International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 2, 2026

Optimizing Retail Inventory With Multi Agents

Mohd Basit Mohiuddin[1], L Rajeshwari[2], B Bhavani[3], Sangireddy Abhinaya[4]

[1]Assistant Professor; Department Of Computer Science And Engineering(AI&ML) Bhoj Reddy Engineering College For Women Hyderabad India

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering(AI&ML) Bhoj Reddy Engineering College For Women Hyderabad India

Mail Id; llakavathrajeshwari@gmail.com[2],bhavaniboyapati99@gmail.com[3],sangireddyabinaya@gmail.com[4]

Abstract

The Multiple Agent system is a machine learning–based time series forecasting framework developed to automate demand prediction and inventory decision-making within retail environments. The proposed system employs a multi-agent architecture composed of a Store Agent, Warehouse Agent, Supplier Agent, and an Orchestrator that collectively manage the workflow of inventory forecasting and stock control. The Store Agent utilizes a Linear Regression model implemented through the scikit-learn library to analyze historical sales data and generate predictions of future product demand. These demand forecasts are evaluated by the Warehouse Agent, which compares predicted sales with the current inventory stored in a SQLite database. Using rule-based threshold logic, the system determines whether the available stock is sufficient or if replenishment is required. When the inventory level falls below the predefined threshold, the Supplier Agent automatically generates purchase orders containing product information, required quantities, and cost estimates. The Orchestrator component coordinates the interaction among all agents, ensuring that forecasting, stock evaluation, and order generation occur in a structured sequence. To enhance system transparency and explainability, the platform integrates a Large Language Model (LLM) through Ollama, which produces reasoning logs that describe the decision-making process of the system. The application is implemented using Python, Flask, Bootstrap, and supporting libraries such as pandas and NumPy for data processing and analysis. Through a web-based dashboard, users can record sales transactions, monitor inventory levels, and track workflow execution. By automating demand forecasting and inventory decisions, the system reduces manual workload, improves prediction accuracy, and supports data-driven inventory management in modern retail operations.

Keywords

Inventory Management, Multi-Agent Systems, Machine Learning, Demand Forecasting, Linear Regression, Supply Chain Automation, Time Series Prediction, Large Language Models (LLM), Retail Analytics

Introduction

In modern retail environments, effective inventory management plays a critical role in maintaining profitability and ensuring customer satisfaction. Retail stores must carefully maintain a balance between product supply and consumer demand. When inventory levels are excessively high, businesses experience increased storage costs and capital becomes locked in unsold goods. Conversely, insufficient inventory may lead to stock shortages, lost sales opportunities, and dissatisfied customers. Therefore, accurate demand forecasting and intelligent stock management are essential components of efficient retail operations.The “Multiple Agent” system proposed in this project addresses these challenges by integrating Machine Learning techniques with AI-based reasoning. The system functions as a time series forecasting platform that predicts future product demand using historical sales information. Based on these predictions, the system automatically makes inventory-related decisions through specialized software agents. This approach demonstrates how automated decision systems can replace traditional manual inventory planning methods.The Store Agent is responsible for predicting product demand. It uses Linear Regression from the scikit-learn library to analyze historical sales data and identify patterns in previous transactions. Based on these patterns, the system generates daily demand predictions that assist in inventory planning.The Warehouse Agent evaluates the predicted demand and compares it with the current stock level. Using rule-based logic, it determines whether the existing inventory is sufficient or if additional stock must be ordered. This proactive approach helps prevent stock shortages before they occur.When replenishment is required, the Supplier Agent automatically generates purchase orders. It determines the required quantity, estimates the cost, and records the order information in the system database. This automation reduces manual effort and improves operational efficiency.A distinctive component of this system is the integration of Ollama with a Large Language Model (LLM). The LLM provides explanations for system decisions by generating reasoning logs. For instance, it can explain why a purchase order was created or why existing inventory is considered adequate. This feature improves transparency and helps users understand the logic behind automated

decisions.The system is implemented using Python 3.x and the Flask web framework. SQLite is used as the database due to its lightweight and efficient structure. The user interface is designed with Bootstrap 5, providing a responsive and user-friendly web dashboard. Supporting libraries such as pandas, numpy, scikit-learn, requests, and tabulate are used for data analysis and processing.

Existing System

In many small and medium-sized retail businesses, inventory management and sales forecasting are performed using manual methods or simple digital tools such as spreadsheets. Sales information is often recorded in notebooks or Excel files, and future demand is estimated based on past experience or basic calculations. These approaches typically lack advanced forecasting techniques, which reduces prediction accuracy and increases reliance on human judgment.Traditional systems also lack **real-time monitoring and automated alerts**, making it difficult to track inventory efficiently as business operations expand. Consequently, managing large datasets and complex supply chains becomes increasingly challenging.

Proposed System

The proposed Multiple Agent system is an intelligent inventory management platform designed to automate demand forecasting and stock control using Machine Learning and multi-agent architecture. The system predicts future product demand and performs automated inventory decisions based on historical sales data.Sales records are stored within a SQLite database, and data processing is carried out using pandas and numpy libraries. The Store Agent applies Linear Regression from scikit-learn to analyze historical sales patterns and generate demand forecasts. This data-driven approach improves prediction accuracy compared to traditional estimation methods.The Warehouse Agent evaluates the predicted demand against current stock levels. Using predefined rule-based logic, the system determines whether inventory replenishment is required..An Orchestrator component coordinates the interaction between different agents to ensure smooth workflow execution. In addition, the system integrates a Large Language Model through Ollama, which generates reasoning logs that explain the decisions made by the agents. These explanations improve transparency and help users understand system behavior.The entire system is accessible through a web-based dashboard built with Flask and Bootstrap, enabling users to monitor inventory status, execute workflows, and review system outputs.

Literature Survey

Recent research in the fields of artificial intelligence, multi-agent systems, and supply chain analytics has contributed significantly to the development of intelligent inventory management solutions.Ayang et al. (2025) proposed a multi-agent deep learning framework for demand forecasting and supply chain optimization. Their study demonstrates how cooperative agents can dynamically respond to changing demand patterns, improving supply chain efficiency. This work supports the use of a multi-agent architecture in automated inventory systems.Quan and Liu (2024) introduced an LLM-driven multi-agent inventory management system that enhances decision-making through communication and reasoning between agents. Their research highlights the role of AI reasoning in improving automated supply chain management.Bastarianto et al. (2023) examined the application of agent-based modeling in complex dynamic systems. Their work emphasizes characteristics such as agent autonomy, cooperation, and system scalability, which are essential in designing multi-agent frameworks.Rasal (2024) explored multi-agent communication supported by large language models. The study highlights the importance of structured interaction and clearly defined agent roles, which are relevant for designing coordinated workflows among different agents.Earlier studies in supply chain management also provide important theoretical foundations. Silver et al. (2016) and Chopra & Meindl (2019) discussed traditional inventory control strategies such as reorder point models, stock balancing, and supply-demand coordination.Box et al. (2015) introduced time series forecasting techniques such as ARIMA, which are widely used for predicting future demand using historical data. Although these models are powerful, the current project adopts Linear Regression due to its simplicity and computational efficiency.Although these studies provide strong theoretical and technical foundations, there remains a need for a simple, integrated, and explainable solution that combines forecasting, automated inventory management, and AI-based reasoning. The proposed system addresses this gap by integrating Machine Learning, multi-agent architecture, and explainable AI into a practical inventory management platform.

Functional Requirements

Functional requirements describe the essential operations and services that the system must perform to achieve its intended objectives. The proposed Multiple Agent inventory management system is designed to support several important functionalities that enable efficient monitoring, forecasting, and decision-making in inventory operations. The system manages product information such as product name, category, price, and available stock levels within a structured database. It records and maintains historical sales data including transaction date, quantity sold, and generated revenue, which is later used for demand analysis. The Store Agent performs demand

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

forecasting by applying a Linear Regression model that analyzes historical sales data and predicts future demand patterns for each product. These predicted values are then evaluated by the Warehouse Agent, which compares the forecasted demand with the existing inventory levels in order to determine whether stock replenishment is necessary. Based on predefined rule-based logic, the system identifies situations where inventory falls below the required threshold and triggers the Supplier Agent to automatically generate purchase orders.

Computational Resource Requirements

The implementation of the Multiple Agent inventory management system requires specific hardware and software resources to ensure smooth operation and effective performance. From a hardware perspective, the system requires a processor equivalent to at least an Intel i3 or higher to support application execution and data processing tasks. A minimum of 4 GB RAM is required for basic system functionality, although 8 GB of memory is recommended to achieve better performance when handling larger datasets and machine learning computations. The system also requires at least 20 GB of available storage space to accommodate application files, database records, and system, and workflow logs. Machine learning capabilities are implemented using the scikit-learn library, while pandas and NumPy are used for data preprocessing and numerical computations. The system also utilizes the Requests library for API communication with the AI reasoning module. Tabulate is used for displaying formatted data when necessary, and Ollama is integrated with a Large Language Model to provide reasoning explanations for system decisions. The frontend interface is designed using the Bootstrap framework to ensure a responsive and visually appealing dashboard. Development and testing can be carried out using development environments such as Visual Studio Code or any other Python-compatible integrated development environment.

Life Cycle Model

Fig: 1 Agile Model

The development of the Multiple Agent inventory management system follows the Agile software development methodology, which supports iterative development and continuous improvement throughout the project lifecycle. Agile allows the development team to divide the project into smaller modules and progressively refine each component through repeated development cycles. The process begins with the requirement gathering and planning phase, during which system requirements such as inventory management functionality, demand forecasting capabilities, and dashboard design are identified and prioritized. These requirements are then divided into smaller modules to simplify implementation.During the system design phase, the architecture of the application is defined, including the structure of the frontend interface, backend server, database schema, multi-agent components, and AI reasoning mechanisms. The design remains flexible so that new features or improvements can be incorporated during later stages of development. In the development phase, the system modules are implemented gradually through multiple iterations. Features such as inventory tracking, manual sales recording, demand forecasting, stock evaluation, and automated purchase order generation are developed step by step. The testing phase ensures that each module functions correctly and interacts properly with other components. During this stage, system functionalities such as sales transactions, forecasting accuracy, stock updates, and agent coordination are carefully evaluated to detect and correct errors. Once testing is completed successfully, the system moves to the deployment phase, where the application becomes accessible through the web-based dashboard. Finally, the feedback and improvement phase allows continuous enhancement of system features. User feedback and performance evaluations are analyzed to refine functionalities such as workflow monitoring, system logs, and AI-based reasoning explanations, ensuring that the system continues to evolve and improve over time.

System Design

Architecture

System architecture describes the structural framework of a software system and explains how its different components interact to process requests and perform operations. It identifies the major modules within the system, defines their relationships, and specifies the sequence in which these modules communicate during execution. A well-structured architecture improves system maintainability, scalability, and overall understanding of the system behavior. In the proposed Multiple Agent Inventory Management System, the architecture is organized into two main perspectives: software architecture and technical architecture, each describing different aspects of the system structure and technology implementation.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

Software Architecture

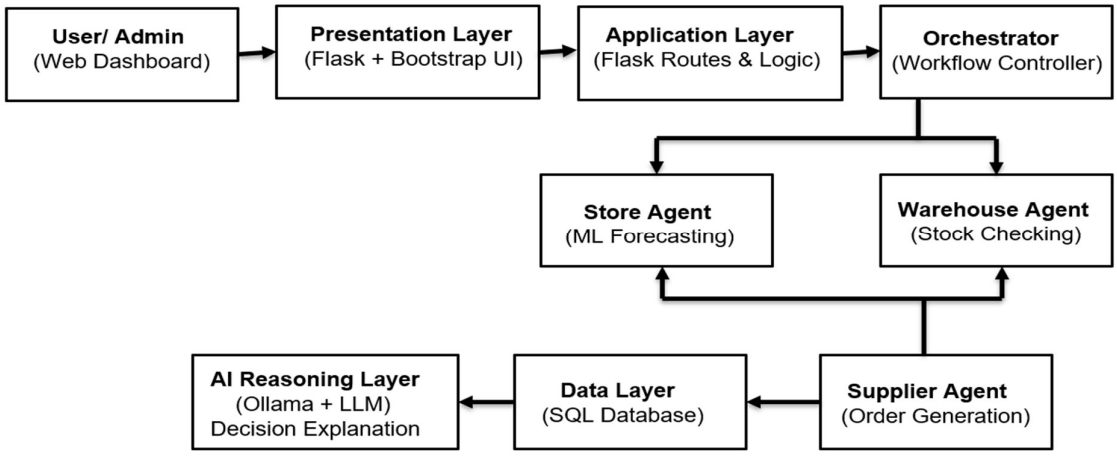


Fig: 2 Software Architecture

software architecture represents the internal organization of the application and illustrates how different system components interact with each other to perform inventory management operations. The system follows a **multi-agent architecture**, where multiple specialized agents perform distinct tasks while cooperating through a centralized coordinator. The system includes a **web-based user interface dashboard** that allows users to monitor inventory levels, simulate sales activities, execute system workflows, and review system logs. The **Orchestrator** acts as the central controller that coordinates the activities of all agents and ensures that each stage of the workflow is executed in the correct order. The **Store Agent** is responsible for performing demand forecasting by analyzing historical sales data using machine learning techniques. The **Warehouse Agent** evaluates the predicted demand and compares it with current inventory levels to determine whether stock replenishment is required. When the inventory falls below the required threshold, the **Supplier Agent** automatically generates purchase orders with the necessary product information and quantity details. All system data, including product records, sales history, forecast results, purchase orders, and reasoning logs, are stored in a **SQLite database**. In addition, an **AI reasoning layer powered by a Large Language Model through Ollama** generates human-readable explanations for system decisions. This modular design allows each component to function independently while still collaborating effectively through the orchestrator, thereby improving system scalability and maintainability.

Technical Architecture

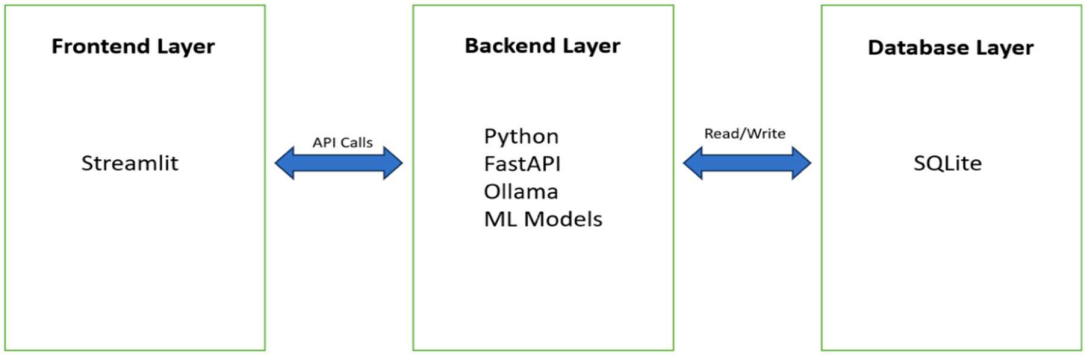


Fig: 3 Technical Architecture

The technical architecture illustrates the technology stack and layered structure used to develop and operate the system. It describes how frontend interfaces, backend services, machine learning models, and data storage components interact within the system environment. The architecture begins with the **presentation layer**, which is implemented using HTML, CSS, Bootstrap, and JavaScript to create a responsive and interactive web dashboard. This interface allows users to access system features such as inventory monitoring, workflow execution, and order tracking. The application layer is built using the Flask web framework, which manages HTTP requests, processes user actions, and coordinates communication between system modules. The **intelligence layer** contains the

machine learning components responsible for demand forecasting using the scikit-learn library, while also integrating a Large Language Model through Ollama to generate reasoning explanations. The **data processing layer** utilizes pandas and NumPy libraries to clean, organize, and analyze historical sales data before it is used for forecasting. Finally, the **database layer** uses SQLite to store structured information such as inventory records, sales transactions, forecast results, purchase orders, and system logs. This layered architecture ensures a clear separation of responsibilities between system components, which improves maintainability and simplifies future system upgrades.

Implementation

The Multiple Agent inventory management system is implemented to automate various retail inventory operations through the integration of machine learning techniques, rule-based decision mechanisms, and AI-powered reasoning capabilities. The system performs important tasks such as forecasting product demand, evaluating stock levels, generating purchase orders, and providing explanations for system decisions through a web-based interface. The implementation process begins with the development of a user interface in the form of a web dashboard that allows users to interact with the system. Through this dashboard, users can monitor inventory data, record sales transactions, simulate daily sales activities, execute automated workflows, and review purchase orders and system logs.All system data is stored in a SQLite database, which maintains records of products, sales history, forecast results, purchase orders, workflow activities, and reasoning logs. Before demand forecasting is performed, historical sales data undergoes preprocessing using pandas and NumPy libraries to ensure that the data is properly structured and cleaned for analysis. The Store Agent applies a Linear Regression model from the scikit-learn library to analyze historical sales trends and predict future demand. These predictions are then evaluated by the Warehouse Agent, which compares forecasted demand with current inventory levels using predefined rule-based thresholds. If the predicted demand exceeds the available stock, the Supplier Agent automatically generates a purchase order containing the required quantity, supplier information, and estimated cost. Communication between these agents is coordinated by the Orchestrator, which ensures that forecasting, stock evaluation, and order generation occur sequentially and efficiently. To enhance transparency and explainability, the system integrates Ollama with a Large Language Model, which produces reasoning logs that explain why certain inventory decisions were made.

Technologies Used

The implementation of the system utilizes a combination of frontend, backend, machine learning, and database technologies. The frontend interface is developed using HTML for structural layout, CSS for styling, Bootstrap for responsive design, and JavaScript for client-side interactivity. The backend logic is implemented using Python, while the Flask web framework manages HTTP requests, application routing, and server-side processing. Machine learning capabilities are implemented using the scikit-learn library, which provides the Linear Regression model used for demand forecasting. The system integrates Ollama to connect with a Large Language Model that generates reasoning explanations for inventory decisions. The SQLite database stores product information, sales transactions, forecasts, purchase orders, and system logs. The application is typically deployed using a local Flask server during development and testing, with the option for future deployment in cloud environments for improved scalability. Session management in Flask is used to maintain user authentication and request validation. Data processing tasks are handled using pandas and NumPy, while the Requests library enables communication with the AI reasoning layer.

Testing

Testing is a crucial stage in the development of the Multiple Agent inventory management system, as it ensures that each module operates correctly and that the complete system workflow performs as expected. The testing process validates functionalities such as demand forecasting, stock evaluation, purchase order generation, and reasoning explanation generation. To achieve comprehensive validation, the system is evaluated using unit testing, integration testing, and system testing approaches. Each agent component—including the Store Agent, Warehouse Agent, Supplier Agent, and Orchestrator—is first tested individually before evaluating the performance of the entire workflow.

Unit Testing

Unit testing focuses on verifying the correctness of individual system modules. The Store Agent is tested to ensure that the forecasting algorithm produces accurate predictions when provided with historical sales data. The Warehouse Agent is evaluated to confirm that the threshold-based logic correctly determines whether inventory levels are sufficient. The Supplier Agent is tested to verify that purchase order quantities and costs are calculated correctly when stock shortages occur. The database management component is also tested to confirm that CRUD operations such as data insertion, retrieval, updating, and deletion are functioning properly. In addition, the integration of the Large Language Model is tested to ensure that reasoning explanations are successfully generated for each decision. Each function is executed using sample input data to verify that the outputs match the expected results.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

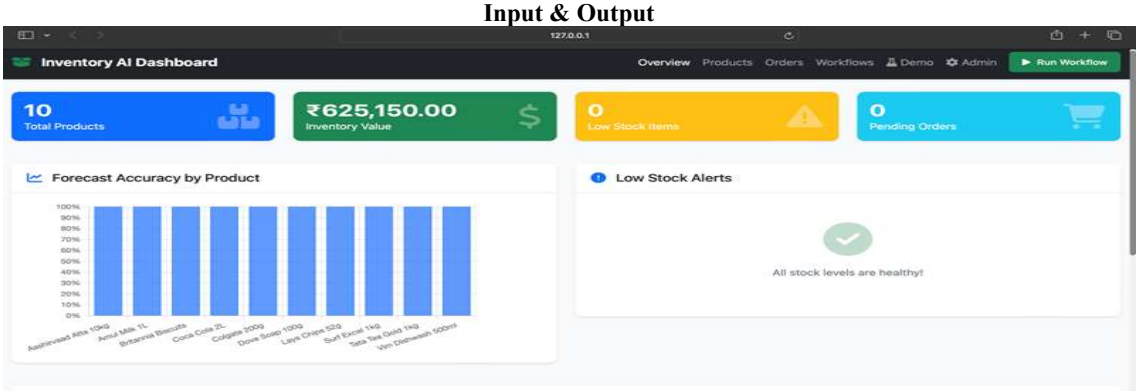
Integration Testing

Integration testing evaluates how different modules communicate with each other and ensures that data flows correctly across system components. In this stage, forecast results generated by the Store Agent are verified to ensure they are correctly transferred to the Warehouse Agent for inventory evaluation. The generation of reorder signals is also tested to confirm that the Supplier Agent is triggered whenever stock levels fall below predicted demand. Additionally, the orchestrator is tested to verify that it executes the agents in the correct sequence. The system also checks whether purchase orders are properly stored in the database and whether reasoning logs are generated after each workflow execution. These tests confirm that the interactions among system modules operate smoothly.

System Testing

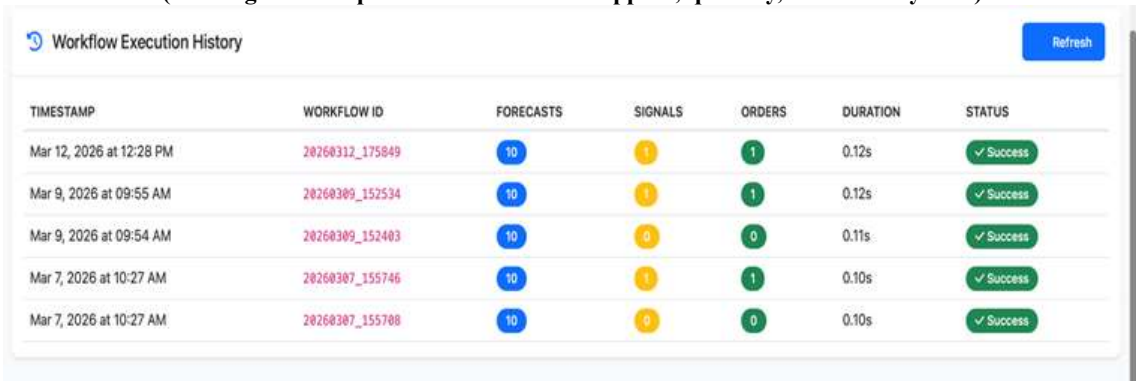
System testing evaluates the complete application to confirm that all modules work together correctly in real operational scenarios. During this stage, users execute the workflow from the dashboard, which triggers the forecasting process, stock evaluation, and automatic purchase order generation when necessary. The system also verifies that reasoning explanations and logs are displayed correctly on the dashboard and that database records are updated after workflow execution. This testing stage ensures that the entire system performs reliably and meets the intended functional requirements.

Input & Output



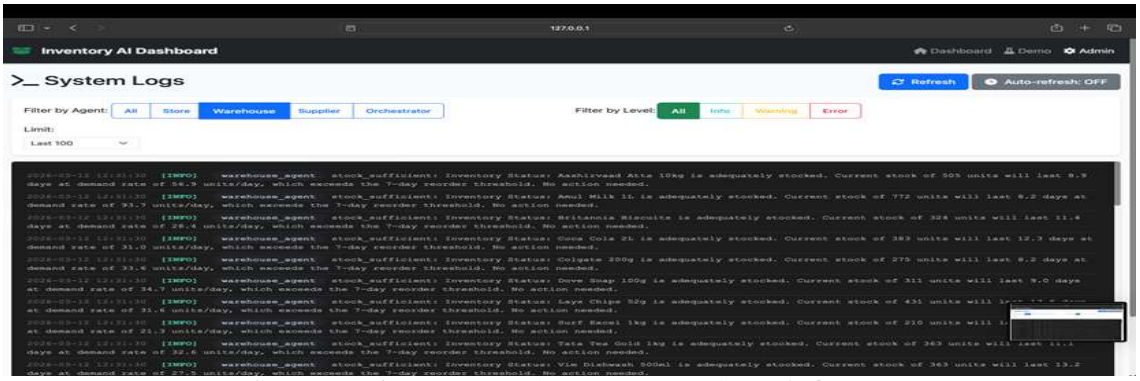
Screenshot 1:InventoryManagementDashboardOutput
(Shows total products, inventory value, low stock items, and forecast accuracy)

Screenshot 2:PurchaseOrderManagementOutput
(Shows generated purchase orders with supplier, quantity, and delivery date)

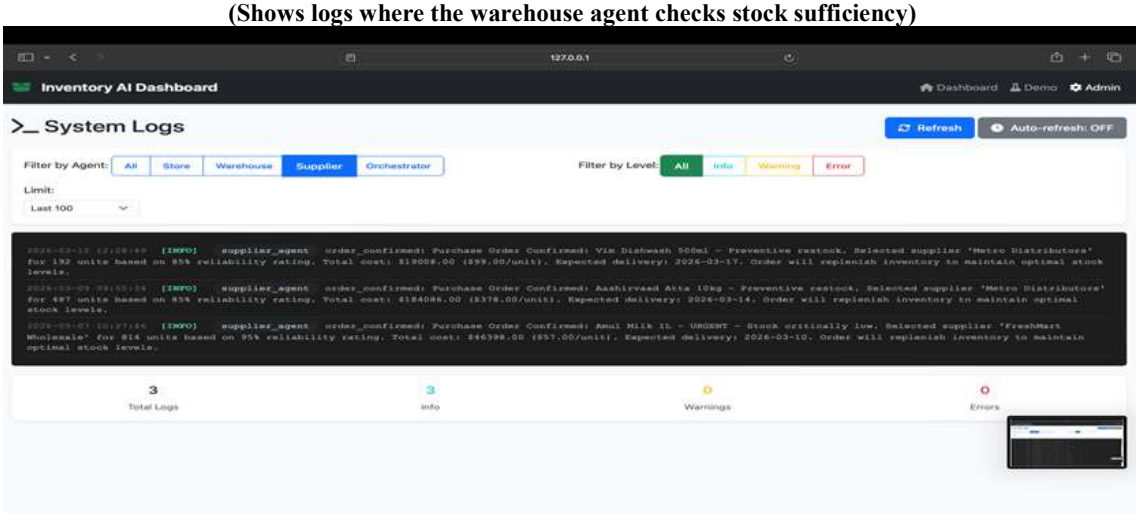


Screenshot 3:WorkflowExecutionMonitoringOutput
(Shows workflow history with forecasts, signals, orders, and execution status)

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



Screenshot 4 : Warehouse Agent Inventory Analysis Output
(Shows logs where the warehouse agent checks stock sufficiency)



Screenshot 5: Supplier Agent Purchase Order Decision Output
(Shows logs where the supplier agent confirms purchase orders)

Conclusion

The proposed **Multiple Agent inventory management system** demonstrates how modern technologies such as machine learning, rule-based decision mechanisms, and AI-based reasoning can be effectively integrated to build an intelligent inventory control platform. The system automates critical inventory operations including demand forecasting, stock evaluation, and purchase order generation through a structured multi-agent framework, thereby reducing manual effort and improving operational accuracy. Demand forecasting is performed using a Linear Regression model that analyzes historical sales data to estimate future product demand. These predictions are then evaluated by the Warehouse Agent, which applies threshold-based decision logic to determine whether current stock levels are sufficient or if replenishment is required. This mechanism helps prevent both stock shortages and excessive inventory accumulation. When inventory levels fall below the required threshold, the Supplier Agent automatically generates purchase orders to ensure timely procurement of products. All operational data, including inventory records, sales transactions, forecast results, and purchase orders, are securely maintained within a SQLite database for reliable data management. An important feature of the system is the integration of a Large Language Model through Ollama, which generates reasoning logs that clearly explain the decisions made by the system. These explanations improve transparency and allow users to better understand the automated processes. The Orchestrator component ensures smooth coordination among the different agents, enabling efficient workflow execution and making the system scalable and maintainable. In addition, the Flask-based web dashboard provides a user-friendly interface through which users can monitor inventory status, simulate sales activities, and track system workflows effectively. Overall, the project presents a cost-effective, reliable, and intelligent solution that enhances decision-making in inventory management and demonstrates the practical application of machine learning-driven automation in real-world retail environments.

Future Scope

Although the proposed system provides a strong foundation for automated inventory management,

several enhancements can be explored in future work to further improve its capabilities and scalability. One potential improvement involves incorporating more advanced forecasting techniques such as ARIMA models, Long Short-Term Memory (LSTM) networks, or other deep learning approaches that can better capture complex patterns and seasonal variations in sales data. These models could significantly improve prediction accuracy, especially in dynamic retail environments. The system could also be extended to support real-time data integration by connecting with sales platforms, point-of-sale systems, or IoT-based inventory sensors through APIs, allowing continuous monitoring of inventory levels and immediate response to demand fluctuations. Deploying the application on cloud infrastructure would further enhance scalability, accessibility, and data security, enabling the system to support larger organizations with extensive product inventories. Additional intelligent agents could also be introduced, such as a Pricing Agent for implementing dynamic pricing strategies or a Demand Optimization Agent to assist in strategic inventory planning and business decision-making. The user interface could be improved with advanced data visualization features and mobile application support to make monitoring and management more convenient for users. Furthermore, integrating the system with enterprise resource planning (ERP) platforms and supplier management networks would enable complete automation of supply chain activities, including procurement and supplier coordination. Enhancing the Large Language Model reasoning module to provide predictive insights, recommendations, and decision-support suggestions could also increase the intelligence of the system. With these improvements, the proposed solution has the potential to evolve into a fully automated, scalable, and enterprise-level inventory management platform capable of supporting complex retail operations and modern supply chain systems.

REFERENCES

[1] Y. Yang, M. Wang, J. Wang, P. Li, and M. Zhou, "Multi-Agent Deep Reinforcement Learning for Integrated Demand Forecasting and Inventory Optimization in Sensor-Enabled Retail Supply Chains," *Sensors*, vol. 25, no. 8, 2025.

[2] Y. Quan and Z. Liu, "InvAgent: A Large Language Model-Based Multi-Agent System for Inventory Management in Supply Chains," *arXiv preprint arXiv:2401.xxxxx*, 2024.

[3] F. F. Bastarianto, T. O. Hancock, C. F. Choudhury, and E. Manley, "Agent-Based Models in Urban Transportation: Review, Challenges, and Opportunities," *European Transport Research Review*, vol. 15, no. 1, pp. 1–19, 2023.

[4] S. Rasal, "LLM Harmony: Multi-Agent Communication for Problem Solving," *arXiv preprint arXiv:2401.01312*, 2024.

[5] E. A. Silver, D. F. Pyke, and D. J. Thomas, *Inventory and Production Management in Supply Chains*, 4th ed. Boca Raton, FL, USA: CRC Press, 2016.

[6] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*, 7th ed. Boston, MA, USA: Pearson, 2019.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[8] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[9] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] 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.

[11] M. Wooldridge, *An Introduction to Multi-Agent Systems*, 2nd ed. Chichester, UK: Wiley, 2009.