EyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores

Final Year Project Report

by

Afifa Masood	**BSEF21M004**
Hamza Arshad	**BSEF21M006**
Urooj Fatima	**BSEF21M019**
Rabail Waseem	**BSEF21M020**

Project Advisor:

Dr. Muhammad Farooq

Faculty of Computing & Information Technology,

University of the Punjab, Lahore, Pakistan.

(2025)

EyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores

Executive Summary

The **EyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores** project aims to revolutionize retail theft prevention by leveraging advancements in artificial intelligence (AI) and computer vision. The system processes live surveillance footage using machine learning models to detect suspicious behaviors indicative of shoplifting in real time. By automating theft detection, it reduces dependency on manual monitoring, minimizes human error, and provides store personnel with timely alerts and actionable insights.

The project encompasses key deliverables, including a detailed Software Requirements Specification (SRS), a comprehensive Software Design Specification (SDS), and a functional prototype. The SRS defines the functional and non-functional requirements, emphasizing scalability, usability, and data privacy. The SDS translates these requirements into modular system components, incorporating state-of-the-art object detection algorithms and behavior analysis models. The prototype demonstrates the integration of these components, validating the system's ability to detect shoplifting activities with high accuracy while maintaining real-time performance. EyeGuard is designed to be cost-effective, scalable, and adaptable to various retail environments, ensuring a secure and efficient shopping experience.

FYDP Overview

FYDP TitleEyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores

Table 1 Project Proposal Summary

Sr. No	Roll Numbers	Name	Signatures
1.	BSEF21M004	Afifa Masood	
2.	BSEF21M006	Hamza Arshad	
3.	BSEF21M019	Urooj Fatima	
4.	BSEF21M020	Rabail Waseem	
5.			

FYDP Goals
To develop an automated shoplifting-theft detection system using computer vision techniques and neural networks.
FYDP Objectives
Objective 1: To design a system that can identify suspicious behavior in real-time from surveillance footage.
Objective 2: To implement image processing and behavior analysis algorithms to recognize theft patterns.
Objective 3: To alert security personnel upon detection of potential theft actions.
FYDP Success Criteria
Successful real-time detection of suspicious activities with a high accuracy rate (e.g., >90%).
Positive feedback from stakeholders regarding usability and effectiveness.
Assumptions:
The system will have access to high-quality surveillance footage.
The neural network will be adequately trained with diverse datasets to recognize various theft scenarios.
Retail environments will permit the installation and use of the detection system without significant interference.
Risks & Obstacles
Inaccurate detections leading to false alarms, which could undermine the system's credibility.
Limited training data may hinder the neural network's performance in recognizing diverse theft behaviors.
Potential privacy concerns from customers and staff regarding surveillance footage.
Organization Address: FCIT, University of the Punjab, Lahore, Pakistan
Target End Users **Retail store owners, security personnel, and loss prevention teams.**
Suggested Project Supervisor: Dr.Muhammad Farooq
Approved By:
Date: October 07, 2024

Table of Contents[1]

[1] Once the document is completed, Right click on table of contents, Choose "Update Field" then choose "Update entire table" to automatically update table of contents

List of Tables

List of Figures

Chapter No 1

Project Proposal

1.1. Introduction

The retail industry faces a persistent challenge: shoplifting, which leads to substantial financial losses and affects customer trust. Traditional theft prevention methods, such as human monitoring and basic surveillance systems, are often labor-intensive, costly, and prone to errors. However, advancements in artificial intelligence (AI) and computer vision provide an opportunity to revolutionize how theft detection is approached. By integrating real-time video analysis and behavioral pattern recognition, AI-powered systems can bridge the gap between outdated techniques and modern, automated solutions. This project focuses on developing an intelligent shoplifting detection system to address the limitations of current methods, offering an efficient and scalable way to monitor retail spaces and prevent losses.

1.2. Background

The retail industry faces significant losses due to shoplifting, which impacts not only profitability but also customer trust. Traditional methods of preventing theft, such as human monitoring and surveillance cameras, are resource-intensive and prone to human error. With advancements in computer vision and machine learning, it is possible to build systems that automatically monitor, detect, and respond to suspicious activity. This project aims to bridge the gap between conventional surveillance methods and automated, real-time detection systems that use artificial intelligence to detect and prevent shoplifting incidents.

1.3. Problem Statement

Retail stores suffer from billions of dollars in losses annually due to shoplifting. Current theft detection relies on manual monitoring by security personnel, which is both expensive and ineffective due to the limitations of human attention and capacity. There is an unmet need for a scalable, automated solution capable of monitoring retail environments in real-time and detecting theft with a high level of accuracy. Addressing this issue can drastically reduce retail theft, improve resource allocation, and prevent revenue loss.

Statistics: Global shoplifting costs the retail industry approximately $50 billion annually.
Challenges: Manual detection methods are time-consuming and unreliable, often leading to missed theft attempts.

1.4. Stakeholders & Interests

Retail Owners: The primary stakeholders who would benefit from reduced theft, increased profit margins, and lowered operating costs.
Market Size: In 2023, the global retail industry was valued at over $26 trillion, with an estimated loss of $50 billion annually to theft. Even a small percentage reduction in theft could lead to significant savings.
Security Companies: These firms could integrate the system into their service offerings for enhanced security packages.
Market Size: The global security market is projected to grow to $133 billion by 2025, driven by demand for better surveillance technologies.
Consumers: Customers also benefit indirectly from reduced prices and increased product availability due to lower theft rates.

1.5. Objectives:

The primary objectives of the shoplifting-theft detection system are:

Real-time Detection: To monitor live video feeds from surveillance cameras and detect suspicious activity.

Behavioral Analysis: To identify potential shoplifting behavior based on predefined patterns, such as unusual movement or handling of merchandise.

Alert System: To immediately notify store management or security personnel when suspicious activity is detected.

Data Logging: To record incidents of detected theft for post-analysis and security audits.

1.6. Scope:

The system will be designed for implementation in medium to large retail stores and is aimed at reducing shoplifting incidents through real-time monitoring and detection. The scope includes:

Surveillance Footage Processing: Analyzing video streams from multiple cameras to detect suspicious activities.

Machine Learning Models: Training and deploying neural network models (CNN) to classify behaviors indicative of theft.

Limitations:

The system may not perform optimally in stores with poor-quality cameras or where shoplifting involves sophisticated techniques that deviate from usual behavior patterns.

The project will focus on indoor environments and will require customization for different store layouts and camera placements.

1.7. Assumptions

The design and implementation of the system are based on the following assumptions:

1. Retail stores have adequate quality surveillance cameras capable of providing clear video footage for analysis.
2. The system will have access to live video feeds from store cameras without network latency issues.
3. The store layout and camera placement will support proper visibility of key areas, such as shelves and checkout counters.
4. The behaviors indicative of shoplifting can be modeled using historical data and predefined patterns.
5. The system will have sufficient computational resources for real-time processing, either on-premises or through cloud services.
6. Store personnel will be trained to respond effectively to alerts generated by the system.
7. There will be compliance with privacy and data protection regulations in the region of deployment.

1.8. Risks

Poor video quality:
Low-resolution or blurry footage from store cameras could reduce the system's ability to accurately detect suspicious activities. This can be mitigated by applying video pre-processing techniques to enhance clarity and implementing fallback measures for ambiguous inputs.
Inadequate training data:
Insufficient or biased data can result in the system failing to recognize shoplifting behaviors. To address this, the team will gather diverse datasets from multiple sources and update the model continuously with real-world scenarios.
False positives and false negatives:
The system might generate incorrect alerts, leading to unnecessary interventions or missed theft attempts. Fine-tuning the machine learning model and introducing multi-step verification processes for high-confidence alerts will help reduce these errors.
Limited computational resources:
Real-time video processing might face performance issues due to hardware or software limitations. To mitigate this, algorithms will be optimized for efficiency, and cloud-based processing solutions will be explored for scalability.
Privacy and legal concerns:
The use of surveillance systems raises concerns about compliance with privacy laws and maintaining customer trust. The project will adhere to data protection regulations like GDPR, implement anonymization techniques, and ensure clear communication about data usage policies.
System integration issues:
Difficulties may arise when integrating the system with existing store infrastructure or security protocols. To overcome this, the team will work closely with store IT teams and conduct rigorous integration tests during pilot phases.
Misinterpretation of behavioral patterns:
Non-suspicious actions might be incorrectly flagged as shoplifting due to limited behavior modeling. This risk can be mitigated by employing advanced detection methods, such as context-aware analysis and fine-grained behavior classification.

1.9. Success Criteria

Table 2 Project Success Criteria

Feature	**Description**
User Registration and Authentication	The system must allow secure access for store managers and authorized personnel to manage alerts and analyze data
Core System Functionalities	Real-time monitoring and detection of suspicious activities are essential features. The system must immediately alert staff upon identifying potential theft.
Data Management	The system should log incidents of detected theft and store them securely for post-analysis and auditing purposes.

Reporting and Visualization	A visual dashboard summarizing incidents, system performance, and trends should be provided to give actionable insights to stakeholders
Search and Filter Options	The system should support searching and filtering incident logs by criteria such as time, location, and type of behavior detected.
Speed and Responsiveness	The system must analyze live video feeds with minimal latency to ensure timely alerts and responses.
Scalability	It should support multiple cameras and larger retail environments without compromising performance or detection accuracy.
Data Privacy and Security	All data must comply with privacy regulations, and customer identities should be anonymized where applicable.
Access Control	Role-based permissions should restrict access to sensitive data and features, ensuring that only authorized personnel can make critical changes.
User Interface Design	The interface should be intuitive and easy to use, even for non-technical users, facilitating efficient interaction with the system.
Cross-Platform Compatibility	The system must support deployment on both on-premises hardware and cloud platforms, offering flexibility to stakeholders.
Error Handling	Robust mechanisms must be in place to handle errors such as interrupted video

1.10. Tools, Libraries & Technologies

Table 3 Tools,Libraries & Technologies

	Tools	Version	Rationale
Tools, Libraries, And Technologies	OpenCV	4.5.2	OpenCV is a powerful computer vision library that provides various functionalities for image processing, object detection, and real-time video analysis. It allows us to implement image processing techniques easily.
	Anaconda	2023.03	Anaconda is a distribution of Python and R for scientific computing and data science. It simplifies package management and deployment, making it easier to manage libraries and dependencies.
	Jupyter Notebook	6.4.0	Jupyter Notebook allows for an interactive coding environment, enabling real-time code execution and visualization. It is ideal for testing algorithms and visualizing results.
	Libraries	**Version**	**Rationale**
	TensorFlow	2.8.0	TensorFlow is a popular deep learning library that provides flexible tools and libraries for building and training neural networks, including Convolutional Neural Networks (CNNs) suitable for image recognition tasks.

	Keras	2.8.0	Keras is an API built on top of TensorFlow that simplifies building and training deep learning models. It allows for

			rapid experimentation and prototyping of neural networks.
	NumPy	1.21.0	NumPy is essential for numerical computations and array manipulations in Python. It serves as the foundation for most scientific computing libraries in Python
	Matplotlib	3.5.1	Matplotlib is a plotting library that enables data visualization, which is crucial for analyzing and presenting results from the detection algorithms. .
	Technology	**Version**	**Rationale**

	Python Deep Learning Frameworks	3.9.12 Various	Python is the primary programming language for this project due to its simplicity and the vast ecosystem of libraries for machine learning and computer vision. Frameworks such as PyTorch and TensorFlow allow for efficient training and deployment of neural networks, making them essential for implementing the detection system.
	Video Surveillance Systems	N/A	Integrating with existing video surveillance systems will allow real-time monitoring and detection of shoplifting activities in retail environments.

1.11. Work Division

Table 4 Project Team Members Work Division

Sr. No	Roll Number	Name	Role Assignment & Work Division
1.	BSEF21M004	Afifa Masood	Model Training Specialist
2.	BSEF21M006	Hamza Arshad	Quality Assurance Engineer
3.	BSEF21M019	Urooj Fatima	User Interface Designer
4.	BSEF21M020	Rabail Waseem	Data Analyst
5.			

1.12. Conclusion

The proposed shoplifting detection system leverages the power of AI and machine learning to provide a scalable, efficient, and reliable solution for combating retail theft. By analyzing surveillance footage in real time and identifying suspicious behaviors, the system aims to reduce dependency on manual monitoring, minimize human error, and improve operational efficiency. This innovative approach not only addresses the significant financial impact of shoplifting but also enhances customer trust and satisfaction by ensuring a secure retail environment. With a strong focus on data privacy, system integration, and continuous improvement, the project is designed to meet the needs of modern retail spaces and set a new standard for theft prevention.

Chapter No 2

Literature Review

2.1 Literature Survey

The literature survey serves as the foundation for the proposed solution, offering insights into existing research and technological advancements in the field of shoplifting detection and retail security. This section critically examines the methodologies, technologies, and limitations of previous approaches, highlighting the challenges of relying solely on traditional surveillance methods. By analyzing various studies and commercial solutions, this review aims to establish the necessity and feasibility of an AI-powered shoplifting detection system that bridges existing gaps in accuracy, scalability, and automation.

2.2 Related Work:

Several existing studies and solutions have attempted to address the issue of shoplifting detection using computer vision and machine learning. Below are key approaches and their limitations:

1. Traditional Video Surveillance Systems:

Traditional systems rely on human operators to monitor CCTV footage in real-time. While effective in some cases, human attention is prone to fatigue, and surveillance personnel can easily miss crucial moments. These systems also tend to be reactive rather than proactive, as they detect theft only after it occurs.

- **Disadvantage**: High labor costs, limited scalability, and human error.

2. Rule-Based Computer Vision Systems:

Early attempts to automate theft detection involved rule-based systems, which relied on manually predefined rules such as the speed of movement or specific gestures (e.g., pocketing items). However, these systems lack the adaptability required to handle complex and varied human behaviors.

- ○ **Disadvantage**: Rigid, unable to learn from new data, high false positives due to non-flexible rules.

3. ### Object Detection Models:

   Several modern systems apply object detection methods to track objects being picked up or moved. For instance, **You Only Look Once (YOLO)** and **Faster R-CNN** have been used to detect objects in retail environments and flag suspicious movements. While these methods are highly effective in object detection, they fall short when tasked with recognizing complex behavioral patterns such as intentional theft.

   - ○ **Disadvantage**: Good for detecting objects but lacks the behavioral context needed for identifying theft patterns.

4. ### Behavioral Analysis Using Computer Vision:

   Recent research focuses on using computer vision for action and behavior recognition. Studies, such as the one conducted by **D. Singh et al. (2019)**, used convolutional neural networks (CNN) and Long Short-Term Memory (LSTM) networks to recognize suspicious activities in surveillance footage. These models show promise in detecting theft-like behavior but face challenges in terms of accuracy when dealing with crowded environments or occluded views.

   - ○ **Disadvantage**: Difficulties in generalizing to diverse environments, occlusions, and varied store layouts.

5. ### Market Solutions:

   Commercial systems such as **RetailNext** and **AxxonSoft** have introduced AI-powered video analytics that include theft detection as part of broader store analytics solutions. These systems are often expensive and are integrated with additional features such as heat maps and customer flow analytics, which may not be necessary for small to
   medium-sized stores.

   - ○ **Disadvantage**: High cost, complex integration, and not focused solely on theft detection.

2.3 Gap Analysis:

Although advancements have been made in theft detection through computer vision and machine learning, significant gaps remain in current research and commercial solutions:

1. #### Lack of Real-Time Behavioral Context:

   Existing object detection models are proficient at identifying items being moved or picked up, but they often fail to consider the broader context of the behavior, such as intent or repeated suspicious actions. This project aims to incorporate behavioral analysis alongside object tracking, which will enhance the ability to distinguish between innocent actions (like browsing) and theft.

2. High False Positives and False Negatives:

   Current solutions face challenges with high false-positive rates, flagging non-theft activities as theft. This project will focus on reducing these errors by training neural networks specifically on shoplifting behavior patterns, making the detection system more accurate and reliable.

1. Scalability and Cost:

   Many existing solutions in the market are expensive and difficult to scale for smaller retailers. This project will address this by offering a more cost-effective and focused solution that can be scaled to different store sizes without the need for additional costly infrastructure.

2. Occlusion and Crowded Environments:

   Current systems struggle with occlusion (when a person's view is blocked) or detecting theft in crowded environments. The use of CNN combined with behavior analysis models in this project aims to address this gap by incorporating multi-camera views and motion pattern analysis.

By filling these gaps, this project will provide a more adaptable, accurate, and cost-effective solution to the shoplifting problem in retail environments.

2.4 Summary

This chapter reviewed existing work in the domain of shoplifting detection, highlighting the limitations of traditional surveillance systems and the potential of AI-powered solutions. Current research and market implementations demonstrate significant progress but are hindered by challenges such as high costs, scalability issues, and behavioral ambiguity. The gap analysis identifies specific areas where the proposed project can make a meaningful contribution, including the development of an affordable, scalable, and privacy-compliant system for real-time theft detection in retail environments. This literature review establishes the necessity and feasibility of the project, providing a solid foundation for the design and implementation phases.

Chapter No 3

Software Requirements Specification

3.1 Requirements Analysis

The **Requirements Analysis** section outlines the key functionalities, user needs, and system behaviors necessary for the development of the **Shoplifting-Theft Detection System**. This analysis is essential to ensuring the system addresses the main objectives of detecting shoplifting incidents in real-time, accurately identifying suspicious behavior, and providing timely alerts to security personnel. This section also discusses the different user types that will interact with the system and the methodologies used to gather the requirements.

The purpose of this section is to document and analyze the necessary requirements that will guide the development process, ensuring that the system meets the stakeholders' expectations and operates effectively within retail environments.

3.2 User classes and characteristics[2]

Table 5 User Classes

User Class	User Characteristics
Store Security Personnel	Experienced in monitoring store activity and handling security incidents. May have limited technical skills. Requires real-time notifications and clear action prompts
Store Manager	Oversees overall store operations. Requires detailed reports and insights into security incidents and system performance. Medium technical proficiency.
System Administrator	Responsible for maintaining system performance, updating software, and ensuring data integrity. High technical expertise in both software and hardware management.
Developers	Software engineers responsible for system updates, bug fixes, and new feature implementation. High technical knowledge of AI, CV, and system architecture.

[2] Identify the various user classes that you anticipate will use this product and describe their pertinent characteristics using 2 column table.

3.3 Requirement Identifying Technique

This section describes the requirements identifying technique(s) which further help to derive functional requirements specification. The selection of the technique(s) will depend on the type of project. For instance,

- **Use case includes detailed use case descriptions & use case diagram)** is an effective technique for interactive end-user applications. Use case name and associated requirements must be presented here. However, use case descriptions must be given in fully dressed format.
- **Storyboarding**
  Visual representation of sequence of events, designs to represent the flow.

3.4 Functional Requirements

This section provides a detailed breakdown of the **functional requirements** for the **Shoplifting-Theft Detection System**, organized by feature. Each requirement defines what the system must accomplish, how it will respond to errors or invalid inputs, and any necessary dependencies or rationale behind the feature.

Functional Requirement-1[3]

Table 6 Function Requriements Specification for Requirement Search Courses

Identifier	FR-1
Title	Real-Time Surveillance Footage Analysis
Requirement	The system shall analyze surveillance camera footage in real-time to detect suspicious activities related to potential shoplifting incidents.
Source	Store Security Personnel, System Administrators
Rationale	To allow security personnel to monitor shoplifting activities continuously without manual intervention.
Business Rule (if required)	All camera footage must be processed within 2 seconds of capture to maintain real-time detection capabilities.
Dependencies	Dependency on surveillance camera feed and system's image processing module.
Priority	High

[3] Fill this template for multiple important functional requirements of your system,

Identifier	FR-2
Title	Detection of Suspicious Behavior
Requirement	The system shall use behavior analysis algorithms to detect suspicious actions such as loitering, concealing items, and abrupt movements..
Source	Store Managers, Security Experts
Rationale	To automate the detection of potential theft-related behaviors, reducing reliance on manual monitoring.
Business Rule (if required)	Behavior analysis must include a range of movements typical of shoplifting scenarios.
Dependencies	Requires integration with the object tracking system.
Priority	High

Identifier	FR-3
Title	Alert Generation for Detected Incidents
Requirement	The system shall generate real-time alerts for security personnel when suspicious behavior is detected.
Source	Store Security Personnel, System Administrators
Rationale	To ensure timely intervention by security staff in the event of suspicious activities.
Business Rule (if required)	Alerts must be generated within 1 second of detecting suspicious activity.

Dependencies	Relies on behavior detection algorithms (FR-2).
Priority	High

Identifier	FR-4
Title	Incident Recording and Reporting
Requirement	The system shall automatically log and store all detected incidents along with time, location, and camera information.
Source	Store Manager, System Administrators
Rationale	To provide a documented history of detected incidents for audit and review purposes.
Business Rule (if required)	All incident logs must be stored securely for at least 90 days.
Dependencies	Requires integration with the alert generation system (FR-3).
Priority	Medium

Identifier	FR-5
Title	System Administration Dashboard
Requirement	The system shall provide a user-friendly dashboard for administrators to manage cameras, configure alerts, and review recorded incidents.
Source	System Administrators
Rationale	To give administrators full control over system configurations and easy access to incident reports.

Business Rule (if required)	Dashboard access must be restricted to authorized personnel only.
Dependencies	Dependency on incident reporting and alert configuration.
Priority	Medium

3.5 Non-Functional Requirements

This section specifies nonfunctional requirements other than constraints, supporting requirements recorded in Functional Requirements section, and external interface requirements. These quality requirements should be specific, quantitative, and verifiable. The following are some examples of documenting guidelines.

1. Reliability

- Uptime: Minimum of 99.8% for continuous surveillance operation.
- Mean Time Between Failures (MTBF): At least 1,000 hours.
- Mean Time to Recover (MTTR): Under 5 minutes after a system failure.
- Backup: Automatic data backup every 24 hours to ensure data integrity.
- Fault Tolerance: System should remain operational in the event of a single-point failure, utilizing redundant servers.

- Real-time error logging and diagnostics to detect and notify any system malfunction.

2. Usability

- Maximum of 3 user interactions to review flagged footage or trigger alerts.
- Simple, intuitive design for security staff with minimal training.
- Real-time audio or visual alerts for immediate response to shoplifting incidents.
- System should support multiple languages for global user base.
- User-friendly dashboard with drag-and-drop functionality for camera management and zone configuration.
- Touchscreen compatibility for ease of use on tablets and other mobile devices.

3. Performance

- Detection and alert issuance within 2 seconds of suspicious activity.
- Ability to process video feeds from 10 to 20 HD cameras without performance loss.
- Continuous operation under peak data loads, ensuring high system performance.
- Minimum latency of 500ms between detection and alert generation.

- System should handle up to 50 concurrent users accessing the video feeds simultaneously.
- Scalability to support up to 100 cameras in large retail environments without a performance drop.

4. Security

- Data encryption using AES-256 for all footage and logs.
- Multi-factor authentication (MFA) for system access by authorized personnel only.
- Full audit trail to track all user activities, including login attempts and system changes.
- Regular penetration testing and security updates to mitigate vulnerabilities.
- Role-based access control to limit system functionalities based on user permissions.
- Automatic session timeout after 15 minutes of inactivity to prevent unauthorized access

3.6 External Interface Requirements

This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements. A complex system with multiple subcomponents should create a separate interface specification or system architecture specification. The interface documentation could incorporate material from other documents by reference. For instance, it could point to a hardware device manual that lists the error codes that the device could send to the software.

1. User Interfaces Requirements

- **GUI Standards**: Follow established security software GUI standards for ease of use and consistency across systems.
- **Fonts, Icons, and Button Labels**: Use clear, readable fonts (e.g., sans-serif, 14pt), easily recognizable icons, and intuitive button labels for efficient navigation.
- **Color Schemes**: Implement color-coded alerts (e.g., red for critical alerts, yellow for warnings) and maintain a neutral background for user focus.
- **Field Tabbing Sequences**: Logical field tabbing for fast interaction (e.g., moving between camera feeds or alert logs).
- **Common Controls**: Utilize dropdown menus, toggle buttons, and sliders for user inputs, following common security system controls.
- **Screen Layout and Resolution**: Ensure adaptable screen layouts that automatically adjust for different screen sizes and resolutions (1080p or higher).
- **Standard Buttons and Functions**: Include always-visible buttons for actions such as “Help,” “Alerts,” “Playback,” and “Live Feed.”
- **Shortcut Keys**: Offer configurable keyboard shortcuts for frequently used actions like pausing a live feed, starting playback, or marking suspicious activity.
- **Message Display Conventions**: Use clear, concise alert messages, showing the date, time, and a description of the suspicious activity.
- **Localization**: Provide multilingual support for international deployment, with UI layout standards to accommodate text expansion in various languages.

- **Accessibility**: Accommodate visually impaired users with adjustable font sizes, screen reader compatibility, and high-contrast mode.

2. Software Interfaces

- **Video Management System (VMS)**: Integrate with existing VMS platforms to access live and recorded video feeds.
- **Database Systems**: Interface with an internal or external database (e.g., MySQL, PostgreSQL) for storing event logs, user activities, and alert data.
- **Operating Systems**: Compatible with Windows and Linux operating systems for maximum flexibility in deployment.
- **AI/ML Libraries**: Utilize AI/ML libraries for enhanced theft detection, such as OpenCV for real-time video processing and TensorFlow for behavior analysis.
- **External Tools**: Integrate with third-party analytics tools for data reporting, such as Power BI or Grafana, for generating insights on theft incidents.
- **Cloud Storage**: Offer integration with cloud storage solutions (e.g., AWS S3, Google Cloud) for offsite backup and data archiving.
- **APIs**: Provide a REST API to allow third-party software to query event logs, video feeds, and system statuses

3. Hardware Interfaces

- **Cameras**: Support for IP and analog cameras, including 1080p HD or higher, with connectivity via Ethernet or Wi-Fi. Support for ONVIF-compliant devices.
- **Sensors**: Connect with external sensors (e.g., RFID, motion detectors) to trigger real-time alerts when anomalous behavior is detected.
- **Servers**: Integration with server hardware for video processing, storage, and system operations, capable of handling high data throughput.
- **Edge Devices**: Support for edge computing devices for real-time video analysis on-site, reducing latency.
- **Input Devices**: Interface with touchscreens, keyboards, and mice for manual control of the system in command centers.
- **Power Systems**: Integration with uninterrupted power supply (UPS) hardware to ensure system operation during power outages.

4. Communications Interfaces

- **Email and SMS**: Enable real-time alerts via email (SMTP) or SMS (via APIs like Twilio) to notify security personnel of detected incidents.
- **Web Browser**: Provide web-based access to the system’s control panel and video feeds using secure HTTPS communication.
- **Network Protocols**: Use RTSP (Real-Time Streaming Protocol) for streaming video feeds and FTP/SFTP for transferring log files and backups.
- **Webhooks**: Support for webhooks to integrate with other security management systems for automated incident response (e.g., sending alerts to a central management dashboard).

3.7 Use case Analysis

1. Use Case #1 (use case name and unique identifier – e.g. UC1)

TO DO: Provide a specification for each use case diagram, add the table caption for each use case description using "References > Insert Caption" option

Table 7 Use case Description for the Use case Recommend Course

UC Identifier	UC1
UC Name	Use case name following convention Verb Noun e.g. Recommend Course
Requirements Traceability	Mention all requirements traced to this use case by referencing with their requirements number mentioned in functional requirements
Purpose	What is the basic objective of the use-case. What is it trying to achieve?
Priority	What is the priority. Low, Medium, High. Importance of this use case being completed and functioning properly when system is deployed.
Preconditions	Any condition that must be satisfied before the use case begins
Post conditions	The conditions that will be satisfied after the use case successfully completes
Actors	Actors (human, system, devices, etc.) that trigger the use case to execute or provide input to the use case
Extends	If this is an extension use case, identify which use case(s) it extends
Main Success Scenario	flow of events normally executed in the use-case (in terms of actor action and system response). It should start describing the events assuming that precondition is already true. Description should end leading to the post conditions specified.
Alternate Flows	a secondary flow of events due to infrequent conditions
Exceptions	Exceptions that may happen during the execution of the use case
Includes	Use case ID of the included function

2. Use Case #2 (Alert Staff U2)

3. UC Identifier	UC2
Requirements Traceability	- REQ-004: The system must trigger alerts based on predefined suspicious behavior patterns. - REQ-005: Alerts should be sent to the appropriate staff members in real-time.
Purpose	The purpose of this use case is to notify security personnel immediately when suspicious behavior is detected, enabling prompt intervention to prevent theft.
Priority	High
Preconditions	- The surveillance system must be actively monitoring the retail environment.

	- Suspicious behavior must be detected by the system.
Post conditions	- Security personnel receive alerts detailing the location and nature of the suspicious behavior. - A log of the alert is created for future reference.
Actors	- Primary: Security Staff - Secondary: Surveillance System (Camera), Alert System
Extends	- UC1: Propose Preventive Actions (if necessary, based on the nature of the alert)
Main Success Scenario	1. The system identifies suspicious behavior from the video feed. 2. An alert is triggered based on the behavior detected. 3. The system sends a real-time notification to security personnel, including location and behavior details. 4. Security personnel review the alert and take appropriate action.
Alternate Flows	- If the alert system is temporarily unavailable, the system logs the incident for later review.
Exceptions	- If the alert fails to send due to network issues, an error message is logged, and the incident is flagged for follow-up.
Includes	- UC1: Propose Preventive Actions

Use Case #3 (Review and Investigate Incident Logs U3)

UC Identifier	UC3
Requirements Traceability	REQ-006: The system must allow security staff to review incident logs. REQ-007: Provide video playback of flagged incidents.
Purpose	Enable security personnel to review logs of detected suspicious activities and investigate incidents using video playback.
Priority	Medium
Preconditions	Incidents have been flagged, and corresponding logs are available in the system.
Post conditions	Security personnel can investigate incidents using stored video feeds and action logs.
Actors	- Primary: Security Staff - Secondary: Surveillance System , Log System
Extends	- UC2: Alert Staff (if suspicious behavior is detected)
Main Success Scenario	1. Security staff accesses the incident log. 2. The system presents a list of flagged incidents with corresponding video clips. 3.The user selects an incident for review. 4. The system plays back the video and provides relevant information about the incident.
Alternate Flows	- If video playback is unavailable, the system provides alternative views (e.g., screenshots or motion summaries).
Exceptions	If the log data is corrupted or missing, an error is reported, and the system prompts for further investigation.
Includes	- None specified. This use case is focused on reviewing logs and may not need to include other use cases.

4. Use Case #4 (System Calibration and Maintenance U4)

UC Identifier	UC4
Requirements Traceability	REQ-008: The system must allow users to calibrate cameras and perform maintenance tasks.
Purpose	: Ensure cameras and detection algorithms are properly calibrated to maintain accurate theft detection.
Priority	Medium
Preconditions	The system must be operational, and cameras should be accessible for calibration.
Post conditions	Cameras are calibrated, and system health is confirmed.
Actors	- Primary: System Administrator - Secondary: Surveillance System
Extends	- None specified. This use case is primarily focused on system maintenance and calibration and may not need to include other use cases.
Main Success Scenario	1. The administrator initiates a system calibration process. 2. The system displays a calibration interface for camera adjustments. 3.The administrator adjusts camera settings as needed. 4. The system confirms calibration success and logs the maintenance action.
Alternate Flows	- If a camera is unreachable, the system prompts the administrator to check the hardware or network connection.
Exceptions	If calibration fails due to hardware malfunction, the system reports an error.
Includes	- None specified. This use case is primarily focused on system maintenance and calibration and may not need to include other use cases.

5. Use Case #5 (User Authentication and Access Control U5)

Table 7 Use case Description for the Use case Recommend Course

UC Identifier	UC5
Requirements Traceability	REQ-009: The system must enforce user authentication for access to critical functions REQ-010: Access control must restrict certain functions based on user roles (e.g., security staff vs. admin)..
Purpose	Ensure that only authorized personnel can access or modify system settings and view sensitive data.
Priority	High
Preconditions	The user must be registered in the system with an assigned role.

Post conditions	Only authorized personnel can perform certain tasks, such as reviewing logs, modifying system settings, or calibrating cameras.
Actors	- Primary: Security Staff, System Administrator - Secondary: Authentication System
Extends	- UC2: Alert Staff (if suspicious behavior is detected)
Main Success Scenario	1. The user attempts to log in to the system. 2. The system verifies the user's credentials. 3.The user is granted access to system functions based on their role. .
Alternate Flows	- If the user enters incorrect credentials, the system prompts for re-entry or account recovery.
Exceptions	If the authentication server is down, access to the system is restricted to previously logged-in sessions.
Includes	- None specified. This use case can be seen as a foundational use case for other operations that require authentication but does not necessarily include other use cases directly.

6. Use Case Diagram[4]

[4] System is not the actor. Use Computer Aided Software Engineering Tool (CASE tool) to design use case diagram. Diagram should be readable. There is a single use case diagram for entire system

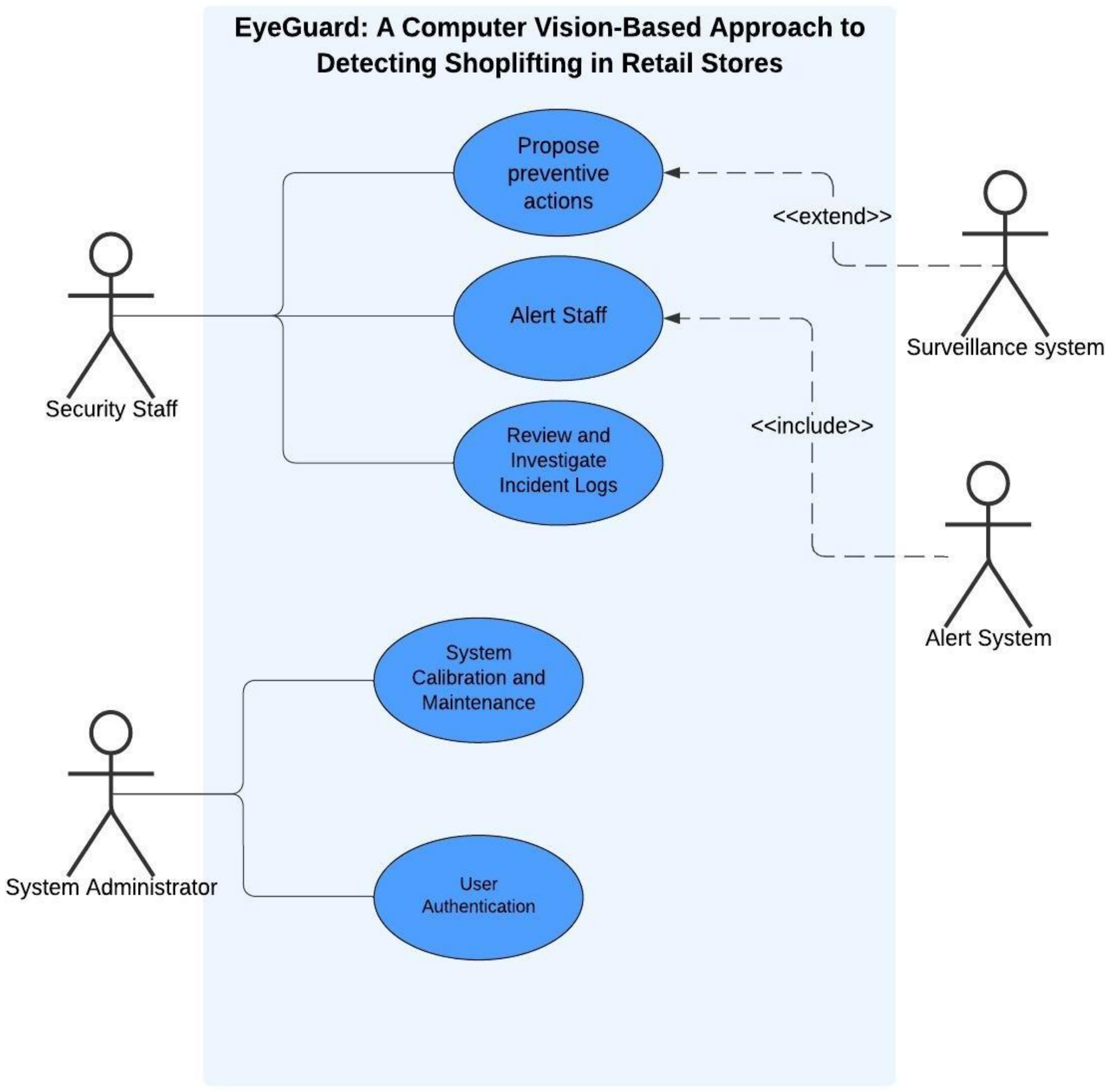

EyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores
Propose preventive actions
<<extend>>
Surveillance system
Security Staff
Alert Staff
Review and Investigate Incident Logs
<<include>>
Alert System
System Calibration and Maintenance
System Administrator
User Authentication

Figure 1 Use Case Diagram

3.8 Storyboards

Storyboard 1: Detecting Suspicious Behavior

Step 1: Surveillance Activation

- *User Action:*
  - Security staff logs into the surveillance system and activates monitoring mode.
- *System Response:*
  - The system begins streaming video in real-time from all connected cameras.
  - A user-friendly interface displays live feeds with camera locations labeled.

Step 2: Behavior Analysis

- *User Action:*
    - Security staff observes the surveillance feed on the display, watching for any unusual activities.
- *System Response:*
    - The system actively analyzes the live feed using algorithms to identify signs of suspicious behavior.
    - Detected behaviors may include:
        - Loitering in restricted areas.
        - Frequent glancing at exits or other suspicious actions.
    - An overlay on the video feed highlights areas of interest, helping staff focus their attention.

Step 3: Alert Generation

- *User Action:*
    - Security staff notices an alert indicator flashing on the screen.
- *System Response:*
    - The system generates a real-time alert, providing:
        - **Details of Suspicious Activity**: Description of the observed behavior, time of detection, and location.
        - **Visual Markers**: A highlighted area on the surveillance feed where the behavior was detected.

      - **Action Prompt**: Options for the staff to review the alert further or take immediate action (e.g., notify other security personnel, switch to the relevant camera).

LOGIN 1
DETECTING SUSPEILLOUSE ANALTIOS
DETECTING SUSPICIOUS ANALYTIOS DETECTED
ACTIVATION
ALERTS
ER 3
ALERT
ALERT 36

Storyboard 2: Proposing Preventive Actions

Step 1: Alert Notification

- *User Action:*
  - Security staff receives a real-time alert about suspicious behavior.
- *System Response:*
  - The alert appears on the staff's mobile device, displaying:
    - **Location**: Exact GPS coordinates or address.
    - **Behavior Description**: Brief details of the suspicious activity (e.g., loitering, unusual movements).

Step 2: Reviewing Recommendations

- *User Action:*
  - Security staff taps on the alert notification to open it.
- *System Response:*
  - The system displays a detailed view of the alert with:
    - *Proposed Preventive Actions:*
      - Increased monitoring of the area.
      - Direct intervention to engage with the suspect.
      - Request for backup from other security personnel.
      - Notification to law enforcement if necessary.

- *Additional Information:*
- Time of the alert.
      - Any previous incidents at the location.

Step 3: Taking Action

- *User Action:*
  - Security staff decides to approach the suspect for direct intervention.
- *System Response:*
  - The system logs the action taken, including:
    - **Action Details**: Description of the intervention (e.g., approached the suspect, asked for identification).
    - **Timestamp**: When the action was taken.
    - **Alert Status Update**: Changes the alert status to "Under Investigation" or "Resolved" based on the outcome.
    - **Follow-Up Recommendations**: Suggestions for further monitoring or incident reporting if necessary.

Storyboard 4: Reviewing Incident Reports

Step 1: Accessing Reports

- *User Action:*
  - Security staff navigates to the incident reports section in the application, typically found in the main menu.
- *System Response:*
  - The system displays a comprehensive list of past incidents, including:
    - **Timestamps**: Date and time of each incident.
    - **Descriptions**: Brief summaries of the incidents.
  - Each entry may have an associated severity level or status indicator for quick reference.

Step 2: Selecting an Incident

- *User Action:*

    - Security staff selects a specific incident from the list to review further.

- *System Response:*
    - The system displays detailed information about the chosen incident, including:
        - **Incident Overview**: Detailed description and context of the event.
        - **Video Footage**: Relevant surveillance video clips related to the incident.
        - **Analysis Data**: Any behavioral analysis or system-generated insights regarding the incident.
        - **Follow-Up Actions**: Summary of actions taken during and after the incident.

Step 3: Evaluating Effectiveness

- *User Action:*
    - Security staff evaluates the response taken during the incident, considering factors such as:
        - Timeliness of response.
        - Effectiveness of actions taken.
        - Any challenges faced during the incident.
    - Staff notes any improvements needed for future responses.
- *System Response:*
    - The system allows staff to add comments or suggestions for future incidents through a user-friendly interface.
    - These comments are logged and saved for review by management, facilitating continuous improvement and training.

INCIDENT REPORTS
INCIDENT REPORTS
Summerces, rie inprecvene,
Comments and ungretavions
suggents for furure review.

3.9 Summary

The processes of requirements identification and use case analysis are critical for the successful development of the Shoplifting-Theft Detection System. They ensure that the system effectively addresses its core objectives, including the real-time detection of shoplifting incidents, accurate recognition of suspicious behavior, and prompt alerts for security staff. Through rigorous requirements engineering, key functionalities and user needs are meticulously documented, encompassing aspects like live surveillance, alert notifications, and detailed incident reporting. By engaging with various stakeholders via interviews and surveys, the project team gathers essential insights that inform the system's design, ultimately enhancing its capability to operate seamlessly within retail environments. This comprehensive analysis not only aligns the system with stakeholder expectations but also strengthens its role in enhancing security measures in retail settings.

Chapter No 4

Software Design

Specification

4.1 System Design

The **Shoplifting-Theft Detection System** operates as an autonomous surveillance solution designed to detect suspicious activities (specifically theft) in retail environments. It integrates computer vision (CV) algorithms and machine learning models to analyze real-time surveillance footage captured by CCTV cameras. The system interacts with the existing retail infrastructure, including cameras, video management systems (VMS), and security personnel dashboards. The goal is to identify potential shoplifting incidents and alert security staff promptly.

4.2 Design Considerations

Assumptions and Dependencies:

- **High-Quality Video Feed**: It is assumed that the video footage captured by cameras is clear enough for the system to accurately analyze and detect suspicious behavior.
- **Internet Connectivity**: The system may require internet connectivity to fetch updates for the machine learning models, which are continually improved.
- **Existing Hardware Infrastructure**: The system assumes that the store's existing camera infrastructure is adequate for the task and may require minimal upgrades.
- **Security Personnel Availability**: The system assumes that security personnel will be available to act on the alerts generated by the system in real-time.

Limitations:

- **Performance of Detection Models**: The system's accuracy is dependent on the quality of the training data used to train the machine learning models. Misclassifications can occur, especially with overlapping object behavior or similar patterns of movement.
- **Processing Latency**: Real-time analysis of video feeds requires significant processing power, and latency could become an issue with large-scale surveillance setups.
- **Limited Detection Range**: The system may not be able to detect theft activities outside the camera's field of view, requiring additional cameras for complete coverage.
- **Compatibility with Legacy Systems**: Integrating the software with older surveillance equipment may present compatibility issues, especially if the cameras or video management systems do not support modern streaming protocols.

Risks:

- **False Positives/Negatives**: The risk of detecting false alarms (false positives) or failing to detect real thefts (false negatives) is a major concern. This can be mitigated by regularly updating the machine learning models with new data and fine-tuning them for better accuracy.
- **Scalability Issues**: As the number of cameras increases, the system may face challenges in processing all video streams in real-time, leading to potential performance bottlenecks. This can be addressed by optimizing the architecture to allow for horizontal scaling.
- **Privacy Concerns**: There is a risk of privacy violations, as the system processes video footage of customers. Measures must be taken to ensure that personal data is anonymized and handled according to applicable data protection regulations (e.g., GDPR).
- **Integration with Legacy Systems:** The system may need to interface with existing, older surveillance infrastructure or software that might not be compatible with newer technologies (e.g., machine learning models or cloud services). This could lead to

integration challenges and potential data loss or corruption. To mitigate this, a robust interface layer should be designed to ensure seamless communication between the new system and legacy platforms, and gradual migration or upgrades to the legacy systems may be required.

4.3 Requirements Traceability Matrix [5]

Shows how each design element traces back to a specific requirement in the Software Requirements Specification (SRS) for the **Shoplifting-Theft Detection System**, ensuring that all requirements are accounted for in the design. The table documents the requirements traceability, where the **Requirement ID** is a unique identifier assigned to each requirement, the **Requirement Description** provides a brief summary of the requirement, and the **Design Specification** lists the components, classes, or algorithms where the requirement is addressed. This ensures that each design element aligns with the corresponding functional requirement defined in the SRS.. [6]

Table 8 Requirements Traceability Matrix

[5] Please don't forget to add the caption of table (Use References -> Insert Caption option in MS Word)

[6] Make sure that the list is complete and consistent with requirements mentioend in SRS

Requirement ID	**Requirement Description**	**Design Specification**
R1	The system must detect suspicious activities such as unusual behavior or concealed items in real-time from CCTV footage.	Component “Suspicious Activity Detection Module”
R2	The system should classify the detected activities into categories like shoplifting, suspicious behavior, etc.	Component “Activity Classification (Using Machine Learning Models)”
R3	The system must trigger an alert when a potential theft is detected.	Component “Alert Notification System”
R4	The system shall automatically log and store all detected incidents along with time, location, and camera information.	Component: Incident Logging and Storage Database: Incident Log Storage
R5	The system should be capable of processing multiple CCTV streams simultaneously for large retail environments.	Component “Video Stream Processing and Multi-Stream Support”
R6	Machine learning models like Faster R-CNN, YOLO, and ConvLSTM should be utilized for detecting and recognizing theft events.	Component “Machine Learning Model Integration”
R7	The system shall provide a user-friendly dashboard for administrators to manage cameras, configure alerts, and review recorded incidents.	Component: “System Administration Dashboard”

4.4 Design Models [7]

Provide the descriptions of the following models used to describe the system design. Also ensure visibility of all diagrams.

The applicable models for the project using object-oriented development approach may include:

1. Design Class Diagram (DCD)[8]

This class diagram represents the core structure of the **EyeGuard** system, a shoplifting detection system leveraging computer vision. The design consists of six key classes: **User**, **Admin**, **Shoplifter Detection System**, **BehaviorDetection**, **Camera**, and **Incident**, each with specific roles and relationships. The **User** class provides fundamental properties like user ID, name, and role and is extended by the **Admin** class, which inherits user features and adds administrative privileges (e.g., managing system settings, viewing incident logs). This relationship is shown via an **inheritance arrow** connecting Admin to User.

The **Shoplifter Detection System** acts as the central control entity, managing incidents and cameras. It aggregates with the **BehaviorDetection** class, which provides functionalities like analyzing behavior and tracking movement, indicating a loosely coupled relationship represented with an **aggregation arrow**. The **Shoplifter Detection System** also has a **composition relationship** with the **Incident** class, signifying a strong lifecycle dependency, as incidents cannot exist independently of the system.

The **Camera** class is associated with the **Shoplifter Detection System** to represent the system's ability to monitor and record activities across multiple cameras. The **one-to-many multiplicity** between Shoplifter Detection System and Camera (denoted by 1 on the system side and * on the camera side) reflects that one detection system can handle multiple cameras. Similarly, the **many-to-one multiplicity** between Incident and Camera signifies that multiple incidents can originate from a single camera. This detailed structure ensures modularity and scalability while clearly defining class roles and relationships.

The **User-Admin relationship** is a **One-to-Many** association, where a single Admin manages multiple Users, overseeing their roles and activities, while each User is connected to only one Admin for approval or monitoring. The **Shoplifter Detection System-Behavior Detectors relationship** can be either **One-to-One** or **One-to-Many**, depending on the system's architecture. A single detection system may either rely on one behavior detection module (One-to-One) or utilize multiple modules to analyze customer behavior across different zones or scenarios simultaneously (One-to-Many). Lastly, the **Shoplifter Detection System-Incidents relationship** is **One-to-Many**, as a single detection system can handle and log multiple incidents over time, with each incident uniquely tied to the context of the specific system where it occurred.

[7] Please don't forget to add the caption of each figure (Use References -> Insert Caption option in MS Word)

[8] Insert caption of each diagram.

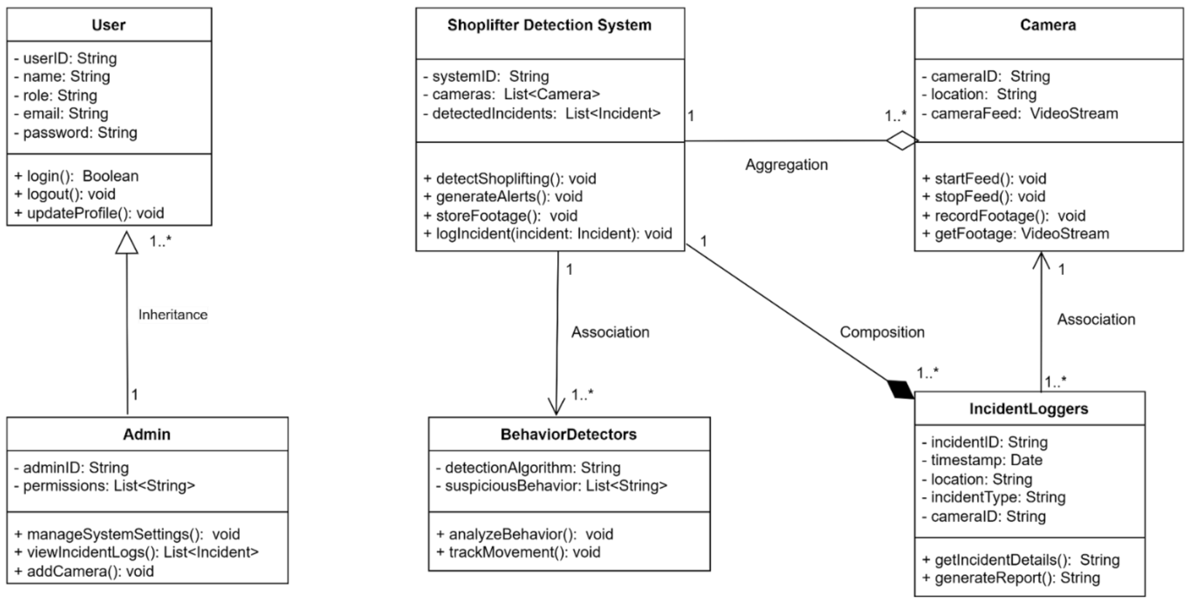


Figure 2 Class Diagram

2. Interaction Diagram (Either sequence or collaboration)

This sequence diagram illustrates the operational flow of the **EyeGuard** shoplifting detection system. It captures interactions between key components: **Cameras**, **Behavior Detectors**, **Incident Loggers**, **Alert Generators**, and **Users**, showing how they collaborate to detect and handle shoplifting incidents.

1. **Camera Feed**: The process begins with cameras continuously sending video feeds to the **Behavior Detectors** for analysis.
2. **AI/ML Analysis Loop**: The **Behavior Detectors** analyze the video using AI/ML algorithms to identify any suspicious behaviors. This is represented in a loop, where real-time analysis occurs repeatedly.
3. **Decision Branch**: If no suspicious behavior is detected, the system continues monitoring. If suspicious behavior is identified, the system proceeds to log the incident.
4. **Incident Logging and Alerts**:
   - The detected incident is logged in the **Incident Logger** with all relevant details.
   - Simultaneously, the **Alert Generator** triggers a notification, sending it to relevant **Users** (e.g., security personnel or store managers).
5. **User Interaction**: Users can view the incident logs and request detailed reports for further action or evidence.

This diagram also includes **alt** sections to represent alternative flows (e.g., suspicious vs. normal behavior) and effectively demonstrates the system's logic and real-time operations, ensuring responsiveness and accuracy in detecting thefts.

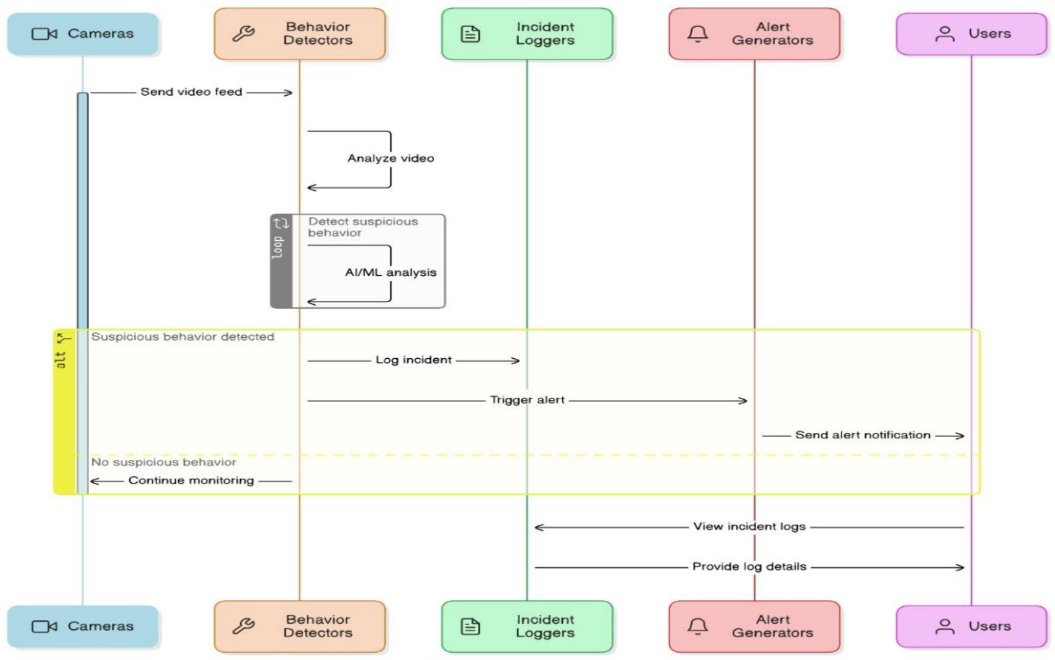


Figure 3 Interaction Diagram

3. State Transition Diagram (for the projects which include event handling and backend processes)

This state diagram represents the operational flow of the **EyeGuard: A Computer Vision-Based Approach to Detecting Shoplifting in Retail Stores**. The system transitions through various states during its lifecycle, starting with **System Initialization**, followed by **Monitoring**, and handling exceptional scenarios like **Maintenance** or **System Errors**.

System Initialization: The system begins by loading the necessary models and configuring parameters for its operation. Once all configurations are complete, the system transitions to the **Ready** state, where it is prepared to monitor activities.

Monitoring Process:

The system remains in the **Idle** state until it detects an activity. Upon detection, it transitions to **ActivityDetected**, where behaviors are analyzed using AI/ML algorithms.

If no threat is identified, the system returns to **Idle** or continues monitoring. However, if a **Potential Threat** is detected, the system raises an alert (**AlertRaised**) and awaits a response (**AwaitResponse**) from the user or an external process.

After resolving the alert, the system resumes monitoring or transitions to the **Maintenance** state for upkeep and debugging.

Error Handling and Maintenance: Any critical issues encountered during operations transition the system to a **SystemError** state. Maintenance ensures the system's stability and readiness for subsequent operations.

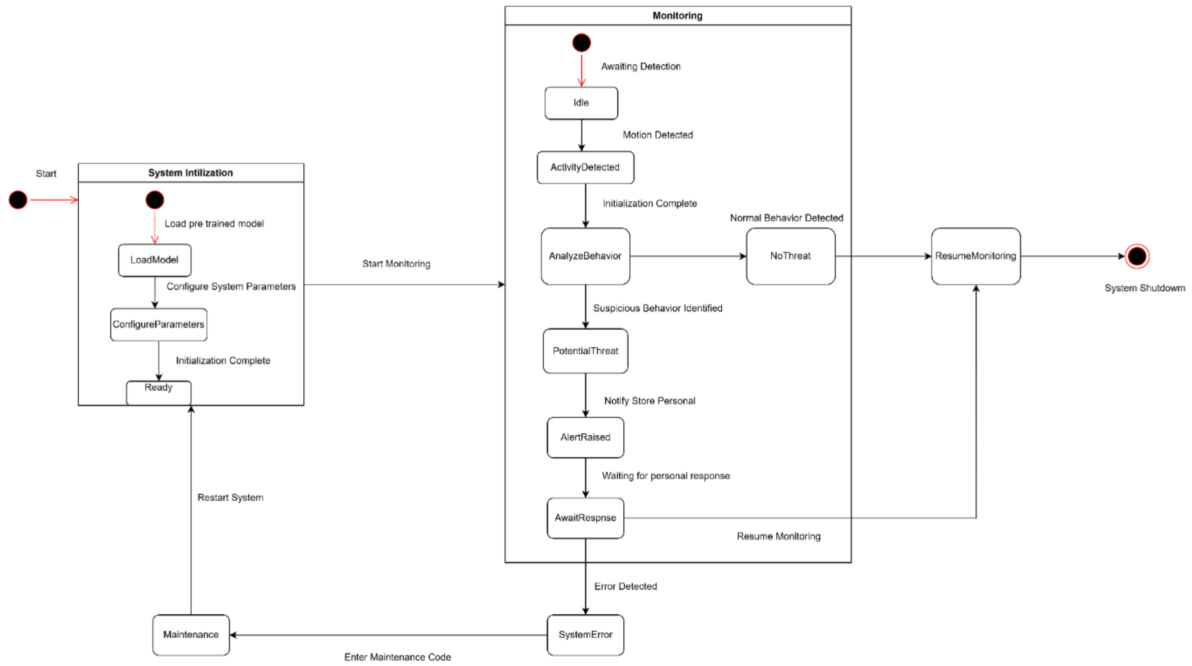


Figure 4 State Transition Diagram

4.5 Architectural Design [9]

Architecture Style: Layered

1. **Presentation Layer (User Interaction)**
   - **Modules/Components:**
     - Users Component: Provides an interface for personnel or stakeholders to interact with the system (e.g., view logs, receive notifications).
     - User and Admin Classes: Handle user-specific functionalities like logging in, managing permissions, and updating profiles.
2. **Business Logic Layer (Processing and Decision-Making)**
   - **Modules/Components:**
     - Behavior Detectors Component: Processes video feeds using AI/ML algorithms to detect suspicious activities.
     - Alert Generators Component: Analyzes detection results and triggers alerts.
     - BehaviorDetection Class: Implements detection algorithms and behavior analysis functions.
     - ShoplifterDetectionSystem Class: Integrates cameras, incidents, and behavior detection as the main system logic.
3. **Data Layer (Storage and Logging)**
   - **Modules/Components:**

[9] Please don't forget to add the caption of each figure (Use References -> Insert Caption option in MS Word)

- Incident Loggers Component: Manages storage and retrieval of detected incidents for record-keeping.
- Incident Class: Represents each logged incident with attributes like ID, timestamp, and type.
- Camera Class: Manages video feeds and recorded footage storage.

4. **Infrastructure Layer (Hardware/Systems Integration)**
   - **Modules/Components:**
     - Cameras Component: Interfaces with physical cameras to provide video feeds for the system.
     - Camera Functions (e.g., startFeed, recordFootage): Directly manage interactions with the camera hardware.

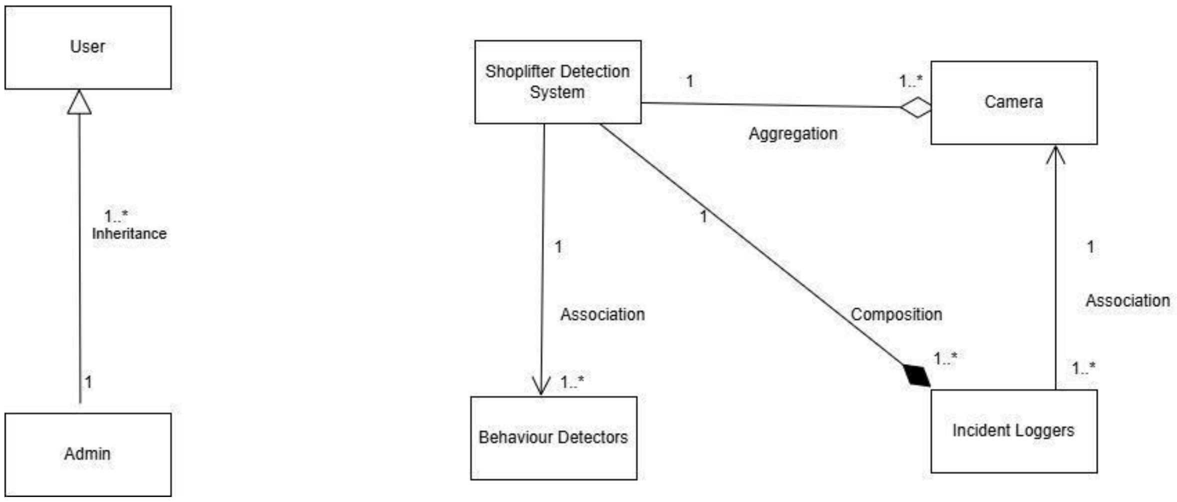


Figure 5 UML Class Relationship Diagram

1. UML Component diagram

This is a **Component Diagram** for the **EyeGuard - Shoplifting Detection System**, illustrating how the system's components interact and exchange data or functionalities. Here's an explanation of each component and its relationships:

Components:

1. **Cameras Component (Pink)**:
   - **Function**: Responsible for video surveillance, capturing live feeds from the monitored area.
   - **Output**: Sends real-time video feeds (CameraFeed) to the **Behavior Detectors Component**.
2. **Behavior Detectors Component (Yellow)**:
   - **Function**: Uses AI/ML algorithms to analyze the video feed and detect suspicious behaviors.
   - **Input**: Receives real-time video feeds (CameraFeed) from the Cameras Component.
   - **Output**: Generates detection results (DetectionResults) and logs suspicious activities (IncidentLog) to the **Incident Loggers Component**. It also triggers alerts to the **Alert Generators Component**.
3. **Incident Loggers Component (Blue)**:
   - **Function**: Maintains logs of detected incidents for record-keeping.
   - **Input**: Receives IncidentLog from the Behavior Detectors Component.
   - **Output**: Provides access to logs for users (ViewLogs) through the **Users Component**.
4. **Alert Generators Component (Orange)**:
   - **Function**: Generates alerts based on detected suspicious behavior.
   - **Input**: Receives detection results (DetectionResults) from the Behavior Detectors Component.
   - **Output**: Notifies users about incidents (AlertNotification) through the **Users Component**.
5. **Users Component (Green)**:
   - **Function**: Acts as the interface for personnel or stakeholders to interact with the system.
   - **Input**:
     - Receives ViewLogs from the Incident Loggers Component.
     - Receives AlertNotification from the Alert Generators Component.

Relationships:

1. **Cameras → Behavior Detectors**:
   - The Cameras Component sends real-time video feeds for analysis.
2. **Behavior Detectors → Incident Loggers**:
   - Logs detected suspicious activities for further review.
3. **Behavior Detectors → Alert Generators**:
   - Provides detection results to trigger alerts.
4. **Incident Loggers → Users**:
   - Allows users to access logs of detected incidents.
5. **Alert Generators → Users**:
   - Sends notifications to users about detected suspicious activities.

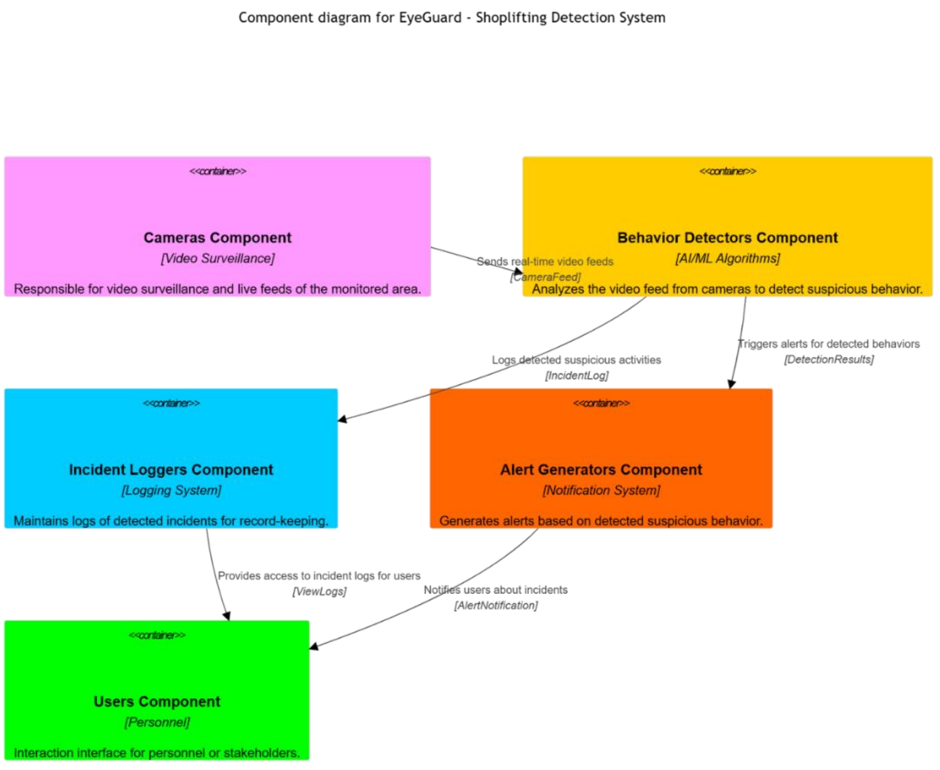

Component diagram for EyeGuard - Shoplifting Detection System
<<container>>
Cameras Component
[Video Surveillance]
Responsible for video surveillance and live feeds of the monitored area.
<<container>>
Behavior Detectors Component
[AI/ML Algorithms]
Analyzes the video feed from cameras to detect suspicious behavior.
Sends real-time video feeds
[CameraFeed]
Logs detected suspicious activities
[IncidentLog]
Triggers alerts for detected behaviors
[DetectionResults]
<<container>>
Incident Loggers Component
[Logging System]
Maintains logs of detected incidents for record-keeping.
<<container>>
Alert Generators Component
[Notification System]
Generates alerts based on detected suspicious behavior.
Provides access to incident logs for users
[ViewLogs]
Notifies users about incidents
[AlertNotification]
<<container>>
Users Component
[Personnel]
Interaction interface for personnel or stakeholders.

Figure 6 UML Component Diagram

4.6 Data Design

The information domain of the system is transformed into well-structured data entities and relationships to ensure efficient storage, processing, and organization. Major entities like Alerts, Cameras, Incidents, Incident Logs, and Users are represented as objects in the database, each with unique identifiers and attributes defining their properties and roles. For instance, the Camera entity stores details such as its location and operational status, while Incident entities capture information about suspicious activities, including timestamps, severity, and the associated camera. These entities are interlinked to maintain relational integrity, such as referencing Camera IDs in Incidents. Functions like detectBehavior() and logIncident() process incoming data—such as video frames or behavior analysis results—while generateAlert() triggers alerts and notifies users in real-time. The database design ensures that large datasets, like logs of incidents, can be efficiently queried and retrieved using filters such as date or camera ID, enabling seamless analysis and reporting.

1. Data Dictionary

- **Alphabetical List of System Entities or Major Data with Types and Descriptions:**

Table 9 9 Major Data with Types and Description Table

Terminology	**Description**
Alert	**Type**: Object. Represents alerts generated by the system in response to events or anomalies.
Alert.timestamp	**Type**: DateTime. Stores the date and time when the alert was generated.
Alert.message	**Type**: String. Descriptive message providing details of the alert.
Camera	**Type**: Object. Represents surveillance cameras integrated into the system.
Camera.id	**Type**: String. Unique identifier for each camera.
Camera.location	**Type**: String. Describes the physical location of the camera.
Camera.status	**Type**: Boolean. Indicates whether the camera is active or inactive.
Incident	**Type**: Object. Represents incidents detected by the system.
Incident.id	**Type**: String. Unique identifier for the incident.

Incident.description	**Type**: String. Detailed explanation of the incident.
Incident.timestamp	**Type**: DateTime. Records the time the incident occurred.
Incident.cameraId	**Type**: String. References the camera that captured the incident.
Incident.severity	**Type**: String. Categorizes the severity level (e.g., low, medium, high).
IncidentLog	**Type**: Object. Stores a collection of incidents detected by the system.
IncidentLog.totalIncidents	**Type**: Integer. Counts the total number of incidents logged.
User	**Type**: Object. Represents users interacting with the system, including admins and operators.
User.username	**Type**: String. Unique name used for authentication.
User.password	**Type**: String. Encrypted password for authentication.
User.role	**Type**: String. Defines user roles (e.g., Admin, Operator).

Structured Approach: Functions and Function Parameters:

Table 10 Function Descriptions Table

Function Name	Description	Parameters (Data Type and Role)
detectBehavior()	Analyzes video frames to identify suspicious behavior based on predefined detection criteria.	- frameData (array): Contains video frame data for analysis, typically processed as image arrays or tensors. - confidenceThreshold (float): The minimum confidence level required to flag behavior as suspicious. For instance, a value of 0.75 ensures only high-confidence behaviors are flagged.
generateAlert()	Triggers an alert when suspicious behavior is detected, notifying the appropriate user or system.	- incidentID (int): A unique identifier for the detected incident, linking it to logged data. - alertMessage (string): A detailed description of the alert, specifying what behavior was detected and any additional metadata (e.g., timestamp, location). Example: "Suspicious loitering detected near entrance camera at 3:45 PM."

logIncident()	Logs detailed information about an incident for future reference, analysis, and auditing.	- cameraID (int): The unique identifier of the camera where the suspicious activity was detected. - details (string): A full description of the incident, including metadata like time, location, and behavior type. Example: "Person entering restricted area without authorization, detected by Camera 5 on 2024-11-23."
addUser()	Adds a new user to the system and assigns them a specific role.	- userID (int): A unique identifier for the user. - name (string): The user's name. - role (string): The user's role within the system (e.g., "admin," "operator," or "viewer"). Example: Admin users have full control over system configurations, while operators can monitor live video feeds and review logs.
configureCamera()	Configures the operational settings of a specific camera in the surveillance system.	- cameraID (int): The unique identifier of the camera to configure. - resolution (string): Specifies the camera's resolution, such as "1080p" or "4K." - location (string): Describes the camera's physical location, e.g., "Main entrance - Gate 3." This ensures that cameras are correctly identified and mapped to their respective areas within the system.

trainDetector()	Updates the behavior detection model by using new training data to improve detection accuracy.	- trainingData (array): A dataset containing annotated examples of behaviors for training the model. Example: Arrays of labeled frames showing normal and suspicious activities. - algorithmName (string): Specifies the detection algorithm being used, such as "YOLOv5" or "ResNet-50," to ensure compatibility with the dataset and system requirements.
retrieveLogs()	Retrieves incident logs based on specific filters, such as date or camera, for analysis or reports.	- cameraID (int, optional): Allows filtering logs to show only incidents recorded by a specific camera. - date (datetime, optional): Filters logs to show only incidents occurring on a specific date. Example: Retrieve all incidents from 2024-11-23 for a system audit or compliance check.
deleteUser()	Removes a user from the system, revoking their access and permissions.	- userID (int): The unique identifier of the user to delete. Example: Deleting a former operator or an unused guest account. Additional safeguards may ensure that admin accounts cannot be deleted without elevated permissions.

 Object-Oriented (OO) Approach: Objects, Attributes, Methods, & Method Parameters:

1. Object: Camera

- **Attributes**:
  - id: String - Unique identifier for the camera.
  - location: String - The physical location of the camera.
  - status: Boolean - Indicates whether the camera is active or inactive.

    - resolution: String - The resolution of the camera feed (e.g., "1080p", "4K").
- **Methods**:
    - captureVideo():
        - **Parameters**: None.
        - **Description**: Captures a live video feed from the camera.
    - updateSettings(newResolution: String, newLocation: String):
        - **Parameters**:
            - newResolution: String - Updates the camera's resolution.
            - newLocation: String - Updates the camera's location.
        - **Description**: Modifies the resolution and location of the camera.

2. Object: BehaviorDetector

- **Attributes**:
    - algorithm: String - Name of the algorithm used for detection (e.g., YOLOv5).
    - threshold: Float - Minimum confidence score to flag suspicious behavior.
    - inputFeed: VideoStream - The live video feed being analyzed.
- **Methods**:
    - analyzeFeed(feed: VideoStream):
        - **Parameters**:
            - feed: VideoStream - The video stream to analyze.
        - **Description**: Processes the video feed to identify suspicious behaviors.
    - setThreshold(newThreshold: Float):
        - **Parameters**:
            - newThreshold: Float - Updates the confidence threshold for detections.

        - **Description**: Adjusts the sensitivity of the behavior detection algorithm.

3. Object: Alert

- **Attributes**:
    - alertID: Int - Unique identifier for the alert.
    - message: String - Detailed description of the alert.
    - timestamp: DateTime - Time the alert was generated.
    - severity: String - Level of alert severity (e.g., "low", "medium", "high").
- **Methods**:
    - sendAlert(recipient: User):
        - **Parameters**:
            - recipient: User - The user to notify about the alert.
        - **Description**: Sends the alert message to a specific user.
    - logAlert():
        - **Parameters**: None.
        - **Description**: Logs the alert details into the incident log.

4. Object: IncidentLog

- **Attributes**:
    - logID: Int - Unique identifier for the incident log entry.
    - description: String - Detailed explanation of the logged incident.
    - timestamp: DateTime - Time the incident occurred.
    - cameraID: String - Identifier of the camera related to the incident.
- **Methods**:
    - addEntry(logDetails: Incident):

        - **Parameters**:
            - logDetails: Incident - The details of the incident to add.
        - **Description**: Adds a new incident entry to the log.
    - retrieveEntries(startDate: DateTime, endDate: DateTime):
        - **Parameters**:
            - startDate: DateTime - Start of the date range for log retrieval.
            - endDate: DateTime - End of the date range for log retrieval.
        - **Description**: Fetches incident entries within a specific date range.

5. Object: User

- **Attributes**:
    - userID: Int - Unique identifier for the user.
    - username: String - The username of the user.
    - role: String - The user’s role in the system (e.g., admin, operator).
    - status: Boolean - Indicates whether the user is active.
- **Methods**:
    - login(credentials: Credentials):
        - **Parameters**:
            - credentials: Credentials - The username and password for login.
        - **Description**: Authenticates the user in the system.
    - updateRole(newRole: String):
        - **Parameters**:
            - newRole: String - The new role to assign to the user.
        - **Description**: Updates the user's role.

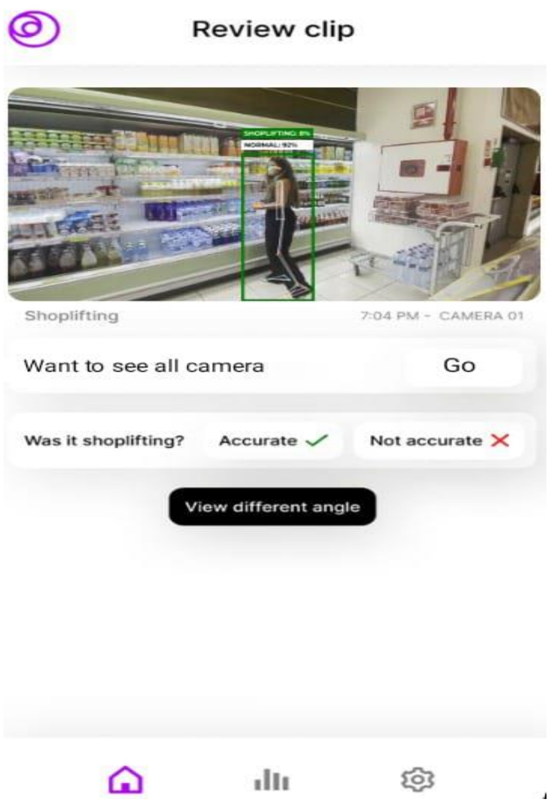
4.7 User Interface Design

Describe overview of the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user. [10]

1. Screen Images

Review Clips Fast and Easy:

Quickly assess potential shoplifting incidents through concise video clips, streamlining decision-making and reducing manual review time.



Purpose: This screen allows the user to review a flagged video clip that has detected a potential shoplifting activity.

Features:

- A clear video feed showing the area of interest with a bounding box around the person flagged for shoplifting.
- A timestamp and camera identifier (e.g., "CAMERA 01").
- Options to verify if the activity was shoplifting, marked as "Accurate" or "Not accurate."
- Buttons to:

[10] [Aligned with User interface design presented in the Deliverable 4 - Prototype]

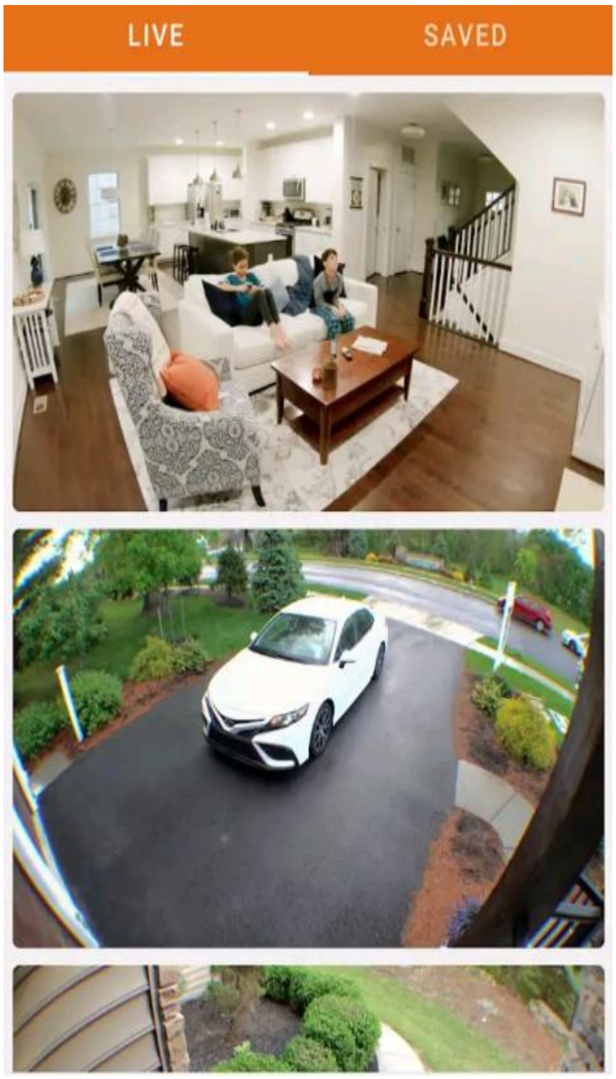
    - Review all camera feeds for additional context ("Go").
    - View the incident from different angles.

Design Elements:

- Emphasis on user interactivity to validate the AI's shoplifting detection.
- Minimalist interface for focused decision-making.

Multi-Camera Monitoring:

Effortlessly oversee multiple camera feeds simultaneously, eliminating the need for manual surveillance and enhancing operational efficiency.



- **Purpose**: This screen displays both live and saved footage from different camera views.
- **Features**:
    - Two tabs, **Live** and **Saved**, for toggling between real-time footage and previously recorded clips.

    - Real-time video monitoring of indoor and outdoor areas, such as a living room and driveway.
    - Organized layout for easy navigation and observation.
- **Design Elements**:
    - Tabbed navigation for seamless switching between modes.
    - High-resolution video previews to enhance clarity for decision-making

Plug and Play:
Just plugs into your existing camera systems without the need of additional hardware. Simply download, setup and start receiving shoplifting notifications within minutes.
Purpose:
This screen allows the user to review a history of flagged incidents where potential shoplifting activity has been detected.
Features:

- A chronological timeline of shoplifting events with timestamps and dates.
- Camera identifiers for each incident (e.g., "CAMERA 01").
- Thumbnail previews of the detected activity for quick visual assessment.

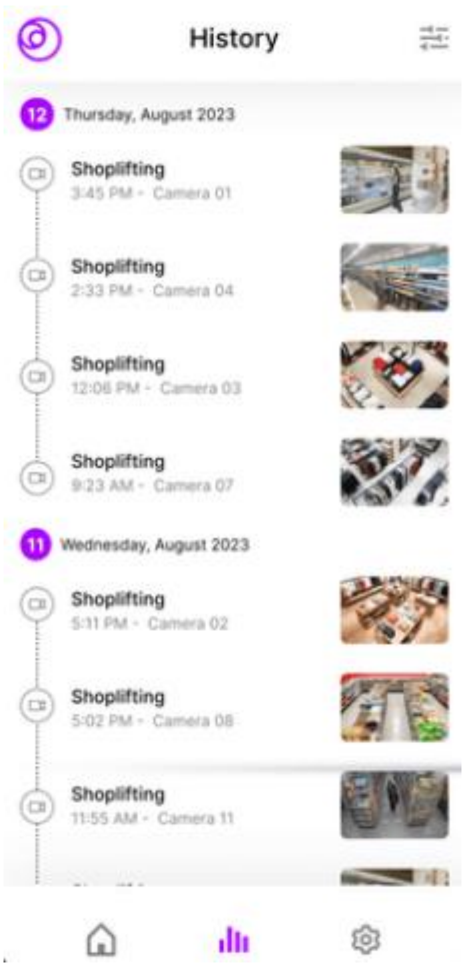
Buttons to:

- Open a detailed view of the flagged activity.
- Filter events based on date or camera.

Design Elements:

- Clear separation between events for better readability.

- Compact and organized layout to accommodate multiple incidents.


History

12 Thursday, August 2023

Shoplifting
3:45 PM - Camera 01

Shoplifting
2:33 PM - Camera 04

Shoplifting
12:06 PM - Camera 03

Shoplifting
9:23 AM - Camera 07

11 Wednesday, August 2023

Shoplifting
5:11 PM - Camera 02

Shoplifting
5:02 PM - Camera 08

Shoplifting
11:55 AM - Camera 11

Real-Time Notifications:

Receive instant alerts upon detection of suspicious behavior, enabling proactive intervention.
Purpose:
This screen promptly alerts the user about a detected shoplifting event, ensuring immediate awareness and action.
Features:

- A real-time notification pop-up with the alert message "Shoplifter detected!"
- Specific details such as the camera ID (e.g., "Suspicious activity detected on CAMERA 01") and timestamp.

Buttons to:

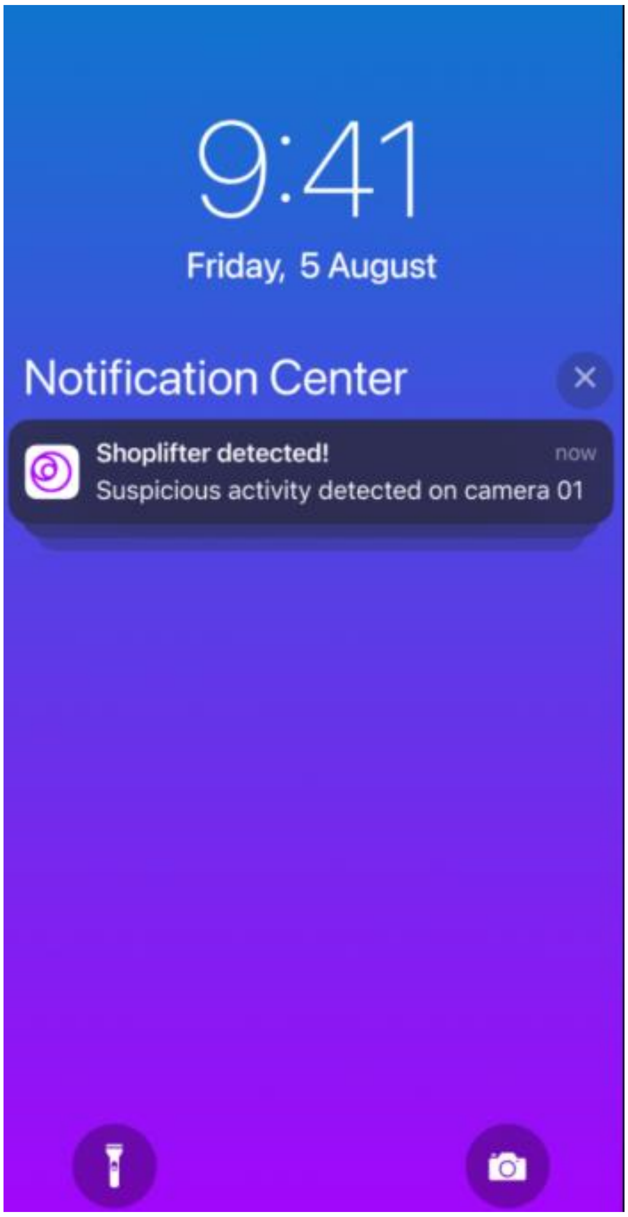
- Open the live feed for the camera where the incident occurred.
- Dismiss the notification after reviewing.

Design Elements:

- High-contrast colors to grab attention.

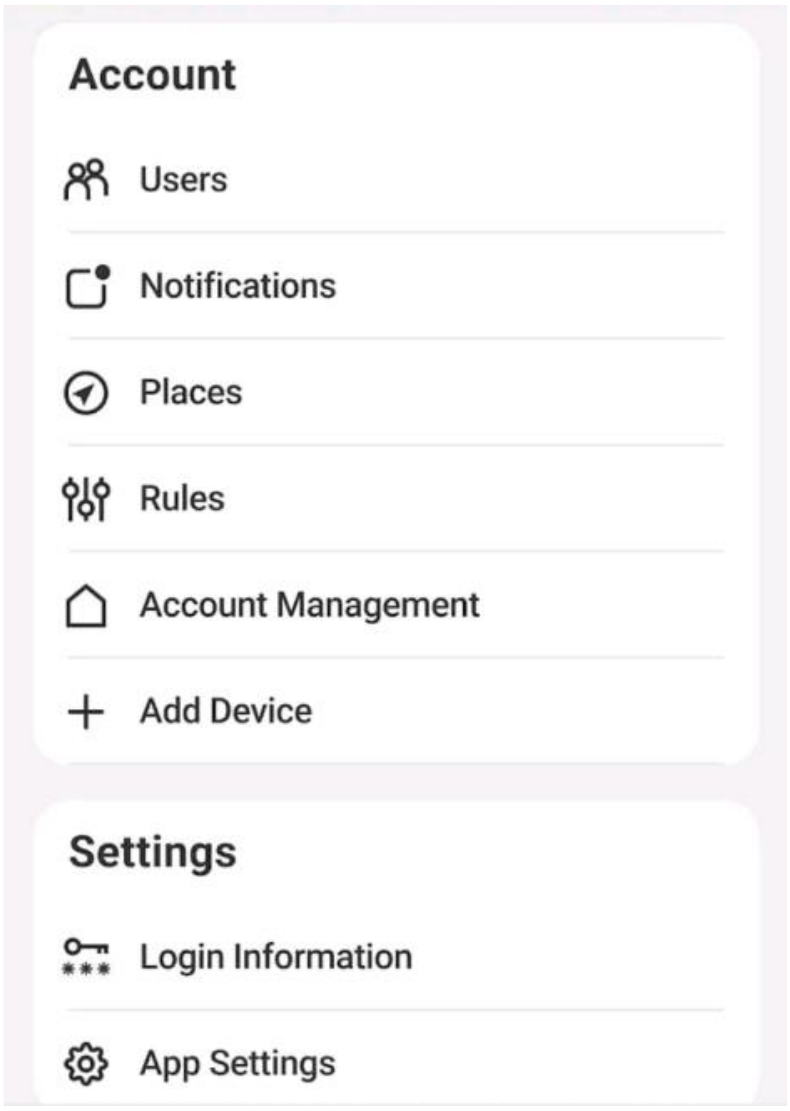
- Simple and clear text for instant understanding.

.



Shoplifter detected! now
Suspicious activity detected on camera 01

Settings:
Allows users to configure their preferences for the application. It includes options for "Login Information," where users can manage their credentials, and "App Settings," where they can customize the app's behavior or appearance to suit their needs.



Purpose:
This screen allows the user to customize system configurations, manage accounts, and add or update devices.

Features:

- Manage user accounts and permissions.
- Set up and configure notifications for different events.
- Assign and organize cameras by location.
- Define shoplifting detection rules and alert thresholds.
- Add new cameras or devices to the system.
- Manage login credentials and account security.

Buttons to:

- Navigate to specific sections like "Users," "Notifications," and "Places."
- Save or reset configurations in each section.

Design Elements:

- Intuitive icons to represent different settings.
- A categorized layout for easy navigation and user convenience.

[11]

2. Screen Objects and Actions

Use Case 1: Reviewing a Shoplifting Incident

- **Screen Objects:**
    1. **Incident Video Clip Preview:**
        - Displays a specific event of suspected shoplifting.
        - Contains a bounding box over the detected subject, labeled with "Shoplifting" and a confidence score.
    2. **Time and Camera Information:**
        - Indicates the timestamp and camera source of the recorded clip.
    3. **Review Options:**
        - Includes buttons for marking the detection as "Accurate" or "Not accurate."
    4. **Additional View Options:**
        - A button labeled "View different angle" for checking other camera angles of the same event.
    5. **Navigation Option:**
        - A button labeled "Want to see all cameras" with an associated "Go" action.
- **Associated Actions:**
    1. **Confirm or Reject Detection:**
        - The user can mark the detection as either accurate or inaccurate, providing feedback to refine the system's performance.
    2. **Switch to Alternate Camera Views:**
        - By selecting "View different angle," the user can verify the incident from other perspectives.
    3. **Navigate to All Cameras:**
        - The "Go" button takes the user to a screen displaying feeds from all available cameras.

Use Case 2: Viewing Shoplifting History

- **Screen Objects:**
    1. **Incident Timeline:**
        - Displays a chronological list of detected shoplifting events grouped by date (e.g., August 12, August 11).
    2. **Event Thumbnails:**
        - Small preview images corresponding to each recorded incident.
    3. **Camera and Time Labels:**
        - Displays the camera ID and timestamp for each event.
    4. **Scroll Navigation:**
        - Allows users to navigate through a timeline of incidents.
- **Associated Actions:**
    1. **Review a Specific Incident:**

[11] Please don't forget to add the caption of each figure (Use References -> Insert Caption option in MS Word)

- Users can click on an incident to open its detailed view for review and validation.

2. **Navigate Through Timeline:**
   - Scrolling through the timeline provides a quick overview of past incidents across cameras and dates.
3. **Filter and Sort Events (if applicable):**
   - A filter/sort option (visible in the top-right corner) allows users to refine the view based on criteria like time, camera, or detection type.

4.8 Design Decisions

1. Object-Oriented Design (OOD) Approach

- **Chosen Approach**: **Object-Oriented Programming (OOP)**
- **Reason for Choice**:
  - **Modularity**: OOP promotes modularity, allowing the system to be broken down into smaller, manageable units (objects) like Camera, User, Alert, and Incident. This modularity makes it easier to develop, maintain, and extend the system.
  - **Reusability**: Components such as detection algorithms and alert generation logic can be reused across different parts of the system.
  - **Maintainability**: The encapsulation of attributes and behaviors within objects leads to better organization and easier debugging and updating. For instance, you can change the behavior of a BehaviorDetector object without affecting other parts of the system.
  - **Scalability**: As the system grows (e.g., adding more cameras or detection methods), OO principles help manage complexity by allowing objects to be extended or modified independently.

---

2. Machine Learning Models for Behavior Detection

- **Chosen Approach**: **Deep Learning Models (Faster R-CNN, YOLO, ConvLSTM, etc.)**
- **Reason for Choice**:
  - **Real-time Performance**: Deep learning models like YOLO (You Only Look Once) and Faster R-CNN are known for their ability to perform fast object detection with high accuracy. These models can analyze video feeds in real-time, which is crucial for detecting theft and other suspicious activities instantly.
  - **Accuracy**: YOLO and Faster R-CNN are well-suited for object detection and behavior analysis, which ensures the system can accurately detect theft activities such as concealed items or unusual behavior.
  - **Adaptability**: The system can be updated with new models or fine-tuned with custom training datasets as the environment changes or if new types of suspicious activities need to be detected.

---

3. Algorithmic Approach for Event Detection

- **Chosen Approach**: **Object Detection via Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)**
- **Reason for Choice**:
    - **Real-time Surveillance**: CNNs are particularly well-suited for image and video analysis. Using CNN-based models like YOLO or Faster R-CNN allows the system to identify theft and suspicious activities quickly and accurately.
    - **Temporal Behavior Analysis**: For detecting complex actions, **ConvLSTM** (Convolutional LSTM) is employed, which combines CNNs for spatial analysis with LSTMs for temporal dynamics, making it effective for analyzing movement patterns (e.g., person following an unusual path or hiding items).
    - **Efficiency**: These models offer a balance between high detection accuracy and computational efficiency, ensuring that the system can process multiple video streams without significant performance degradation.

---

4. Database Design and Normalization

- **Chosen Approach**: **Relational Database with Normalization (Up to 3NF)**
- **Reason for Choice**:
    - **Data Integrity and Consistency**: A relational database ensures that the system's data is structured and easily accessible while maintaining integrity. Normalization helps eliminate redundant data and ensures efficient storage.
    - **Efficiency**: Normalization up to **Third Normal Form (3NF)** ensures that each piece of data is stored only once, which reduces the chances of data anomalies and enhances query performance.
    - **Scalability**: Relational databases can easily scale by adding more tables and relationships as the system grows (e.g., adding new camera feeds, incidents, or logs).
    - **Security**: A relational database offers strong security features like encryption, access control, and audit trails, which are essential for protecting sensitive video footage and incident data.

---

5. Cloud Integration for Scalability and Storage

- **Chosen Approach**: **Cloud Storage and Cloud Computing (AWS, Azure, etc.)**
- **Reason for Choice**:
    - **Storage of Large Volumes of Data**: Given the large amount of video footage generated, cloud storage offers a scalable and cost-effective solution for storing video files and incident logs.

   - **Computational Scaling**: For tasks like training or retraining machine learning models, cloud computing services provide the computational power needed to handle these heavy tasks without overloading local systems.
   - **Redundancy and Backup**: Cloud services offer redundancy, ensuring that critical data (like surveillance footage) is backed up and available even if local systems fail.
   - **Accessibility**: Cloud-based systems enable remote monitoring and alerting capabilities for security personnel, allowing them to access data from anywhere.

---

6. Real-time Processing and Latency Considerations

- **Chosen Approach**: **Edge Computing (for Video Analysis)**
- **Reason for Choice**:
   - **Low Latency**: Real-time video analysis demands low latency, and processing video feeds at the **edge** (close to the cameras) ensures quick detection and alerting, minimizing delays in identifying potential threats.
   - **Reduced Network Load**: Edge computing allows for preprocessing (such as detecting suspicious events or identifying items) at the camera or local server level, thus reducing the bandwidth required to send video data to the central server.
   - **Scalability**: This approach allows adding new cameras and processing units to scale the system across different store locations without overloading central servers.

---

7. User Interface and Dashboard Design

- **Chosen Approach**: **Web-based Dashboard for Admin and Security Personnel**
- **Reason for Choice**:
   - **Accessibility**: A web-based dashboard makes it easy for security personnel and administrators to monitor the system from any device, anywhere in the store or remotely.
   - **Real-time Alerts**: The dashboard provides a user-friendly interface for viewing live camera feeds, monitoring detected incidents, and responding to alerts.
   - **Ease of Integration**: The web interface can be easily integrated with various backend components (e.g., incident logs, machine learning models, camera feeds), allowing security personnel to control all aspects of the surveillance system through a single platform.

---

8. Integration with Existing Retail Infrastructure

- **Chosen Approach**: **Modular Integration with Legacy Systems (CCTV and VMS)**
- **Reason for Choice**:
   - **Compatibility**: The system is designed to interface with existing CCTV and Video Management Systems (VMS) to avoid expensive upgrades or replacements. A modular approach ensures minimal disruption during implementation.

- **Gradual Transition**: Legacy systems can be gradually upgraded or replaced without affecting the overall functionality of the shoplifting-theft detection system.
- **Cost-Efficiency**: By leveraging existing hardware, the system can be deployed with lower initial investment and faster deployment times.

4.9 Summary

In this chapter, the design of the **EyeGuard Shoplifter Detection System** has been outlined, emphasizing the integration of key components such as **Cameras**, **Behavior Detectors**, **Incident Logs**, **Alert Generators**, and **Users**. The focus was on transforming the system's information domain into a structured data architecture that ensures efficient storage, processing, and retrieval. Data entities like **Alerts**, **Incidents**, and **Cameras** were modeled as objects with clearly defined attributes. Relationships between entities were established to reflect real-world connections, such as the **One-to-Many** relationship between **Users** and **Admin**, and the **One-to-Many** relationship between the **Shoplifter Detection System** and **Incidents**. The design ensures that these components work seamlessly together to detect suspicious behavior, generate alerts, and log incidents, creating a comprehensive system for real-time monitoring and management.

The chapter also highlights the use of modularity in the system's architecture, particularly with the **Behavior Detectors** and their relationship with the **Shoplifter Detection System**. The design accommodates flexibility in the number of detectors used and the possibility of multiple detectors managing different zones. Additionally, the role of functions such as `detectBehavior()`, `generateAlert()`, and `logIncident()` has been clearly defined, each with specific parameters to ensure accurate detection and alerting. The database design ensures that critical data like **Incident Logs** can be filtered and retrieved efficiently for analysis, making it easier to monitor and audit the system's performance.

This chapter is integral to the overall project as it provides a clear blueprint for how data will be structured and managed, ensuring that the EyeGuard system is capable of meeting its objectives. The design addresses the need for real-time detection and response to shoplifting activities, enabling efficient monitoring by administrators and operators. Furthermore, it ensures that data is organized in a way that supports scalability, easy maintenance, and future enhancements, contributing to the system's long-term effectiveness and adaptability in a retail environment.

References

References

1. Book [1]

[1] A. S. T. a. H. Bos, Modern Operating Systems, 4th ed., vol. 1, Upper Saddle River, NJ, USA: Pearson Education,, 2014.

[2] Y. B. a. A. C. I. Goodfellow, Deep Learning, 1st ed., vol. 1, Cambridge: MA, USA: MIT Press,, 2016.

2. Article in a Journal

[3] C. S. e. al., "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* Vols. vol. 1, , no. no.1, pp. pp. 1-9, June 2015.

[4] C. E. Shannon, "A mathematical theory of communication," *he Bell System Technical Journal*,, vol. vol. 27, no. no. 3, pp. pp. 379-423, July 1948.

3. World Wide Web

[5] " OpenAI," ChatGPT: Language model, [Online]. Available: Available: https://chat.openai.com/..

[6] N. [Online], "National Retail Federation," 2023 National Retail Security Survey, [Online]. Available: Available: https://nrf.com/.

[7] RetailNext, "AI-powered retail theft detection solutions," RetailNext. [Online], [Online]. Available: Available: https://retailnext.net/.

4. Articles from Conference Proceedings (published)

[8] N. D. a. B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),*, pp. pp. 886-893, 2005,.

[9] M. A. e. al, "TensorFlow: A system for large-scale machine learning," *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, pp. pp. 265-283., 2016.

[10] A. S. T. a. H. Bos, Modern Operating Systems, 4th ed., vol. 1, Upper Saddle River, NJ, USA: Pearson Education,, 2014.

[11] Y. B. a. A. C. I. Goodfellow, Deep Learning, 1st ed., vol. 1, Cambridge: MA, USA: MIT Press,, 2016.

5. ChatGPT

[12] "ChatGPT,"ChatGPT Response on shoplifting detection systems," OpenAI,, [Online]. Available: Available: https://chat.openai.com/.