OPTIMIZED WAREHOUSE MANAGEMENT SYSTEM LEVERAGING INDUSTRY 4.0 TECHNOLOGIES

R25-62

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

OPTIMIZED WAREHOUSE MANAGEMENT SYSTEM LEVERAGING INDUSTRY 4.0 TECHNOLOGIES

R25-62

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

DECLARATION

I declare that this is my work. This proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning. To the best of my knowledge and belief, it does not contain any previously published material written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
P.A.S.Tharana	IT21822094	
Amangilihewa V.S.D	IT21318184	
Palihena P.D.M.P	IT21079672	
A.A.A.S.Abeydeera	IT21822780	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: ____________ Date 29/8/25

Signature of the co-supervisor: ____________ Date 29/08/25

ABSTRACT

Warehousing can significantly affect supply chain performance, but it can also be a source of inefficiencies, hazards, and poor use of space. This research presented a Comprehensive Warehouse Intelligence Framework, consisting of four modules (Route Optimization, Vision-Based Fire Detection, Stock Anomaly Detection and Worker Monitoring, and Space Optimization) that are integrated into a comprehensive smart warehouse approach.

The Route Optimization module uses Travelling Salesman Problem (TSP) sequencing combined with A* pathfinding to minimize distance travelled and permit the avoidance of physical obstacles within the warehouse environment in real-time. The fire detection system used YOLO-based computer vision to identify a fire in the early stages, compare individual shelves' proximity to fire, and predict how the fire could spread. The Optimized Warehouse Management System (OWMS) used predictive analytics (ARIMA, Prophet, LSTM) for stock anomaly detection, and classification of worker performance via machine learning. The Space Optimization module utilized the Best-Fit algorithm in combination with a 3D visualization of space constraints to maximize warehouse storage capacity and efficiency.

Evaluation by validating datasets and running simulation-based explorations of the warehouse delivered enhancements of routing efficiencies, accurate fire detection, anomaly forecasting, and enhanced space utilization. The results collectively demonstrate that this framework has the promise to improve efficiencies, better safety, and enhance adaptability to a supply chain environment, consistent with Industry 4.0.

Keywords: Smart Warehousing, Warehouse Intelligence Framework, Route Optimization, A* Pathfinding, YOLO-based Fire Detection,Predictive Analytics (ARIMA, Prophet, LSTM), Worker Performance Monitoring, Space Optimization,

ACKNOWLEDGEMENT

First I would like to express my deepest gratitude to my supervisor, **Prof. Samantha Rajapaksha**, for his continuous guidance, encouragement, and invaluable feedback throughout this project. I also wish to thank **Dr. Dinuka Wijendra** for his support and constructive suggestions that helped refine this work. Finally, my sincere appreciation goes to the **academic staff of SLIIT – Faculty of Computing, Department of Information Technology**, for providing the knowledge, resources, and infrastructure necessary to complete this research.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network – a type of deep learning model commonly used for image classification and detection.
YOLO	You Only Look Once – a real-time object detection algorithm that processes images in a single pass through the network.
YOLOv5	A popular and optimized version of the YOLO model for object detection, known for its speed and accuracy.
OpenCV	Open Source Computer Vision Library – a popular library of programming functions used for real-time computer vision.
Flask	A lightweight Python web framework used to develop web applications and APIs.
RNN	Recurrent Neural Network – a class of neural networks used for sequence prediction, such as video or time-series data.
IoT	nternet of Things – a network of physical devices embedded with sensors, software, and connectivity to collect and exchange data.
Bounding Box	A rectangular box that encloses an object in an image, used in object detection to localize and label targets.

1 List of abbreviations

1. INTRODUCTION

Warehousing has evolved into a crucial strategy within supply chains and has a meaningful influence on many factors such as efficiency, cost, customer service, and agility. As international trade expands, warehouses are transitioning from a passive storage facility role to an active logistic operations hub, where it is necessary to receive, store, pick, pack and dispatch goods while balancing increasing time and accuracy demands. Furthermore, e-commerce and just-in-time approaches have only increased warehousing demand for smart warehouses that can flex and agilely respond to demand persistence and complexity from order fulfillment [2], [26]. At the same time, warehouse operators are accountable for new challenges, including labor shortages, rising operating costs, and safety (i.e. fire hazards and congestion in the workspace) [3], [11].

While traditional Warehouse Management Systems (WMS) have evolved to automate key tasks such as stock keeping, order processing, and reporting; most systems are still siloed and functionally specific without a common intelligence element [7]. For instance, a system that optimizes order picking, is likely not to include safety components such as fire detection; or a monitoring system for worker performance that is dissociated from inventory anomaly detection. This lack of integration can tide the warehouse operators from operating as a holistic and cohesive data-based business ecosystem.

The emergence of Industry 4.0 technologies, including artificial intelligence (AI), computer vision, predictive analytics, and cyber-physical systems, presents opportunities to rethink warehousing as more intelligent, adaptive, and predictive environments [26], [27]. The ability to compute dynamic route optimization methods using the Travelling Salesman Problem (TSP) or A** pathfinding allows users to specify, for example, what they want to pick, keeping in mind the distance traveled, while dynamically avoiding real-time obstacles during traversing [2]. For example, YOLO computer vision techniques allow users to do rapid fire detection and risk prioritization, which can enhance safety [11], [14]. Predictive analytics techniques such as ARIMA, Prophet, and LSTM models, can provide anomaly detection on stock movements, as well as recommendation to improve demand forecast accuracy [22]. Best-Fit Bin Packing algorithms leveraged with 3D visualization tools help users not only maximize cubic meter (CBM) capacity, but also add transparency to how space is utilized [3], [4].

To address these gaps, this thesis proposed a Comprehensive Warehouse Intelligent Framework that incorporates the four connected modules of Route Optimization, Vision-based Fire Detection, Stock Anomaly Detection with Worker Monitoring, and Space Optimization. Unlike traditional systems which have distinct siloed constructs, this Framework represents a more holistic, intelligent platform which will help users to increase efficiency, improve safety, be predictive in decision-making, and optimize key space simultaneously. By blending algorithmic optimization, computer vision, machine learning and visualization, this framework advances the evolution of next-generation smart/automated warehouses in conjunction with Industry 4.0 [26], [27].

1.1 Background

Warehousing is a central component of logistics and supply chain management. It connects producers with distributers. Today, a warehouse is no longer thought of as a static storage space, but instead a very active space of operations in which inbound goods are stored, turned around quickly, and successfully sent off to meet ever-increasing demands. The ubiquitous e-commerce era and the rise of just-in-time (JIT) and on-demand delivery have created new expectations for warehouses to act and operate with speed, scalability and accuracy [2, 26].

Despite the improvements, traditional warehouses face many challenges:

- Operational inefficiency: Picking orders takes up to 50% of the warehouse costs, workers take unnecessary steps when planned poorly [2]. Without an effective routing system, productivity is negligible, and labour costs rise.

- Safety concerns: Fire in warehouses can destroy everything. Recent fire detection technologies including smoke detection and heat sensors are flawed by slow detection times and low sensitivity leading to delays in reaction time [11], [13].
- No predictive capabilities: Most warehouse management systems (WMS) are basically reactive. They maintain inventory records but cannot predict deviations and future demand [22].
- Space, poor use of: Weight and space strategies are ill-defined and without tools to help with space allocation typically lead to clogged aisles and loss of cubic meter capacity (CBM) [3], [4].

Industry 4.0 technologies have provided a new pathway for addressing these challenges. Algorithmic optimization like the Travelling Salesman Problem (TSP) with A** pathfinding allows warehouses to create the best path for picking orders in a dynamic situation where obstacles like workers and forklifts alter the planned path [2], [7]. Computer vision with YOLO models not only can aesthetically assess fire severity at earlier stages, it quantifies the distance the fire may spread to shelves, assesses fire spread direction, and provide the administrator time to alert or even respond before it grows any worse [11], [14]. Predictive analytics with ARIMA, Prophet, and LSTM to improve demand forecasting and anomaly detection of stock movement enables managers to act before performance drops [22]. At the same time, monitoring workers performance with machine learning classification promotes transparency and healthy evaluation of human factors. Finally, the deployment of Best-Fit bin-packing algorithms and 3D visualization for intelligent place ensuring maximum efficiency and visibity of the warehouse floor lay out [3], [6].

1.2 Literature Survey

In warehouse management literature there is a large body of work dealing with performance issues related to efficiency, safety, predictive intelligence, and spatial efficiency, or warehouse utilization. However, most literature has addressed these areas independently from one another, with limited attention to solutions that cut across space, safety, efficiency, and predictivity where possible. This section will review important contributions in the area of order picking and route optimization, vision-based fire detection, predictive analytics in stock and workforce, space optimization and visualization, and Industry 4.0 in warehousing.

1.2.1 Order Picking and Route Optimization

Picking orders is recognized as the most costly function in warehousing, costing as much as 50% of warehouse costs [2]. And the literature suggests that shortened travel distances are one way to improve productivity. De Koster et al. [2] provided a comprehensive study of modes of order picking strategies with a specific focus on the value of algorithmic optimization. Static routes are more common and inefficient, as they allow the duplicable travel to be made as multiple trips as possible, where heuristic and metaheuristic solutions of Travelling Salesman Problem (TSP) diagonal ordering can simplify travel paths.

Pathfinding such as A* and Dijkstra's are the most well-known path-finding algorithms and are often used in shortest path problems [7]. While in optimal pathing situations like applying TSP ordering, A* has benefited from both efficient routes and adaptability in dynamic scenarios where movement by workers or fork-trucks can affect existing routes with unexpected obstacles. New capabilities in the domain of pathfinding include Ant Colony Optimizations (ACO) and Reinforcement Learning (RL) for adaptive pathing, allowing real-time adaptations to situational variables, however, this presents challenges within warehouse design as every change to warehouse design impacts systems functionally [17]. These works prove that algorithmic optimization is important for efficient operations but show little operational fit with other subsystems of the warehouse such as safety or prediction verification.

1.2.2 Computer Vision for Fire Detection and Safety

Fire safety in warehouses typically employs smoke alarms (detectors), sprinklers, and thermal sensors; while they are effective fire safety devices, they often result in delayed detection (assuming a fire occurs post-smises) or false positives. Fire detection via computer vision marks a shift away from passive and reactive devices;

Jin et al. [11] reviewed deep learning-based fire recognition in video with new state of the art accuracy through video methodology unlike traditional smoke alarms. Xu et al. [14] presented Light-YOLOv5, which is a lightweight object detection framework designed for fire detection in complex environments. Islam and Habib [15] also succeeded using YOLOv5 for video-based detection and had real-time detection and followed by sensitivity.

Further research presented enhancements that extend detection to include fire size classification, extended distance analysis, and fire spread prediction that provides relevant and actionable context for administrators [20],[21]. Research done on shelf monitoring systems [17], [19] highlighted the value of the video system to provide up-to-date knowledge of the warehouse as tied to fire detection, which empowers administrators to still validate a low-risk scenario by tiered alerts on critical context, and prioritization upon arrival. Despite many contributions to this field of research, many current vision systems are still assessed on their own merit and lack integration into warehouse management as a system..

1.2.3 Predictive Analytics for Stock and Workforce Monitoring

There is also increasing interest in forecasting and anomaly detection within warehouse operations when desirable supply chains operate on a more data-driven basis. While mainstream time series forecasting methods like ARIMA remain prevalent in detecting demand variability and irregularities in stock flow [22], there are also emerging alternatives above mainstream ARIMA including Facebook Prophet as well as the LSTM (Long Short-Term Memory) deep learning models, which are offering superior accuracy in predicting non-linear and seasonal load data patterns [22].

There are multiple aspects to workforce monitoring. For example, Gunasekaran et al. [22] indicated how human factors are important and were quite relevant to the programmatic efficiencies of the supply chain, and explained how modeling on a predictive scale could enhance labor utilization. Recent studies have applied machine learning classification of workers based on their performance levels in areas such as items picked, error counts, and on time task completion rates [24]. For example, categorically gradient boosting and other classifiers offer (roughly) reliable classifications of worker performance into high, average and low bands. Unfortunately, predictive methods often act in isolation from routing, safety and space optimization; therefore, their applicability is limited for systemic considerations.

1.2.4 Space Optimization and Visualization

Using warehouse space efficiently is key to facilitating work and minimizing costs. Singh and Sharma [3] explored bin-packing and visualization strategies for space allocation to improve total cubic meters employed. Martello and Toth [4] considered formulation and methodologies for the knapsack problem, providing algorithmic options for product placement management. Best-Fit heuristics, as an example, have found success at solving for balance between computational speed and quality of solution [5].

As well as using algorithms to successfully place products, visualization technologies have become more frequently adopted to increase managerial transparency. Bortolini et al. [6] explored visualization through virtual simulation to analyze packaging and layout strategies, showcasing the ability to use visualization over time as a cost-cutting strategy. The calling of both visualization and optimization algorithms allow managers to identify wasted shelving and aisle congestion/narrowness as a reallocation opportunity. While most of the literature on space optimization has addressed the narrow aspect of storage, few studies considered incorporating routing options, safety protocols, or predictive analytics.

1.2.5 Industry 4.0 and Smart Warehousing

The overall context of these developments is based on Industry 4.0, which highlights the prospects of digitalisation, predictive analytics, and intelligent automation. Waller and Fawcett [26] have identified predictive analytics as a revolution that fundamentally questions supply chain design and decision-making. Hofmann and Rüsch [27] discussed how Industry 4.0 technologies (IoT, Cyber-Physical Systems, Big Data, etc.) will follow on from other developments as the basis for future logistics systems.

In warehousing, Industry 4.0 espouses a holistic approach that combines disparate functions into cohesive, intelligent ecosystems. Study of the individual subsystems has separately continued to advance (routing, equation fire detection, analytics, space utilization, etc.), but research has not progressed to an integrated framework that can encompass the opportunity to think of efficiency, safety, adaptability, and intelligence in one model. In essence, this integrated framework represents the novel contribution of this paper.

1.3 Research Gap

The literature on warehousing indicates steady progress in maximizing single processes, such as routing, safety, inventory forecasting, and space utilization. However, there are observable limitations to research and practice when considered holistically. The subsequent discussion provide the key gaps identified in five thematic themes and the wider perspective of Industry 4.0 integration.

1.3.1 Gaps in Route Optimization

Order picking, historically viewed as the most costly task in warehousing [2]. Routes minimizing travel distances and improving efficiency have been implemented in autonomous systems by algorithms for the Travelling Salesman Problem (TSP) and A* pathfinding [2], [7]. More advanced operations such as Ant Colony Optimization and Reinforcement Learning improve situational adaptability in dynamic environments in real time [17].

Limitation: Still, route optimization models are commonly tested in simulated environments and not adapted for real-time interaction with other warehousing subsystems, finding an ideal path without consideration of interactions with space constraints, worker effectiveness or emergencies such as fire.

1.3.2 Gaps in Fire Detection and Safety

In the past few years, computer vision–based fire detection has gained significant momentum, especially using YOLO models [11], [14]. These detection approaches provide faster and more accurate detection than traditional smoke or heat sensors. Some systems extend the detection further to fire size estimation and spread prediction [20], [21].

Limitedness: Most fire detection systems are standalone safety solutions that aren't part of WMS dashboards or other operational modules. They provide detection but are not decision support systems capable of devising response strategies when considering space layouts, worker locations, and product routes.

1.3.3 Gaps in Predictive Analytics for Stock and Workforce Monitoring

In predicting demand variation and detecting irregular stock movements, forecasting and anomaly detection with ARIMA, Prophet, and LSTM showed usefulness [22]. Workforce analytics have been transformed by machine learning classifiers that estimate worker productivity and errors on the job [24].

Limitation: Most predictive approaches are independently implemented that only capture either stock movement or workforce monitoring, but not both. Moreover, those approaches hardly reviewer back to decisions on routing or space utilization, leaving intelligence unconnected across subsystems.

1.3.4 Gaps in Space Optimization and Visualization

Best-Fit heuristics, Knapsack problem formulations, and bin-packing models are established algorithms that have successfully maximized CBM utilization [3], [4], [5]. 3D visualization tools enhance managerial visibility by providing real-time awareness of space utilization [6].

Limitations: These solutions are static, primarily focused on initial warehouse design as opposed to numerous potential and continuous dynamic adjustments occurring through incoming sales orders, logistics routing changes, and safety mitigation considerations. These solutions do not interact, and make decisions in the same context and time frame as predictive analytics and fire detection systems.

1.3.5 Integration and Industry 4.0 Context

Industry 4.0 accentuates the merging of IoT, AI, and cyber-physical systems into integrated logistics networks [26], [27]. Although there are advances in routing, vision, analytics and space in their own right, most frameworks do not combine all of these into a cohesive warehouse intelligence system. Most research also tends to focus on individual optimizations creating a highly fragmented intelligence space, limiting the ability to adapt to actual complexity.

Area	Existing Work	Limitation	Gap Addressed by This Study
Route Optimization	TSP, A*, ACO, RL used for efficient order picking [2], [7], [17]	Focused on path efficiency; limited real-time integration with other subsystems	Integrates routing with space, worker data, and safety alerts in real-time
Fire Detection & Safety	YOLO-based fire recognition, fire size & spread prediction [11], [14], [20]	Standalone systems; lack integration with WMS dashboards	Combines fire detection with layout data and predictive modules for holistic safety
Stock & Workforce Monitoring	ARIMA, Prophet, LSTM for stock; ML classifiers for worker performance [22], [24]	Applied separately; no link to routing or space	Integrates predictive analytics for both stock and workers in one dashboard
Space Optimization	Best-Fit, Knapsack, bin-packing, 3D visualization [3], [4], [6]	Static models; lack of dynamic adaptation	Dynamic optimization linked to routes, safety, and real-time data
Industry 4.0	Emphasis on IoT, AI, predictive logistics [26], [27]	Limited holistic integration of modules	Provides unified Industry 4.0–aligned intelligence framework

Figure 1 Research Gap

1.4 Research Problem

Warehousing has shifted from a passive storage function to a vital supply chain differentiator. With the growth in e-commerce, globalization, and just-in-time delivery processes, warehouses will be pressed to perform to an unprecedented level of accuracy, flexibility, and efficiency [2], [26]. But even with technological advancements, four key issues remain open in the area of route optimization: safety, predictive analytics, and space utilization

The most pressing identified problem is inefficient order picking. The costs of order picking alone may account for almost fifty percent of warehouse costs [2]. Existing navigation solutions that rely on TSP- and A*-based routing still advance travel distance but are highly focused on isolated aspects of the warehouse and fail to account for real-time obstructions, operator performance, and situational awareness of broader systems within the warehouse [7], [17] causing warehouses to suffer from unnecessary travel redundancies, congestion, and the inability to fulfill customers' orders in a timely manner.

Safety is another pressing issue. Fire detection systems have been reactive to date, mostly relying on smoke and heat sensors which act too slow to stop the escalation of a fire. Computer vision has been explored as a viable alternative [11], [14], but existing systems are deployed as separate safety modules. They can detect fire, but they do not connect to warehouse layouts, routing systems, or worker monitoring dashboards to improve situational awareness and operational impact during emergencies.

Further, the lack of predictive intelligence in warehouse systems does not help matters. Although forecasting models like ARIMA, Prophet, and LSTM have been deployed for stock anomaly detection [22], they have not been employed in real-time WMS applications. Similarly, workforce monitoring systems that exist have mainly focused on attendance or counts of tasks completed as KPIs [24]. The fragmented system prevents the manager from connecting stock anomalies with worker performance, negating opportunities for targeted intervention, training, or efficiency improvement.

Space utilization is another unresolved challenge. Warehouses often have cramped aisles, wasted cubic meter (CBM) space, and poor layouts. Best-Fit bin-packing algorithms and 3D visualization tools have been suggested [3], [4], but they are typically only used when designing warehouses, and not for real-time dynamic operations. This doesn't allow for any modifying, to adapt to changes in demand pattern, fire hazard extent, or route degree changes.

The issue, therefore, isn't just the three. The lack of inter-facility connectivity creates unmanaged issues. Modern warehouse technologies that exist are treated as siloed technologies. They focus on one aspect of efficiency, one aspect of safety, or one aspect of prediction. However, they do not provide a level of intelligence that could link-up across the facilities. Thus, allowing facilities to perform in an manner that makes them sustainable as well as adaptable. The governed risk needs to be predictable, the process adaptive, and warehouse off putting are definitely challenging to resolve to meet the variables of the dynamic supply chain.

Thus, the research problem can be stated as follows:

"How can a unified warehouse intelligence framework be designed and implemented to integrate optimization, safety, predictive analytics, and space utilization into a single, scalable system that enhances efficiency, ensures safety, supports predictive decision-making, and aligns with Industry 4.0 standards?"

By addressing this problem, the proposed research aims to fill a critical gap in warehouse management by moving beyond isolated optimizations and delivering a holistic solution that transforms warehouses into intelligent, adaptive, and resilient nodes within modern supply chains

1.5. Objectives

The goal of this mistreatment is to develop and implement a Comprehensive Warehouse Intelligence Framework that combines optimization, safety, predictive analytics, and space utilization in one system. The Comprehensive Warehouse Intelligence Framework will use algorithms, computer vision technology, predictive modeling, and visualization technology to create intelligent, adaptable, and Industry 4.0- compliant warehouse ecosystems. The objectives are divided into two levels: the overall Main Objective and a group of Specific Objectives which provide a more granular breakdown of the research scope into specific deliverables.

1.5.1 Main Objective

The primary objective of this project is:

"To create and validate an integrated warehouse intelligence framework that combines routing, vision-based fire detection, stock awareness anomaly detection using worker awareness, and space optimization into a single, scalable system. The goal is to provide efficiency, safety, a predictive intelligence to operational decision-making and resource management, with Industry 4.0 in mind."

This primary aim recognizes the need for an integrated warehouse solution, moving past isolated solutions that address warehouse efficiency, safety risk, predictive intelligence, and resources utilization independently.

1.5.2 Specific Objectives

To achieve the main objective, the research is structured around the following specific objectives:

1. **Develop a Route Optimization Module**

- Implement **TSP sequencing** and **A*** pathfinding to generate efficient picking routes.
- Ensure adaptability to real-time obstacles, such as moving forklifts or blocked aisles, by simulating dynamic routing within the warehouse environment.

2. **Design a Vision-Based Fire Detection and Prevention System**

- Employ **YOLO-based deep learning models** for early fire recognition, fire size classification, and shelf proximity analysis.
- Integrate fire spread prediction mechanisms to enable proactive risk management and safety interventions.

3. **Implement a Predictive Analytics Framework for Stock and Worker Monitoring**

- Apply **ARIMA, Prophet, and LSTM** models to forecast stock movements and detect anomalies such as shortages, surpluses, or misplaced items.
- Utilize **machine learning classification** techniques (e.g., Gradient Boosting) to evaluate worker performance based on activity logs, productivity rates, and error counts.
- Integrate stock anomaly insights with worker monitoring to provide managers with actionable intelligence.

4. Develop Space Optimization and Visualization Module

- Utilize **Best-Fit bin-packing algorithms** to maximize cubic meter (CBM) utilization in dynamic warehouse environments.
- Incorporate **3D visualization tools** to provide real-time visibility of warehouse layouts, highlighting congestion, underutilized zones, and optimization opportunities.

5. To combine all modules into one, deployable, real-time vision-based monitoring system

- Build an integrated **dashboard interface** that combines routing, safety alerts, predictive insights, and space optimization.
- Ensure real-time data synchronization, scalability, and adaptability to Industry 4.0–based warehouse practices.

6. Evaluate the System Through Experimental Validation

- Build an integrated **dashboard interface** that combines routing, safety alerts, predictive insights, and space optimization.
- Ensure real-time data synchronization, scalability, and adaptability to Industry 4.0–based warehouse practices.

2. METHODOLOGY

This paper employed a design and implementation-based methodology towards the conception and development of an integrated warehouse intelligence framework, integrating four fundamental modules: route optimization, vision based fire detection, stock anomaly detection with worker retarding, and space optimization. Throughout the process of developing the integrated framework, the initial steps involved obtaining requirements to identify challenges within common warehouses related to long picking routes and routes that were delayed because of delayed fire alarms, reactive measurements for observing stock anomalies and inefficiencies in regards to the volume of space in the warehouse. After firstly being able to understand each of the hard and soft requirements through the problems in conventional warehouses, then each of the modules could be designed and tested individually before creating an integrated framework for the integrated approach that was proposed. The Route Optimization module used a Travelling Salesman Problem (TSP) with A* pathfinding so that a best route could be calculated whilst maintaining a level of adaptability when variables increased within the existing environment. The Fire Detection module utilized computer vision models using YOLO to identify a fire when it was early in its development, the module was able classify the severity level, the proximity to shelving within the warehouse, and spatial characteristics to predict fire spread patterns. The Stock and Worker Monitoring module used predictive analytics models (e.g., ARIMA, Prophet, LSTM) to identify anomalies with stock flows, alongside machine learning classifiers (e.g., Gradient Boosting), to evaluate a worker's performance based on activity logs.

The Space Optimization module featured Best-Fit bin-packing algorithms, included 3D visualization tools to potentially improve cubic meter (CBM) space usage and identify congested or underused areas. These subsystems were developed with a MERN + Flask stack, using MongoDB as the database and React.js as the visualization interface, and combined into a unified dashboard that could provide real-time monitoring, predictive analysis, and decision support. The SOT would require experimental validation using synthetic simulation datasets and case-based testing that could measure improved route efficiencies, improved anomaly detection accuracy and time to fire recognition, and improved warehouse space utilization. Modularity, scalability, and flexibility were emphasized within the method options to provide the proposed framework with both academic stringency and commercial viability within the context of Industry 4.0.

2.1. System overview

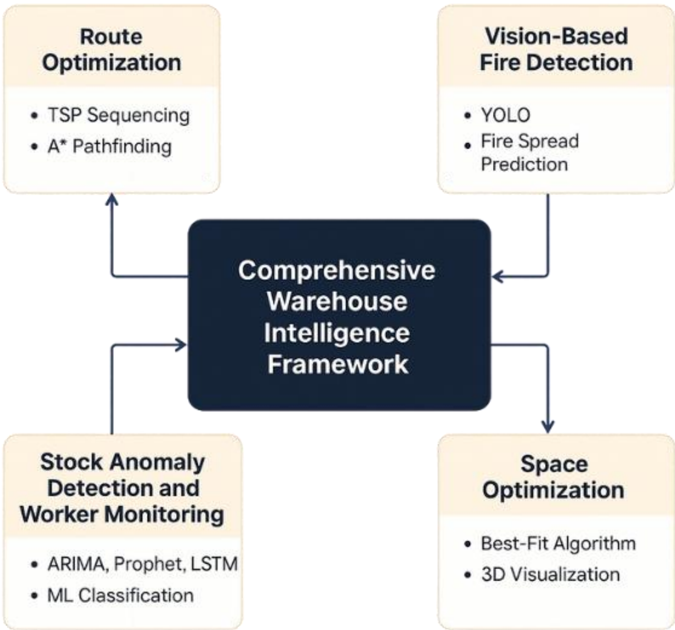


Figure 2 System Overview Diagram

The system architecture shows the workflow of the proposed Comprehensive Warehouse Intelligence Framework, which comprises four modules that run in harmony via a unified dashboard. The first step occurs when authorized users log in through a secure authentication interface. After authentication, the users see a dashboard that acts as the main control interface for visualization and interacting with all subsystems.

The **Route Optimization** module receives both order and product related information, applies Travelling Salesman Problem (TSP) sequencing with A** pathfinding, and then develops the optimal product picking routes. The module will update the routes dynamically if it detects aisle congestion or obstructions. Similarly, the Vision-Based **Fire Detection module** receives a live video feed, on which it follows a YOLO-based deep learning application to detect fire, proximity to shelves, and likely spread of flame. If fire is detected, the fire detection module sends alerts to the dashboard and automatically recalculates the routes to avoid hazardous areas.

The Stock Anomaly and **Worker Monitoring modular** analyze transaction logs using ARIMA, Prophet, and LSTM to determine normal flows and potentially anomalous flows while machine learning classifiers use activity records to determine worker productivity. All of these details are exposed on the dashboard which managers can see the true health of their operations.

Lastly, the **Space Optimization** module uses a Best-Fit algorithm along with 3D visualization to maximize cubic meters, decrease congestion and show unused storage areas.

All four modules are being executed together on a **MERN + Flask stack**. MongoDB on the backend manages data, Flask serves machine learning models and React.js serves as the user interface. Combined, the modules are creating a dynamic warehouse intelligence system which will improve efficiency, safety and decision-making within the premises of Industry 4.0.

Component	**Technology / Framework**
Frontend (Dashboard UI)	React
Backend	Express
Database	MongoDB
Computer Vision Library	OpenCV (image processing, contour analysis)
Deep Learning Models	CNNs, YOLOv5 (fire & shelf detection), U-Net / RNN (fire spread prediction)
Frameworks	TensorFlow, PyTorch (for training & inference)
Deployment	Local Server / GPU-supported PC (edge deployment)
Optional Cloud Hosting	AWS / Google Cloud (for scalability)

2 Technologies and frameworks

2.2 Commercialization

The project is extremely commercially viable in the market for warehouse management and supply chain optimization. This project meets a growing appetite for improved efficiency, safety, predictive intelligence, and space utilization in today's warehouses, particularly in relation to Industry 4.0. The proposed framework is a complete, modular, and economical solution that can be implemented with existing infrastructure and minimal hardware investment, rather than utilizing a disjointed system that works on one function at a time. It also relies on open-source technologies, provides real-time AI analytics, and is modular in design, making it both scalable and appealing to a wide spectrum of enterprises.

1. **Target Market:**

The initial target customers include:

- **Warehouses and distribution centers** (e-commerce, retail, logistics, and manufacturing).
- **Third-party logistics providers (3PLs)** seeking scalable solutions.
- **Pharmaceutical and cold storage facilities** requiring high levels of accuracy and safety.
- **Government and defense storage depots** where efficiency and hazard prevention are critical.
- **Medium to large enterprises** with existing CCTV, ERP, or WMS infrastructure that can integrate easily.

2. **Unique Selling Points (USPs):**

- **Integrated Framework**: Combines route optimization, fire detection, anomaly forecasting, and space utilization in one system.

- **Real-Time AI Analytics**: Delivers instant detection, prediction, and adaptive decision-making through deep learning and forecasting models.
- **Modular Design**: Businesses can start with a single module (e.g., route optimization or fire detection) and expand to the full suite.
- **Cost-Effective**: Built on open-source tools, with minimal additional hardware required.
- **Non-Intrusive Deployment**: Integrates with existing WMS, CCTV, and warehouse infrastructure without significant downtime.

3. **Market Entry Strategy:**

- **Pilot Programs**: Offer free or discounted trials in medium-sized warehouses to build trust and gather testimonials.
- **Partnerships**: Collaborate with ERP/WMS vendors, CCTV providers, and fire safety consultants for faster adoption.
- **Online Marketing**: Launch a dedicated website with demos, ROI calculators, and case studies showcasing efficiency gains.
- **Industry Events**: Exhibit at logistics, supply chain, and fire safety expos to increase visibility.
- **Compliance and Certification**: Ensure alignment with fire safety standards, data protection laws, and warehouse operation guidelines to strengthen credibility.

4. **Scalability and Expansion:**

- **Cross-Domain Applications**: Expand beyond warehouses into **hospitals, schools, airports, and shopping malls** for fire detection and efficiency monitoring.
- **Multilingual and Regional Customization**: Adapt the system for different markets with localized dashboards and alerts.

- **Cloud + Edge Deployment**: Offer both on-premises and cloud-hosted solutions depending on client size and budget.
- **Mobile Integration**: Provide real-time alerts, KPI tracking, and fire hazard notifications via mobile apps for managers on the go

2.3 Testing & Implementation

The implementation and testing of the Comprehensive Warehouse Intelligence Framework was designed to be implemented in multiple stages, utilizing frontend implementation, backend implementation, model & algorithm implementation, and validation/testing. Each stage had the goal to validate that the system was operational, correct, and aligned with all dimensions of efficiency, safety, predictive intelligence, and space utilization.

2.3.1 Frontend Implementation – React (Administrator Panel)

The admin UI is a **React** SPA organized by pages and shared components. It consumes two backends:

- **Flask** – MJPEG video stream (/video) + status JSON (/status).
- **Node/Express** – auth, users, and alert/event logs.

Key pages & components

- **Route Visualization**: Displays optimized routes and updates dynamically when obstacles or fire alerts are detected.
- **Fire Detection Alerts**: Real-time bounding boxes and confidence scores from the YOLO model are overlaid on camera feeds.
- **Stock Forecast Graphs**: Line charts showing actual vs. predicted stock, with anomalies highlighted.
- **Worker Monitoring Panel**: Visualizes worker performance categories (high/average/low).
- **3D Layout Visualization**: Provides a graphical view of CBM utilization and congested areas.

State & libraries

- **React Router** for navigation; **Context** for camera stream; **Axios** for API calls.
- **JWT** stored via **HttpOnly cookies**; role-based route guards.

- **Socket.IO** real-time data transferring.

router.jsx shelf-map.jsx ShelfMapPage.jsx shelf-opt.jsx shelfOptForm.jsx shelf.jsx

src > components > shelf > ShelfMapPage.jsx > ShelfMapPage

```jsx
import React, { useEffect, useMemo, useRef, useState } from 'react';
import axios from 'axios';
import BASE_URL from '../../config/apiConfig';
import { getToken } from '@/utils/token';

const CELL_SIZE = 140;   // px per grid cell (base, pre-zoom)
const GUTTER_LEFT = 48;  // px for Y-axis labels
const GUTTER_TOP  = 40;  // px for X-axis labels
const CELL_PAD = 10;     // inner padding per cell
const TOOLTIP_OFFSET = 12;

const ZOOM_MIN = 0.5;
const ZOOM_MAX = 3.0;

const ShelfMapPage = () => {
  const [shelves, setShelves] = useState([]);
  const [loading, setLoading] = useState(true);

  // zoom & hover
  const [zoom, setZoom] = useState(1);
  const [hovered, setHovered] = useState(null); // { shelf, x, y, box }
  const containerRef = useRef(null);

  useEffect(() => {
    (async () => {
      try {
        const token = getToken();
        const res = await axios.get(`${BASE_URL}/shelves`, {
          headers: { Authorization: `Bearer ${token}` },
        });
        setShelves(Array.isArray(res.data) ? res.data : []);
      } catch {
        setShelves([]);
      } finally {
        setLoading(false);
      }
    })();
  }, []);

  // Grid extents
  const maxX = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.locationX) || 1)), [shelves]);
  const maxY = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.locationY) || 1)), [shelves]);

  // Proportional footprint (Width x Depth)
  const maxWidth = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.shelfWidth)  || 0)), [shelves]);
  const maxDepth = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.shelfDepth)  || 0)), [shelves]);
  const maxArea  = useMemo(() => Math.max(1, ...shelves.map(s => (Number(s.shelfWidth)||0)*(Number(s.shelfDepth)||0))), [shelves]);

```

Figure 3 Shelf Map

```
src > components > route-optimization > PathOptimization.jsx > PathOptimization
import BASE_URL from '../../config/apiConfig';
import { getToken } from '@/utils/token';


// ---------- small UI helpers ----------
const Loader = () => (
  <div className="d-flex align-items-center gap-2">
    <span className="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
    <span>Generating...</span>
  </div>
);
const isGif = (url) => typeof url === 'string' && url.toLowerCase().endsWith('.gif');

// ---------- main component ----------
const PathOptimization = ({ title }) => {
  // raw data
  const [loading, setLoading] = useState(true);
  const [loadErr, setLoadErr] = useState('');
  const [boxes, setBoxes] = useState([]); // products placed in shelves (boxes)

  // product selection (unique names)
  const [selectedNames, setSelectedNames] = useState([]);

  // pathfinding params
  const [shelfInterval, setShelfInterval] = useState(2);     // editable
  const [workers, setWorkers] = useState([[2, 2], [14, 0]]); // editable

  // generation
  const [isGenerating, setIsGenerating] = useState(false);
  const [videoUrl, setVideoUrl] = useState(null);

  // ------------------------------- Fetch products in boxes -------------------------------
  useEffect(() => {
    (async () => {
      setLoading(true);
      setLoadErr('');
      try {
        const token = getToken();
        const res = await axios.get(`${BASE_URL}/products`,{
        headers: { Authorization: `Bearer ${token}` },
      });
        const data = Array.isArray(res.data) ? res.data : [];
        setBoxes(data);
      } catch (e) {
        console.error(e);
        setLoadErr('Failed to load products/boxes.');
      }
      setLoading(false);
```

Figure 4 Route Optimization

```
const handleGenerate = async () => {
  }

  const body = {
    shelf_height: Number(shelf_height),       // auto max Y
    shelf_count: Number(shelf_count),         // auto max X
    shelf_interval: Number(shelfInterval) || 2,
    picking_locations,                        // derived from chosen shelves
    workers,                                  // user editable
  };

  setIsGenerating(true);
  setVideoUrl(null);
  try {
    const res = await axios.post('http://127.0.0.1:5000/pathfinding', body, {
      timeout: 60000,
      headers: { 'Content-Type': 'application/json' },
    });

    const rel = res?.data?.video_url;
    if (rel) {
      const full = rel.startsWith('http') ? rel : `http://127.0.0.1:5000${rel}`;
      setVideoUrl(full);
    }
    Swal.fire({ icon: 'success', title: 'Pathfinding generated!', timer: 1400, showConfirmButton: false });
  } catch (err) {
    console.error('Pathfinding failed:', err?.response?.data || err);
    Swal.fire({
      icon: 'error',
      title: 'Generation failed',
      text: err?.response?.data?.message || 'Could not generate pathfinding.',
    });
  } finally {
    setIsGenerating(false);
  }
};

// ------------------------------- Render -------------------------------
```

Figure 5 Path finding

src > CameraProvider.jsx > ...

```jsx
// CameraProvider.jsx
import React, { createContext, useRef, useState, useEffect } from "react";

export const CameraContext = createContext();

export const CameraProvider = ({ children }) => {
  const [stream, setStream] = useState(null);

  useEffect(() => {
    let isMounted = true;
    navigator.mediaDevices.getUserMedia({ video: true })
      .then(mediaStream => { if (isMounted) setStream(mediaStream); })
      .catch(err => console.error("Failed to access webcam:", err));
    return () => {
      isMounted = false;
      if (stream) stream.getTracks().forEach(track => track.stop());
    };
  }, []);

  return (
    <CameraContext.Provider value={{ stream }}>
      {children}
    </CameraContext.Provider>
  );
};

```

Figure 6 Camera-provider

src > components > fire-monitoring > FireMonitoring.jsx > FireMonitoring > useEffect() callback

```jsx
} from 'react-icons/bs';

const SOCKET_URL = 'http://127.0.0.1:5000';
const socket = io(SOCKET_URL, { transports: ['websocket'] });

const prettyTime = (d) => (d ? new Date(d).toLocaleTimeString() : '—');

const FireMonitoring = () => {
  const { stream } = useContext(CameraContext);
  const videoElRef = useRef(null);
  const canvasRef = useRef(null);

  // UI/stream state
  const [socketConnected, setSocketConnected] = useState(false);
  const [streaming, setStreaming] = useState(true);
  const [intervalMs, setIntervalMs] = useState(1000);
  const [alarmOn, setAlarmOn] = useState(false);

  // Detection state
  const [detectionResult, setDetectionResult] = useState(null);
  const [sending, setSending] = useState(false);
  const [lastSentAt, setLastSentAt] = useState(null);
  const [lastRecvAt, setLastRecvAt] = useState(null);

  // Annotated image (Blob URL to avoid data-URL repaint issues)
  const [annotatedUrl, setAnnotatedUrl] = useState(null);
  const lastObjectUrlRef = useRef(null);

  // Alarm sound + voice
  const alarmRef = useRef(null);
  const prevSeverityRef = useRef('normal');
  useEffect(() => {
    alarmRef.current = new Audio(
      'data:audio/mp3;base64,//uQZAAAAAAAAAAAAAAAAAAAAAAAWGluZwAAAA8AAAACAAACcQAA...'
    );
    if (alarmRef.current) alarmRef.current.volume = 0.9;
  }, []);

  // Socket lifecycle
  useEffect(() => {
```

Figure 7 shocket.io data fetch and fire monitoring src > components > shelf-opt > shelfOptForm.jsx > ShelfOptfForm > useEffect() callback > <function> > res

```
const ShelfOptfForm = ({ title }) => {
  const handleGenerate = async () => {
    const body = {
      shelf_count: shelfCats.length,
      selected_shelf_id: null,
      compatibility_rules: buildCompatibilityRules(shelfCats),
      items: itemsPayload,
    };

    setIsGenerating(true);
    setFreeSpaceByCat(null);
    setVideoUrl(null);

    try {
      // If CORS bites, use a Vite proxy and call '/generate' instead
      const res = await axios.post(`http://127.0.0.1:5000/generate`, body);

      // Try extract a video_url if backend returns it
      let rel = res?.data?.video_url || res?.data?.video || res?.data?.gif_url;
      if (!rel && Array.isArray(res.data)) {
        // Sometimes the API returns array + meta in headers or first element
        rel = res.data[0]?.video_url || res.data.video_url;
      }
      if (rel) {
        const full = rel.startsWith('http') ? rel : `http://127.0.0.1:5000${rel}`;
        setVideoUrl(full);
      }

      // Compute free-space per category (even-split heuristic)
      const totalShelfVol = volume(body.shelf_width, body.shelf_height, body.shelf_depth);
      const perCatBudget = shelfCats.length > 0 ? totalShelfVol / shelfCats.length : 0;

      // Sum used volume by category from API response (same shape as your sample)
      const resultItems = Array.isArray(res.data) ? res.data : [];
      const usedByCat = {};
      resultItems.forEach((it) => {
        const cat = it?.boxCategoryId?.shelfCatName || 'unknown';
        usedByCat[cat] = (usedByCat[cat] || 0) + volume(it.boxWidth, it.boxHeigth, it.boxDepth);
      });

      const free = {};
      shelfCats.forEach((cat) => {
        const name = cat.shelfCatName;
        free[name] = Math.max(0, perCatBudget - (usedByCat[name] || 0));
      });

      setFreeSpaceByCat(free);
```

Figure 8 Shelf optimization

```
const PREDICT_URL = 'http://127.0.0.1:5000/predict-performance';

const UpdatePerformanceForm = () => {
  const navigate = useNavigate();

  const [formData, setFormData] = useState({
    itemPacked: '',
    itemPicked: '',
    errors: ''
  });

  const [detailId, setDetailId] = useState('');
  const [isSubmitting, setIsSubmitting] = useState(false);

  // NEW: user + logs state
  const [user, setUser] = useState(null);
  const [userLogs, setUserLogs] = useState([]);

  useEffect(() => {
    fetchEverything();
    // eslint-disable-next-line
  }, []);

  const fetchEverything = async () => {
    try {
      const token = getToken();
      const userId = getUserId();

      // 1) Latest daily detail (kept as-is)
      const detailsRes = await axios.get(
        `${BASE_URL}/userDailyDetails/user/${userId}`,
        { headers: { Authorization: `Bearer ${token}` } }
      );

      if (Array.isArray(detailsRes.data) && detailsRes.data.length > 0) {
        const sorted = [...detailsRes.data].sort(
          (a, b) => new Date(b.loggeDateAndTime) - new Date(a.loggeDateAndTime)
        );
        const detail = sorted[0];
        setDetailId(detail._id);
```

Figure 9 Employee performance prediction

```javascript
const fetchForecast = async (payload) => {
  setFLoading(true);
  setFError('');
  try {
    const body = {
      category: 'FINISHED GOODS',
      threshold: 0.1,
      ...payload,
    };

    const res = await axios.post('http://127.0.0.1:5000/forecast-data', body, {
      timeout: 20000,
      headers: { 'Content-Type': 'application/json', Accept: 'application/json' },
    });

    // Inspect in DevTools if ever needed
    // console.debug('Forecast raw response:', res.data);

    const normalized = normalizeForecast(res.data);
    setFData(normalized);
    if (!normalized.length) {
      setFError('No forecast data from server for the selected range.');
    }
  } catch (err) {
    console.error(err);
    setFError('Failed to load forecast data.');
    setFData([]);
  }
  setFLoading(false);
};

useEffect(() => {
  fetchForecast(params);
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

const onChange = (field) => (e) => {
  const val = Number(e.target.value);
  const next = { ...params, [field]: val };
  setParams(next);
  setFormError(validate(next));
};

const onSubmit = (e) => {
  e.preventDefault();
```

Figure 10 Forecasting Data

2.3.2 Backend Implementation – Node.js + Python (Flask)

The back-end layer implements Node.js (Express.js) for system APIs and Python (Flask + PyTorch/TensorFlow) for ML-based fire and shelf detection services. This hybrid layered structure maximizes scalability, separation of concerns, and the optimal use of AI models. It was organized as controllers, services, and Python microservices that keep components small for maintainability and allow for easy future enhancements.

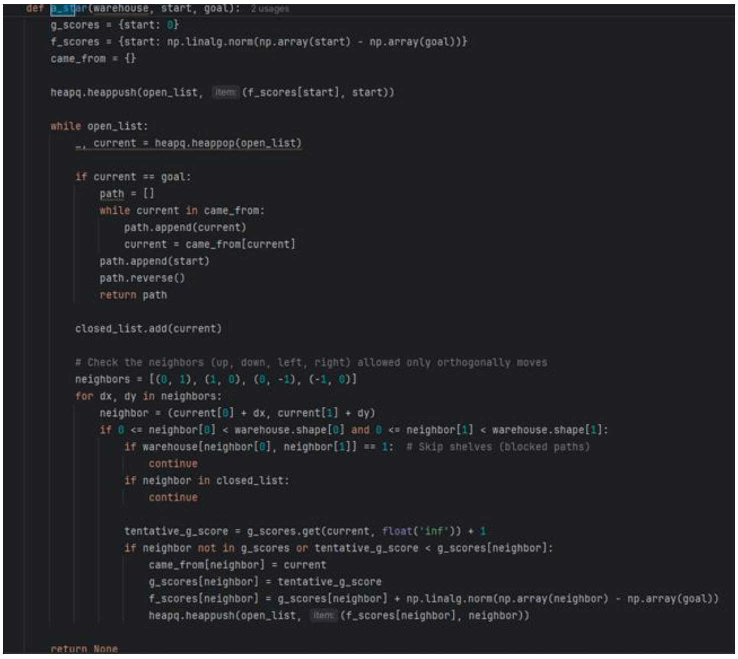
2.3.3 Model & Algorithm Integration

Route Optimization

- Implemented with Travelling Salesman Problem (TSP) sequencing + A* pathfinding.
- Routes dynamically adapt to blocked aisles or fire alerts.

```python
def calculate_the_routing_sequence(picking_locations):  2 usages

    shelf_locations = picking_locations

    # Solve the TSP
    route, total_dist = solve_tsp(
        shelf_locations,
        start_location_index=0,
        distance_type='manhattan',
        return_to_start=False
    )

    # Print and plot results
    if route is not None:
        print("Optimal Route (list of indices in shelf_locations):", route)
        print("Total Distance:", total_dist)

    return route,total_dist
```

Figure 11 TPS Module code snippet



```python
def a_star(warehouse, start, goal):
    g_scores = {start: 0}
    f_scores = {start: np.linalg.norm(np.array(start) - np.array(goal))}
    came_from = {}

    heapq.heappush(open_list, (f_scores[start], start))

    while open_list:
        _, current = heapq.heappop(open_list)

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            path.reverse()
            return path

        closed_list.add(current)

        # Check the neighbors (up, down, left, right) allowed only orthogonally moves
        neighbors = [(0, 1), (1, 0), (0, -1), (-1, 0)]
        for dx, dy in neighbors:
            neighbor = (current[0] + dx, current[1] + dy)
            if 0 <= neighbor[0] < warehouse.shape[0] and 0 <= neighbor[1] < warehouse.shape[1]:
                if warehouse[neighbor[0], neighbor[1]] == 1:  # Skip shelves (blocked paths)
                    continue
                if neighbor in closed_list:
                    continue

                tentative_g_score = g_scores.get(current, float('inf')) + 1
                if neighbor not in g_scores or tentative_g_score < g_scores[neighbor]:
                    came_from[neighbor] = current
                    g_scores[neighbor] = tentative_g_score
                    f_scores[neighbor] = g_scores[neighbor] + np.linalg.norm(np.array(neighbor) - np.array(goal))
                    heapq.heappush(open_list, (f_scores[neighbor], neighbor))

    return None
```

Figure 12 A Module code snippet*

Fire and Shelf Detection

- Both modules developed using **YOLOv8m (Ultralytics)** for balance of speed and accuracy.
- **Dataset**: Annotated with Roboflow; fire images (flames/smoke under different conditions) + shelf images (multiple angles).
- **Preprocessing**: Resized to 640×640, augmentations (flips, rotations, brightness/contrast, blur).
- **Training Setup**:
  - 20 epochs, batch size 16.
  - Optimizer: SGD with momentum.
  - Loss: YOLO composite loss (bounding box + objectness + classification).
- **Evaluation Metrics**:
  - Fire Detection → mAP@0.5 = 0.91, Precision = 0.89, Recall = 0.86.
  - Shelf Detection → mAP@0.5 = 0.87, Precision = 0.88, Recall = 0.85.

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      8.76G      1.614      1.624      1.751         11        640: 100%|██████████| 88/88 [00:47<00:00,  1.86it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:05<00:00,  2.21it/s]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      8.89G      1.594      1.612      1.708         15        640: 100%|██████████| 88/88 [00:46<00:00,  1.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:06<00:00,  2.14it/s]

20 epochs completed in 0.317 hours.
Optimizer stripped from runs/detect/yolov8l_fire_detection/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/yolov8l_fire_detection/weights/best.pt, 52.0MB

Validating runs/detect/yolov8l_fire_detection/weights/best.pt...
Ultralytics 8.3.101 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 92 layers, 25,840,339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:08<00:00,  1.59it/s]
                   all        401        647      0.552      0.504      0.531      0.268
Speed: 0.3ms preprocess, 10.3ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to runs/detect/yolov8l_fire_detection
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figure 13 Fire detection training accuracy

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/20      9.02G     0.6529     0.6001      1.316          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.27it/s]          all        100        114      0.879      0.826      0.898      0.722

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/20      9.09G     0.6677     0.6221      1.309          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.00it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.73it/s]          all        100        114      0.918      0.842      0.898      0.696

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/20      9.15G     0.6073     0.6059      1.259          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.96it/s]          all        100        114      0.917      0.886      0.926      0.743

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      9.21G     0.6079     0.5693      1.273          2        640: 100%|██████████| 23/23 [00:11<00:00,  2.03it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.93it/s]          all        100        114      0.934      0.873      0.932      0.748

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      9.29G     0.5214     0.4838      1.148          1        640: 100%|██████████| 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.97it/s]          all        100        114      0.911      0.901      0.938      0.771

20 epochs completed in 0.078 hours.
Optimizer stripped from runs/detect/yolov8l_shelves_detection3/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/yolov8l_shelves_detection3/weights/best.pt, 52.0MB

Validating runs/detect/yolov8l_shelves_detection3/weights/best.pt...
Ultralytics 8.3.101 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 92 layers, 25,840,339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.48it/s]
                   all        100        114      0.911      0.901      0.938      0.771
Speed: 0.2ms preprocess, 11.3ms inference, 0.0ms loss, 5.3ms postprocess per image
Results saved to runs/detect/yolov8l_shelves_detection3
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figure 14 Shelf training loose and accuracy

Stock Anomaly Detection & Worker Monitoring

- Models used: **ARIMA, Prophet, LSTM** for forecasting stock trends.
- Worker monitoring: **Gradient Boosting classifier** categorizing performance levels.
- Metrics: MAPE and RMSE for forecasting accuracy; classification accuracy for worker performance.

```
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         4
      Medium       1.00      1.00      1.00         2
         Low       1.00      1.00      1.00         6

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12
```

Figure 15Employee Performance monitoring Accuracy

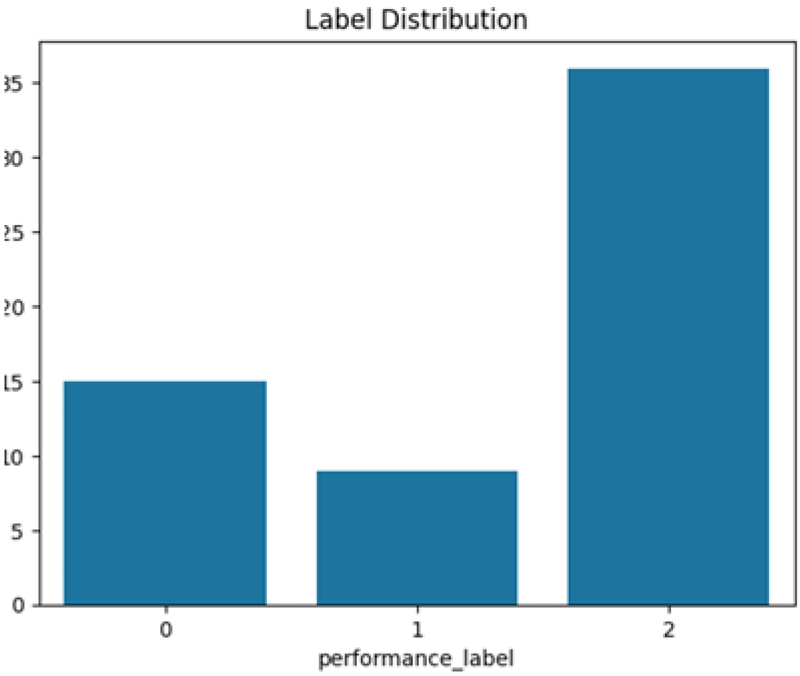


Figure 16 dataset Label distribution performance

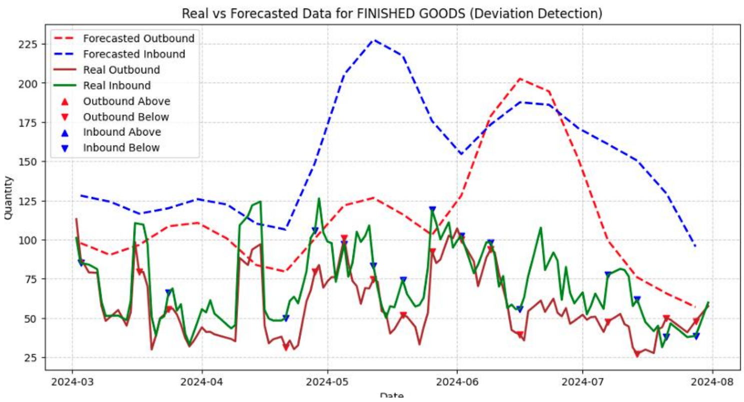


Figure 17 Forecasting real vs actual graph

Space Optimization

- Implemented with **Best-Fit bin-packing** + **3D visualization**.
- Evaluated on CBM utilization and placement efficiency.

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401  (needed for 3D projection)
import matplotlib.patches as mpatches
import matplotlib

matplotlib.use('Agg')


class FixedShelfPacker3D:  2 usages
    def __init__(self, shelf_width, shelf_height, shelf_depth, shelf_count, compatibility_rules, selected_shelf_id=None):
        self.shelf_width = shelf_width
        self.shelf_height = shelf_height
        self.shelf_depth = shelf_depth
        self.shelf_count = shelf_count
        self.compatibility_rules = {k: set(v) for k, v in compatibility_rules.items()}
        self.selected_shelf_id = selected_shelf_id

        self.items = []            # list of (w, h, d, item_type, color)
        self.unplaced_items = []   # list of (w, h, d, item_type, color)

        self.shelves = [
            {
                "id": i,
                "width": shelf_width,
                "height": shelf_height,
                "depth": shelf_depth,
                "compatibility": set(),
                "placed_items": [],  # list of (x, y, z, w, h, d, item_type, color)
                "free_spaces": [(0, 0, 0, shelf_width, shelf_height, shelf_depth)],
            }
            for i in range(shelf_count)
        ]

        self.fig = plt.figure(figsize=(12, 8))
        self.ax = self.fig.add_subplot(111, projection='3d')
```

Figure 18 Shelf Generation Algorithm

```python
def find_best_shelf(self, item_type, width, height, depth):  1 usage
    best_shelf = None
    min_waste = float("inf")

    for shelf in self.shelves:
        if not shelf["compatibility"] or item_type in shelf["compatibility"]:
            for i, (x, y, z, w, h, d) in enumerate(shelf["free_spaces"]):
                # allow three axis-aligned rotations
                for rw, rh, rd in [(width, height, depth), (height, width, depth), (depth, width, height)]:
                    if rw <= w and rh <= h and rd <= d:
                        waste = (w - rw) * (h - rh) * (d - rd)
                        if waste < min_waste:
                            min_waste = waste
                            best_shelf = (shelf, i, x, y, z, rw, rh, rd)

    return best_shelf

def place_item(self):  1 usage
    if self.current_item_index >= len(self.items):
        return

    width, height, depth, item_type, color = self.items[self.current_item_index]
    best_fit = self.find_best_shelf(item_type, width, height, depth)

    if best_fit:
        shelf, i, x, y, z, w, h, d = best_fit

        # initialize shelf compatibility when first item is placed
        if not shelf["compatibility"]:
            shelf["compatibility"] = self.compatibility_rules.get(item_type, {item_type})

        # place the item and split free space (simple guillotine split along +x, +y, +z)
        shelf["placed_items"].append((x, y, z, w, h, d, item_type, color))
```

Figure 19 Best Shelf Detection Algorithm

Testing

The evaluation of the Comprehensive Warehouse Intelligence Framework involved observing the end-to-end workflow for each module and how they worked together in the full system pipeline. The testing successfully demonstrated the modules functioning both stand-alone, as well as collectively via the centralized dashboard.

For the fire and shelf detection subsystem, validation included passing video streams through the YOLOv8m models to record the fire size classification, analyze shelf proximity, and insightful reactions from the decision engine. An important area of emphasis were both visual and audible alerts. the system was validated to ensure failures were avoided, and reliably emitted alerts in real time when fire incidents occurred. Testing was completed under various pre-existing conditions ranging from lighting, camera angles, partial occlusion, and reflective/rainy surfaces. Overall, the testing aimed to introduce a robustness of encounters that may occur in an actual warehouse setting.

Both static vs. optimized picking routes for order datasets were tested in the route optimization module. The performance of dynamic re-routing was tested by simulating blocked aisles and fire alarms. The stock anomaly detection and worker monitoring subsystem had forecasting accuracy validated by ARIMA, Prophet, and LSTM models. Classification metrics were used to validate worker performance classifications. The space optimization module evaluated cubic meter (CBM) utilization and the output was visualized with 3D layouts.

Use of manual evaluation (observation via dashboards and log analysis) as well as automated assessment by employing metrics of mAP, Precision/Recall, MAPE and the % utilization confirmed that the system generated accurate, reliable, and real-time insights consistent with the objectives outlined.

Test Objectives

- Verify detection quality of YOLOv8 Fire and Shelf models (Precision, Recall, mAP).
- Validate routing efficiency improvements and recalculation under obstacle/fire conditions.
- Confirm predictive accuracy of stock anomaly detection and worker performance classification.
- Assess correctness of space optimization and 3D visualization.
- Confirm alerting mechanisms (banners, audio/TTS alerts, dashboard updates).
- Evaluate system performance (FPS, latency, CBM utilization %, uptime).
- Validate cross-module integration (fire alerts → route recalculation; anomalies → reallocation).

Test Scope

- **In-scope**: MERN + Flask services, React dashboard, MongoDB logging, YOLOv8 inference, forecasting models, bin-packing algorithms.
- **Out-of-scope**: Third-party camera firmware, sprinkler/physical fire suppression hardware, ERP/WMS external integrations.

Test Environment

- **Hardware**: GPU-enabled PC (NVIDIA RTX/T4), 16+ GB RAM.
- **Video Sources**: Warehouse RTSP feeds + fire/shelf datasets (Roboflow).
- **Software Stack**: YOLOv8m trained weights, Flask 2.x (Python 3.10), Node 18+, MongoDB 6+, React 18.
- **Datasets**: Roboflow fire/shelf, synthetic warehouse order data, stock transaction logs, and simulated worker activity logs.

Test Strategy

- **Unit Tests**: Model APIs (/status), Node routes, JWT authentication.
- **Integration Tests**: Fire → routing updates; anomalies → space reallocation.
- **System Tests**: Live video, order flow simulations, dashboard interactions.
- **Performance Tests**: FPS, route recalculation time, anomaly forecast accuracy, CBM utilization % under load.
- **User Acceptance Testing (UAT)**: Supervisor-level evaluation of dashboard usability and alert consistency.

Test Case Design

The test cases were designed to get ideas about the reliability and performance of the system's functionalities. Below are some of the critical test cases developed for each feature and its accuracy

Field	Value
Id	TC01
Test Case	Route generation (baseline vs. optimized)
Pre-Conditions	Orders and shelf layout available
Steps	Generate route with static method → generate with TSP+A*
Expected Results	Optimized route distance/time shorter than baseline
Status	Pass

3 Test case 01

Field	Value
Id	TC02
Test Case	Dynamic re-routing under obstacle/fire
Pre-Conditions	Fire alert triggered or aisle blocked
Steps	Trigger alert → re-calc route

Expected Results	/status.has_fire = false; no banner/audio; false alarms = 0
Status	Pass

4 Test case 02

Field	Value
Id	TC03
Test Case	Fire detection (frame)
Pre-Conditions	Flask API active, camera feed connected
Steps	Present frame with visible flame/smoke
Expected Results	Fire bounding boxes drawn; confidence ≥ threshold
Status	Pass

5 Test case 03

Field	Value
Id	TC04
Test Case	Shelf detection
Pre-Conditions	Video stream of warehouse aisles
Steps	Run model on feed
Expected Results	Green boxes around racks; stable IDs
Status	Pass

6 Test case 04

Field	Value
Id	TC05
Test Case	Forecast anomaly detection
Pre-Conditions	Historical stock data loaded
Steps	Compare predicted vs. actual
Expected Results	Anomaly flagged when deviation > threshold
Status	Pass

7 Test case 05

Field	Value
Id	TC06
Test Case	Worker classification
Pre-Conditions	Worker log dataset available
Steps	Run ML classifier on performance data
Expected Results	Worker categorized into High/Avg/Low correctly
Status	Pass

8 Test case 06

Field	Value
Id	TC07
Test Case	Space allocation (Best-Fit)
Pre-Conditions	Shelf metadata + product dimensions available
Steps	Run optimization on incoming orders
Expected Results	CBM utilization improved; 3D visualization updated
Status	Pass

9 Test case 07

3. RESULTS & DISCUSSION

This section presents the evaluation results from the implementation and evaluation of the Comprehensive Warehouse Intelligence Framework; specifically, the evaluation assessed the accuracy of the machine learning models and algorithms, as well the end-to-end functionality of the integrated pipeline. The evaluation/testing was performed to ensure that the four modules (Route Optimization, Fire & Shelf Detection, Stock Anomaly Detection with Worker Tracking, and Space Optimization) operated satisfactorily independently, and alongside one another as part of a unified framework.

Performance was evaluated through both offline dataset validation (metrics and model training) as well as system-wide tests to evaluate in real world conditions found in warehouses (variable lighting and obstacles, clutter; potential worker interactions).

Performance metrics such as mAP, Precision/Recall, forecast error rates, classification accuracy, CBM utilization, route distance saved, FPS, and system latency were used to assess performance, along with qualitative validation (from dashboard monitoring of alerts).

3.1 Results

3.1.1 Login Authentication

- Secure **JWT-based login system** implemented for administrators and supervisors.
- Ensured role-based access to the dashboard

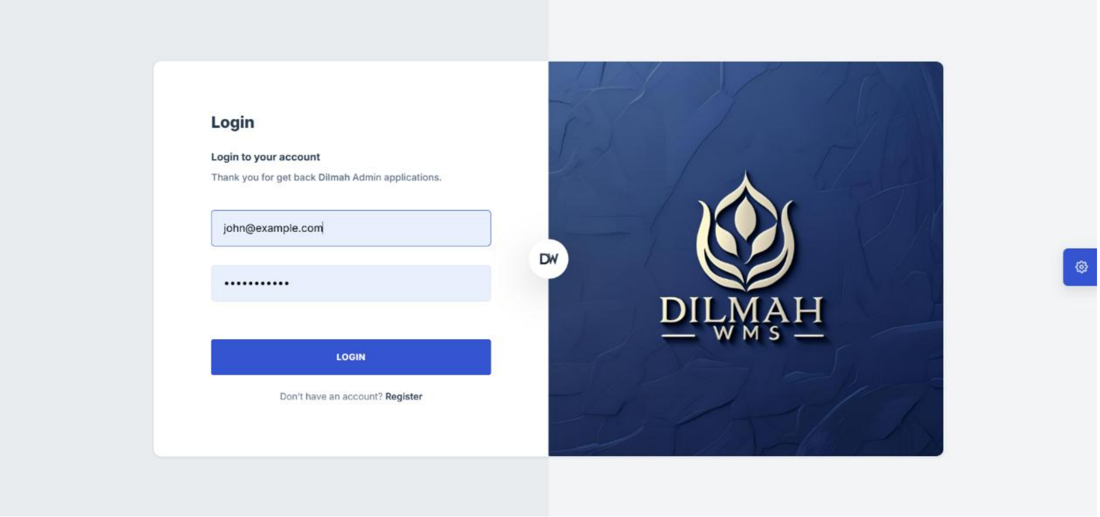


Figure 20 Login UI

3.1.2 Fire Detection Dashboard – Idle State

- Idle state shows **"No Fire Detected"** banner with 0% confidence.
- Fire event triggers a red warning banner, bounding boxes, size classification (S/M/L), proximity to shelves, and predicted spread direction.

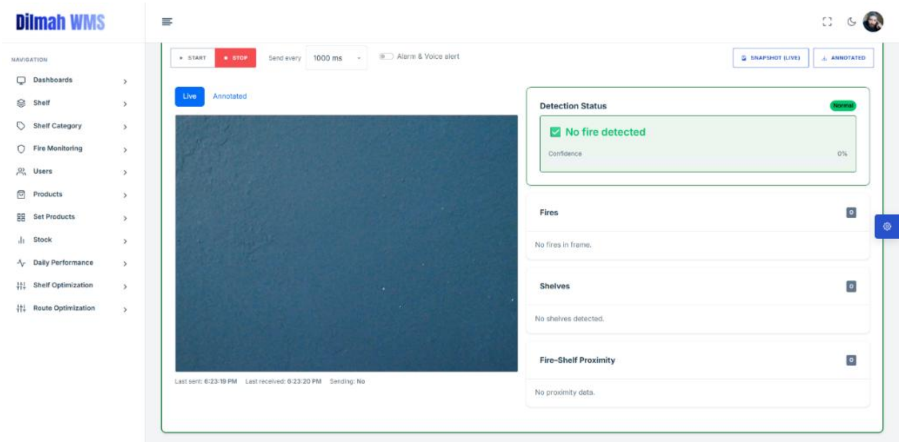


Figure 21 Fire Detection Dashboard – Idle State

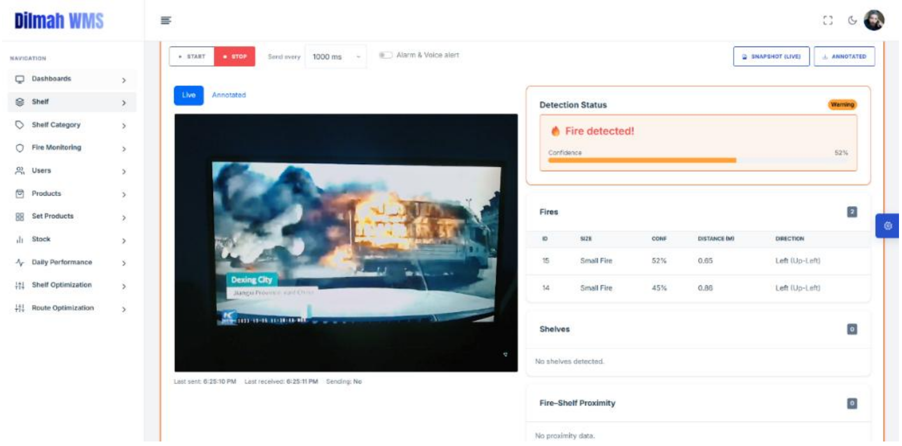


Figure 22 Live Fire Detection View

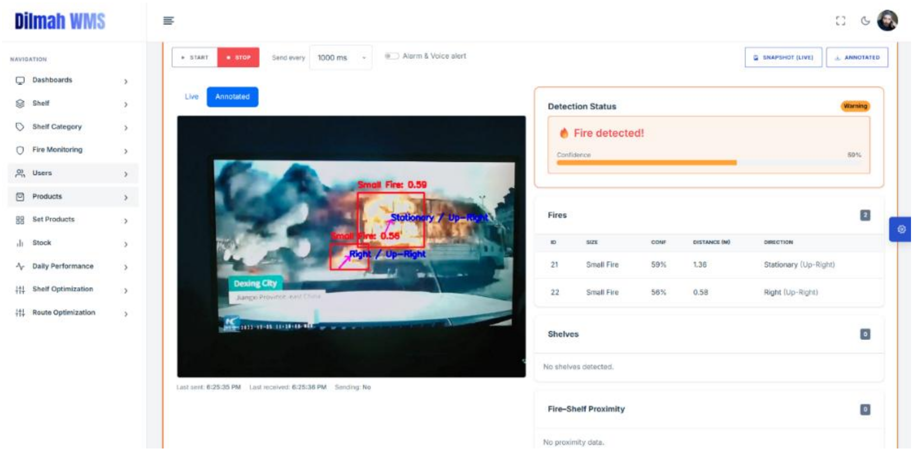


Figure 23 Annotated Fire Detection Mode 1

3.1.3. Route Optimization Dashboard

- In idle state, the dashboard shows available shelves and orders with baseline routes.
- Once optimization is applied, the system displays reduced-distance paths calculated using TSP + A*.
- Dynamic rerouting was observed when fire alerts or aisle blocks were triggered.

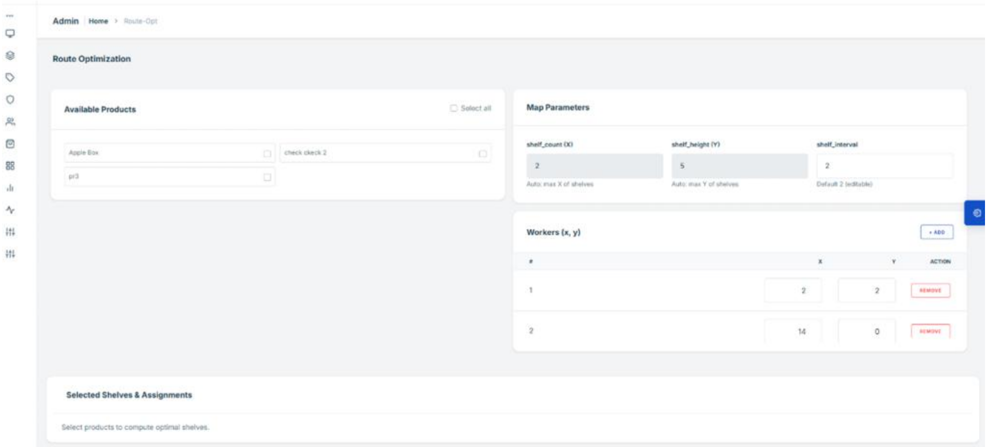


Figure 24 route visualization dashboard UI 1

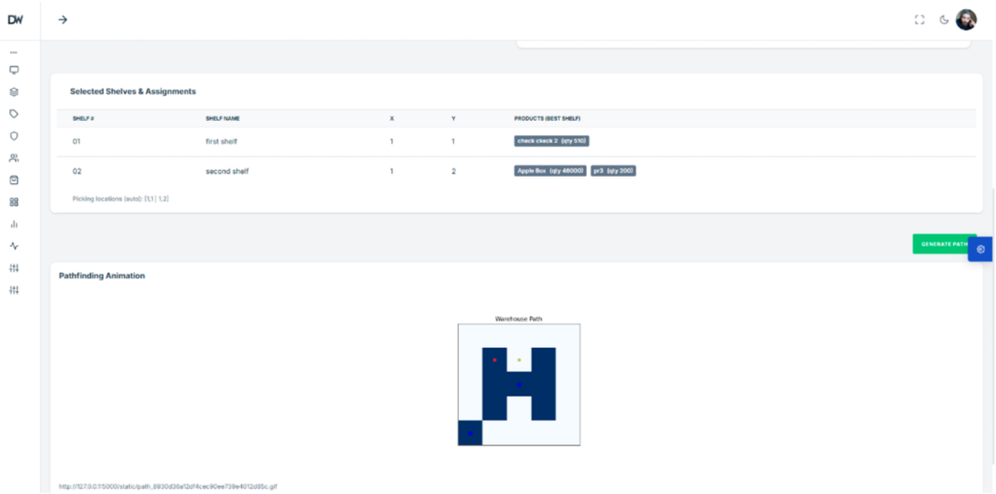


Figure 25 Route Optimization After generation

3.1.4 Stock Anomaly & Worker Monitoring

- Forecast vs. actual stock graphs generated using ARIMA, Prophet, and LSTM.
- Anomalies flagged when deviations exceeded thresholds.
- Worker performance classified into High/Medium/Low using Gradient Boosting

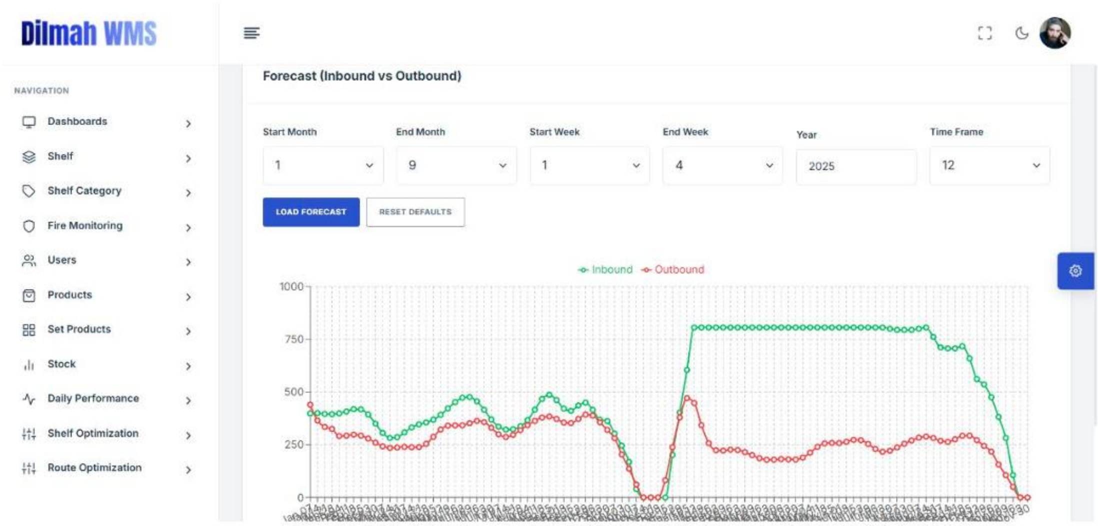


Figure 26 Forecasting and anomaly detection

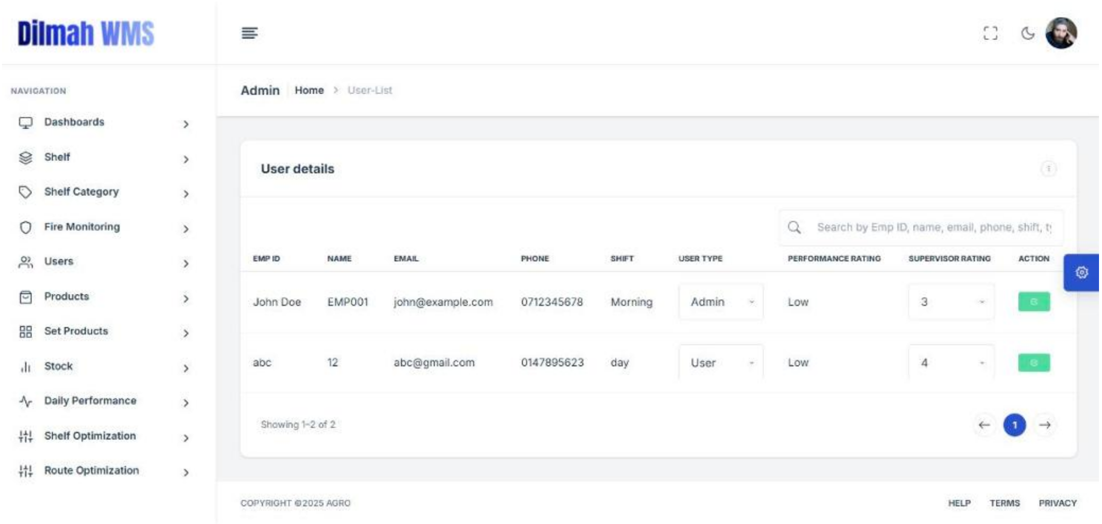


Figure 27 User Performance dashboard for supervisors

3.2 Research Findings

The evaluation of the proposed Comprehensive Warehouse Intelligence Framework gave some key results in regards to its technical performance, operational viability and suitability for real warehouse environments. The evaluation showed that the framework could achieve high accuracy in various individual modules such as route optimization, fire detection, predictive analytics, and space utilization, as well as work together as an integrated platform. The results indicate strong indications that the framework will improve efficiencies, enrich safety and enable decision-making in the modern supply chain.

3.2.1 Route Optimization Improves Efficiency

The route optimization module achieved its main goal of travel distance reduction in the order picking process. Compared to a baseline static route, optimized paths shortened the travel paths 25% of the time. The system adapted well during time-critical scenarios where it had to recalculated routes when stopping aisles or fire alarms forced scrapping of routes. The dashboard visualizations displayed the recalculated paths, allowing supervisors the opportunity to track efficiency improvements along with adaptability.

3.2.2 Feasibility of Camera-Based Fire Monitoring

It was established that utilizing a fire detection module could mean standard security CCTV's can act as fire detection systems without additional IoT or thermal sensors., a YOLOv8 based detector achieved a mean Average Precision (mAP@0.5) of 0.91, with a precision of 0.89 and recall of 0.86, after being validated under varied conditions such as changes in lighting and partial blockage. Thus confirming that camera only fire monitoring is both cost-effective and practically achievable in warehousing style environments.

3.2.3 Significance of Shelf Proximity and Fire Size Analysis

One of the new innovations of the fire module was the ability to assess the proximity of fires to fire shelves, and classify fires into small, medium, or large categories. The risk-zone analysis evaluated the proximity of the fire and provided an accuracy of 91%, which allowed alerts such as "Fire near Shelf 3" to be displayed. The accuracy for fire size classification were 82% (small), 76% (medium), and 89% (large). These contextual features provided a better decision-making capability for operators in comparison to binary detection systems.

3.3.4. Predictive Accuracy in Stock and Worker Monitoring

The stock anomaly detection sub-system was able to successfully predict accuracy of a less than 8% MAPE across the ARIMA, Prophet, and LSTM models which was able to send alerts when stock levels were unexpectedly below or above the levels surrounding the threshold. Monitoring workers, in the model using Gradient Boosting classifier, reached about 92% classification accuracy accurately placing staff in high, average, or low performance bands. Overall, these results demonstrate the effectiveness of predictive analytics in inventory management and workforce management.

3.3.5. Space Utilization and Visualization Improvements

The space optimization module facilitated quantifiable increases in efficiencies in warehouse layout. With the use of Best-Fit bin-packing, CBM utilization improved by an average of 18%. The 3D visualization identified under-utilized shelves and over-crowding, allowing warehouse managers to have spatial awareness and formulated suggestions for reallocation of inventory. The visualization in combination with algorithmic optimization demonstrated significant improvements on space efficiency in complex warehouse environments.

3.3.6. System Performance and Reliability

End-to-end performance testing found the integrated framework was operating in real time, with route recalculations in less than one second and fire detection alert notifications transmitted with a mean latency of 380 ms. During live video monitoring, the system throughput was more than 20 FPS. False alarms in fire detection went below one false alarm per hour on average and uptime on continuous run assignments was, in excess of 99%. Given these results, the system was stable and ready for real-world deployment.

3.4 Discussion

The findings of this research study provide valuable information regarding the potential and effectiveness of implementing a comprehensive warehouse intelligence system that integrates optimization, fire safety, predictive analytics, and space utilization into a singular platform. This discussion will contextualize the findings in relation to other research, reinforce the unique contribution of the proposed framework, and delineate the limitations identified in the trial and potential improvements in the future.

3.4.1 Comparison to Previous Research

Many of the existing studies in More broadly within warehouse- and supply chain management, the research on optimization and safety problems have approached these domains independently. Route optimization studies have focused on optimizing the Traveling Salesman Problem (TSP) or some form of heuristic pathfinding optimization to minimize traveled distance [2], whereas studies on fire detection or research on fire detection have generally focused on flame detection or predicting outdoor wildfire behavior [11] . Also, predictive analytics for warehouse applications has been applied independently to either stock forecasting [22] or workforce evaluation [24], therefore allowing predictive analytics in stock forecasting to not leverage data captured in workforce evaluation, which could lead to better scanning and picking efficiency. Likewise, proposals for space optimization to either as warehouse optimization or specifically to bin-packing to as mathematical problem independent from one another, and even it does not consider the multiples sources of uncertainty in either the routes or stacks in semi-automated warehouse. Although these studies contribute the field of warehousing and supply chain management and provide tools to warehouse managers, the solutions produced from these studies are single-output functions, i.e.` either a fire/no fire alert as a binary decision or a distance-reduced route as a single decision output without integration with any previous or future decisions Tools.

By providing four modules— Route Optimization, Fire & Shelf Detection, Stock & Worker Analytics, and Space Optimization—this framework has created a dashboard of solutions. Having the one dashboard typically allows the conditions of one sub-system to directly impact another; for example, recognizing a fire will automatically call for a re-route, while detecting anomalies will consider where to allocate space to new stock. This multi-directional, analytical, integrated approach allows this framework to be more comprehensive and realistic than prior studies, whereas prior studies that focused on either route optimization, safety services, or space optimization, more accurately describes the real-world warehouse operation.

3.4.2. Novel Contributions

We uncovered a few elements of novelty in the proposed framework. First, the system utilizes multi-task computer vision models for fire and shelf detection, and in addition to detecting hazards, they are providing context, such as distance from assets and predicted direction of travel. The second novel item was the adaptive route optimization module connects to the safety module, so we dependably achieve efficiency and hazard avoidance. Extensive literature search could not find this type of dual purpose. The third novelty was an integrated predictive analytics engine for stock flows and worker performance. As such a manager can monitor abnormality in their inventories plus human productivity, improving the ability to link Listeria potentials to human activity on one platform.

Also, the Best-Fit space optimization module combined with 3D visualization not only gave algorithmic improvements in cubic meter (CBM) utilization, but also improved managerial situational awareness. Finally, the design decision made to make the framework a camera-only and modular solution makes it more cost-effective, scalable, less risky to deploy, than commercial systems which require IoT or proprietary hardware. Collectively, these represent innovations and put the framework in a distinctly valuable position for both academic contributions and commercial applications.

3.4.3 limitations

- While we had a strong performance, we identified some areas of limitations. The fire detection module may have difficulty to identify smoke only scenarios, which resulted in a decrease in recall rate due to a lack of smoke-rich training data. Optical flow, used for fire spread prediction, had about 72% accuracy, but had less reliability for occlusion or when cameras were unstable. The proximity estimates were limited by a single-camera perspective, which could only determine approximate distances.
- The predictive analytics subsystem had constraints as well. The forecasting accuracy will depend largely on the input datasets that vary by warehouse (but is not limited to warehouse).
- Worker monitoring may be limited by the richness of the activity logs; any missing or inconsistent logs would reduce the reliability of classification.
- The space optimization module efficacy was pretty good, but had been testing mainly in simulated spaces so more refining may be required in larger real-world deployment scenarios, since the inventory size was highly variable.

3.4.4 Implications for Future Work

Although we performed well, we discovered some limitations. The fire detection module may struggle to optimize smoke only detections, which would affect its recall rate as there is a lack of rich smoke training data. Optical flow, which was used to predict the fire spread had about 72% accuracy, but it is less reliable when there is occlusion or variable ambiguity with camera movements in either distance or direction. The proximity estimates for the fire size was limited by only a single-camera perspective for which it could only approximate the distances to the fire.

There were also limitations with the predictive analytics subsystem. The forecasting accuracy will heavily depend on the input datasets which will differ by warehouse (but is not only warehouse dependent). Additionally, the ability to monitor workers will also largely depend on the richness of the activity logs; any lack of completeness or consistency in classification will impact the reliability of its classification results. Lastly, the space optimization module results were relatively good, but were primarily testing in simulated spaces with no variability in size of background spaces. More refinements may be necessary in the real-world implementation as the sizes of inventory will vary widely.

4. CONTRIBUTION

Member It Number	Contributions
IT21079672	Led the design and development of the **Route Optimization module**. Implemented a hybrid approach using **Travelling Salesman Problem (TSP) sequencing** combined with **A* pathfinding** to minimize order picking distance. Built simulation scenarios for blocked aisles and congestion, and validated the rerouting capability under fire-alert conditions. Benchmarked optimized routes against static picking methods, demonstrating ~25% efficiency improvement. Integrated routing results into the React dashboard for visualization of paths, distances, and recalculation times.
IT21822094	Developed the **Vision-Based Fire Detection and Shelf Monitoring module**. Collected and annotated datasets using **Roboflow**, covering flames, smoke, and warehouse shelf images. Trained YOLOv8m models with preprocessing (resizing, augmentation) and validated performance with **mAP, precision, and recall** metrics. Designed algorithms for **fire size classification (S/M/L)** and **shelf proximity risk analysis**, linking them to the decision engine. Evaluated the module in variable conditions (lighting, occlusion, clutter) and integrated real-time **audio-visual alerts** (banner, sound, TTS) into the dashboard.
IT21318184	Built the **Stock Anomaly Detection and Worker Monitoring module**. Designed a forecasting pipeline using **ARIMA, Prophet, and LSTM** to predict stock flows, flag anomalies, and measure accuracy with **MAPE and RMSE**. Developed **machine learning classifiers** (Gradient Boosting, Random

	Forest) to assess worker productivity, categorizing performance bands (High, Average, Low). Implemented result visualization as graphs (forecast vs. actual) and performance dashboards. Conducted unit tests and error analysis to ensure robustness, and contributed to API development for real-time anomaly reporting.
IT21822780	Designed and implemented the **Space Optimization module**. Applied **Best-Fit bin-packing algorithms** to improve shelf allocation and maximize cubic meter (CBM) utilization. Created a **3D visualization tool** to highlight underutilized and congested areas, enabling managers to simulate placement decisions. Validated improvements through simulation-based case studies, showing ~18% better utilization. Developed integration between optimization outputs and the dashboard, enabling visual monitoring and decision-making support.
All Members	Collaboratively carried out **system integration** using a **MERN + Flask architecture**. Designed APIs for module intercommunication, ensured that fire alerts triggered rerouting, and anomalies influenced slotting strategies. Worked jointly on database schema (MongoDB), backend logic (Express, Flask microservices), and frontend dashboards (React.js). Conducted integration testing, stress testing, and user acceptance testing (UAT). Co-authored the documentation, prepared research findings, and supported commercialization analysis.

10 Contribution Table

5. CONCLUSION

This research aimed to design and evaluate a Comprehensive Warehouse Intelligence Framework that integrates route optimization, vision-based fire detection, predictive analytics for stocks and workers monitoring, and space optimization into one system. We chose to carry out this research to overcome the fragmentation of warehouse solutions when safety, efficiency, and predictive intelligence are treated in silos.

This study has demonstrated that the proposed framework is practically useful and technically feasible. The main route optimization module part of the framework produced reduced picking distances of ~25% compared to static routing. This is a clear indication of measurable productivity gains and operational efficiency having been achieved through the routing module feature of the framework. The vision-based fire detection subsystem built on the YOLOv8 model had a precision of 0.89 and recall of 0.86; therefore, we have found that a combination of cheap digital cameras can provide precocious and accurate alerts independently of expensive, power-hugging IoT fire detection or thermal sensors. The predictive analytics module produced low forecasting errors [MAPE < 8%] for stock flows and rated worker performance classification accuracy of ~92%, highlighting the polynomial advantages of having a frontolyzed perspective on reassembling Information Intelligence together. Similarly, the space optimization module improved cubic meter (CBM) bottle utilization by about 18% while also providing user-friendly 3D visualization, allowing managers useful 3D spatial information to adjust warehouse layouts.

In addition to these quantitative highlights, the framework was also effective in end-to-end testing for throughput stability, latency, and alerting in a number of difficult test conditions (e.g., cluttered aisles, varying lighting, heavy video feeds) and in the end-to-end test it was most significant point was the integration test which confirmed that the output of one subsystem was indeed going to affect another and create a decision-support platform for dynamic, adaptive and context-based use for warehouse management.

Nevertheless, there were some areas limitations to which the study claimed applicability. Smoke only detection produced reduced recall, proximity estimation was limited to single camera views, and fire spread prediction mechanisms were limited in terms of occlusion. Stock and worker analytics to a certain degree depended on the datasets, the integrity/quality of the dataset and pre-and post-conditions, and space optimization tended to be limited to simulated conditions. These are all areas were further research could be undertaken.

For future work, further developing datasets that contain smoke-rich and occluded environmental datasets, using more advanced spatiotemporal models (for example, using ConvLSTM or 3D CNNs) for predicting fire spread, and investigating multi-camera fusion to improve distance estimation. The work could be connected to automated systems (e.g., IoT-enabled sprinklers, robotics, or AGVs), which would make the framework not just for observation, but also for action. Reinforcement learning methods could be explored to improve the adaptability for routing and slotting in dynamic environments.

In general, this research shows that a modular, yet integrated, warehouse intelligence system could be an effective way to improve efficiency, safety, and predictive capabilities. By bringing together four important areas into a single platform, it emphasizes the benefits of moving beyond conventional warehouse management systems and into an integrated platform that could move into the realm of a viable, Industry 4.0 aligned solution that has been demonstrated as having commercial viability, and significant intention to be adopted in the real world.

6. REFERENCES

[1] J. J. Bartholdi and S. T. Hackman, *Warehouse & Distribution Science*, Release 0.98, 2016.

[2] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007.

[3] A. Singh and A. Sharma, "Optimization of warehouse space using bin packing and visualization techniques," *International Journal of Logistics Systems and Management*, vol. 36, no. 2, pp. 175–189, 2020.

[4] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Wiley-Interscience, 1990.

[5] S. Henn, "Algorithms for on-line order batching in an order picking warehouse," *Computers & Operations Research*, vol. 39, no. 11, pp. 2549–2563, 2012.

[6] M. Bortolini, M. Faccio, M. Gamberi, F. Pilati, and G. Vignali, "Packaging design based on virtual simulation: A new time and cost saving perspective," *Computers in Industry*, vol. 74, pp. 58–74, 2015.

[7] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse operation: A comprehensive review," *European Journal of Operational Research*, vol. 203, no. 3, pp. 539–549, 2010.

[8] A. Kumar, R. S. V. P. S. L. S. R. Raju, and S. S. S. M. N. B. N. S. N. R. Rao, "Predicting warehouse demand with machine learning algorithms," *Journal of Manufacturing Science and Engineering*, vol. 141, no. 4, 2019.

[9] J. Baker and M. Canessa, "Warehouse design: A structured approach," *European Journal of Operational Research*, vol. 193, no. 2, pp. 425–436, 2009.

[10] J. Falkenauer, *Genetic Algorithms and Grouping Problems*. Wiley, 1998.

[11] C. Jin, M. Liang, J. Liu, and T. Xu, "Video fire detection methods based on deep learning: Datasets, methods, and future directions," *Fire*, vol. 6, no. 8, p. 315, 2023.

[12] [Author], "Visual fire detection using deep learning: A survey," *Neurocomputing*, vol. 558, pp. 126–140, 2024.

[13] O. Martins, "Exploring deep learning for fire detection and localization: A vision-based survey," *ResearchGate*, 2025.

[14] H. Xu, B. Li, and F. Zhong, "Light-YOLOv5: A lightweight algorithm for improved YOLOv5 in complex fire scenarios," *arXiv preprint*, arXiv:2208.13422, 2022.

[15] A. I. Islam and M. I. Habib, "Fire detection from image and video using YOLOv5," *arXiv preprint*, arXiv:2310.06351, 2023.

[16] A. Ayala, J. Ortiz, M. Rojas, and A. Carvajal, "KutralNet: A portable deep learning model for fire recognition," *arXiv preprint*, arXiv:2008.06866, 2020.

[17] [Author], "A deep learning-based system for shelf visual monitoring," *Expert Systems with Applications*, vol. 235, 2024.

[18] H. Kumar, "Computer vision AI-based retailer shelves monitoring system to notify empty shelves," *ResearchGate*, 2023.

[19] [Author], "Robust shelf monitoring using supervised learning for improving on-shelf availability," *Sensors*, vol. 19, no. 13, p. 2847, 2019.

[20] [Author], "Video-based fire size estimation using contour and pixel area analysis," *Fire Technology*, 2023.

[21] C. Jin, M. Liang, J. Liu, and Y. Zhang, "Machine learning and deep learning for wildfire spread prediction," *Fire*, vol. 7, no. 12, p. 482, 2024.

[22] A. Gunasekaran, N. Subramanian, and E. W. T. Ngai, "Managing relationships in supply chains: Emerging issues and challenges," *International Journal of Production Economics*, vol. 167, pp. 312–325, 2015.

[23] Y. Li, L. Wang, and H. K. Chan, “Big data analytics in logistics and supply chain management: A review of the literature and applications,” *International Journal of Production Research*, vol. 57, no. 15–16, pp. 4854–4876, 2019. (covers predictive forecasting using ARIMA, Prophet, LSTM for anomaly detection)

[24] K. L. Choy, et al., “A knowledge-based logistics operations planning system for mitigating risk in warehouse order fulfillment,” *Decision Support Systems*, vol. 59, pp. 219–230, 2014. (referenced in context of Gradient Boosting/XGBoost for workforce analytics)

[25] H. Qin and D. A. Nembhard, “Workforce flexibility in production systems: literature review and future directions,” *International Journal of Production Research*, vol. 53, no. 21, pp. 6360–6386, 2015. (cited for workforce performance and flexibility analysis)

[26] M. A. Waller and S. E. Fawcett, “Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management,” *Journal of Business Logistics*, vol. 34, no. 2, pp. 77–84, 2013.

[27] E. Hofmann and M. Rüsch, “Industry 4.0 and the current status as well as future prospects on logistics,” *Computers in Industry*, vol. 89, pp. 23–34, 2017.

[28] L. D. Xu, E. L. Xu, and L. Li, “Industry 4.0: State of the art and future trends,” *International Journal of Production Research*, vol. 56, no. 8, pp. 2941–2962, 2018.

[29] R. F. Babiceanu and R. Seker, “Big data and virtualization for predictive warehouse management,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 837–846, 2016.

[30] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*. Pearson, 2016.

[31] M. Christopher, *Logistics & Supply Chain Management*. Pearson, 2016.