

# *AI-Powered Compliance Monitoring for Retail Staff*

## **Detailed System Design and Implementation Report**

### *Group 4*

<b>Name</b>	<b>Position</b>	<b>Email</b>	<b>Phone</b>
<b>Vuong Khang Minh</b>	Document Researcher, Document Organizer, AI Specialist	104179690@student.swin.edu.au	0967572933
<b>Nguyen Dang Duc Anh</b>	Software Specialist, Requirement Analysis	104182520@student.swin.edu.au	0947023256
<b>Nguyen Ha Huy Hoang</b>	Team leader/Project Manager	103487444@student.swin.edu.au	0358791781
<b>Nguyen Dang Khanh Toan</b>	Quality Assurance, Domain Expert, Software Specialist	103487389@student.swin.edu.au	0888752003
<b>Nguyen Cuong Nhat</b>	AI Specialist, Requirement Analysis	104178590@student.swin.edu.au	0974247189

## 1 DOCUMENT CHANGE CONTROL

Version	Date	Authors	Summary of Changes
1	20/7/2024	Vuong Khang Minh	Initial version of the document, detailing the system architecture, definitions, assumptions, and simplifications.
1.1	21/7/2024	Nguyen Dang Duc Anh	Updates to software requirements and detailed design sections.
1.2	22/7/2024	Nguyen Ha Huy Hoang	Updated system architecture.
1.3	25/7/2024	Nguyen Dang Khanh Toan	Quality assurance checks and additional domain expert insights integrated.
1.4	25/7/2024	Nguyen Cuong Nhat	Final revisions, and requirement analysis improvements.

### - DOCUMENT SIGN OFF

Name	Position	Signature	Date
Vuong Khang Minh	Document Researcher, Document Organizer, AI Specialist	<i>minh</i>	10/7/2024
Nguyen Dang Duc Anh	Software Specialist, Requirement Analysis	<i>anh</i>	10/7/2024
Nguyen Ha Huy Hoang	Team leader/Project Manager	<i>hoang</i>	10/7/2024
Nguyen Dang Khanh Toan	Quality Assurance, Domain Expert, Software Specialist	<i>toan</i>	10/7/2024
Nguyen Cuong Nhat	AI Specialist, Requirement Analysis	<i>nhat</i>	10/7/2024

- **CLIENT SIGN OFF**

Name	Position	Signature	Date
Nguyen Vo Thanh Khang	AI Researcher	<i>khang</i>	10/7/2024
Organisation			
QUY NHON AI CREATIVE VALLEY			

## 1. Introduction

The Employee Compliance Checker (ECC) is a software system designed to enhance employee monitoring in retail environments. By leveraging AI and video analysis technologies, the ECC collects and analyzes CCTV feeds to monitor employee compliance, generate detailed reports, and provide timely notifications.

This Detailed Design and Implementation Report aims to guide the development team by presenting the system's architecture, detailed design, and implementation approach. This report is intended for project stakeholders, developers, testers, and anyone involved in the system's lifecycle.

### 1.1 Overview

This document provides a comprehensive overview of the Employee Compliance Checker, covering its architecture, detailed design, and implementation strategy. It outlines the assumptions and simplifications made during the design process, presents the system architecture, and offers a detailed design using an object-oriented approach. Additionally, the document discusses design verification and the current state of system implementation.

### 1.2 Definitions, Acronyms and Abbreviations

**AI:** Artificial Intelligence

**CCTV:** Closed-Circuit Television

**ECC:** Employee Compliance Checker

**GUI:** Graphical User Interface

**HTTPS:** Hypertext Transfer Protocol Secure

**POS:** Point of Sale

**SRS:** Software Requirements Specification

**YOLO:** You Only Look Once - A state-of-the-art, real-time object detection system developed by Joseph Redmon and Ali Farhadi.

**STGCN:** Spatial Temporal Graph Convolutional Networks - A framework for skeleton-based action recognition, which captures spatial and temporal features for analyzing human actions from video data

### 1.3 Assumptions and Simplifications

In developing the detailed system design for the Employee Compliance Checker, several assumptions and simplifications were made to ensure the system is both effective and manageable. These assumptions are crucial to understand the scope and functionality of the system, as well as to acknowledge any limitations.

**Assumptions:**

**1. Employee Awareness and Consent:**

- It is assumed that all employees are informed of the compliance check policy and have consented to the monitoring. This ensures transparency and adherence to ethical standards in the workplace.

**2. Camera Positioning:**

- Cameras are strategically positioned to cover all necessary areas within the retail environment. This placement is designed to maximize the effectiveness of monitoring while respecting employees' privacy in non-work-related areas (e.g., restrooms, break rooms).

**3. Notification and Reporting:**

- The system is primarily designed to provide notifications and generate reports based on analyzed data. Real-time interventions by the system are optional and not mandatory. This approach balances the need for timely alerts with the practicality of automated monitoring.

**4. Exclusion of Facial Recognition:**

- Due to privacy concerns and the need for legal clearance, facial recognition or personal identification features are excluded from the system. This ensures that the system complies with privacy laws and regulations while maintaining the integrity of the compliance monitoring process.

**5. Retail Environment Focus:**

- The system is designed exclusively for use in retail environments for employee monitoring. This focus allows for a more tailored design that meets the specific needs and challenges of retail operations, such as customer service and loss prevention.

**Simplifications:**

**1. Limited Scope of Real-Time Interventions:**

- The decision to make real-time interventions optional simplifies the system design. This allows the system to focus on data collection, analysis, and reporting, reducing the complexity and potential issues related to real-time decision-making and actions.

**2. Predefined Camera Locations:**

- By assuming predefined camera locations, the design does not account for dynamic or user-defined camera positioning. This simplification streamlines the design process and ensures consistent coverage across all monitored areas.

**3. Privacy Compliance:**

- Excluding facial recognition from the system design simplifies the development and implementation process, as it avoids the complexities of handling sensitive biometric data and navigating privacy regulations.

**4. Single Domain Operation:**

- Limiting the system's operation to retail environments simplifies the design by focusing on a specific set of requirements and challenges. This targeted approach allows for more effective and efficient system development.

By clearly stating these assumptions and simplifications, we aim to provide a transparent and realistic understanding of the system's design and capabilities. These considerations are essential for stakeholders to evaluate the system's applicability and to plan for any future enhancements or adjustments.

## 2. System Architecture Overview

The Employee Compliance Checker system is designed to ensure retail employees adhere to company policies using AI-powered analysis of CCTV feeds. The architecture is crafted to provide real-time monitoring and analysis while leveraging cloud-based scalability for comprehensive data management.

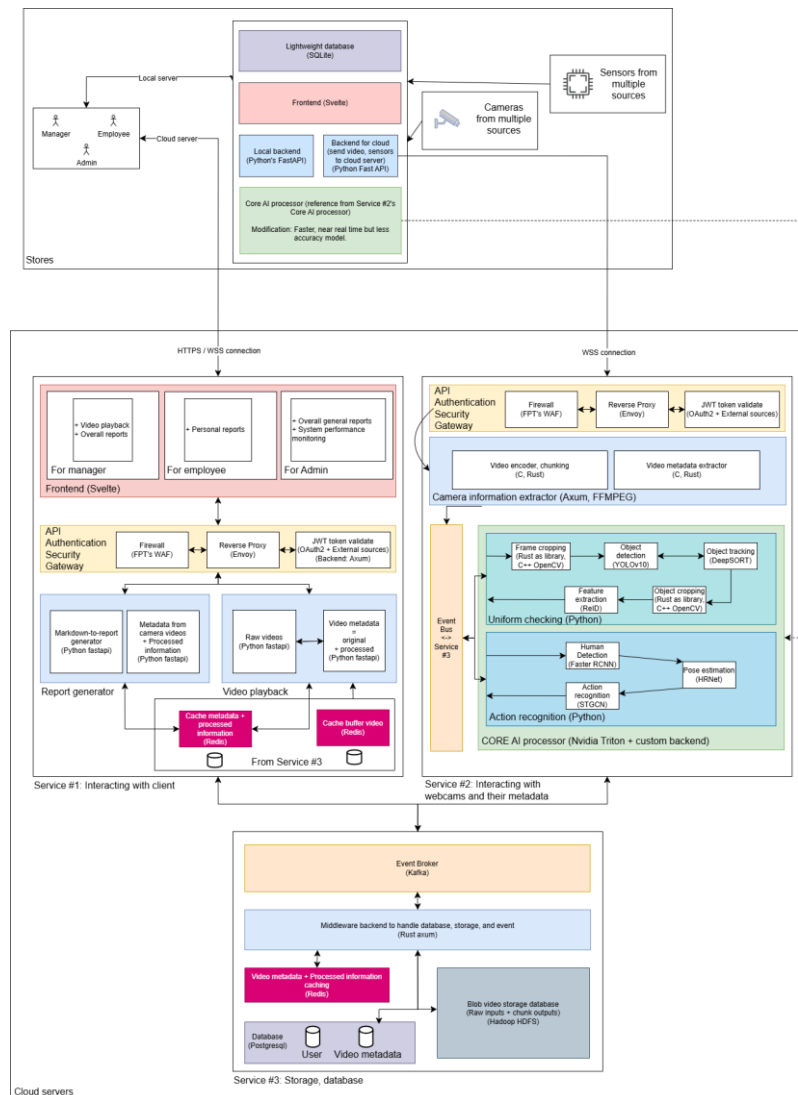


Figure 1. High-Level System Architecture

The system architecture consists of several key components, each playing a crucial role in ensuring efficient operation and data processing:

### Local Servers in Stores:

- **Local Database:** Stores critical data locally to ensure quick access and initial processing.
- **Frontend Interface:** Built with Svelte, this interface allows managers, employees, and administrators to interact with the system. Managers can view compliance reports and video playback, employees can check their compliance status, and administrators can monitor system performance.

- **Local Backend:** Uses FastAPI to handle on-site processing tasks, including preliminary analysis of video feeds.

**Cloud Infrastructure:**

- **Cloud Server:** Provides robust computing power for advanced AI analysis and centralized data storage. It communicates with local servers using secure HTTPS connections.
- **Cloud Backend:** Manages detailed data analysis and long-term storage, ensuring that the system can handle extensive data from multiple locations.

**AI Processing Components:**

- **Frame Cropping:** Extracts relevant parts of video frames for further analysis.
- **Object Detection:** Identifies objects and actions within the video frames.
- **Action Recognition:** Determines specific employee actions to check compliance with protocols.
- **Pose Estimation:** Analyzes human poses to ensure safety and adherence to guidelines.
- **Uniform Checking:** Verifies if employees are correctly wearing their uniforms.

**Event Bus Service:**

- Ensures smooth data flow between various system components and facilitates real-time processing.

**Database Systems:**

- **PostgreSQL:** Handles structured data storage.
- **Redis:** Provides caching for fast data access.
- **Hadoop HDFS:** Manages large-scale video file storage.

**API Security and Management:**

- **Firewall and WAF:** Protects against external threats.
- **Reverse Proxy:** Manages traffic between clients and services.
- **JWT Token Validation:** Secures access to system resources.

### 3. Detailed System Design

#### 3.1. The Detailed Design and Justification

##### UML Diagram

The system architecture involves various components interacting seamlessly to ensure real-time compliance monitoring and reporting. The following UML diagram presents a high-level overview of these interactions.



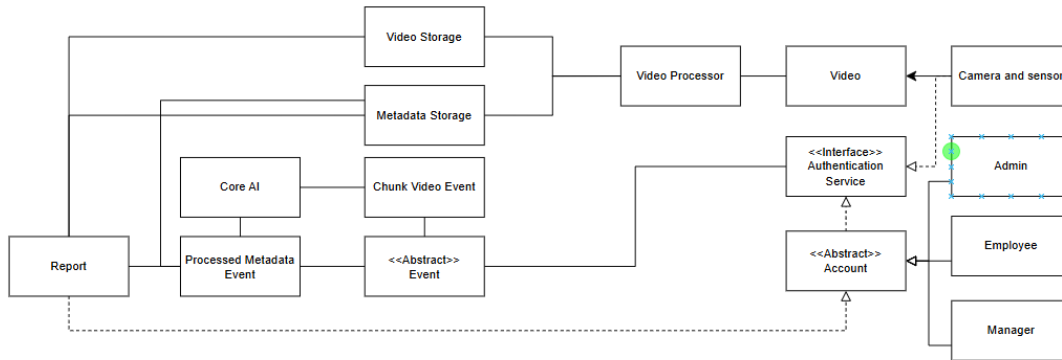


Figure 2. UML diagram

**UML Diagram:**

- **Video Storage:** Stores the recorded video footage.
- **Metadata Storage:** Holds metadata associated with video events.
- **Core AI:** Processes the video and metadata to generate compliance events.
- **Video Processor:** Handles the video streams from cameras and sensors.
- **Authentication Service:** Manages user authentication for accessing the system.
- **Admin, Employee, Manager Interfaces:** Different user roles interacting with the system.
- **Report Generation:** Compiles and generates compliance reports.

**Design Patterns and Heuristics**

In designing the Employee Compliance Checker system, several design patterns and heuristics have been employed to ensure scalability, maintainability, and efficiency:

**1. Observer Pattern:**

- **Justification:** This pattern is used to ensure that the system components can react to events in real-time. For example, when a new video is processed, the system can automatically update the compliance reports.
- **Implementation:** The Event Bus Service acts as the subject, notifying various services (observers) about new events.

## 2. Singleton Pattern:

- Justification:** Ensures that certain components, such as the Core AI processor and authentication service, are single instances to avoid inconsistent states and ensure centralized control.
- Implementation:** The Core AI processor is instantiated once and reused across different components for uniform data processing.

## Bootstrap Sequence

The system's bootstrap sequence ensures that all components are initialized in the correct order, and dependencies are resolved before the system becomes operational. The sequence diagram illustrates the interactions during the system startup.

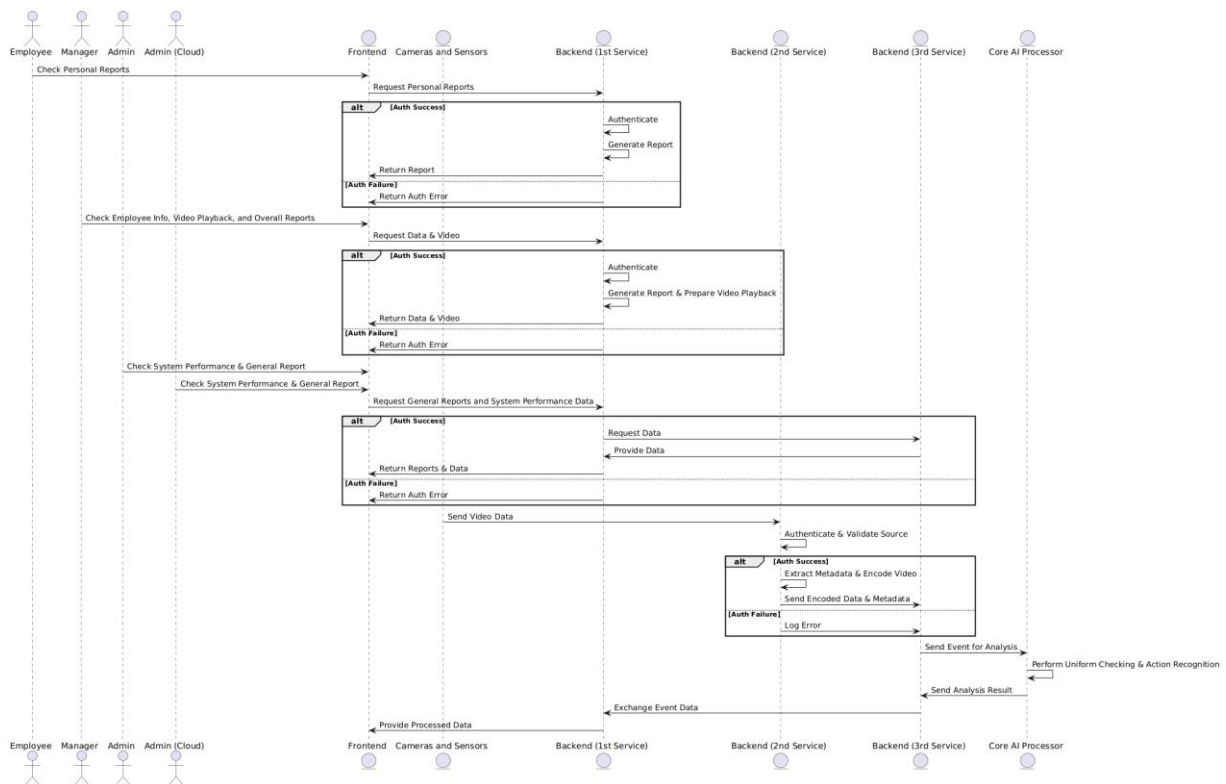


Figure 3. Bootstrap Sequence

### **Bootstrap Sequence Steps**

#### **1. Check Personal Reports:**

- Employee/Manager/Admin/Admin (Cloud) sends a request to check personal reports.
- Frontend requests personal reports from Client Service.
- Client Service authenticates the request.
  - On success: Client Service generates and returns the report.
  - On failure: Client Service returns an authentication error.

#### **2. Check Employee Info, Video Playback, and Overall Reports:**

- Manager/Admin/Admin (Cloud) requests data and video.
- Frontend sends a request to Client Service.
- Client Service authenticates the request and prepares the video playback.
  - On success: Client Service returns the data and video.
  - On failure: Client Service returns an authentication error.

#### **3. Check System Performance & General Report:**

- Admin/Admin (Cloud) requests system performance and general reports.
- Frontend sends the request to Client Service.
- Client Service requests data from Storage Service.
- Storage Service provides the required data to Client Service.
- Client Service authenticates the request.
  - On success: Client Service returns the reports and data.
  - On failure: Client Service returns an authentication error.

#### **4. Send Video Data:**

- Cameras and Sensors send video data to Client Service.
- Client Service authenticates and validates the video source.

- On success: Camera Service extracts metadata, encodes the video, and sends encoded data and metadata.
- On failure: Camera Service logs an error.

#### 5. Exchange Event Data:

- Client Service sends event data for analysis to Core AI Processor.
- Core AI Processor performs uniform checking and action recognition.
- Core AI Processor sends the analysis result back to Client Service.
- Client Service provides the processed data to Frontend.

This sequence ensures that all components are properly initialized and ready for operation, enabling a seamless and secure user experience.

#### Justification for the Chosen Design Approach

The Responsibility-Driven Design (RDD) approach has been chosen for this system. RDD focuses on assigning responsibilities to various objects, ensuring that each component has a clear and distinct role. This approach is particularly suitable for the Employee Compliance Checker system due to the following reasons:

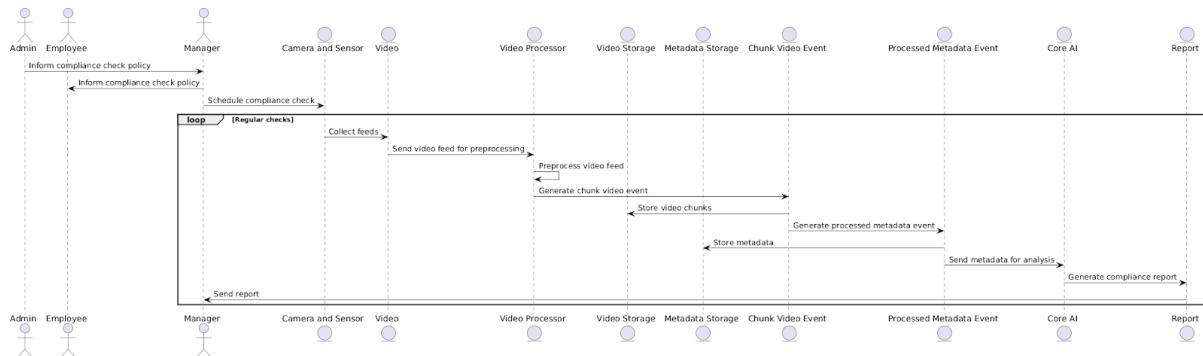
- **Modularity:** By clearly defining responsibilities, the system components can be developed, tested, and maintained independently.
- **Scalability:** The system can easily accommodate new features or components without significant changes to the existing architecture.
- **Maintainability:** Responsibilities are encapsulated within individual components, making it easier to identify and address issues.

In conclusion, the detailed design of the Employee Compliance Checker system leverages robust design patterns and a well-defined bootstrap sequence to ensure efficient and reliable operation. The Responsibility-Driven Design approach aligns with the system's requirements for modularity, scalability, and maintainability, making it the most suitable choice for this project.

### 3.2. Design Verification

The proposed approach was validated by simulating many use cases indicated below.

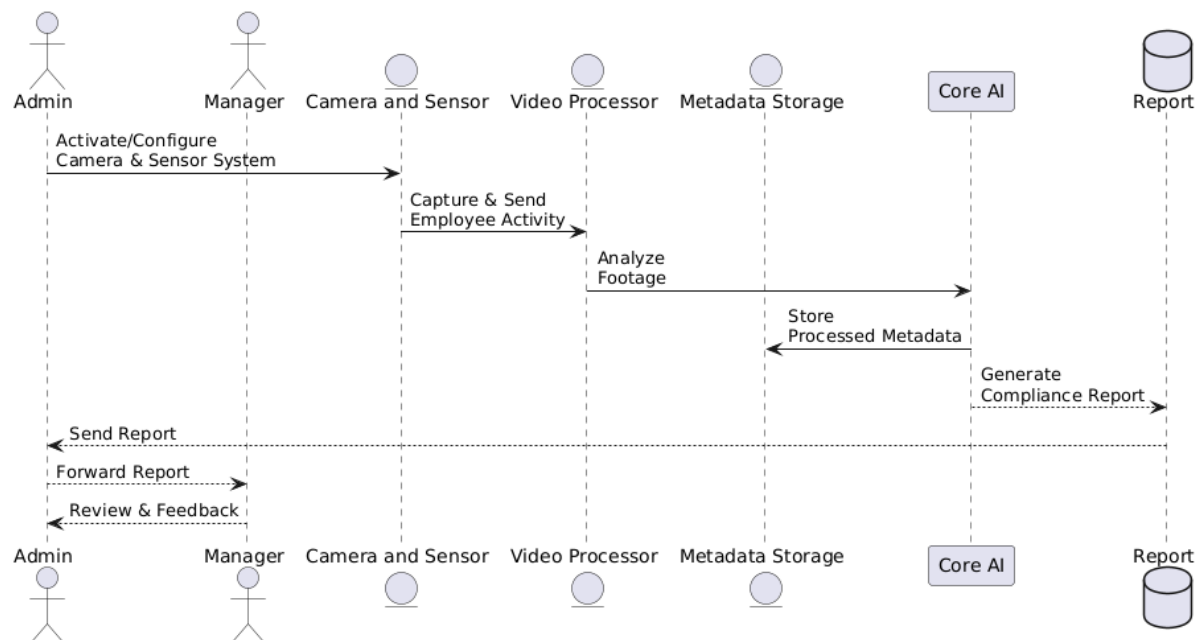
#### Camera Feed Collection



The system design ensures that camera feeds are collected and processed efficiently:

- **Scenario:** Admin schedules compliance checks and informs employees.
- **Design Support:** The system utilizes strategically placed cameras and sensors to capture video feeds. The Video Processor preprocesses these feeds, which are then stored in the Metadata Storage for subsequent analysis by the Core AI. This process is validated through simulated compliance checks, confirming that video feeds are collected, processed, and stored accurately.

#### Employee Analysis

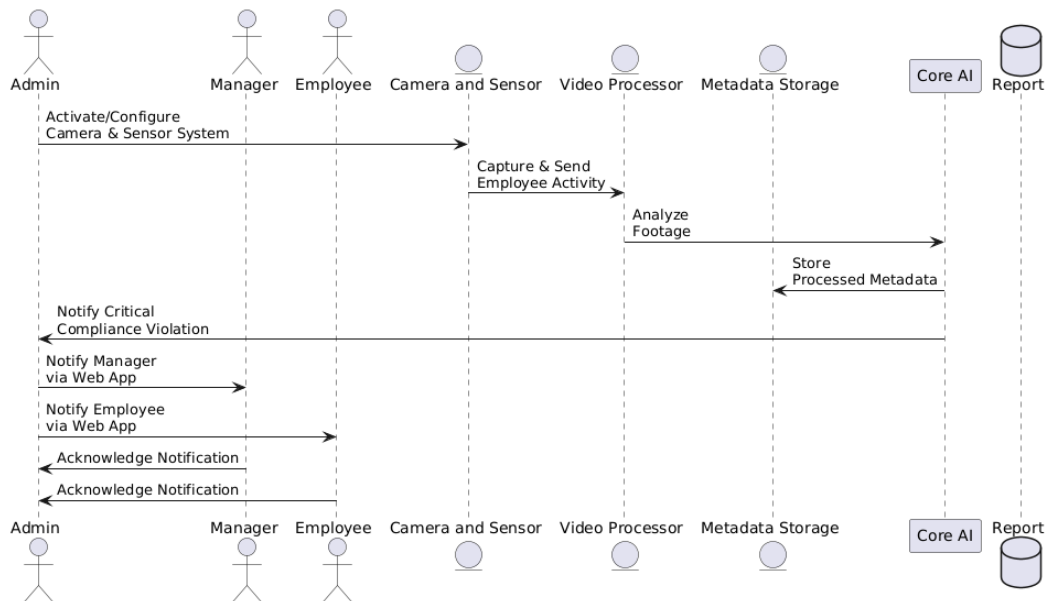


The system design effectively supports detailed employee analysis:

- **Scenario:** Admin activates and configures the camera system for employee monitoring.

- **Design Support:** The Camera and Sensor systems capture detailed footage of employee activities. This footage is analyzed by the Video Processor, and the resulting metadata is stored in the Metadata Storage. The Core AI utilizes this metadata to generate comprehensive compliance reports. Simulations of employee activities validate that the system can accurately capture and analyze employee behavior, ensuring compliance with company policies.

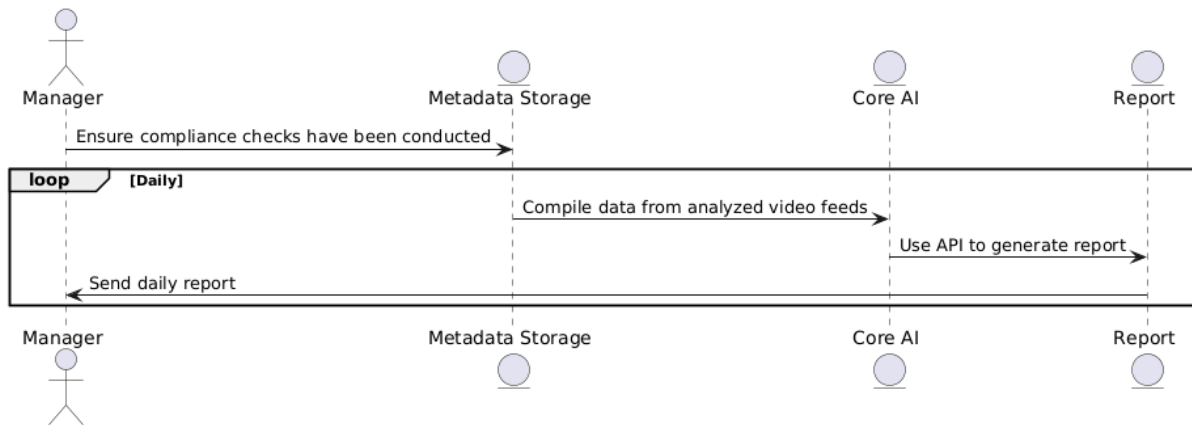
## Notifications



The design includes a robust notification system for critical compliance violations:

- **Scenario:** Admin configures the system to send notifications for compliance breaches.
- **Design Support:** Upon detecting critical compliance violations, the system sends real-time notifications to managers and employees through the web app. This process involves capturing video footage, processing it to extract relevant metadata, and storing this data in the Metadata Storage. The notification system is tested by simulating various compliance breaches, ensuring timely and accurate alerts.

## Generate Report



The system design supports the generation of detailed compliance reports:

- **Scenario:** Manager initiates daily compliance reporting.
- **Design Support:** The system compiles data from the analyzed video feeds stored in the Metadata Storage. The Core AI processes this data to generate detailed reports, which are then reviewed and sent by the manager. Simulations of daily report generation confirm the system's ability to compile, process, and present compliance data effectively.

## 4. Implementation

### 4.1. Overview of the State of the System Implementation

The implementation of the Employee Compliance Checker (ECC) system is progressing in alignment with the Software Requirements Specification (SRS), architecture design, and detailed design. The system development is structured in phases to ensure each component functions correctly before integration. The current implementation status is as follows:

#### 1. Completed Stages:

- **Staff Detection:** Implemented using YOLOv10 (Wang et al., 2024) for robust object detection.
- **Initial Uniform Detection:** Initial models trained to identify specific uniform features.
- **Basic Web-Based GUI:** Implemented to alert non-compliance.
- **Initial Violation Action Detection:** Initial STGCN model (Yan et al., 2018) trained for action recognition.

#### 2. Ongoing Stages:

- **Advanced Uniform Detection:** Enhancing models for more accurate identification of uniform features.
- **Advanced Violation Action Detection:** Refining the detection models for better accuracy.
- **Complete Website:** Developing a comprehensive website for system interaction.
- **Report Generation:** Implementing features to generate detailed compliance reports.
- **Scalability Enhancements:** Ongoing work to ensure the system can handle multiple video feeds and expand as needed.
- **Usability Improvements:** Developing a more comprehensive user interface for better monitoring and managing compliance status.

### 3. Future Stages:

- **Integration with Existing Systems:** Planning to integrate with HR and security systems for seamless operation.
- **Advanced Notification System:** Expanding notification options and integrating with additional communication platforms.
- **Comprehensive Reporting System:** Developing detailed reporting functionalities, including daily and weekly compliance reports.
- **Security Enhancements:** Implementing advanced security measures like end-to-end encryption and data anonymization techniques.

## 4.2. Key Implementation Decisions

### 1. Event-Driven Architecture

- **Decision:** Adopted an event-driven architecture to handle real-time data processing and responsiveness.
- **Rationale:** Ensures efficient handling of events such as compliance violations, with immediate notifications and updates.

### 2. Edge Computing for Initial Processing

- **Decision:** Use edge computing devices for initial video processing.
- **Rationale:** Reduces latency and bandwidth usage by processing video feeds locally before sending relevant event data to the central server.

### 3. AI Algorithms for Analysis

- **Decision:** Implement AI algorithms on the central server for deeper event analysis.



- **Rationale:** Leverages AI for accurate and efficient detection of compliance violations, improving the reliability of the system.

#### 4. Web-Based GUI for User Interaction

- **Decision:** Develop a web-based graphical user interface.
- **Rationale:** Provides an accessible and user-friendly platform for managers to view reports, receive notifications, and interact with the system.

#### 5. Security Measures

- **Decision:** Implement end-to-end encryption and data anonymization techniques.
- **Rationale:** Ensures data privacy and security, critical for handling sensitive compliance information.

### 4.3. Detail implementation

#### Detection Module

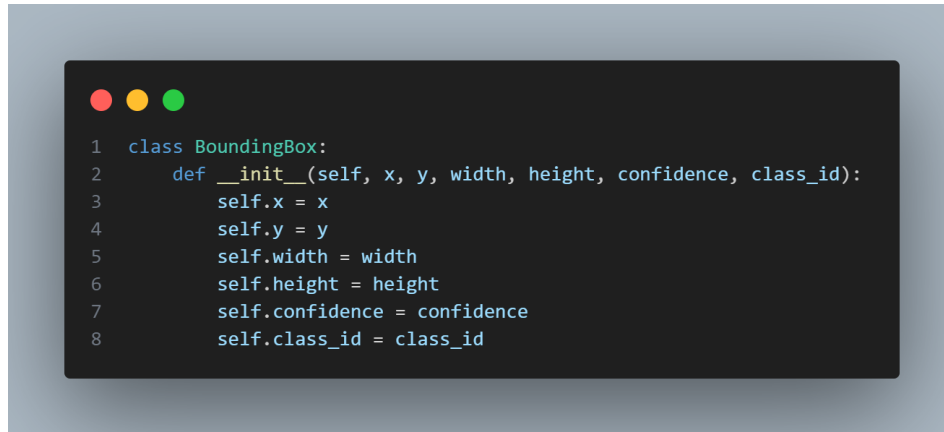
**Overview:** The Detection Module is responsible for identifying staff members in the video feed using the YOLOv10 object detection algorithm. It outputs bounding boxes around detected individuals for further processing.

#### Algorithms:

- **YOLOv10:**
  - A state-of-the-art object detection algorithm known for its accuracy and speed.
  - **Steps:**
    1. **Image Preprocessing:** Resizes and normalizes input frames.
    2. **Detection:** Processes the frames through a convolutional neural network to predict bounding boxes, class probabilities, and confidence scores.
    3. **Post-Processing:** Applies non-maximum suppression to filter out overlapping bounding boxes.

#### Data Structures:

- **Bounding Box:** Represents the detected region.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains the Python code for the BoundingBox class.

```
1 class BoundingBox:
2     def __init__(self, x, y, width, height, confidence, class_id):
3         self.x = x
4         self.y = y
5         self.width = width
6         self.height = height
7         self.confidence = confidence
8         self.class_id = class_id
```

Figure 4. Code for class BoundingBox

## Tracking Module

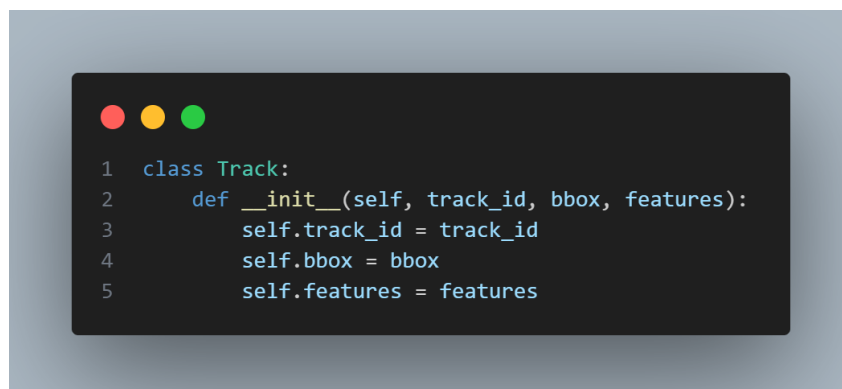
**Overview:** The Tracking Module uses DeepSORT to maintain the identities of detected individuals across frames, ensuring continuous monitoring.

### Algorithms:

- **DeepSORT:**
  - Combines Kalman filtering and Hungarian algorithm for data association.
  - **Steps:**
    1. **Feature Extraction:** Extracts features from detected bounding boxes.
    2. **Prediction:** Predicts the next position of existing tracks.
    3. **Association:** Matches detected bounding boxes with existing tracks using the Hungarian algorithm.
    4. **Update:** Updates the state of matched tracks and creates new tracks for unmatched detections.

### Data Structures:

- **Track:** Represents an individual track with its state

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains the Python code for the Track class.

```
1 class Track:
2     def __init__(self, track_id, bbox, features):
3         self.track_id = track_id
4         self.bbox = bbox
5         self.features = features
```

Figure 5. Code for class Track

## 5. References

1. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G 2024, Yolov10: Real-time end-to-end object detection, arXiv, viewed 26 July 2024, <https://arxiv.org/abs/2405.14458>.
2. Yan, S., Xiong, Y. & Lin, D 2018, Spatial temporal graph convolutional networks for skeletonbased action recognition, arXiv, viewed 26 July 2024, <https://doi.org/10.48550/arxiv.1801.07455>.