SMARTSTOCK: Real-Time AI-Powered Vision System for Automated Warehouse Inventory Management

Lum Fu Yuan, Yun-Huoy Choo*, Azah Kamilah Muda

Faculty of Artificial Intelligence and Cyber Security, University Technical Malaysia Melaka (UTeM), Durian Tunggal, Melaka, 76100 Malaysia

***Corresponding Author**

DOI: https://dx.doi.org/10.47772/IJRISS.2025.91100289

Received: 12 November 2025; Accepted: 18 November 2025; Published: 08 December 2025

ABSTRACT

SMARTSTOCK is an intelligent warehouse management system developed to overcome long-standing challenges in stock visibility, manual inventory counting, and inefficient resource utilization. The system integrates artificial intelligence, computer vision, and image processing to enable real-time stock detection, customer presence tracking, and parking lot traffic monitoring. Designed to enhance operational transparency and decision-making efficiency, **SMARTSTOCK** automates key warehouse functions traditionally dependent on manual supervision. The architecture comprises two core modules: an AI-based Video Recognition Module and a Real-Time Visibility and Reporting Module. The AI-based Video Recognition Module incorporates three sub-modules—Stock Detection and Tracking, Client Presence Counting, and Car Park Counting—which employ YOLOv8 models for object detection and the SORT algorithm for object tracking and counting. Experimental evaluation demonstrated high reliability and accuracy, achieving 90.16% for stock tracking, 95% for client presence counting, and 100% for car park occupancy detection. The Real-Time Visibility and Reporting Module provides a unified dashboard for data visualization, live monitoring, and decision support, significantly reducing human error and out-of-stock occurrences. Despite its strong performance, **SMARTSTOCK** faces limitations related to hardware dependency and difficulty in detecting low-stock items under full occlusion. Future enhancements will focus on cloud-based implementation, model optimization, and integration with point-of-sale systems to achieve comprehensive inventory intelligence. Overall, **SMARTSTOCK** represents a robust, explainable, and scalable AI-driven framework that advances warehouse automation, improves resource utilization, and strengthens real-time decision-making within retail environments.

Keywords: Warehouse Automation, Computer Vision Object Tracking, Real-Time Inventory Monitoring, Explainable AI, AI in Logistics.

INTRODUCTION

Effective warehouse management is critical for modern retail operations, yet many warehouses continue to rely on manual inventory counting and inventory matching procedures. These traditional methods are labor-intensive, error-prone, and often fail to provide real-time visibility into stock levels, resulting in frequent out-of-stock (OOS) conditions. Such inefficiencies can lead to substantial sales losses, operational disruptions, and diminished customer satisfaction. The challenge is further compounded when products are poorly positioned, partially visible, or misaligned, making identification difficult using conventional approaches. Without immediate insights into inventory levels, timely response to stock changes is nearly impossible, often resulting in overstocking, delayed replenishment, and difficulty locating OOS areas. Additionally, monitoring in-store customer traffic and parking availability is essential to optimize warehouse operations. Understanding traffic patterns allows warehouse staff to allocate resources effectively, anticipate demand, and ensure that high-demand products are replenished before shelves run empty. A lack of real-time traffic monitoring during peak periods or promotional events may exacerbate temporary OOS conditions, affecting both sales and customer loyalty.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

To address these challenges, the **SMARTSTOCK** system leverages artificial intelligence (AI), computer vision, and image processing technologies, including YOLO (You Only Look Once) for object detection and SORT (Simple, Online, and Realtime Tracking) for tracking and counting. The system provides a comprehensive, automated solution for monitoring inventory, detecting OOS products, and analyzing in-store and parking traffic. By automating critical warehouse processes, **SMARTSTOCK** reduces reliance on manual operations, minimizes human error, and improves operational efficiency. The system integrates three core modules: stock detection and tracking, customer presence monitoring, and car park counting, all supported by a user-friendly desktop application that provides real-time reporting and analytics. A robust database underpins the system, enabling efficient storage, retrieval, and historical analysis of inventory and traffic data.

The expected outcomes of **SMARTSTOCK** include accurate real-time detection and tracking of warehouse stock, automated identification of low-stock and OOS items, monitoring of customer presence and parking availability, and provision of actionable insights through an intuitive desktop interface. These capabilities allow warehouse managers and staff to optimize resource allocation, expedite replenishment, and make informed operational decisions. By enhancing inventory visibility, improving traffic monitoring, and automating repetitive tasks, **SMARTSTOCK** aims to strengthen overall warehouse efficiency, reduce operational costs, and improve customer satisfaction. Its scalability and adaptability make it suitable for deployment across diverse retail and warehouse environments, offering a sustainable and intelligent approach to modern warehouse management.

LITERATURE REVIEW

The development of the **SMARTSTOCK** system is underpinned by advancements in three key technical areas: the challenges and digitization of inventory management, the application of computer vision in logistics, and the evolution of real-time object detection models.

The Need for Automated Inventory Management

Traditional inventory systems, reliant on manual processes such as periodic audits and barcode scanning, are inherently inefficient and introduce susceptibility to significant human transcription errors and data latency. The integration of cyber-physical systems, a core tenet of Industry 4.0, mandates a shift towards automated, continuous monitoring solutions. Research in smart warehousing highlights the critical need for systems that provide real-time visibility into stock levels to minimize holding costs and prevent stockout scenarios [1]. Studies have explored utilizing Radio-Frequency Identification (RFID) and sensor networks; however, these technologies often require direct tagging of every item, presenting scalability and cost challenges for high-volume, dynamic environments [2]. The limitations of conventional methods establish a clear requirement for non-intrusive, vision-based alternatives.

Object Detection in Logistics and Supply Chain

Computer vision has emerged as a disruptive technology in supply chain optimization, offering solutions for quality inspection, package sorting, and inventory assessment. Early implementations primarily used traditional image processing techniques, but modern approaches leverage deep learning for superior performance under complex conditions. Convolutional Neural Networks (CNNs) have facilitated significant breakthroughs, classifying detection models into two main categories: two-stage detectors (e.g., R-CNN, Faster R-CNN) and single-stage detectors (e.g., SSD, YOLO) [3]. While two-stage models typically yield marginally higher localization accuracy, single-stage detectors are demonstrably favored in industrial applications where high inference speed is a non-negotiable prerequisite for real-time operation.

Evolution of the YOLO Framework

The You Only Look Once (YOLO) architecture, as analyzed in Table 1 and Figure 1, has become the de facto standard for real-time object detection due to its ability to frame object detection as a single regression problem, vastly improving speed while maintaining competitive accuracy.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

TABLE I. YOLO Technology's Evolution and Performance Analysis

Version	**Performance Analysis**
YOLOv1(2015)	Achieved 63.4 mean Average Precision (mAP) and 45 frames per second (FPS) on the Pascal VOC 2007 dataset, surpassing earlier methods like R-CNN.
YOLOv2 (2016)	Introduced anchor boxes and could detect over 9000 object categories, achieving 76.8 mAP at 67 FPS. This version improved both the accuracy and the speed of the model.
YOLOv3 (2018)	Used Darknet-53 as the backbone and logistic classifiers for improved accuracy, achieving an mAP of 28.2 at 22 milliseconds per image. This version continued to build on the speed and accuracy improvements.
YOLOv4(2020)	Introduced concepts like Bag of Freebies and Bag of Specials, achieving 43.5 mAP at 65 FPS on the COCO dataset. This version focused on enhancing the model's performance without increasing computational cost.
YOLOv5(2020)	Implemented in PyTorch and utilized a Cross-stage Partial (CSP) connection for better performance. YOLOv5 is known for its ease of use and rapid adoption in the industry.
YOLOv6 (2022)	An unofficial version released by Meituan, designed for industrial applications with superior performance. This version emphasized robustness and application in diverse industrial contexts
YOLOv7 (2022)	Noted for its speed and accuracy, achieving a mAP of 56.8% at various FPS ranges. YOLOv7 continued the tradition of improving both the speed and accuracy of the model.
YOLOv8 (2023)	Known for its high accuracy and speed, featuring a Python package and CLI-based implementation for ease of use. YOLOv8 represents the latest advancements in the YOLO family, integrating state-of-the-art techniques for superior performance.

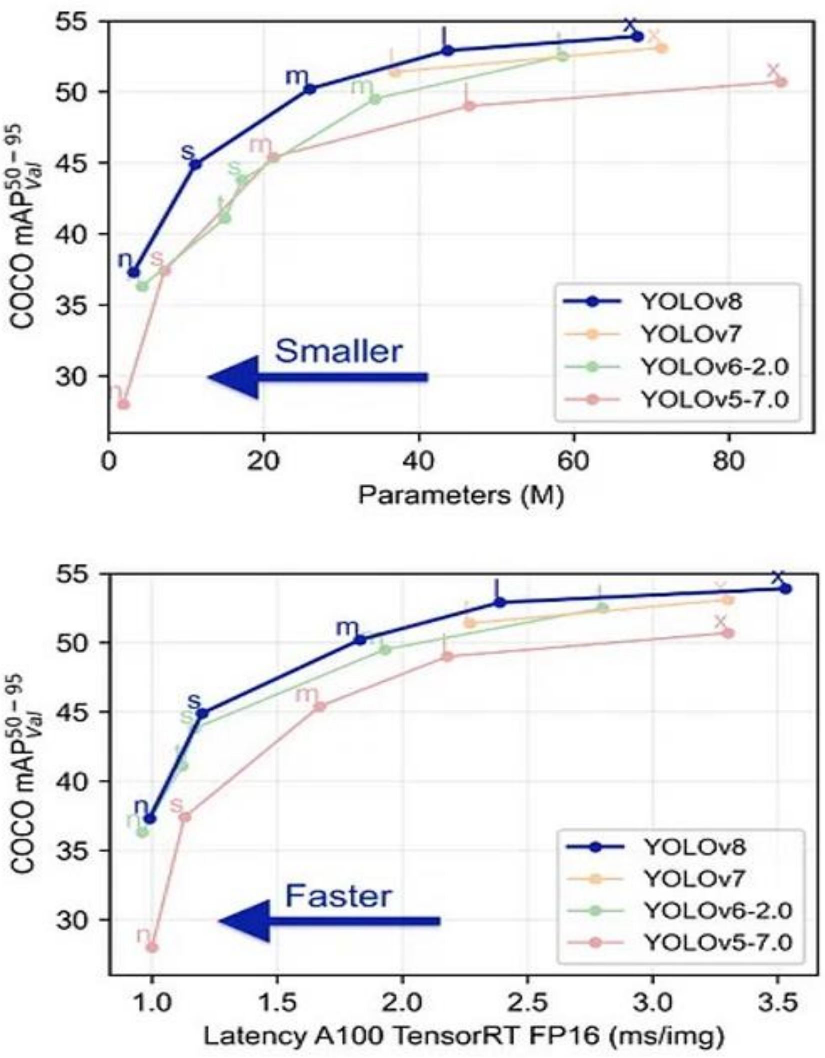

Figure 1: YOLOv8 Performance Against Other Prominent Versions [7]

Successive iterations have introduced significant architectural refinements:

YOLOv3 and YOLOv5: These versions introduced improvements like feature pyramid networks (FPN) and enhanced data augmentation, which were previously applied in logistics for monitoring assembly lines and simple shelf checks [4]. However, these versions often required substantial computational resources, limiting their deployment on edge devices or modest server infrastructure.

YOLOv8: This latest generation, developed by Ultralytics, represents a significant leap forward in optimizing the balance between speed and accuracy. YOLOv8 features a decoupled head for separate classification and localization tasks, alongside anchor-free detection, which contributes to faster training and improved performance on small objects. These advancements make YOLOv8 particularly suitable for the challenging task of inventory counting where items are often dense and partially visible.

Multi-Object Tracking and Counting

Advanced object counting and tracking techniques are critical for real-time applications, extending beyond logistics into traffic management and public monitoring. Mandal and Adu-Gyamfi's study [8] provided a comprehensive analysis of algorithms like YOLO, Faster R-CNN, SSD, SORT, and DeepSORT, concluding that the combination of the YOLOv3 detector with the SORT tracking algorithm offers the optimal balance between detection accuracy and real-time performance for vehicle counting applications. Building on tracking performance, [9] proposed a novel scenario-based object counting method that introduces scene awareness by integrating re-identification (Re-ID) with object detection (using CenterNet), specifically segmenting regions and clustering targets to reduce ID swapping and improve accuracy in densely populated areas. Furthermore, the problem of accurate counting in highly complex environments was addressed by [10], who introduced the Locount task, which explicitly integrates object location and counting to robustly handle severe occlusion among similar targets in retail environments. More recent work by [11] further refined counting accuracy in crowded urban settings with the YOLO-PC method, which adjusts edge values and utilizes a high-resolution dataset to detect more objects with lower error. These studies collectively confirm the trend of integrating high-speed object detection (like YOLO) with sophisticated tracking and counting logic to ensure reliability in real-world, dynamic scenarios.

Image Masking

Effective real-time inventory and traffic management relies on a combination of robust image processing techniques. This project's system integrates YOLOv8 for detection, SORT for tracking, and layer masking for focused image processing. YOLOv8 is selected as the primary object detection algorithm due to its superior speed and accuracy, evidenced by research showing a mAP50 score of 0.933 on retail products, ensuring fast detection of items and traffic [12]. For sustained tracking across frames, the lightweight SORT algorithm is employed; it uses the Kalman filter and the Hungarian algorithm to efficiently predict and associate object motion, maintaining high accuracy even with fast movement and occlusion. Complementing these, layer masking, supported by the work of [13], is crucial for enhancing robustness by selectively hiding or showing parts of the frame to isolate noise and irrelevant details, e.g., masking external streets during car lot monitoring, ensuring that detection and counting are highly precise and task-specific.

Research Gap

While existing literature confirms the efficacy of computer vision in general logistics, a significant gap remains in the practical deployment of state-of-the-art, high-speed models, specifically YOLOv8, for continuous, real-time inventory management across diverse, densely-packed stock items, validated by a custom, real-world warehouse dataset. This study addresses this gap by developing and validating the **SMARTSTOCK** system, focusing on operational efficiency and highly accurate, real-time quantification.

Proposed Solution

The proposed solution adopts a dual methodological framework to address its core components. The AI-based Video Recognition Module follows the CRISP-DM framework, comprising business understanding, data

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

understanding, preparation, modeling, evaluation, and deployment. Objectives focus on automated front-face product counting and OOS detection to enhance warehouse efficiency. Data collection included 935 images from diverse shelf conditions, followed by augmentation (rotation, brightness, blur) and labeling via Roboflow, as shown in Figure 2.

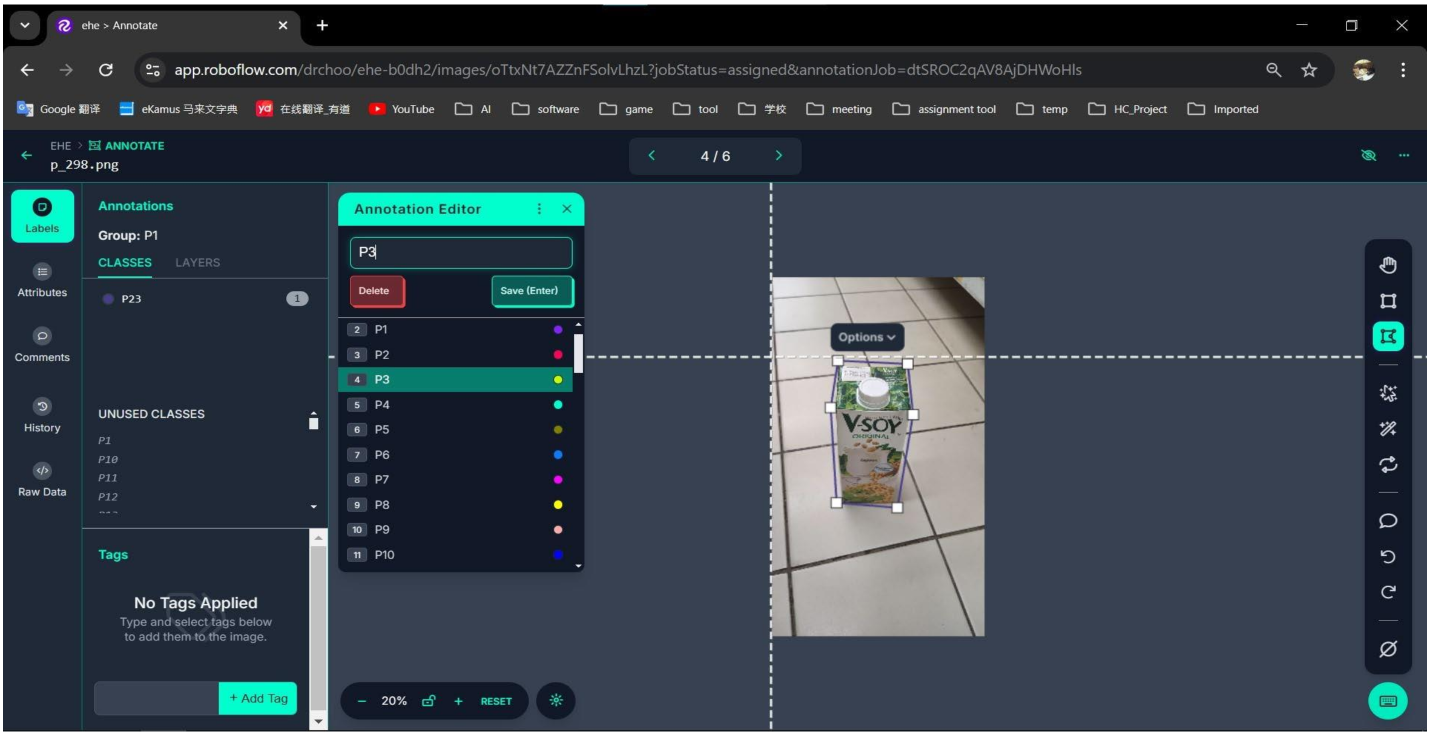

Figure 2: Example of Annotation in Roboflow

YOLOv8 was employed for multi-object detection, with performance evaluated using precision, recall, confusion matrices, and mAP50. Deployment integrates SORT for object tracking and counting, ensuring each item is detected once. CRISP-DM facilitated structured, iterative refinement for reliable, real-time warehouse monitoring.

The Development Design and Specification

The **SMARTSTOCK** Warehouse Management System is developed using the Agile methodology, enabling iterative cycles of planning, designing, developing, testing, reviewing, and launching. The Plan and Design stages define objectives, critical modules (Real-time Visibility, Reporting, Desktop Application), and system architecture, including ERDs and UI wireframes. Development employs C#/.NET for the desktop application, WebSockets for Python-AI integration, and MySQL for database management. Testing ensures functionality, performance, accuracy, and API reliability. Deployment, review, and launch phases implement the system in real-world scenarios, incorporate user feedback, and provide training, maintaining adaptability and responsiveness to evolving warehouse requirements.

The Technical Specification of the solution development is in Table 2.

TABLE 2. Solution Development Specification

Category	**Specification**
Hardware	
Processor	Intel Core i5-8250u
Installed RAM	20GB
GPU	NVIDIA MX150
Storage	500GB NVMe SSD
Software	
IDE for Backend	Visual Studio Code

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

IDE for Frontend	Visual Studio 2022
GPU Acceleration	NVIDIA CUDA and cuDNN
Cloud-based Environment	Google Colab Jupyter Notebook
Diagram Tool	Draw.io
GPU Monitor	MSI Afterburner
CPU Monitor	Intel® Extreme Tuning Utility
Database Manager	XAMPP
Python Environment Manager	Anaconda
Documentation Tool	Microsoft Office
Image Labelling Tool	RobotFlow
Version Control	GitHub
Frameworks	
Deep Learning	Tensorflow and Pytorch
GUI Development	.NET (WinForms)
Video Streaming API	WebSocket
Programming Language	
Python	Primary language for Backend
C#	Developing the frontend side

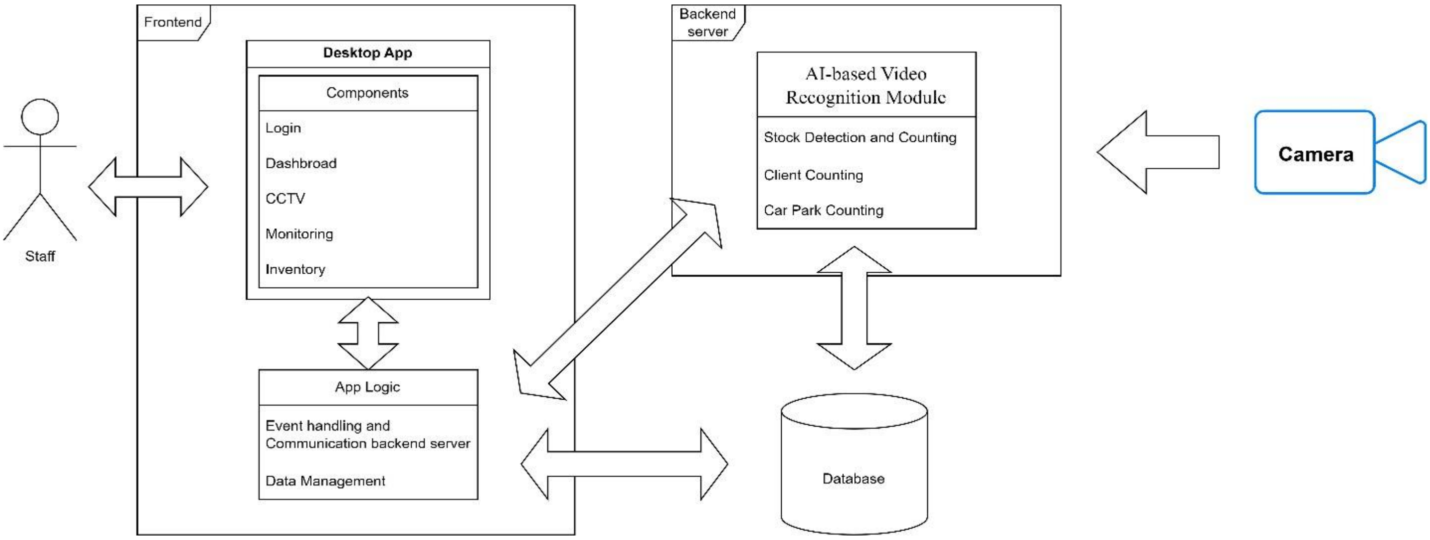

Figure 3: **SMARTSTOCK** System Architecture

System Architecture

This system architecture, as shown in Figure 3, is divided into two major sections: the frontend and the backend. The frontend is the desktop program that employees use to interface with the system, while the backend includes the AI-based video recognition module and the database. It employs a dual methodological framework to address its core components: the AI-based Video Recognition Module follows the CRISP-DM methodology, while the Warehouse Management System follows the Agile methodology. The business understanding defines objectives for automated front-face product counting and Out-of-Stock detection. Data collection involved 935 shelf images under varying lighting and angles, augmented via rotation, brightness adjustment, blur, and other transformations to improve model robustness. YOLOv8 (n, s, m, l weights) was employed for object detection, integrated with the SORT algorithm for unique object tracking and counting. Model performance was evaluated using precision, recall, confusion matrices, and mAP50.

The Warehouse Management System development enable iterative planning, designing, developing, testing, and reviewing. The Plan and Design stages established objectives, critical modules (Real-time Visibility, Reporting, Desktop Application), system architecture, ERDs, and UI wireframes. Development utilized C#/.NET for the desktop application, WebSockets for Python-AI integration, and MySQL for database management. Rigorous testing ensured functionality, performance, API reliability, and robustness, while Deployment, Review, and

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

Launch stages incorporated real-world implementation and user training, allowing the system to adapt to evolving requirements.

To enhance the technical rigor, scalability, and operational transparency of the **SMARTSTOCK** system, the proposed solution incorporates a structured architecture that integrates backend services, AI inference modules, and user-facing interfaces into a unified real-time pipeline. The proposed architecture integrates a modular MVC-based desktop application with a powerful backend. The GUI provides login, dashboard, live CCTV feeds, real-time shelf monitoring, and inventory management. The backend hosts the AI module, comprising stock detection and tracking, client presence counting, and car park monitoring. Scalability is addressed through containerized deployment of inference services, enabling horizontal expansion and efficient resource allocation under fluctuating workloads.

The database stores real-time and historical data, enabling reliable storage, retrieval, and processing. This integrated approach ensures efficient, real-time warehouse monitoring and management, reduces manual errors, optimizes resource allocation, and enhances overall operational efficiency.

AI-Based Video Recognition Model

The **SMARTSTOCK** system's stock detection and tracking uses a YOLOv8-based custom model trained on images collected from MYDIN Melaka, while client presence and car park models use pre-trained networks. Data preprocessing in RoboFlow includes rotation, brightness adjustment, and blur to enhance robustness. Data is split 80/10/10 for training, validation, and testing. Real-time object detection is performed using YOLOv8, and SORT using Kalman filter and Hungarian algorithm to tracks objects across frames. Linear counting ensures each object is counted once as it crosses a defined line. Results are continuously updated to the database for real-time monitoring.

Stock Detection and Tracking Algorithm Design

The stock detection model used YOLOv8 custom training model to detect the front-face of a product. It captures the real-life shelf state video as input. The trained model is used for real-time object detection. YOLO divides the image into a grid and predicts bounding boxes and class probabilities directly from the full images in one evaluation. The bounding box coordinates are given as in Equation (1):

$$(X_1, Y_1), (X_2, Y_2) \quad (1)$$

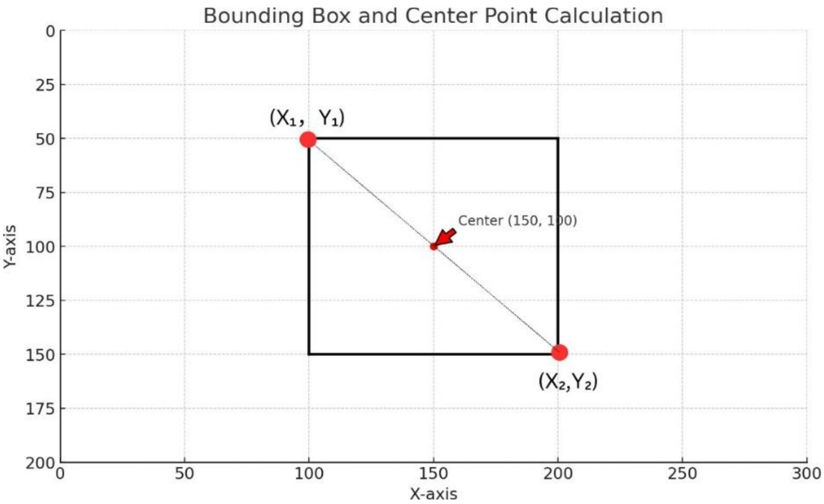

Figure 4: Center Point Calculation

Where (X_1, Y_1) and (X_2, Y_2) are the coordinates of the top-left and bottom-right corners of the bounding box respectively. The center point of the detected object is calculated as shown in Figure 4.

The object tracking uses the SORT approach to track objects across frames based on their bounding boxes, using the Kalman Filter and the Hungarian Algorithm. The Kalman Filter is a recursive algorithm used to estimate the state of a linear dynamic system from a series of noisy measurements. It involves two main steps, i.e. the prediction step to estimates the state at the next time step, and the update step to corrects the state estimate using the measurement at the current time step. The state vector typically includes the position and velocity of the object as in equation (2).

$$X_k = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}]^T \qquad (2)$$

where

x, y represents the object's center coordinates

s is the scale area

r is the aspect ratio

$\dot{x}, \dot{y}, \dot{s}$ are the respective velocities

The Hungarian Algorithm is used for data association, solving the assignment problem to optimally match predicted positions of tracked objects with new detections. This minimizes the total distance (measuring as cost) between the predicted and the detected object positions.

In linear counting, defining a line and to check if the center point of the detected object crossing the particular line is crucial to accurately counting each object only once as it passes a predefined virtual line. The key steps involve:

1. Define the line using two coordinate pairs, linear [x1, y1, x2, y2].
2. Determine if the center point of the object crosses the line, is_crossing =(x1-20<cx<x2+20) & (y1-20<cy<y2+20)
3. Avoid double counting using SORT approach. If an object crosses the line and has not previously been counted, update the count for that object class.
4. Continuous looping the detecting, tracking and counting procedures until the end before sending the results to the database.

The stock detection and tracking algorithm effectively integrates object detection using YOLO and object tracking using SORT. By defining a line and checking for crossing events, it accurately counts objects, avoiding double counting through the use of unique IDs assigned to tracked objects. Hence, the proposed design can be efficiently used for real-time stock detection and tracking in this **SMARTSTOCK** System.

Customer Counting Algorithm Design

The customer counting model uses YOLOv8 with a pre-trained model for object detection sharing similarities to the stock detection and tracking algorithm design, differ primarily in the linear counting method and the pre-trained YOLO model. Camera is recommended to be positioned at a high angle at the store's entrance to capture the customer flow data. The pre-trained YOLOV8 model works well without fine tuning. A two-liner counting algorithm as described in the previous section were used to detect the number of customers entering and exiting the store entrance.

Car Park Counting Algorithm Design

This model uses video footage captured through a high-angle camera, combined with a masking technique to exclude irrelevant areas from analysis. The YOLO model was then employed to detect and classify parking spots either free or occupied. The algorithm provides accurate counts of available and occupied parking spots, enhancing monitoring of parking to understanding the traffic pattern, as shown in Figure 5.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

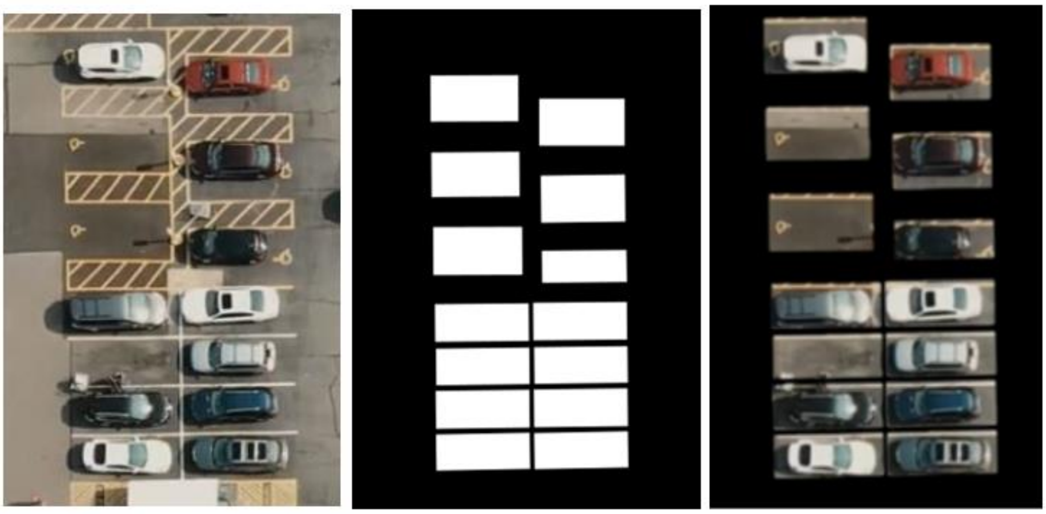

Figure 5: (from left to right) The Input Image Frame, The Mask, and The Masking layer.

The User Interface Design

The proposed user interface as shown in Figure 6 to Figure 10 represents the important modules in the **SMARTSTOCK** system. Figure 6 provides a consolidated real-time overview of warehouse operations through an intuitive visual interface. Key indicators at the top summarize out-of-stock items, client activity, parking availability, and shelf inventory levels.

Figure 6: Main Dashboard of the **SMARTSTOCK** System

The central statistical chart illustrates temporal trends in restock quantities and item counts, supporting performance monitoring. A donut chart highlights the top five restocked products, enabling quick identification of high-demand items. The lower panels present total operational counters and a detailed low-stock area table to prioritize replenishment tasks. Navigation on the left sidebar allows seamless access to CCTV, monitoring, stock, and inventory modules.

Figure 7 shows a "Low Stock Area" management interface. The main panel lists products by area, name, and quantity, highlighting items with critically low stock. On the right, detailed fields display the selected product's image, ID, supplier, quantity, and unit price. A "Restock" button allows quick replenishment, providing a clear, organized way to monitor and manage inventory efficiently.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

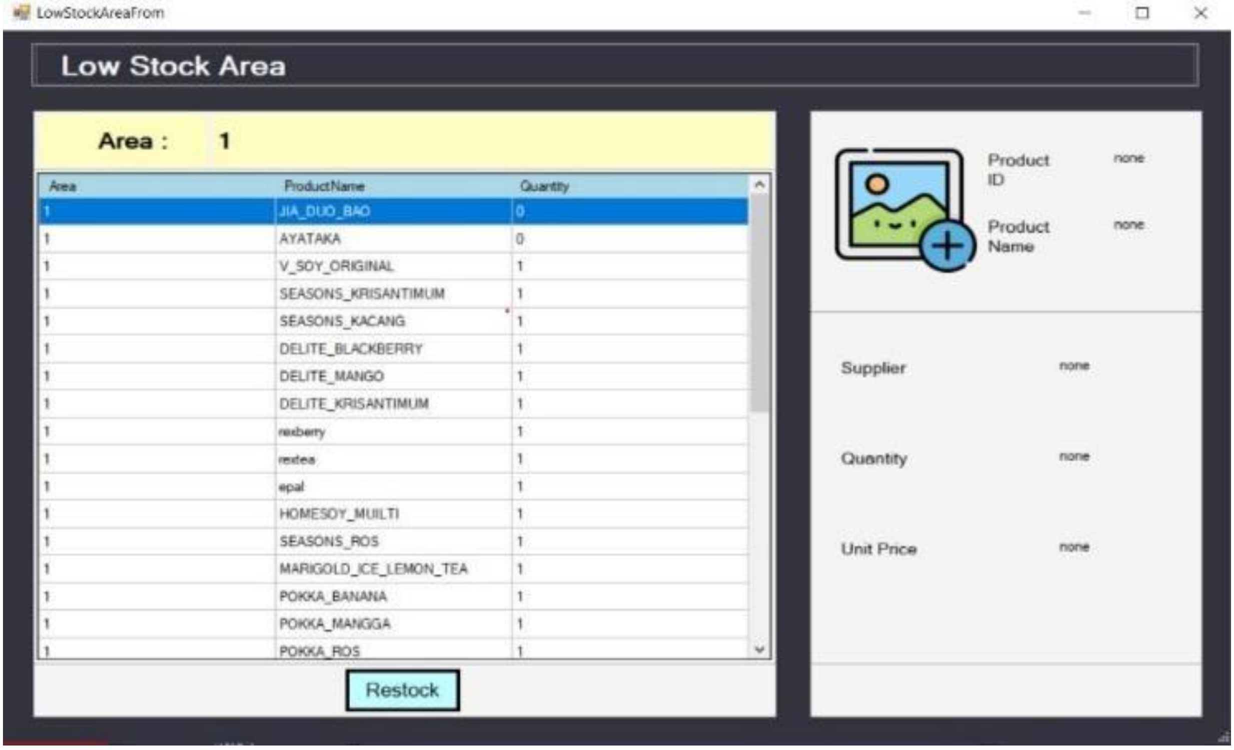

Figure 7: Low Stock Reporting Component

Figure 8 shows a real-time computer vision monitoring dashboard integrating four critical surveillance feeds: two for pedestrian flow analysis at the Main and Back Entrances, tracking IN/OUT counts and individual locations, and two for parking management across different areas, which visually segment occupied (red) and available (green) spaces with precise occupancy statistics. The unified interface, accessible via the CCTV sidebar selection, demonstrates an effective application of object detection algorithms for simultaneous crowd counting and dynamic vehicle parking resource allocation, providing immediate operational intelligence.

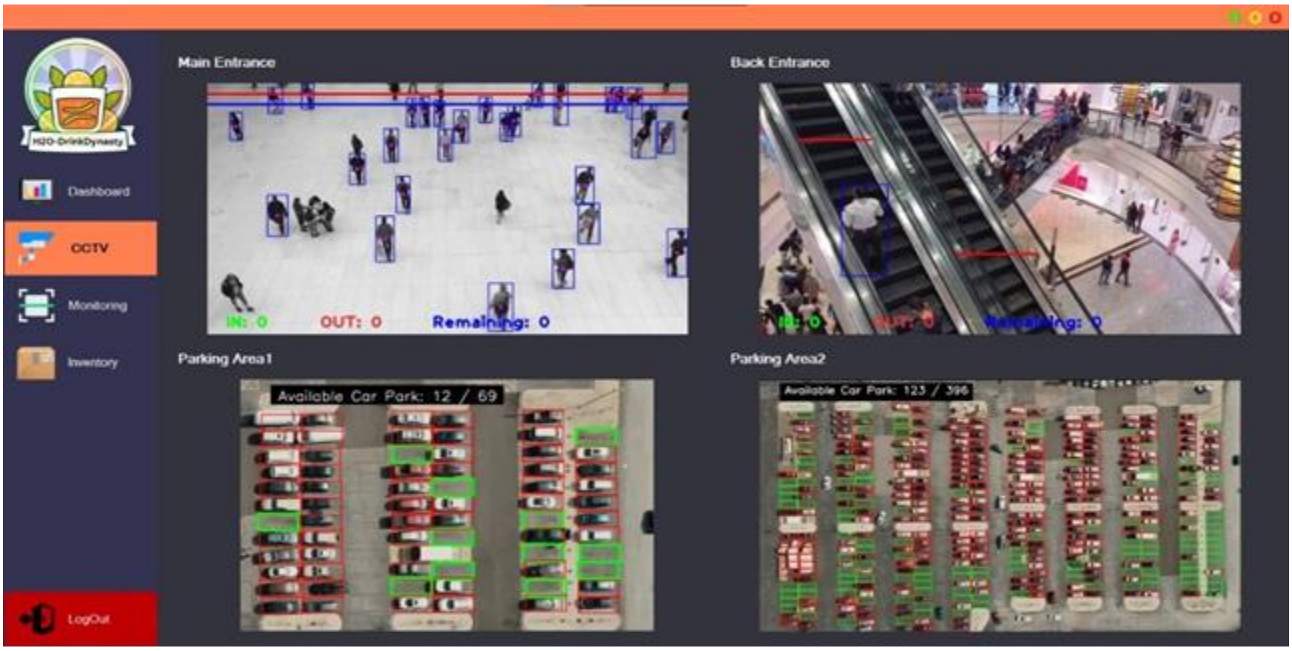

Figure 8: Customer Counting at the Store Entrance and Parking Lot Availability Counting at the Car Park Component

Figure 9 illustrates a computer vision retail shelf management focusing on a beverage rack. This allows operators to visually verify product placement, identify out-of-stock items, monitor planogram compliance, and ensure merchandise is correctly faced. It facilitates operational efficiency by streamlining inventory checks and reducing manual audit labor, providing immediate visual data on shelf conditions across multiple aisles.

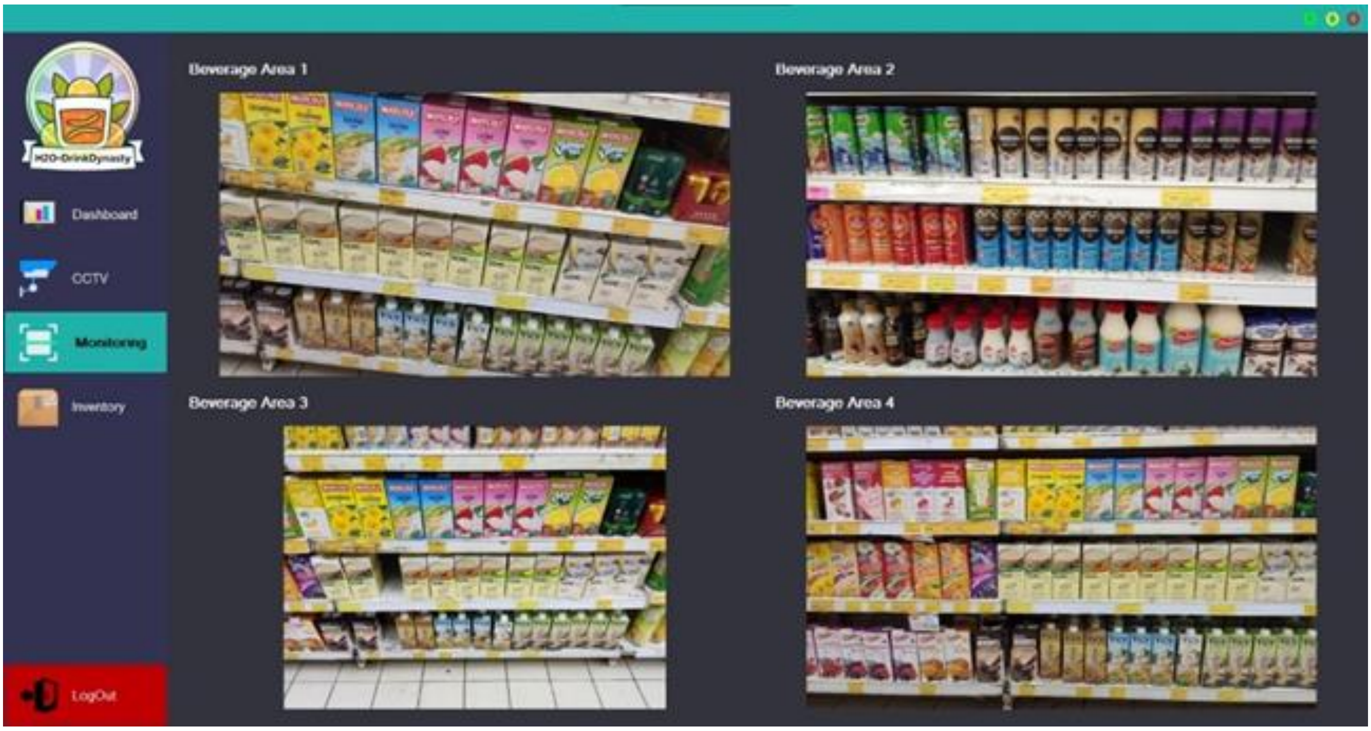

Figure 9: Shelf Product Quantity Counting Component

Figure 10 displays the Inventory Management interface designed for administrative oversight of product stock. This dedicated screen enables efficient data entry, stock auditing, and maintenance of the product catalog.

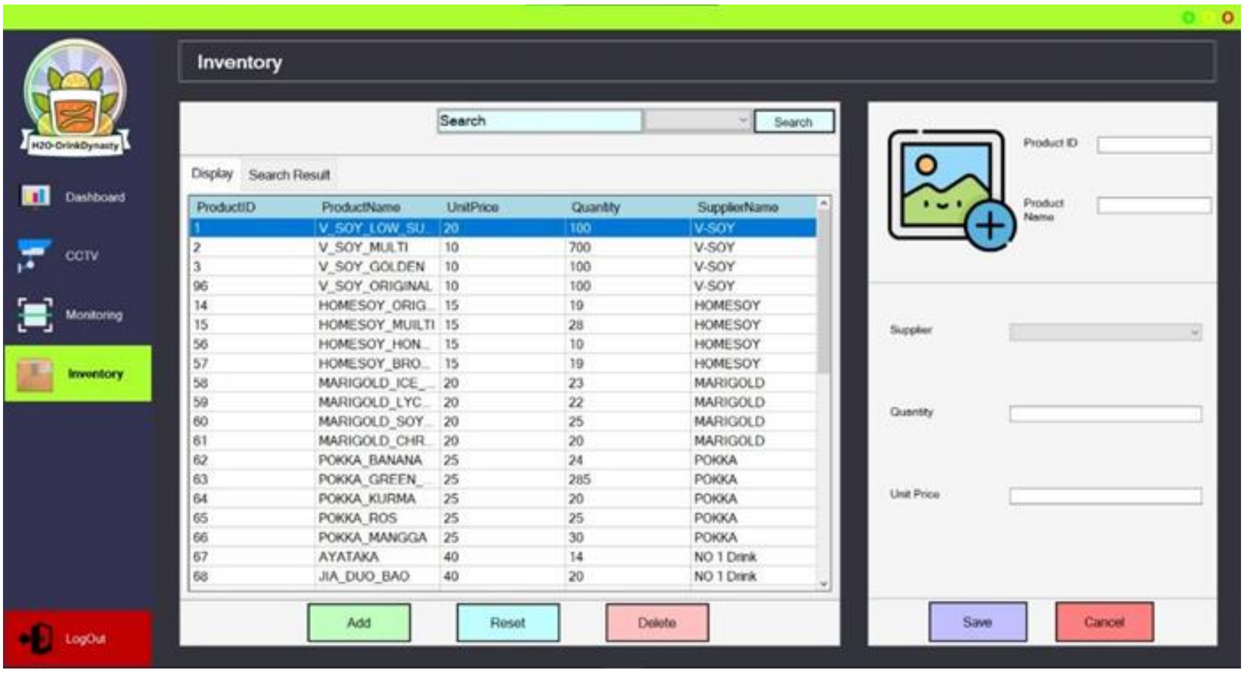

Figure 10: Inventory Management Component

YOLO Model Performance

The proposed YOLO model was trained using two datasets containing 30 and 60 objects. The performance of different YOLOv8 variants—YOLOv8n, YOLOv8s, YOLOv8m, and YOLOv8l—was evaluated. Results showed that larger models, such as YOLOv8l, which contain more hidden layers, achieved the highest average precision but exhibited slower inference speed and lower frames per second (FPS), requiring greater GPU resources. Conversely, YOLOv8n, with fewer hidden layers, offered the fastest processing speed and highest FPS but at the cost of lower detection accuracy. These trade-offs highlight the balance between precision and computational efficiency in selecting an appropriate.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume IX Issue XI November 2025

Figures 11 and 12 compare the performance of YOLOv8 models trained on 30 and 60 classes across different weight variants. Overall, there is minimal difference between the two class sets, except for the Nano models, which show slightly lower initial accuracy due to their limited hidden layers, making them less capable of handling larger class sets and greater variability. Table 3 reports the average precision for each model; precision generally decreases with 60 classes due to increased complexity, and smaller models exhibit larger performance drops. Considering the minimal performance difference between YOLOv8m and YOLOv8l, and the need to handle more classes in warehouse environments, YOLOv8m was selected for the 60-class Stock Detection model. YOLOv8m achieved an average precision of 0.950, recall of 0.98, and F1-score of 0.96, while offering faster processing and higher FPS than YOLOv8l, making it better suited for real-time stock detection and monitoring.

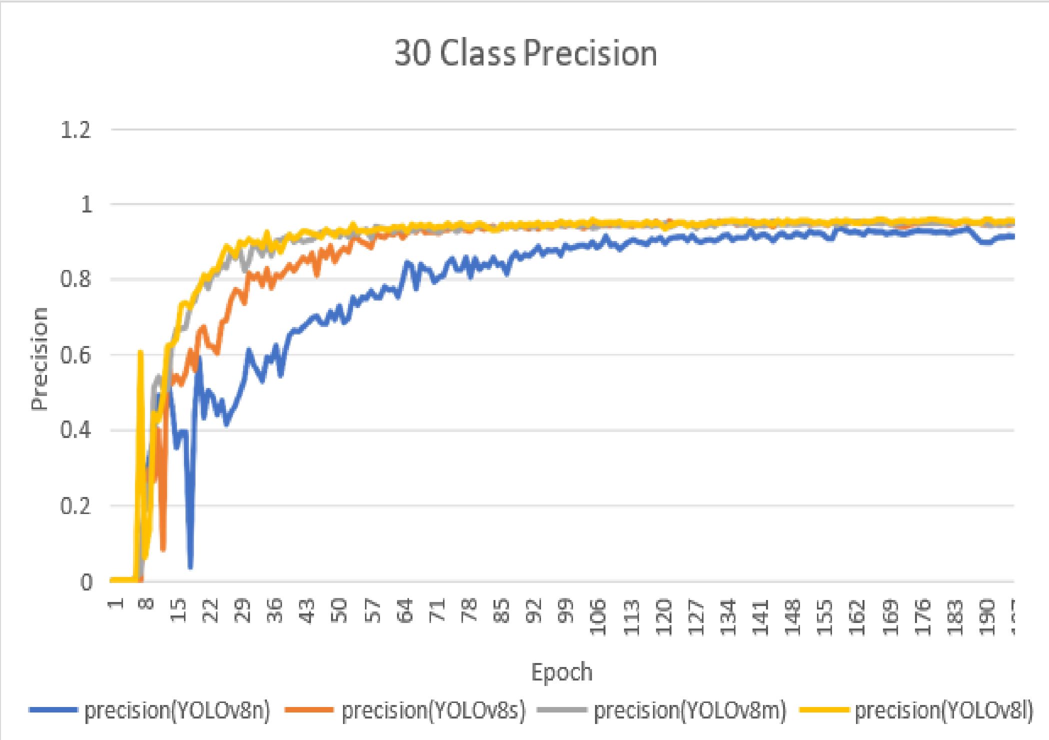

Figure 11: YOLO Model Performance on 30 Object Types

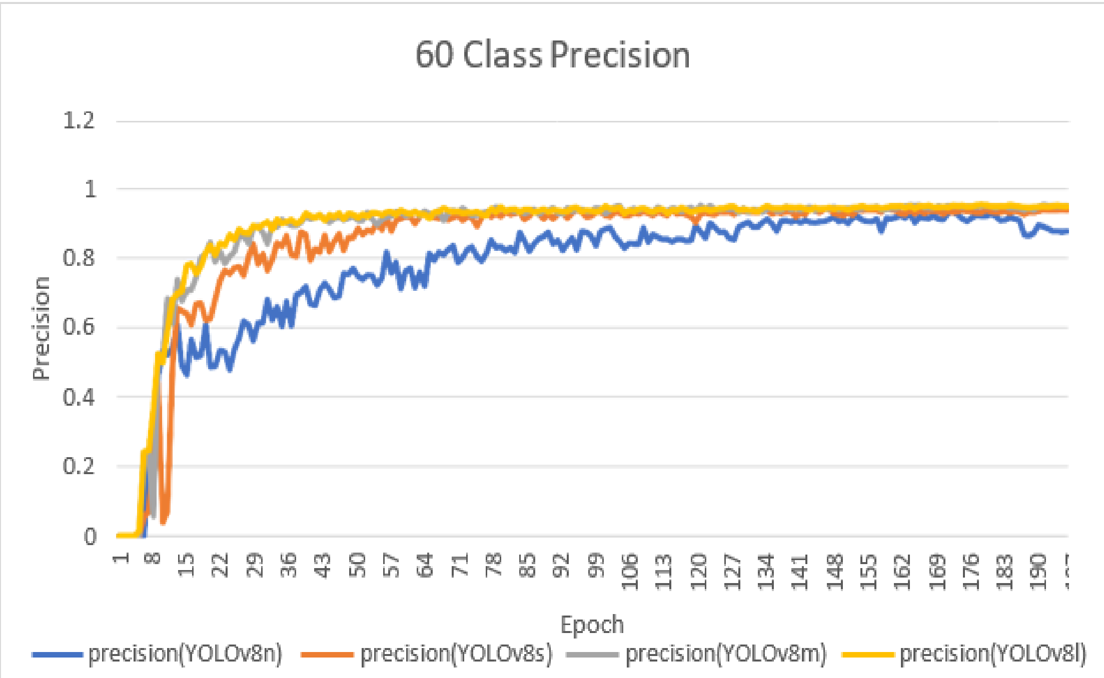

Figure 12: YOLO Model Performance on 60 Object Types

TABLE 3. Differences in YOLOv8 Model Average Precision with Different Numbers of Object Set

Weight / Class Set	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l
30 Objects Set	0.94309	0.95309	0.95569	0.95937
60 Objects Set	0.93173	0.9436	0.95027	0.95452
Percentage Difference	1.21974	1.00572	0.57036	0.50811

CONCLUSION

The **SMARTSTOCK** system represents a significant advancement in warehouse management by leveraging AI-based video recognition and real-time monitoring to address key operational challenges. Its AI module enables accurate detection and tracking of on-shelf products, client presence, and parking availability, providing reliable data for decision-making. The intuitive desktop interface allows staff to interact seamlessly with the system, enhancing productivity and simplifying routine inventory management tasks. Real-time tracking of stock levels improves operational efficiency, prevents Out-of-Stock situations, and facilitates timely replenishment, ensuring a consistent product supply. Despite these strengths, certain limitations exist. The system's performance is dependent on medium-to-high-performance hardware, which may introduce lag under high data loads. Additionally, product obstruction by clients can reduce the accuracy of stock detection, affecting OOS reporting and overall reliability.

Future enhancements can address these limitations and further expand **SMARTSTOCK**'s capabilities. Upgrading to high-performance or cloud-based infrastructure can offload computational demands, ensuring smooth real-time performance even during peak operations. Expanding the stock detection model into multiple smaller, area-specific models can improve detection accuracy while maintaining high frames per second (FPS) and processing speed. Integrating **SMARTSTOCK** with Point of Sale (POS) systems would allow inventory levels to be updated automatically in response to sales, providing actionable insights into high-demand products and enabling more efficient resource allocation.

Overall, **SMARTSTOCK** delivers a scalable, versatile, and innovative solution that enhances warehouse management for various retail environments. With continued development, optimization, and integration, it has the potential to become a primary tool for real-time inventory monitoring, operational efficiency, and effective decision-making across diverse industries.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to all individuals who contributed to the successful completion of this project. Special thanks are extended to team members and collaborators for their guidance, insights, and support throughout the research and development process. The authors also acknowledge the use of free open-source version of ChatGPT, an AI-based language model, which assisted solely in refining grammar, sentence structure, and overall clarity of the manuscript. All ideas, findings, figures, tables, and technical content presented in this paper are the original work of the authors and were not generated by ChatGPT. Its use was limited to improving the readability and presentation of the text.

REFERENCES

1. Gu, J., & Liu, F. (2022). A literature review of smart warehouse operations management. Frontiers of Engineering Management, 9, 31–55.
2. Budiyanto, A., & Muslim, M. (2024). Optimizing Inventory Systems with RFID: A Narrative Review of Integration, Efficiency, and Barriers. Sinergi International Journal of Logistics, 2(2), 133-146.
3. Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., & Ling, H. (2022). Detection and Tracking Meet

Drones challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 7380–7399.
4. Tepteris, G., Mamasis, K., & Minis, I. (2025). Logistics Hub Surveillance: Optimizing YOLOv3 Training for AI-Powered Drone Systems. Logistics, 9(2), 45.
5. Koteswara Rao, M., & Ashok Kumar, P. M. (2025). Advanced Object Tracking in Video Surveillance Systems with Adaptive Deep SORT Enhancement. Engineering, Technology & Applied Science Research, 15(2), 20871–20877.
6. Li, Y., Qu, L., Cai, G., Cheng, G., Qian, L., Dou, Y., ... & Wang, S. (2023). Video Object Counting with Scene-Aware Multi-Object Tracking. Journal of Database Management (JDM), 34(3), 1-13.
7. Maurya, A. S. (2023). Vehicle Detection in Autonomous Vehicles Using Yolov8. International Research Journal of Modernization in Engineering Technology and Science, 5(10), 3108-3113.
8. Mandal, V., & Adu-Gyamfi, Y. (2020). Object detection and tracking algorithms for vehicle counting: a comparative analysis. Journal of big data analytics in transportation, 2(3), 251-261.
9. Li, Y., Qu, L., Cai, G., Cheng, G., Qian, L., Dou, Y., ... & Wang, S. (2023). Video Object Counting with Scene-Aware Multi-Object Tracking. Journal of Database Management (JDM), 34(3), 1-13.
10. Zhang, Y., Li, L., & Qu, L. (2020). Video object counting with scene-aware multi-object tracking. IGI Global.
11. Ren, P., Fang, W., & Djahel, S. (2017, September). A novel YOLO-Based real-time people counting approach. In 2017 international smart cities conference (ISC2) (pp. 1-2). IEEE.
12. Fudholi, D. H., Kurniawardhani, A., Andaru, G. I., Alhanafi, A. A., & Najmudin, N. (2024). YOLO-based small-scaled model for On-Shelf availability in retail. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 8(2), 265–271.
13. Balasubramanian, S., & Feizi, S. (2023). Towards improved input masking for convolutional neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (pp. 1855–1865).