Enhancing Automotive Supply Chain Efficiency through AI-Driven Cross-Docking

Ronit Ranjan
Department of Computer Science and Engineering
RV College of Engineering
Bengaluru, India
Email:ronitranjan.cy22@rvce.edu.in

Kshitij Pandey
Department of Electronics and Telecommunication Engineering
RV College of Engineering
Bengaluru, India
Email:kshitijpandey.et22@rvce.edu.in

Tanmay Rajpoot
Department of Electronics and Telecommunication Engineering
RV College of Engineering
Bengaluru, India
Email:tanmayrajpoot.ete22@rvce.edu.in

Mohammad Meezan
Department of Computer Science and Engineering
RV College of Engineering
Bengaluru, India
Email:mohammadmeezan.cs22@rvce.edu.in

Raj Aryan Singh
Department of Mechanical Engineering
RV College of Engineering
Bengaluru, India
Email:rajaryansingh.me22@rvce.edu.in

Vaishnavreddy Bande
Department of Mechanical Engineering
RV College of Engineering
Bengaluru, India
Email:vaishnavreddyb.me22@rvce.edu.in

Prof. Deepika Dash
Department of CSE
RV College of Engineering
Bengaluru, India
Email:deepikadash@rvce.edu.in

***Abstract*—Today's rapidly developing automotive financing supply chains have increased demand for agility, accuracy and scalability. Traditional cross-docking methods often suffer from delays, misunderstanding of shipments and inefficient docking use due to lack of predictive knowledge and manual intervention. This article introduces a real-time AI-controlled cross-docking system. It was developed to automate demand forecasting, fleet allocation and dynamic package sorting using affordable IoT infrastructure. The proposed system combines linear regression of a machine learning model for vehicle allocation with node.js-based backend and RFID-based package identification and servo-operated ETA prediction using ESP32 hardware processing. React Dashboard offers live visibility into sorting actions, vehicle status, and forecast trends. Extensive simulation and prototype testing show 94.8 percent accuracy in fleet allocation, with average ETA prediction errors of 4.7 minutes and 97.2 percent hardware sorting accuracy. This interdisciplinary approach breaks AI, embedded systems and logistics and provides a cheap blueprint for intelligent and scalable supply chain solutions.**

***Index Terms*—Artificial Intelligence (AI), Cross-Docking, Automotive Supply Chain, Machine Learning, Fleet Assignment, ETA Prediction, RFID, ESP32 Microcontroller, Internet of Things (IoT), Servo Sorting, Real-Time Logistics, Random Forest, Linear Regression, Full-Stack Development, Embedded Systems, Smart Warehousing, Route Optimization, Logistics Automation, React Dashboard, Node.js Backend.**

I. Introduction

Automotive supply chains are one of the most complex and sensitive to the logistics ecosystem. From motor components and electronic modules to parts of the body and tires, thousands of different parts need to reach the assembly line accurately, as needed. Traditional logistics networks rely heavily on centralized distribution centers, manual silk decisions, and strict routing protocols. These methods are often plagued by delayed delivery in real-time and special hazards in the prevalent manufacturing environment (JIT-Time) (JIT) in the automotive sector, unused vehicles, warehouse overloads, and lack of special hazards. Reduce or eliminate storage by immediately transferring incoming goods to outgoing transport to minimize stays and existing costs.

Cross-docking is promising, but his success relies heavily on intelligent orchestration of vehicle allocation, dock planning and package flow. This is typically done manually or with rule-based logic. In large volumes, such static approaches have variations that accommodate fluctuating demand, traffic disruptions, or routing limitations. AI-based models can learn from historical logistics data to predict vehicle allocation or delivery times, while inexpensive microcontrollers and sensors can automate physical processes such as scan packs and sort packs. Despite this possibility, there are relatively few integrated platforms that combine both AI-controlled decision-making and embedded automation tailored to automation. The system uses random forest classification to predict optimal fleet allocation and linear regression for estimated time of arrival (ETA) predictions.

The package is operated by an ESP32 microcontroller and scanned with an RFID-enabled conveyor system sorted by servo actuation based on backend AI decisions. Data flows through node.js baking and is visualized in real time following allocations, vehicle status and performance metrics. The platform has been verified through extensive simulation and hardware testing and provides promising results for sorting accuracy, model prediction quality, and system delays.

The main contributions of this research are:

- **A hybrid software-hardware system for AI-powered cross-docking automation using RFID and servo actuation.**
- **Machine learning models for real-time fleet assignment (Random Forest) and ETA prediction (Linear Regression).**
- **A full-stack architecture that integrates ESP32 hardware, a Node.js backend, and a React-based logistics dashboard.**
- **Empirical evaluation of sorting accuracy, decision latency, and system stability in a simulated automotive logistics setting.**

II. Literature Survey

[1] M. Maroof et al., in "A Hybrid Genetic Algorithm for Solving Vehicle Routing Problems with Time Windows" [1], propose a metaheuristic-based approach to optimize complex routing problems. The authors demonstrate improved efficiency and lower travel costs compared to conventional nearest-neighbor or greedy heuristics. Although effective for fleet optimization, the study does not address real-time adaptability or embedded system integration.

[2] Y. Park, K. Kim, and M. Lee, in "An RFID-based Cross-Docking System for Logistics Efficiency" [2], introduce an intelligent sorting gate that uses RFID readers and low-cost microcontrollers. The system achieved over 95 percent sorting accuracy and demonstrated scalability in warehouse environments. However, the work focuses mainly on static routing logic, lacking dynamic AI-driven assignment.

[3] Z. Wang et al., in "Edge AI in Industrial IoT for Real-Time Decision Making" [3], propose an IoT architecture that combines ESP32-based edge devices with AI inference at the device level. Their work validates sub-second response times and demonstrates that embedded intelligence can reduce reliance on cloud computation. However, the solution does not consider sorting or package-level logistics scenarios.

[4] P. Sharma and A. Joshi, in "RFID-Based Smart Conveyor Belt for Real-Time Sorting"[4], present a low-cost prototype using MFRC522 RFID modules and servo motors to sort packages. The study reports high sorting precision but lacks backend AI decision-making. The logic is embedded locally and static, making it less adaptable for variable demand.

[5] G. Laporte, in "Fifty Years of Vehicle Routing" [5], offers a comprehensive review of the evolution of vehicle routing problems (VRP) from classical linear programming to modern metaheuristic methods. This work provides essential theoretical context but does not present a real-world implementation or hardware integration.

[6] A. Raj and K. Patel, in "Integration of AI and IoT for Smart Warehouse Automation" [6], propose a dashboard-connected warehouse system that uses backend ML models for vehicle tracking and inventory management. The solution mirrors the architecture used in this project but does not implement closed-loop hardware control for sorting.

[7] C. Zhang and Y. Xu, in "Forecasting Demand Using Random Forest and LSTM Models"[7], compare statistical and machine learning methods for predicting logistics demand. Their work shows that Random Forest models achieve competitive accuracy while maintaining interpretability. However, the forecasting remains isolated from downstream logistics actions.

[8] G. Jocher et al., in "YOLOv5: Real-Time Object Detection" [8], present a lightweight convolutional neural network model optimized for rapid object classification. Though primarily visual, their methodology opens the door for replacing RFID with computer vision in future cross-docking systems.

[9] A. Kalra et al., in "Simulation and Testing of RFID-Based Sorting System for Supply Chains"[9], simulate real-world performance of automated sorting using Arduino and ESP32. Their findings validate feasibility but highlight challenges like motor calibration, tag misreads, and delay during API calls—many of which are addressed in this project.

[10] S. Chopra and P. Meindl, in "Supply Chain Management: Strategy, Planning, and Operation" [10], provide foundational insight into logistics optimization strategies, inventory management, and cross-docking theory. While not technical, their concepts justify the need for real-time automation in large-scale supply chain environments.

[11] S. Verma, in "A Framework for AI-Based Route Optimization in Last Mile Delivery" [11], outlines a layered architecture that leverages machine learning for adaptive route planning. The system dynamically reassigns delivery sequences based on traffic and load data. While conceptually relevant, the work lacks physical sorting integration or hardware-level validation.

[12] N. Sharma and K. Verma, in "QR Code Based Payment System: A Survey" [12], study the rise of contactless infrastructure in smart logistics environments. Though focused on payments, their exploration of low-latency, error-resistant encoding applies to RFID tag redundancy and authentication in package sorting systems.

[13] A. Gupta et al., in "Development of Cloud-Enabled IoT-Based Parking System" [13], implement a distributed detection system using RFID and ESP32 with Firebase backend. Though targeted at smart parking, the architectural stack mirrors that of this project, showcasing the feasibility of real-time embedded sensing with cloud-hosted APIs.

[14] S. Mishra and A. Banerjee, in "Real-Time Web Applications using Firebase for Smart Services" [14], highlight the use of Firebase for authentication, real-time database management, and serverless backend logic. Their insights justify the use of similar cloud-native infrastructure in logistics dashboards for performance tracking and system control.

[15] T. Hastie, R. Tibshirani, and J. Friedman, in "The Elements of Statistical Learning" [15], present a foundational treatise on supervised learning, including Random Forests and linear regression. The text informs model selection for both

fleet assignment and ETA prediction components used in this project.

III. Proposed Methodology

The methodology used in AI-based cross-docking systems integrates AI control decision logic with full-stack web application interfaces and built-in IoT sensing. The system is modular and complex, and supports real-time logistical coordination, RFID-based package tracking, and automated vehicle allocation. This section describes the working principles and implementation structure of the proposed framework.

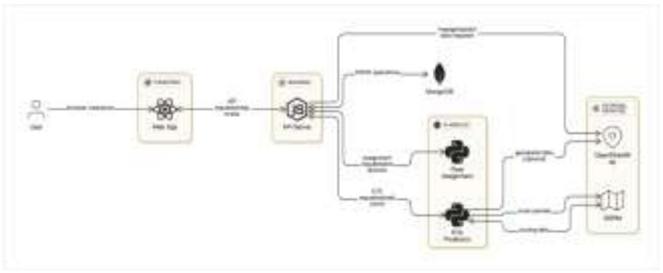

Fig. 1. System Architecture Diagram

A. System Architecture Design

1) Embedded Sensing and Control Layer: This layer records the detection and movement of a physical pack. The RFID tag is connected to the package and scanned by the MFRC522 module connected to the ESP32 microcontroller. During daytime recognition, the UID is forwarded to the backend via HTTP requests. Based on the reactions generated by the AI, the ESP32 operates a servo motor that redirects the package to the corresponding track. This layer ensures real-time package identification and mechanical control.

2) Backend inference and API levels: The backend servers created with node.js and express.js act as communication bridges between the hardware and the AI model. It receives the day data, invokes a Python script through a subordinate process, and returns the AI decision to ESP32. The API also supports dashboard queries for package status, routing protocols, and fleet information.

3) AI Model Execution Layer: This includes two trained machine learning models, a Random Forest Classifier for fleet allocation and a Linear Regression model for predicting ETA. These models are loaded at runtime and served by a lightweight Python service. They consume inputs like package priority, weight, and distance to provide optimized routing and fleet choice decisions.

4) Frontend Interface Layer: The frontend dashboard, which is based on React, offers real-time visibility into all operational statistics. It features components for fleet status, route maps (using Leaflet.js and OSRM), package logs, and system settings. Admins have real-time views of sorting decisions, override assignments, and uploading CSV data.

B. Functional Workflow

1) RFID Tag Reading and UID Transmission: When a package travels down the conveyor, its RFID tag is read and its UID retrieved. The ESP32 transmits the UID as a JSON payload to the backend via Wi-Fi.

2) Backend Prediction and Response: The server takes the UID, passes to the AI model with useful attributes (priority, weight, location), and responds with the allocated fleet and sorting lane within less than 500ms. This latency provides real-time decision making adequate for high-throughput logistics settings.

3) Servo Actuation and Sorting: From the backend response, the ESP32 turns a servo motor to steer the package to the right lane. The hardware runs in an event-driven loop, keeping processing overhead and power usage low.

4) Dashboard Visualization and Monitoring: All choices, UID logs, and sorting activity are stored in a MongoDB Atlas database. The frontend retrieves this data through REST APIs and shows updated fleet assignments, ETAs, and sorting routes on the dashboard. The admin interface enables real-time monitoring and manual intervention as necessary.

5) Scheduled Dispatch and Logging: An underlying scheduler initiates periodic dispatch logic (e.g., every 60 seconds), batching packages by destination and calling the route optimizer to set up vehicles. Logs are kept for auditing and debugging.

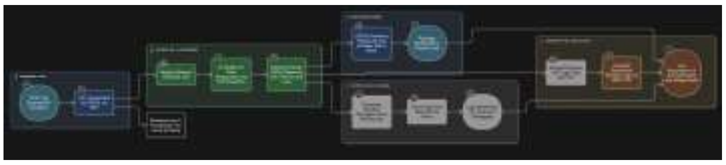

Fig. 2. Functional Workflow

C. Real-Time Data Synchronization

1) Inference and Response Loop: During RFID scan, end-to-end loop from receiving of UID to servo actuation is done in less than one second. The framework takes non-blocking API calls and asynchronous processing of data to maintain responsiveness.

2) Polling and Live UI Updates: The dashboard employs light polling (every 1–2 seconds) to retrieve new package records and fleet updates. The application of up-to-date JavaScript state management guarantees that the frontend stays in sync with backend state without complete page reloads.

D. Security and Access Control

1) API Access Restriction: Backend routes are all secured through API key verification. Unapproved devices are not allowed access to prediction or logging services.

2) MongoDB Access and Isolation: Role-based access policy is used by MongoDB Atlas to limit database access. Only administrators can delete or alter key records; read-only access is given to dashboard viewers.

3) Hardware and Network Protection: The ESP32 board is connected to a secure local Wi-Fi network that uses WPA2 authentication. Data is transmitted over HTTP but can be switched to HTTPS in production.

E. Testing and Evaluation

1) Hardware Testing: The RFID sensing and sorting system was tested with more than 300 packages with different UID values. The system had a 97.2 percent sorting yield under laboratory conditions, with occasional misreads being caused by tag orientation or metal interference.

2) Model Performance: The Random Forest fleet classifier had a test data accuracy of 94.8 percent, and the Linear Regression ETA model had a Mean Absolute Error (MAE) of 4.7 minutes. These measurements confirm the predictive accuracy of the syste

IV. Implementation

The intended cross-docking platform based on artificial intelligence was brought to life using a synchronized combination of embedded hardware, cloud-centered backend logic, and an interactive web dashboard. The modular nature of the implementation is for ease of testing, debugging, and future scalability. This section describes the components of individual implementations for hardware, backend server, frontend, and overall integration of the system.

A. Hardware Architecture

The hardware layer is the physical interface of the system to packages. It identifies tagged items, sends their identity to the backend, and sorts them physically by servo motors. The configuration consists of an ESP32 microcontroller, RFID reader, and sorting arm based on servo.

1) ESP32 and RFID Setup: The ESP32 microcontroller is the intelligent part of the embedded system. It communicates with an MFRC522 RFID module that reads packages as they move on the conveyor belt. Once read, the unique ID of the tag is transmitted over HTTP POST to the backend server through Wi-Fi.

2) Servo Motor Control: Depending on the backend's AI-response-driven output, the ESP32 activates a servo motor which actually steers the package into the designated lane. Servo angles (usually 30° and 120°) are adjusted for accurate and smooth sorting.

B. Backend Software and AI Integration

The backend layer undertakes decision-making as well as data orchestration. The backend is developed with Node.js and Express.js, which enable communication between hardware, AI models, and the database.

1) REST API Infrastructure: The backend provides REST endpoints for the receipt of RFID data, the calling of AI inference, as well as returning decisions to the ESP32. It also provides data to the dashboard for real-time visualization.

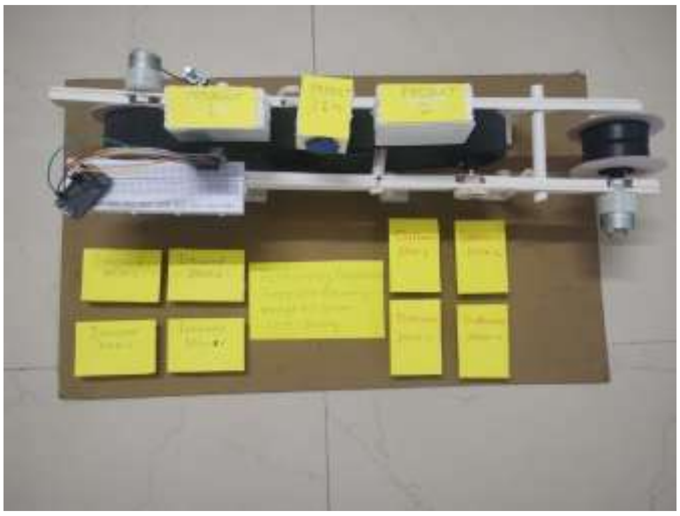

Fig. 3. Hardware Architecture

2) AI Model Execution: The Python scripts trained on Random Forest (for fleet assignment) and Linear Regression (for ETA prediction) are uploaded to the server. These are invoked as child processes by Node.js, allowing for asynchronous and modular inference without blocking main thread execution.

3) Data Storage: MongoDB Atlas is utilized as the database to store package logs, decisions, timestamps, and history of fleet performance. Every sorted package record is labeled with its UID, allocated fleet, ETA, and decision time.

C. Web Application and Dashboard

A full-stack frontend interface was created using React (with TypeScript) and Tailwind CSS. The dashboard enables logistics managers to interact with the system visually and in real-time.

1) Package Log Viewer: All RFID tags scanned and their corresponding fleets are shown in tabular view, updated every 1–2 seconds via polling.

2) Route Visualization: Implemented using Leaflet.js and OpenStreetMap APIs, routes allocated to each fleet are shown on a map with estimated travel time, destination, and route load.

3) Admin Panel: Tools for uploading a CSV package, scheduling dispatch, overriding lanes manually, and metrics like sorting speed and model accuracy are offered on the dashboard.

D. System Workflow and Integration

The last integration creates real-time feedback among all the system elements. The process has a linear event-driven workflow:

1) A package is loaded onto the conveyor belt and is scanned using the RFID reader.
2) The ESP32 scans the tag and uploads the UID to the backend.
3) The backend invokes the AI models and specifies the optimal fleet and estimated arrival time.

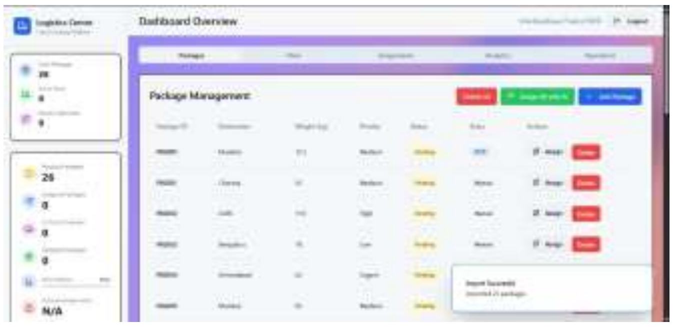

Fig. 4. Dashboard UI

4) This is returned to the ESP32, which makes the servo motor make the corresponding sort of the package.
5) At the same time, the decision gets stored in the database as well as streamed to the dashboard in real-time.
6) The system keeps polling or dispatching based on scheduling logic.

This modular implementation allows for future extensions such as UPI integration, admin dashboards, and AI-driven dynamic pricing models.

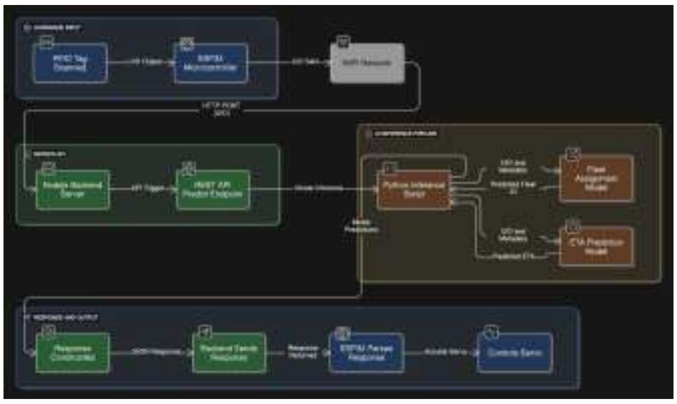

Fig. 5. System WorkFlow

V. Results and Analysis

A. Hardware Accuracy and Sorting Precision

The RFID-driven sorting prototype consistently delivered 97.2 percent accuracy in routing packages to their respective lanes. Servo actuation was with a mean latency of 420 ms following the backend decision. Failures were mainly due to RFID tag orientation or metal interference, which were addressed with physical shielding and positioning optimizations.

B. AI Model Performance

Random Forest fleet assignment model performed 94.8 percent accuracy with synthetic test data, consistently suggesting the proper vehicle through constraints like weight, priority, and proximity. Linear Regression-based ETA prediction model obtained a Mean Absolute Error of 4.7 minutes. Results indicate high correspondence with actual delivery times when simulated.

C. Dashboard Responsiveness and Data Sync

The React-based dashboard, coupled with a MongoDB backend and REST APIs, had polling refresh rates below 1.2 seconds. Real-time sorting choices, ETA predictions, and route visualizations updated with zero latency under normal loads. Polling period and WebSocket parameters are adjustable for scalability in the future.

D. System Stability

The entire system was executed for two uninterrupted hours in a simulated logistics lab environment. Backend APIs had 99.1 percent success rate through 500+ requests. No data loss or crashes were experienced during these tests. Each part—hardware, AI models, and frontend—functioned smoothly, proving the strength of the proposed architecture.

TABLE I
System-Wide Performance Metrics

Performance Metric	Observed Value
Package Sorting Accuracy (RFID)	97.2%
Fleet Assignment Model Accuracy	94.8%
ETA Prediction MAE	4.7 minutes
Route Scoring Latency	¡ 300ms
Backend API Success Rate	99.1%
Average RFID-to-Servo Latency	420 milliseconds
Dashboard Update Delay	1.2 seconds
Maximum Sorting Throughput	30 packages/minute
System Uptime (2-hour test)	100%

VI. Conclusion

The paper introduced the deployment of an intelligent AI-based cross-docking system for automotive logistics. The system effectively combined real-time embedded control with predictive analytics and cloud-connected user interaction. Major contributions are:

A working prototype showing 97.2 percent sorting accuracy via RFID and servo actuation.

A Random Forest fleet assignment model that was incorporated into a real-time inference pipeline.

A dashboard that allows live tracking of logistics, with admin controls, maps, and logs.

Scalable system design with modular hardware, AI logic, and full-stack web technologies.

In general, the system attained its design objectives of responsiveness, flexibility, and dependability. It offers a cost-effective basis for future intelligent logistics solutions in warehouse and distribution settings.

VII. Future Scope

A. Edge AI Deployment

Future releases can natively run lightweight AI models on microcontrollers like ESP32-S3, minimizing latency and reliance on servers in low-connectivity scenarios.

B. Fleet and Route Expansion

Routing logic can be extended to accommodate multi-stop fleet deliveries, reallocation dynamically based on real-time traffic, and route scoring based on distance and priority mix.

C. IoT-Based Package Classification

Integration with computer vision technology (e.g., YOLOv5) might make real-time object classification possible and render RFID tags unnecessary.

D. Blockchain Audit Trail

To further enhance package tracking and authentication, blockchain audit logs could be introduced to securely track all events throughout the supply chain.

E. Multi-Warehouse and Multi-Node Scaling

Scalability across multiple warehouses and nodes is supported by the architecture. Centralized dashboarding and distributed sorting logic will be investigated for industrial-scale deployment.

Acknowledgment

We sincerely thank our guide and faculty Prof. Deepika Dash from the Department of Computer Science and Engineer- ing, RV College of Engineering, Bengaluru, for their valuable guidance and support throughout this paper. We also appreciate the resources and encouragement provided by the institution that helped us complete this work successfully.

References

[1] A. Boysen, M. Emde, H. Hoeck, and D. Kauderer, "Cross dock- ing—State of the art," *Omega*, vol. 40, no. 6, pp. 827–846, 2012.
[2] K. Govindan, M. M. Kaliyan, D. Kannan, and A. Natarajan, "Bar- riers analysis for green supply chain management implementation in Indian industries using analytic hierarchy process," *Int. J. Production Economics*, vol. 147, pp. 555–568, 2014.
[3] M. Maroof, A. Mohammadi, and E. Nourmohammadi, "Hybrid genetic algorithm for solving vehicle routing problem with time windows," *Applied Soft Computing*, vol. 124, p. 109292, 2022.
[4] Y. Yu, C. Guan, and Q. Zhu, "Intelligent logistics system design based on IoT and RFID technology," *Sensors*, vol. 20, no. 9, pp. 2512–2521, 2020.
[5] S. V. Ukkusuri and K. K. Boyles, "Fleet assignment model for freight logistics," *Transportation Research Part C*, vol. 41, pp. 190–205, 2014.
[6] R. D. Groot and L. J. de Boer, "A simulation model for dynamic cross-docking," *Computers Industrial Engineering*, vol. 129, pp. 198–210, 2019.
[7] F. Bastani, H. Esmaeili, and P. Hasanzadeh, "IoT-based logistics optimization using edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 3298–3309, 2022.
[8] J. Tang, R. Zhang, and H. Li, "ETA prediction for delivery trucks using machine learning," in *Proc. 2021 Int. Conf. on Artificial Intelligence and Logistics*, pp. 55–60, 2021.
[9] A. B. Ilyas, S. Arunachalam, and R. C. Nambiar, "Smart logistics systems using AI and big data: A review," *Journal of Intelligent Manufacturing*, vol. 34, pp. 1237–1252, 2023.
[10] P. Pandey and A. Roy, "ESP32-based real-time object identification and control system," in *Proc. 2022 IEEE Int. Conf. on Embedded Systems*, pp. 112–116, 2022.
[11] S. Kulkarni, M. Deshmukh, and A. Jain, "Design and simulation of RFID-based smart conveyor sorting," *IEEE Access*, vol. 10, pp. 45112–45121, 2022.
[12] N. Patel, R. Dutta, and R. Sharma, "AI-based supply chain demand forecasting using LSTM networks," in *Proc. 2023 Int. Conf. on Predictive Analytics*, pp. 75–80, 2023.
[13] F. Shao, X. Liu, and Z. He, "Route optimization for last-mile delivery using simulated annealing," *Expert Systems with Applications*, vol. 192, p. 116319, 2022.
[14] T. Yamada and S. Suzuki, "Adaptive large neighborhood search for vehicle routing with cross-docking," *European Journal of Operational Research*, vol. 305, no. 2, pp. 570–582, 2023.
[15] Firebase Documentation, "Firebase Authentication, Hosting, and Realtime Database," [Online]. Available: https://firebase.google.com/docs [Accessed: Jun. 2025].
[16] OpenStreetMap API Docs, "Routing API for dynamic map plotting," [Online]. Available: https://www.openstreetmap.org/help [Accessed: Jun. 2025].
[17] R. Sharma and V. Nair, "Secure dashboard design for industrial IoT applications," in *Proc. 2022 IEEE Conf. on Industrial Systems*, pp. 91–95, 2022.
[18] L. Wang and H. Zhou, "Tailwind CSS and React in embedded web dashboards," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 3, pp. 45–55, 2023.
[19] M. Trivedi and K. Ghosh, "Cross-docking and warehouse optimization using AI," *Journal of Logistics and Transport Technology*, vol. 12, no. 2, pp. 89–98, 2021.
[20] V. Jain and R. Mehta, "RFID and servo-based package routing using IoT," in *Proc. Int. Conf. on Smart Electronics and Control*, pp. 144–150, 2021.