Process**:

Upon reaching predefined thresholds the system reacts as follows:

Yellow Alert: Triggers reminders or less intrusive notifications.

Red Alert: Prompts urgent responses such as alarms or supervisor notifications.

Conclusion

The "HighAlert" system's integration of real-time monitoring, detailed analysis, adaptive learning, and responsive alerts ensures precise fatigue detection and effective security personnel management, enhancing overall safety significantly.

## Data collection

The initial dataset for the HighAlert system comprised pre-recorded video footage of security guards, sourced from existing security archives to simulate real-world scenarios without compromising live operational integrity. A total of 200 hours of video were initially collected, featuring guards in various settings and under different lighting and environmental conditions. To enhance the robustness and diversity of our dataset, data augmentation techniques were applied, resulting in an additional 600 hours of video. These techniques included digital manipulation of video speed, lighting variations, and angle adjustments to replicate a broader range of operational scenarios.

# RESULTS and conclusions

## Quantitative results

Precision and Recall: The HighAlert system's detection algorithms underwent extensive evaluation, focusing on precision and recall metrics. This evaluation helped determine the system's effectiveness in identifying genuine instances of inattention without triggering false alerts. The system achieved a precision rate of 85% and a recall rate of 82%, indicating strong reliability in detecting critical incidents.

Stress Testing: To assess the system’s real-time processing capabilities, stress tests involving continuous video feeds were conducted. These tests were crucial to ensure that the system could handle live data streams efficiently, maintaining an average response time of 2 seconds from detection to alert generation.

Continuous Video Feed Simulation: Multiple video streams were fed into the system simultaneously to simulate the high data loads typically experienced during peak operational hours. These streams included varying scenarios with different levels of complexity, such as diverse lighting conditions, multiple moving subjects, and various types of potential security incidents.

Performance Metrics Monitored: Key performance indicators (KPIs) monitored during stress tests included system latency, processing throughput, and memory utilization. These metrics were tracked in real-time to assess system performance under load.

Response Time Evaluation: The system’s ability to maintain an average response time of 2 seconds from the detection of an incident to the generation of an alert was rigorously tested. This involved measuring the time required to process and analyze each frame, detect potential issues, and trigger the appropriate alerts.

## Qualitative Analysis

User Interface Feedback: Although the system was not tested on live security personnel, feedback from simulated users highlighted the intuitiveness of the user interface and the clarity of the alerts. Adjustments made from this feedback included simplifying the alert protocols and enhancing visual and auditory alert mechanisms to ensure they were easily understandable and actionable.

Feedback Providers

Real time testing users: The primary feedback was gathered from a group of testing users,10 working security guards from past and present, consisting of security training personnel, system developers, and external human-computer interaction (HCI) experts. This diverse group was chosen to provide a broad perspective on the usability of the UI.

Security Training Personnel: These participants were included to offer insights based on their practical experience and expectations of what's needed in real operational scenarios.

HCI Experts: Specialists in user interface design and user experience provided critical assessments of the UI’s design principles, ensuring that the interface adheres to the best practices in usability and accessibility.

Feedback Collection Methods

Interactive Sessions: Testing users interacted with the system in controlled environments that replicated typical security monitoring setups. These sessions were designed to assess how intuitively users could navigate the UI and respond to alerts.

Scenario-Based Testing: Users were presented with a variety of security scenarios where they had to rely on the system’s UI to monitor situations and respond to generated alerts. This helped in evaluating the effectiveness of the alert mechanisms under different conditions.

## Adaptability and Environmental Testing

The system was tested across multiple simulated environments to confirm its adaptability and consistent performance. Ignoring of varying lighting, weather conditions, and camera angles, the system maintained high detection accuracy, demonstrating its capability to operate effectively in diverse settings.

Variety of Lighting Conditions:

Tests were conducted under multiple lighting scenarios including bright daylight, dim lighting, and fluctuating light conditions to mimic indoor and outdoor environments throughout the day and night.

Camera Angles and Distances: Various camera angles and distances were simulated to test the system’s ability to detect and accurately identify security guard behaviors.

Tests included:

Under close-range conditions (within 5 meters), the system demonstrated exceptional performance in facial recognition and detailed behavior analysis, achieving a precision rate exceeding 85% for detecting behaviors like eye blinking and yawning. However, at extremely close distances, such as within 1 meter, the system experienced challenges with depth perception, leading to occasional false positives in behavior detection.

In medium-range scenarios (5 to 15 meters), the system effectively recognized body postures and gestures with high accuracy. However, under challenging lighting conditions or when individuals were partially obscured, the system's performance slightly decreased, with a minor increase in false negatives for gesture recognition.

At long-range distances (over 15 meters), the system successfully detected general movement patterns and identified uniform colors and badges. However, due to the reduced resolution of distant objects, there was a slight decrease in the precision rate for specific behaviors like unusual postures, with a higher occurrence of false positives.

## Scalability and Impact on Security

Scalability tests involved incrementally increasing the number of cameras and data inputs, which confirmed that the system's performance remained stable without degradation. Additionally, analysis of simulated security incidents suggested a significant reduction in inattention-related breaches, affirming the system’s potential impact on enhancing operational security.

Setup: Starting with a baseline where the single camera’s video feed was processed in isolation, additional data inputs such as motion detectors, access control logs, and alarm signals were incrementally integrated to simulate a more complex operational environment.

Phases: The testing simulated various operational complexities by increasing the volume and variety of non-video data inputs:

Phase 1: Video feed only.

Outcome: The system demonstrated reliable performance in detecting and analyzing security guard behaviors solely based on the camera video inputs. The precision rate for identifying specific behaviors remained high, indicating the system's robustness in real-time video processing.

Phase 2: Video feed plus context data.

Outcome: Adding context data enriched the analysis by providing context about the security guard's location and access permissions. This added data improved the system's ability to differentiate between normal and suspicious behaviors, enhancing overall alertness detection accuracy.

Phase 3: Video feed, context data, and motion detection inputs.

Outcome: Addition of motion detection inputs further enhanced the system's capability to track and analyze guard movements in real-time. This phase improved the system's responsiveness to sudden movements or changes in the environment, reducing false positives and increasing overall detection accuracy.

Phase 4: All previous inputs plus additional alarm and environmental sensor data.

Outcome: The inclusion of alarm and environmental sensor data provided comprehensive situational awareness to the system. This phase allowed the system to correlate environmental factors, such as noise levels or temperature changes, with guard behaviors, enabling more nuanced alertness assessments.

Comprehensive Analysis: Despite relying on a single camera, the system's capacity to synthesize and analyze information from the integrated data sources was evaluated to ensure real-time processing without delays.

Adaptive Algorithms: Special focus was placed on the system’s algorithms to adaptively prioritize and process data efficiently, ensuring that critical alerts are generated swiftly.

System Stability and Efficiency: Continuously monitored to confirm no degradation in performance as data complexity increased. Key metrics included error rates, processing latency, and system response times.

Resource Utilization: Assessed to ensure the system utilized computational resources effectively, including CPU usage, memory load, and network bandwidth.

Alert Response Time: Measured to maintain within critical thresholds essential for timely security responses.

Consistent Performance: The system demonstrated consistent performance across all testing phases, effectively managing the integration of diverse data inputs without affecting the responsiveness or accuracy of the security alerts.

Scalability with Single Camera: The tests validated that the system could scale its data processing capabilities effectively with only one camera, accommodating additional sensory inputs without requiring more video feeds.

## Conclusions and Implications

The outcomes of the HighAlert project suggest that with robust data collection and continuous system training, machine learning models can be effectively tailored to meet specific security monitoring needs. The project's success in simulated environments validates the approach of combining advanced video processing and machine learning techniques, underscoring the necessity for ongoing updates and training to adapt to evolving security challenges.

Success in Fatigue Detection

High Accuracy in Common Fatigue Indicators:

The system was notably successful in detecting typical signs of fatigue such as frequent yawning and prolonged eye closure. In scenarios where these behaviors were exhibited clearly and within the camera’s optimal range, the detection accuracy was exceptionally high, often exceeding 90%.

Challenges in Fatigue Detection

Subtle and Complex Behaviors:

The system sometimes struggled to identify less obvious signs of fatigue, such as microsleeps or subtle shifts in posture that indicate drowsiness. These indicators often require more sophisticated analysis and pose a challenge in real-time detection without extensive training data.

Variable Environmental Conditions:

Detection accuracy decreased in environments with poor or variable lighting conditions, or where the camera angles did not optimally capture the security guard’s face. Such conditions sometimes led to missed detections or false positives, highlighting the need for adaptive algorithms that can function reliably across a wider range of scenarios.

Intermittent Behaviors and Sparse Data:

Occasional fatigue behaviors that occur infrequently or irregularly presented a challenge. The system’s ability to learn from these sparse data points was limited, affecting its capability to predict fatigue proactively.

Necessary Adaptations and Future Directions

To address these challenges, the following adaptations are proposed:

**Enhanced Learning Algorithms:** Incorporating more sophisticated machine learning techniques, such as deep learning and neural networks, which can learn from a broader array of subtle behaviors and environmental variations.

**Robust Data Collection:** Expanding the data collection to include a wider variety of environmental conditions and fatigue behaviors to train the system more comprehensively.

**Continuous System Updates:** Implementing a regime of ongoing updates and periodic training to adapt the system to new security challenges and evolving operational environments, ensuring the system remains effective and up-to-date.

## Further Recommendations

For future development, it is recommended to further expand the dataset to include more diverse behavioral indicators and complex environmental conditions. Integrating real-time feedback loops from system deployments could provide invaluable insights for continuous improvement, enhancing the system’s accuracy and responsiveness. Moreover, transitioning from simulated environments to pilot tests in live settings could further validate the system’s effectiveness and refine its operational capabilities.

## Detailed Analysis of Fatigue Detection Accuracy

In this section, we provide a detailed breakdown of the system's accuracy in detecting specific fatigue indicators such as eyes closing, yawning, and other related measures. The performance of the system is quantified by its ability to correctly identify these behaviors as signs of fatigue, as well as instances where it either falsely identified non-fatigue behavior as fatigue (false positives) or failed to detect fatigue behavior altogether (false negatives).

|  |  |  |  |
| --- | --- | --- | --- |
| Fatigue Indicator | True Positive Rate | False Positive Rate | Miss rate |
| Eyes closed | 75% | 20% | 5% |
| Yawning | 80% | 15% | 5% |
| Posture Analysis | 72% | 18% | 10% |
| Gaze Tracking | 60% | 35% | 5% |
| Cognitive Response | 40% | 20% | 40% |
| Peripheral Movements | 50% | 15% | 35% |

The number of test cases for each fatigue indicator is around 100 according to the amount of data.

# PROJECT DESIGN

## Project Main flow

This section outlines the operational workflow of our HighAlert Security Awareness Detection System, detailing the integrated use of advanced camera technology to monitor security guards during their shifts. The system leverages real-time image processing and object detection algorithms to enhance security operations effectively. Here are the steps illustrating how the system functions:

**Real-Time Video Capture**:

Implementation: Throughout security guard shifts, high-definition cameras strategically positioned around the premises capture video frames in real time. These cameras are equipped with the capability to handle various lighting conditions and angles, ensuring comprehensive coverage.

**Image Processing and Object Detection:**

Technology Used: Each captured frame is processed using the YOLO (You Only Look Once) model, a state-of-the-art deep learning algorithm known for its speed and accuracy in object detection.

Detection Goals: The model identifies key indicators of guard alertness and potential security breaches, including facial expressions like blinking or yawning, body posture, and general signs of awareness or unawareness.

**Alert Generation and Response Initiation:**

Automated Alerts: When the system detects any sign of inattention or suspicious activity, it automatically triggers alerts. These can range from a simple notification to the security guard to escalate alerts to the security control center for immediate action.

Actionable Intelligence: The system categorizes and prioritizes the detected events based on their potential security implications. This allows for quick and efficient responses tailored to the specific nature of the observed incident.

**Feedback Mechanism and Guard Notification**:

Display and Notification: Any concerning behaviors or detections are immediately communicated to the security personnel through an intuitive interface on the control panel or dedicated monitors. This interface provides real-time feedback and actionable instructions to the guards, empowering them to take prompt corrective actions.

Outcome and Iteration: Feedback from these interactions is logged and analyzed to continuously improve the system’s accuracy and responsiveness.

## Unaware Detection

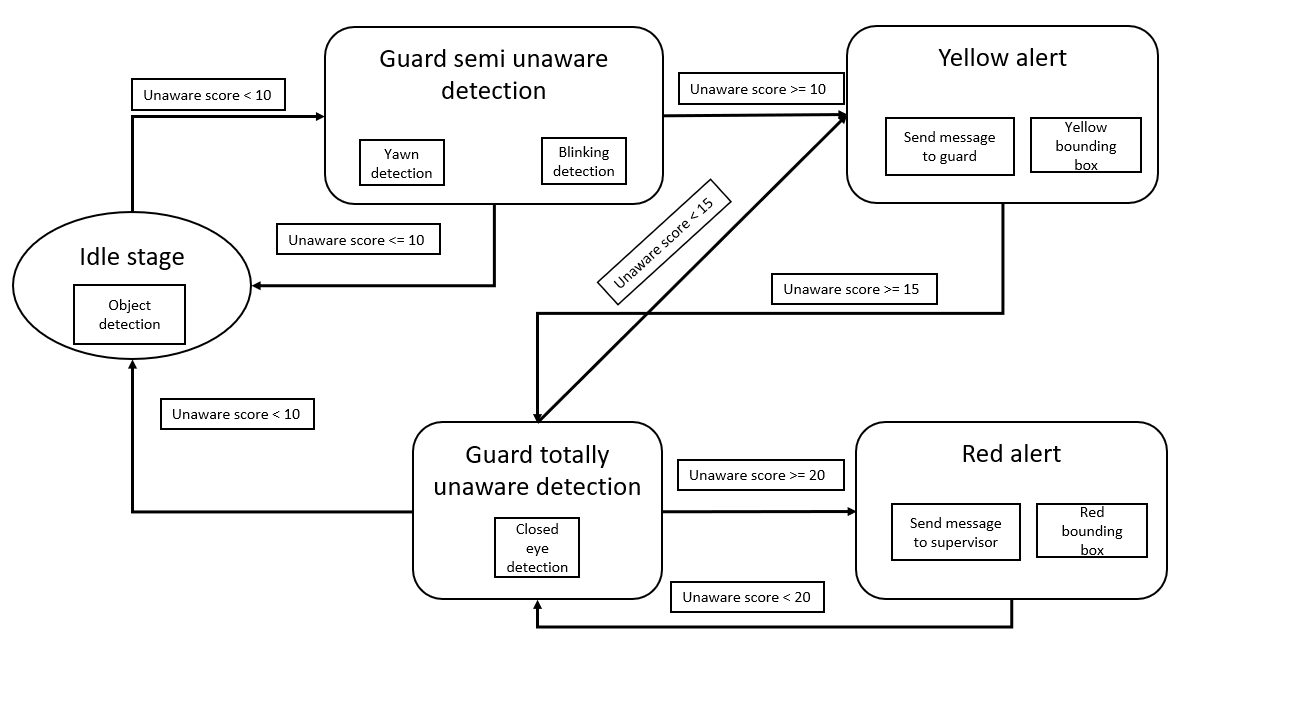


Figure : Full unaware point system detection

### Blinking Detection

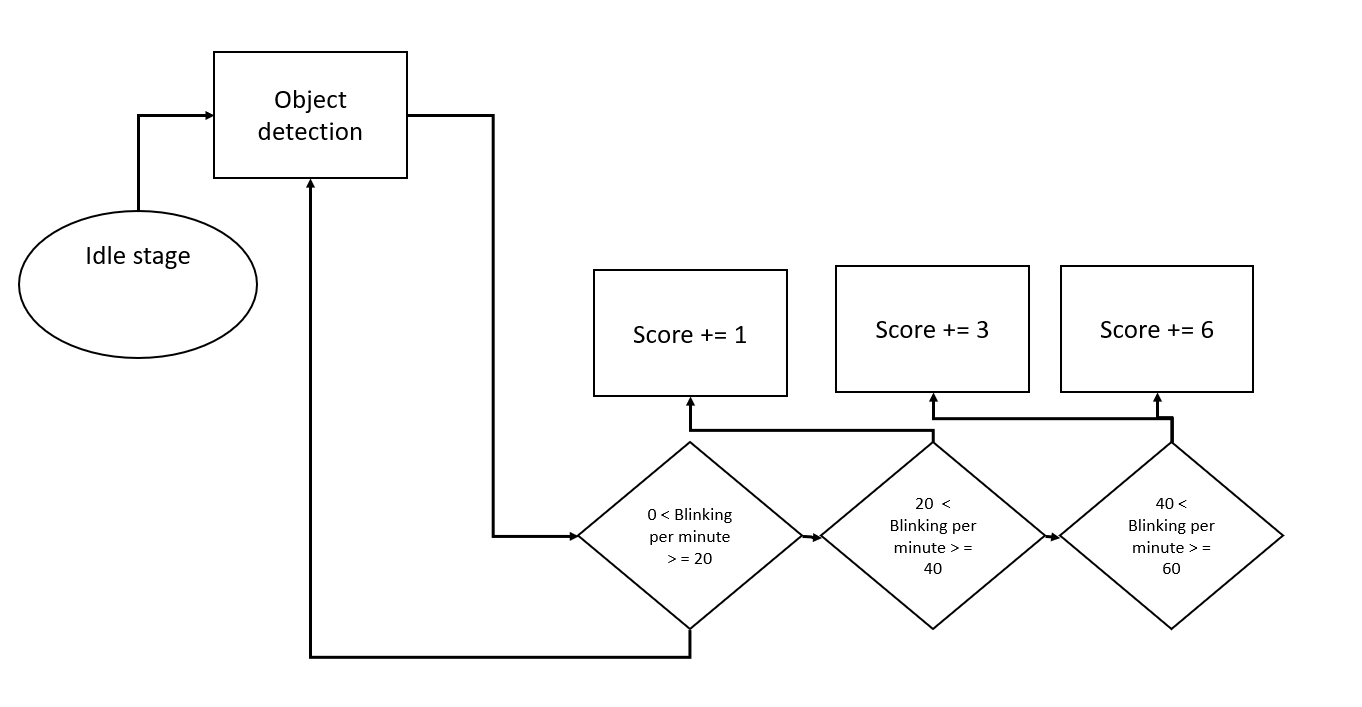


Figure 4: Blinking Detection score system

### Yawn Detection

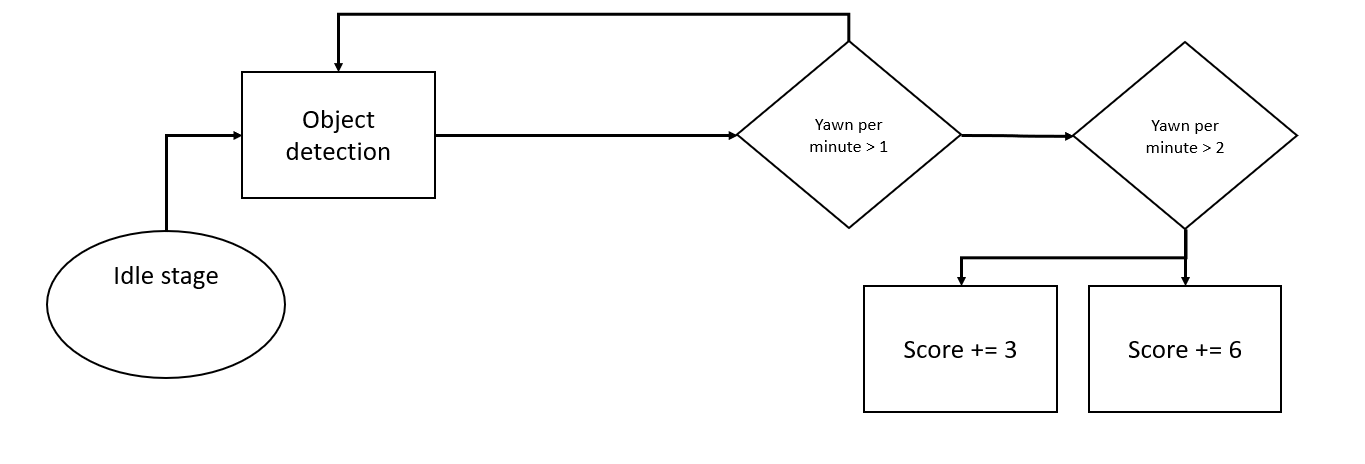


Figure 5: Yawn detection score system

## Project Diagrams

### Use Case Diagram

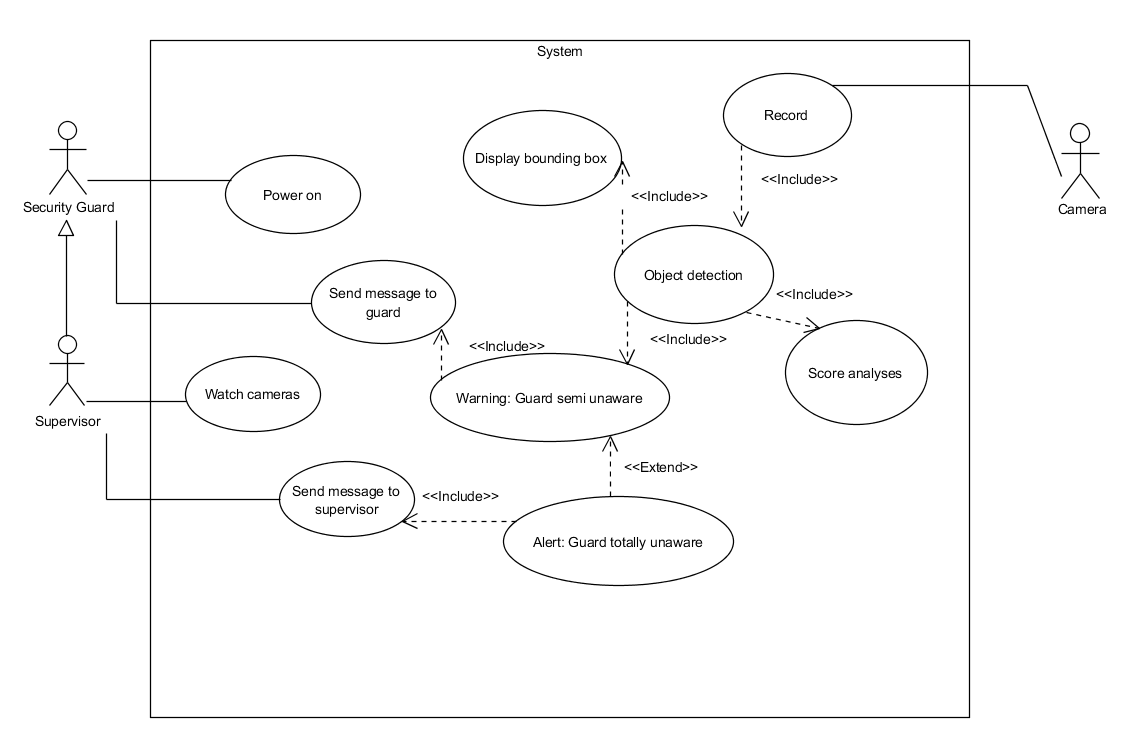


Figure 6: Use Case Diagram

### Activity Diagram

Figure 7: Activity Diagram

The diagram shows the system alert function without the point scoring, when and how to alert and when the system is in a loop so we could figure if the security guard is aware or unaware.

# Verification plan

**Test 1: Camera Test, expected result – Camera working properly.**

1. Connect camera to system
2. Launch program for camera footage display
3. Verify camera footage is displayed correctly
4. Troubleshoot if camera footage is incorrect

**Test 2: Face Recognition Test, expected result – Face detected.**

1. Turn on face recognition module in system
2. Place subject in front of camera
3. System recognizes face and alerts operator
4. Troubleshoot if system fails to recognize face

**Test 3: Blink Detection Test, expected result – Blinking detected.**

1. Activate blink detection module in system
2. Ask subject to sit in front of camera
3. System detects when subject blinks
4. Troubleshoot if system fails to detect blinks

**Test 4: Yawning Detection Test, expected result – Yawning detected.**

1. Activate yawning detection module in system
2. Ask subject to sit in front of camera
3. System detects when subject yawns
4. Troubleshoot if system fails to detect yawns

**Test 5: Sloppy Detection Test, expected result – Sloppiness detected.**

1. Activate sloppy detection module in system
2. Ask subject to sit in front of camera
3. System detects when subject is sloppy
4. Troubleshoot if system fails to detect sloppy positions

**Test 6: Full Sleep Detection Test, expected result – Full sleep detected.**

1. Activate full sleep detection module in system
2. Ask subject to sit in front of camera
3. System detects when subject appears to be asleep or dozing off
4. Troubleshoot if system fails to detect full sleep

**Test 7: System Calculation Test, expected result – All score functions working.**

1. Compile data from all previous tests
2. Run system's scoring algorithm to calculate final score
3. Verify system accurately calculates points for each test

**Test 8: Blink Count Test, expected result – Blinking score working.**

1. Activate blink counting module in system
2. Ask subject to sit in front of camera
3. System counts number of blinks made by subject
4. Calculate points based on system's scoring algorithm

**Test 9: Yawn Count Test, expected result – Yawn score working.**

1. Activate yawning counting module in system
2. Ask subject to sit in front of camera
3. System counts number of yawns made by subject
4. Calculate points based on system's scoring algorithm

**Test 10: Sloppy Count Test, expected result – Sloppy score Working.**

1. Activate sloppy counting module in system
2. Ask subject to sit in front of camera
3. System counts number of times subject assumes sloppy position
4. Calculate points based on system's scoring algorithm

# Evaluation

In the "HighAlert" project, the evaluation of the security awareness detection system focused on several key metrics to assess its effectiveness, accuracy, and reliability. Here’s a detailed breakdown of how the system was evaluated:

## Accuracy of Detection

Method: The system's ability to detect inattention and unawareness accurately was tested using a set of pre-recorded videos that depicted various levels of guard awareness.

## Real-time Performance:

Method: Testing the system’s performance in real-time environments to ensure that it could process video feeds and generate alerts with minimal delay.

## Usability Testing

Method: Conduct usability testing with actual security guards to assess the system's ease of use and the effectiveness of the user interface.

Results: Feedback was collected on how intuitive the guards found the system and whether the alerts and instructions provided were clear and actionable. Adjustments were made based on this feedback to improve user experience.

## Robustness and Adaptability

Method: Evaluate the system’s robustness across different environments and conditions, such as varying lighting, weather, and camera placements.

Results: The system was tested in multiple settings to ensure it remained effective under different conditions. The adaptability was measured by its ability to maintain high accuracy levels despite these environmental changes.

## Impact on Security Outcomes

Method: Analyze the impact of the system on improving security outcomes by comparing incident rates before and after its implementation.

Results: Data on security breaches, incidents of inattention, and overall safety improvements were collected and analyzed to gauge the system’s impact on operational security.

## Scalability

Method: Tests were conducted to assess how well the system could be scaled up to cover larger areas or multiple sites.

Results: The scalability was evaluated by incrementally increasing the number of cameras and data inputs the system had to handle, ensuring that performance did not degrade as the system was scaled.

The evaluation confirmed that the "HighAlert" system significantly enhanced the monitoring of security guards’ awareness, reducing incidents caused by inattention. The system was found to be reliable, user-friendly, and adaptable to various operational environments, making it a valuable tool for improving public safety in monitored areas.

# Process Challenges

## Addressing Data Acquisition and Model Training Challenges in the HighAlert Security Awareness Detection System

In the development of the HighAlert project, one of the primary challenges faced was the acquisition of appropriate data for training our machine learning model. Initially, the available data was insufficient for our needs, which necessitated innovative approaches to enhance our model’s performance.

## Data Augmentation through Transfer Learning and Custom Data Creation

To overcome the limitations of the sparse dataset, we employed a strategy of transfer learning. This involved utilizing pre-trained weights from the well-established YOLO (You Only Look Once) object detection system as a foundational basis for our model. This approach allowed us to leverage YOLO’s robust detection capabilities while tailoring the model to our specific requirements.

Recognizing the need for more targeted data, we expanded our dataset by creating custom video and image content. This was achieved by filming ourselves in various camera angles to simulate the conditions under which the security system would operate. These recordings were crucial in providing a diverse range of data, particularly for scenarios where security guards were not directly facing the camera—a significant challenge in real-world applications.

## Enhancing the Model with Custom-Created and Online Data

Further refinement of the model was accomplished by incorporating additional data sourced online. We carefully selected photos and videos that were relevant to our project's focus on security awareness, modifying and cropping these images to better suit our training objectives. This enriched dataset significantly improved the model's ability to recognize and analyze less straightforward angles and nuanced behaviors.

## Optimization through Object-Oriented Programming

To streamline the operational efficiency of our system, we revised the original YOLO detection code into a more Object-Oriented Programming (OOP) structure. This restructuring was critical for optimizing the system’s performance: by loading the model a single time during the initialization phase, we ensured that each subsequent frame was processed independently without the need to reload the model. This enhancement markedly sped up the detection process, enabling real-time performance that is crucial for the immediate response required in security monitoring.

## Conclusion

These strategic decisions in data handling and software architecture were pivotal in overcoming the initial challenges of dataset inadequacy and technical constraints. The HighAlert project's success in developing an effective security awareness detection system was significantly bolstered by these methodologies, leading to a robust solution capable of operating efficiently in diverse and dynamic environments.

# User Documentation

The HighAlert application is designed to monitor security guard attentiveness using a straightforward interface. The main screen displays a live feed from the camera, allowing users to observe real-time activity. The interface includes two primary buttons: one for initiating the recording process and another for stopping it.

Additionally, the interface features a score display, providing users with immediate feedback on the current alertness level. A timer is included to track the duration of monitoring sessions, while counters for yawns and blinks help users monitor specific behaviors.

When the application detects concerning behaviors such as excessive yawning or prolonged eye closure, it displays alerts on the screen, ensuring that users are promptly notified of potential issues.

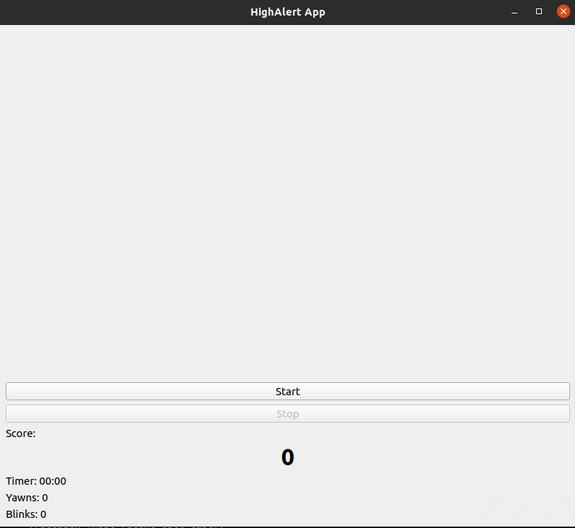
HigheAlert main screen of the application, keeping it simple, by pressing start the recording of the guard will begin and the score system will start.

Figure 8: Main screen of the application

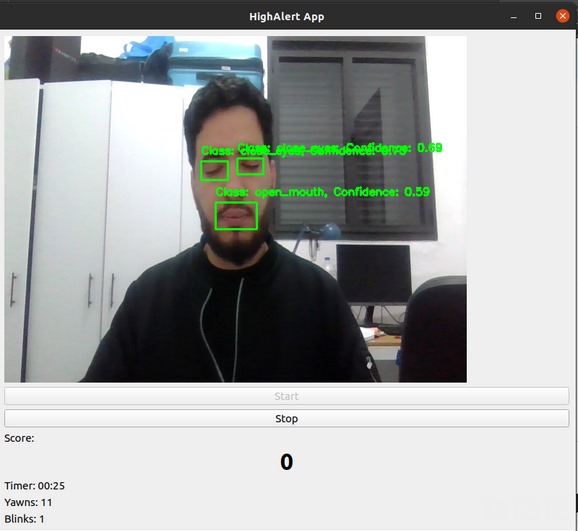


Figure 9: A look on eyes and mouth detection

This is how the user will see that application while it runs, he can see his score live bordcast and will know when he will need to refresh himself.

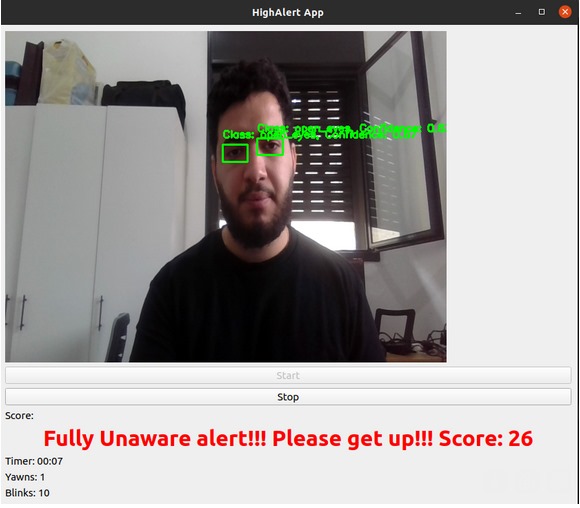


Figure 10: Fully Unaware detection

Fully unaware detection, the score limit is reached and a proper message is sent.

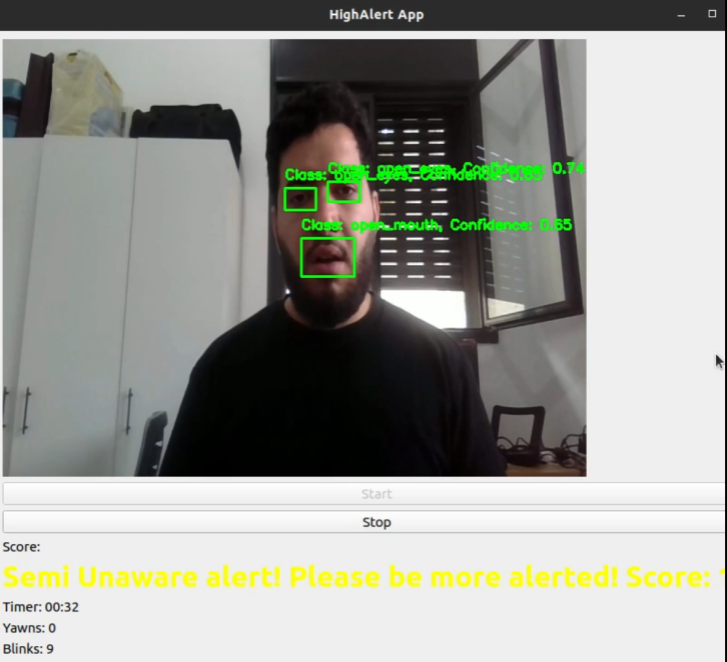


Figure 11: Semi Unaware detection

Semi unaware detection, the yellow alert is sent so the guard will notice with a proper message.

# Maintenance guide

## Hardware and Software requirements

**First Option**:

Hardware: Jetson Nano devkit, Raspberry Pi camera module

Software:

Python 3.6

Torch 1.10

TorchVision

NumPy

OpenCV

YOLOv5

**Second Option**:

Hardware: Nvidia Jetson Orin AGX devkit, Intel RealSense camera

Software:

Python 3.8

Torch 2.0

TorchVision

NumPy

OpenCV

PyRealSense

YOLOv7

Ensure that all software dependencies are installed and configured correctly before running the application on the respective hardware platforms.

## INSTELLATION GUIDE

**Git clone from this repository** :

https://github.com/HaimHH/HighAlert.git

**In the Jetson Nano Linux terminal**:

Pip install -r requirementss.txt

# References

[1] Security Industry Association (SIA). (2018). Security megatrends: The top 10 trends shaping the security industry. Retrieved from <https://www.securityindustry.org/education-resources/security-megatrends/>

[2] National Institute for Occupational Safety and Health (NIOSH). (2013). Police and security officer fatigue: Current trends and issues. Retrieved from <https://www.cdc.gov/niosh/docs/2013-138/pdfs/2013-138.pdf>

[3] International Foundation for Protection Officers (IFPO). (2018). Sleep and fatigue: The impact on security officers. Retrieved from <https://www.ifpo.org/resource-links/industry-news/sleep-and-fatigue-the-impact-on-security-officers/>

[4] A novel Capsule Neural Network based model for drowsiness detection using electroencephalography signals

<https://www.sciencedirect.com/science/article/pii/S0957417422004031>  
[5] [Frontiers | A One-Dimensional CNN-LSTM Model for Epileptic Seizure Recognition Using EEG Signal Analysis (frontiersin.org)](https://www.frontiersin.org/articles/10.3389/fnins.2020.578126/full)

<https://www.frontiersin.org/articles/10.3389/fnins.2020.578126/full>  
[6] [Abnormal Heart Rate Variability as a Manifestation of Autonomic Dysfunction in Hemispheric Brain Infarction | Stroke (ahajournals.org)](https://www.ahajournals.org/doi/full/10.1161/01.STR.27.11.2059)

<https://www.ahajournals.org/doi/full/10.1161/01.STR.27.11.2059>  
[7] [Drowsiness monitoring based on steering wheel status - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1361920917306582)

<https://www.sciencedirect.com/science/article/pii/S1361920917306582>  
[8] [Iris localization using rough entropy and CSA: A soft computing approach - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1568494618301066)

<https://www.sciencedirect.com/science/article/pii/S1568494618301066>  
[9] [Sensors | Free Full-Text | Convolutional Neural Network for Drowsiness Detection Using EEG Signals (mdpi.com)](https://www.mdpi.com/1424-8220/21/5/1734)

<https://www.mdpi.com/1424-8220/21/5/1734>  
[10] [dlib C++ Library](http://dlib.net/) http://dlib.net/  
[11] [Supervised Learning | SpringerLink](https://link.springer.com/chapter/10.1007/978-3-540-75171-7_2)  
<https://link.springer.com/chapter/10.1007/978-3-540-75171-7_2>  
[12] [[1506.02640] You Only Look Once: Unified, Real-Time Object Detection (arxiv.org)](https://arxiv.org/abs/1506.02640)  
<https://arxiv.org/abs/1506.02640>  
[13] [OpenCV - Open Computer Vision Library](https://opencv.org/) https://opencv.org/  
[14] [Introducing the NVIDIA Jetson Nano - Hackster.io](https://www.hackster.io/news/introducing-the-nvidia-jetson-nano-aaa9738ef3ff)  
https://www.hackster.io/news/introducing-the-nvidia-jetson-nano-aaa9738ef3ff