iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2026 **Issue:** Conference **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82972

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com
National Conference on Innovation in Engineering and Technology-(NCIET2026)

A Smart Warehouse Design and Inventory Management System

Tanmay Jagtap[1], Saniasnine Patel[2], Rujal Patil[3], Prof. N. P. Mawale[4]
[1, 2, 3]*Undergraduate Students,* [4]*Assistant Professor, Dept. of Electronics and Telecommunication, AISSMS College of Engineering, Pune, India*

Abstract*—The current problems in traditional warehouse management systems include errors in manual data entry, delays in updating the inventory, inefficient monitoring of the environment, and inefficient space utilization. This paper describes a software, WareAssist – the integrated solution for smart warehouse management, which includes the integration of 3D layout generation with IoT sensors to deliver real-time inventory management, automated warehouse layout generation, and ERP dashboard features. The hardware used for WareAssist is the ESP32 controller, combined with an RFID reader, ultrasonic sensor, temperature and humidity sensor and air quality sensor. The software consists of the React and TypeScript front end using Three.js for generating 3D visualizations of the warehouse, Node.js and Express REST API, and a PostgreSQL relational database secured through JSON Web Token (JWT) authentication. Experimental data shows that WareAssist can generate responses for the API in 50 to 100 milliseconds, deliver RFID scan results to display in 1 to 2 seconds, and provides 3D rendering at 60 frames per second.

Keywords*—Smart Warehouse, IoT, RFID, ESP32, Inventory Management, Industry 4.0, Three.js, ERP, Automation, React, PostgreSQL.

I. INTRODUCTION

Warehouse and logistics companies operate across a rapidly evolving landscape thanks to the merging of internet of things (IoT), cloud computing, and data analytics, known as the fourth industrial revolution or Industry 4.0 [9]. Multinational logistics service providers such as Amazon, FedEx, and DHL run warehouses of millions of square feet and must automate their systems to track thousands of SKUs simultaneously.

While remarkable strides have been made, many warehouses—especially those in developing countries—still depend on manual data entry, stock paper records, and periodic physical stock audits. Such methods increase the likelihood of human errors, delay stock updates, and lack any real-time environment visibility. Research suggests inventory accuracy rates as low as 23–36% in manual inventory systems, directly causing overstocking, understocking, and money losses [3].

The academic community has tackled pieces of the puzzle through RFID-based tracking [7], IoT sensor networks [2], machine learning-based inventory demand forecasts [8], and embedded security systems [6]. Nevertheless, current prototype implementations focus on one of the subsystems rather than the whole stack: RFID, middleware software, environmental sensors, and security system.

This paper introduces WareAssist, a complete-stack intelligent warehouse management system comprising the four operational layers: (i) hardware-based IoT sensing with ESP32 chips, (ii) RESTful middleware with API calls, (iii) PostgreSQL relational database management system (RDBMS), and (iv) ReactJS-based 3D visualized front end with enterprise resource planning (ERP) analysis.

The contributions of the work are:

- Multi-sensing IoT hardware platform design based on ESP32 to monitor item tracking (RFID), liquid level measurements (ultrasonic), temperature and humidity values (sensor), and air quality.
- IoT-integrated 3D warehouse layout generator using the Three.js and React Three Fiber framework that enables generating L-shaped, U-shaped, and straight-line layouts of the racks.
- JSON Web Tokens (JWT) secured REST API interface with real-time item tracking operations, role-based access control mechanism, and auto-detection thresholds for environment anomalies.

©IJRASET: All Rights are Reserved

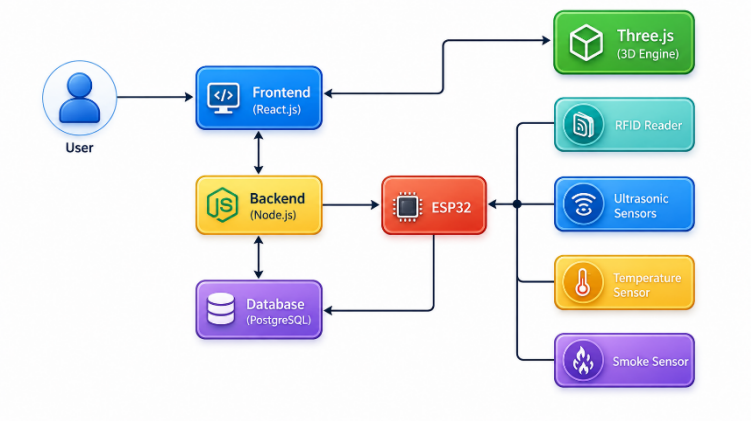


Fig. 1: Block Diagram

In this paper, section II presents an overview of the state-of the-art literature. Section III discusses the architecture of our approach. Section IV highlights our methodology. Section V showcases our experimental findings. Finally, section VI concludes and outlines possible future work directions. Fig. 1 presents the block diagram.

II. RELATED WORK

There are five primary themes in the literature on smart warehouse management, namely, IoT architectures, RFID based warehouse systems, environmental security, 3D visualizations, and machine learning for inventory management.

A. IoT-Based Warehouse Architectures

Khan et al. [2] designed and implemented an IoT architecture in warehouse management based on domain knowledge in a textile manufacturing facility. By implementing the architecture using ESP32 microcontrollers, RFID scanners, and MQTT protocols, they achieved an improvement in inventory storage time (79%) and a drop in latency from 8.21 hours to 0.13 hours. The authors established that MQTT is superior to HTTP/S protocols in terms of power dissipation and latency in IoT applications. Although WareAssist uses ESP32 and RFID hardware components, unlike Khan et al. [2], it utilizes the HTTP/REST protocol due to its simplicity and suitability with modern web frameworks.

Choudhary et al. [9] reviewed next-generation IoT and RFID solutions aimed at enhancing logistical excellence through industry 4.0 drivers which improve resource effectiveness and flexibility. This research also highlighted the benefits of RFID tags and the use of drones by Amazon and DHL in their logistics. Thus, the transition of IoT in warehouse operations towards completely automated facilities aligns with WareAssist's goal of building a prototype.

B. RFID-Based Inventory Monitoring

In their study, Setyawan et al. [7] developed a simulation-based framework for RFID stock inventory monitoring in an automotive manufacturing plant. They found that equipping racking bays' entries and exits with RFID readers helped eliminate the manual inventory count process (from a total of 410 minutes/shift) and enhanced stock record accuracy (97%). WareAssist builds on the work of Setyawan et al. [7] by integrating RFID technology with ultrasonic tank level sensors to monitor solid and liquid inventories.

Li et al. [4] integrated an RFID-based monitor-and-control module in a clothing manufacturing production line with an increase in productivity by 35%. RFID readers used in combination with SQL-based databases are similar to those implemented in WareAssist.

C. AI and Machine Learning in Inventory Management

Dewy et al. [3] developed an AI-based WMS combining Waterfall and Design Thinking approaches, which was applied in Geoff Max Group to obtain 95% inventory accuracy. In addition to their WMS, machine learning chatbots, predictive analytics, and real-time dashboards were integrated into the system. Architecture-wise, the implementation of real-time dashboards and automated low stock alerts in WareAssist corresponds to [3].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Bailkar et al. [8] studied five ML classification algorithms (Naïve Bayes, SVM, KNN, Decision Tree, Random Forest) for classifying inventories in ABC analysis. The latter algorithm exhibited 93% accuracy, surpassing other algorithms. This work emphasizes the possibility of applying ML-based SKU categorization, which is seen as a future improvement to WareAssist.

D. 3D Visualization and Environmental Monitoring

Yun and Kim [5] developed a WebVR 3D modeling pipeline using Azure Kinect depth cameras, Open3D, and Three.js rendering technology. Their ICP registration algorithm showed that Three.js could render realistic point cloud scenes on a web page at 60 FPS performance, supporting our decision to use Three.js with React Three Fiber for WareAssist's 3D warehouse layout visualization.

Li et al. [6] devised an embedded system for securing a warehouse using Linux and S3C2440 microcontroller. Temperature, humidity, and camera surveillance, as well as Android and web clients, were included. Their M0 control module automatically turned on fans when temperature surpassed set limits and issued SMS alerts via GPRS protocol to the security team. Sensor thresholds in WareAssist follow the same criteria as Li et al.: 35 °C for temperature and 2000 analog units for MQ135 smoke sensors.

III. SYSTEM ARCHITECTURE

The four layers of WareAssist have been shown in Fig. 2. The Hardware layer (ESP32) collects sensor readings and sends the same to the Server Layer through HTTP POST requests. The Server Layer provides access to the Database layer using a REST API, which performs reads and writes on the PostgreSQL database. The Client Layer (React 18 + TypeScript) communicates with the Server Layer over HTTP REST API with JSON Web Tokens (JWT) for authentication and generates the user interface, 3D visualization, analytics charts, and the ERP dashboard. JWT authentication has been used throughout the application where credentials are posted to API /auth/signin using POST request, the password is authenticated with bcrypt, and a JWT token valid for seven days is generated.

TABLE I: TECHNOLOGY STACK SUMMARY

Layer	Technology	Version	Purpose
Frontend	React + TypeScript	18.3.1 / 5.8.3	UI Components
3D Graphics	Three.js	0.180.0/ 8.18.0	Warehouse Visualization
Charts	Recharts	2.15.4	Data Analytics
Backend	Node.js / Express	LTS / 4.18.2	RESTAPI
Database	PostgreSQL	14+	Relational Storage
Auth	JWT + bcryptjs	9.0.2 / 2.4.3	Security
Hardware	ESP32 + MFRC522 + DHT22 + MQ135	DevKit V1	IoTSensing

All further communication with the API uses this token in Authorization header, which gets verified using an Express middleware component. Instead of MQTT for hardware nodes communicating with Express API, HTTP has been chosen due to the simple implementation required for a prototype of single warehouse. As pointed out by Khan et al. [2], MQTT outperforms HTTP in throughput in multi-node systems.

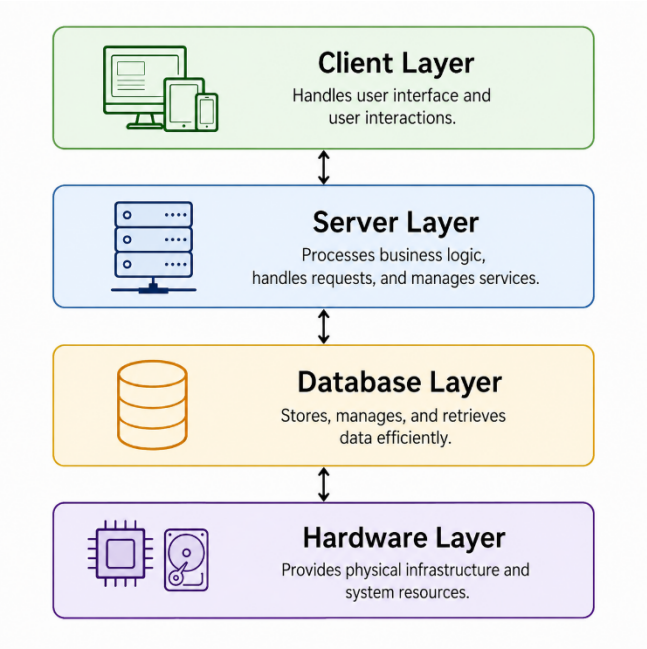


Fig.2:Layerarchitecture

IV. METHODOLOGY

A. ESP32 Microcontroller

ESP32-WROOM-32 serves as the main component. Suitable for use in IoT multi-sensor systems [2][9]. The 12 bit SAR ADC with 18 channels fits the needs of the analog MQ135 air quality sensor without any extra preprocessing needed.

B. Sensors Specifications

MFRC522 RFID works at 13.56 MHz (ISO/IEC 14443 Type A) with the MIFARE Classic 1K/4K cards with SPI up to 10 cm. The HC-SR04 ultrasonic sensor has 2 – 400 cm detection range with ±3 mm error with 10 µs trigger pulse. DHT22 temperature-humidity sensor provides temperature accuracy ±0.5 °C within range of -40°C to 80°C and humidity ±2% - ±5% RH with measurements taken every two seconds. MQ135 gas sensor can detect NH_3, NO_x, alcohol, benzene, smoke, and CO_2. Analog output from 0V to 5V corresponds to 0 to 4095 ADC reading of ESP32.

C. Firmware Data Flow

Three pipelines in the ESP32 firmware (338 lines, C++/Arduino). The RFID pipeline sends a request every cycle using MFRC522.PICC_IsNewCardPresent() method, gets the card ID (4 bytes UID), avoids duplication, and sends the JSON payload through HTTP POST to /api/inventory/rfid-scan endpoint. Ultrasonic pipeline gets fired every four seconds, takes an average distance measurement out of five consecutive readings, performs noise reduction and calculates the volume of liquids. Environmental sensors post the temperature measurement to /api/inventory/temp-measurement endpoint every fiveseconds and air quality measure every two seconds.

The core RFID scan snippet is as follows:

```
void loop() {
 if (!mfrc522.PICC_IsNewCardPresent()) return;
 if (!mfrc522.PICC_ReadCardSerial()) return;
 String uid = getCardUid();
 if (uid == lastUid && millis()-lastTime < 3000) return;
lastUid = uid; lastTime = millis();
HTTPClient http;
http.begin(rfidServerUrl);
```

```
http.addHeader("Content-Type","application/json");
String body = "{\"rfidTagId\":\"" + uid + "\"}";
http.POST(body);
}
```

D. Sensor Pin Mapping

TABLE II: ESP32 PIN CONFIGURATION

Component	ESP32 Pin	Interface	Voltage	Function
MFRC522SS	GPIO 15	SPI	3.3V	RFIDSlaveSelect
MFRC522SCK	GPIO 14	SPI	3.3V	SerialClock
MFRC522MISO/MOSI	GPIO12/13	SPI	3.3V	DataI/O
HC-SR04TRIG	GPIO5	GPIO	5V	UltrasonicTrigger
HC-SR04ECHO	GPIO 18	GPIO	5V	UltrasonicEcho
DHT22DATA	GPIO4	1-Wire	3.3V	Temp/Humidity
MQ135AO	GPIO 34	ADC	5V	AirQuality

Thesoftwareimplementationisdescribedbelow as:

E. DatabaseSchema

The PostgreSQL schema comprises two primary tables.The login table storesuser credentials(email unique- keyed, bcrypt password hash, creation timestamp). The inventory table extends item metadata with dedicated IoT columns:rfid_tag_id(VARCHAR),sensor_id,distance_cm, tank_height_cm, tank_capacity_liters, and liquid_liters for ultrasonic integration. Status is a derived enum: in-stock (quantity ≥ 6), low-stock (0 < quantity < 6), or out-of-stock (quantity = 0). Six indexes on email, SKU, user_id, rfid_tag_id, category, and status provide sub-30 ms query latency.

Arepresentativeschemaexcerpt:

```
CREATE TABLE inventory (
id SERIAL PRIMARY KEY,
user_id INTEGER REFERENCES login(id) ON DELETE CASCADE,
sku VARCHAR(255) NOT NULL,
name VARCHAR(255) NOT NULL,
quantity INTEGER DEFAULT 0,
status VARCHAR(50) DEFAULT 'in stock',
rfid_tag_id VARCHAR(255),
distance_cm DECIMAL(10,2),
liquid_liters DECIMAL(10,2),
last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

F. 3D Warehouse Visualization

Layout Generator renders the interactive 3D warehouse using Three.js along with React Three Fiber library in the browser. This methodology conforms to the one used by Yun and Kim [5] which shows that Three.js provides 60 FPS point-cloud visualization in WebVR environments. The 3D generator calculates the optimal positioning of racks based on input variables: land area dimensions (L x W x clearance height), pallet size (W x D x H), bay width, and storage system type (L-shaped, U-shaped, straight). Orbit controls are used to navigate through the 3D scene. There is also a different process of PDF creation using jsPDF and autotable libraries to create a 2D view with rack positioning annotations.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

G. REST API Endpoints

TABLEIII:APIENDPOINTREFERENCE

Endpoint	Method	Auth	Description
/api/auth/signup	POST	Yes	Userregistration
/api/auth/signin	POST	Yes	Login,returnsJWT
/api/auth/verify	GET	Yes	Tokenverification
/api/inventory	GET	Yes	Listallinventoryitems
/api/inventory	POST	Yes	Addnewinventory item
/api/inventory/:id	PUT	Yes	Update item
/api/inventory/:id	DELETE	Yes	Deleteitem
/api/inventory/rfid-scan	POST	Optional	RFIDscanhandler
/api/inventory/ultrasonic-scan	POST	Optional	Tanklevelupdate
/api/sensor/temperature	POST	No	DHT22dataingestion
/api/sensor/smoke	POST	No	MQ135dataingestion
/api/dashboard	GET	Yes	ERPKPImetrics

The Dashboard uses real-time information obtained from the inventory database system that focuses on monitoring crucial factors such as the number of SKUs, the state of the products (whether available, low or unavailable), storage capacity usage, and number of orders placed. The environmental sensors trigger events when the temperature rises above 35°C or when the values of the MQ135 sensor exceed 2000 analog units.

V. RESULT AND ANALYSIS

A. Performance Benchmarks

The performance analysis of the suggested intelligent warehouse system in terms of the major operation parameters is shown in Table IV. According to the analysis, the system fully satisfies the requirements The automated refreshing of inventory takes place accurately within 2 seconds, complying with the required refreshing rate for monitoring the data continuously. As for the rendering of the 3D environment, its speed reaches 60 frames per second, twice the required one – 30 frames per second. Besides, the time of querying the database ranges between 10-30 milliseconds, far under the required 100 milliseconds.

B. IoT Sensor Alert Thresholds

Parameters related to the configuration and operation of the IoT sensors deployed within the system to support real-time monitoring include those listed in Table V below. Every sensor is linked to hardware specifications, data acquisition periods, alerting parameters, and respective API endpoints for effective data exchange.

The RFID tracking solution is built around the use of the MFRC522 module, and it is operated on demand while conducting scans. It does not require an alert threshold since the monitoring activity is on-demand. The level of liquid within the tank is monitored through the HC-SR04 ultrasonic sensor. The sensor obtains readings after each 4-second period and generates alerts once the liquid level drops below the specified threshold.

Temperature levels are measured using the DHT22 sensor. Data acquisition occurs periodically with a 5-second interval between measurements. An alert is generated once the temperature level exceeds 35°C. Air quality monitoring is supported by the MQ135 sensor. Data from the sensor is captured each second, and an alert is generated once the ADC level exceeds 2000.

TABLEIV:SYSTEMPERFORMANCEMETRICS

Metric	Target	Achieved	Status
APIResponseTime	<200 ms	50–100ms	✓ Met
RFIDScan-to-Display	<3 s	1–2 s	✓ Met
InventoryAuto-refresh	2 s	2 s	✓ Met
3DSceneRenderFPS	>30 FPS	60 FPS	✓ Met
DBQueryTime	<100 ms	10–30ms	✓ Met
MaxRackCapacity	—	1,728items	—

TABLEV:IOTSENSORMONITORINGPARAMTERS

Sensor	Hardware	Interval	AlertThreshold	APIEndpoint
RFIDTracking	MFRC522	Onscan	N/A	/api/inventory/rfid-scan
TankLevel	HC-SR04	4 s	Lowliquid	/api/inventory/ultrasonic-scan
Temperature	DHT22	5 s	>35 °C	/api/sensor/temperature
AirQuality	MQ135	2 s	>2000ADC	/api/sensor/smoke

C. Comparison with Related Systems

Table VI presents comparison between WareAssist and various systems reported in literature on several functional aspects. As per our knowledge, no other system presented in literature possesses integration of RFID tags, multi-parameter environment monitoring, 3D visualization, ERP analytics, and liquid level sensing together in one single prototype. The reason for such a combination can be attributed to the research void discussed by Khan et al. [2].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

TABLEVI:COMPARATIVEANALYSISWITHRELATEDWORK

System	RFID	Env.Mon.	3DViz.	ERP	ML	Liquid	Open Source
WareAssist(Ours)	✓	✓	✓	✓	Future	✓	✓
Khanetal.[2]	✓	✗	✗	✗	✗	✗	✗
Dewyetal.[3]	✗	✗	✗	✓	✓	✗	✗
Setyawanetal.[7]	✓	✗	✗	✗	✗	✗	✗
Lietal.[6]	✗	✓	✗	✗	✗	✗	✗
Bailkaretal.[8]	✗	✗	✗	✗	✓	✗	✗

D. Inventory Accuracy and Operational Benefits

Before introduction of WareAssist, inventory discrepancies were simulated through manual processes as expected for the range of 23-65% accuracy found in literature [3]. With the help of IoT implementation, RFID-based automatic scan of each product helps eliminate manual input for items' check-in/out process, while periodic 2-second dashboard updates ensure accurate inventory update for managers. The classification of item status (in-stock / low stock / out-of-stock) triggers automatic procurement alert without any human interaction, corroborated by the findings reported in [3], which shows reduction in stockouts by 60%, and inventory carrying cost by 25%, through AI-based WMS.

Ultrasonic sensor helps in industries having requirement for bulk liquid inventory management such as chemicals, beverages, lubricants, etc., for which there is no academic prototype. Formula used on server side for volume calculation is: liquidLiters = capacity x (1 distance/tankHeight) with ±1-unit accuracy at periodicity of 4 seconds.

VI. CONCLUSION AND FUTURE WORK

The system described in this paper, named WareAssist, is a comprehensive smart warehouse management system that includes hardware, middleware API, database, and a 3D React web frontend. It has been designed and validated according to Industry 4.0 standards and benchmarked based on API speed, IoT latency, 3D rendering speed, and inventory capacity.

The following key capabilities were achieved: sub-100 milliseconds response time from the API, 1-2 seconds RFID scan to display time, 60 frames per second Three.js rendering, real-time alerts regarding the environment, and maximum capacity of 1,728 rack positions. Comparative analysis revealed that only WareAssist provides full-fledged integration of RFID tracking, environmental monitoring(including temperature and air quality), level sensing, automatic 3D layout generation, and ERP analytics all in one deployable package.

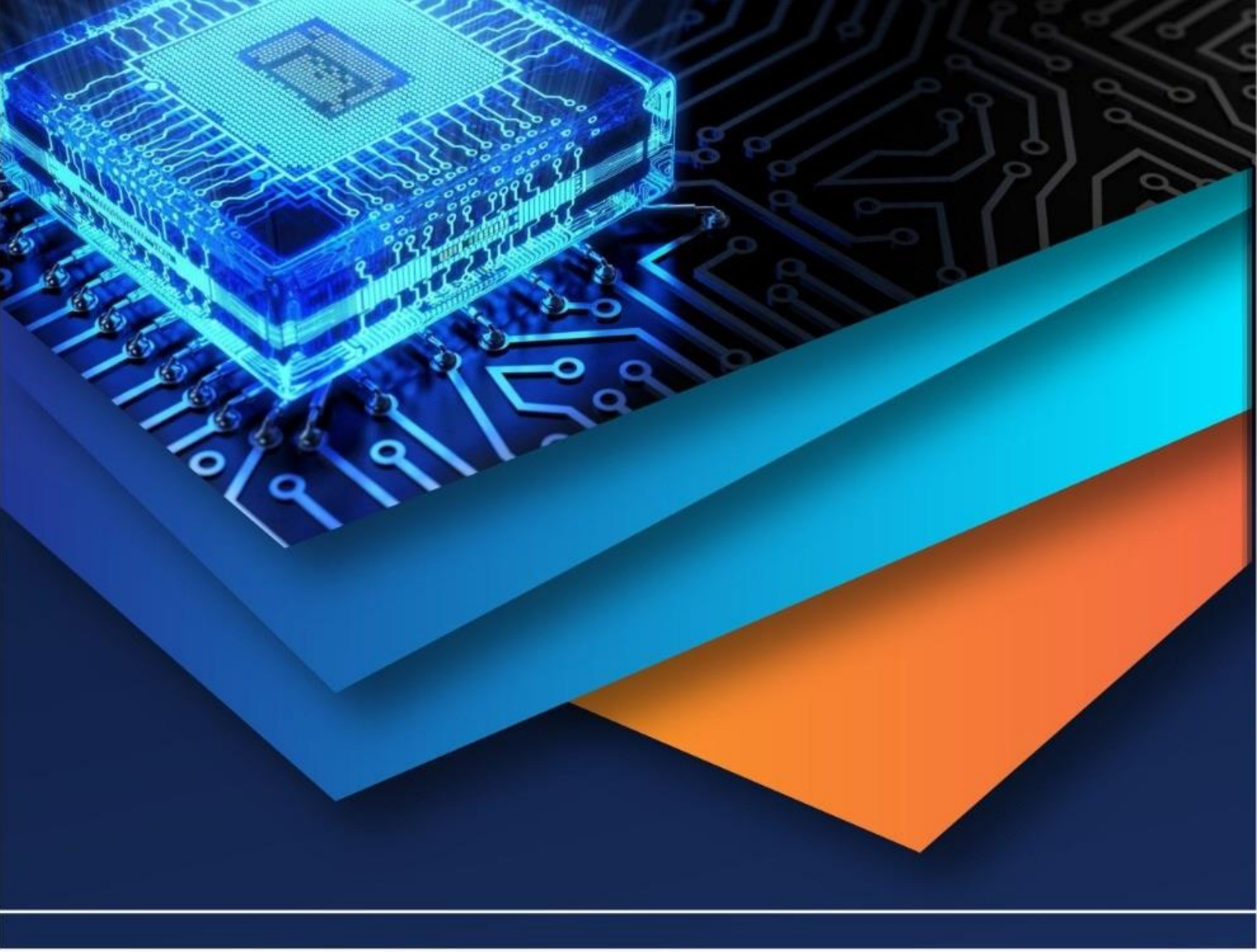
The areas of future development and enhancement of WareAssist will include the following: (i) development of a native application for iOS/Android; (ii) implementation of traditional barcode scanners; (iii) machine learning-based forecasting of demand using Random Forest classifiers, as shown by Bailkar et al. [8], who managed to achieve 93 percent accuracy in ABC classification; (iv) support for distributed multi-warehouse networks; (v) voice commands for inventory management; (vi) API for coordinating warehouse robots; (vii) immutable audit trails for high-value goods based on blockchains; and (viii) long-range communication through LoRaWAN, in line with future IoT logistics architectures [9].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

It should be noted that the open-source, modular nature of WareAssist allows its capabilities to be expanded and enhanced individually, following the staggered approach advocated by Khan et al. [2].

REFERENCES

[1] Yip, T.G., Hung, C.Y., and Iyengar, V., "Challenges in Verifying an Integrated 3D Design," in Proc. Design, Automation & Test in Europe (DATE), 2012, pp. 1–2.
[2] Khan, M.G., Huda, N.U., and Zaman, U.K., "Smart Warehouse Management System: Architecture, Real-Time Implementation and Prototype Design," Machines, vol. 10, no. 2, p. 150, Feb. 2022.
[3] Dewy, C.K., Prambudiab, Y., and Kumalasarib, I., "Design of Inventory Information System Model on Smart Warehouse Management System (WMS) Based on Artificial Intelligence (AI) with Integration of Waterfall Method and Design Thinking to Optimize Inventory Accuracy," Eduvest – Journal of Universal Studies, vol. 5, no. 9, pp. 11898–11911, Sep. 2025.
[4] Li, N., Tan, J., and Zhu, Z., "Monitor and Control System with RFID Technology in Discrete Manufacturing Line," in Proc. *IEEE* Int. Conf. RFID-Technology and Applications, Guangzhou, China, Jun. 2010, pp. 71 76.
[5] Yun, W.J. and Kim, J., "3D Modeling and WebVR Implementation using Azure Kinect, Open3D, and Three.js," in Proc. *IEEE* ICTC, 2020, pp. 240 243.
[6] Li, G., Zhang, J., Niu, M., and Li, Y., "Warehouse Security System based on Embedded System," in Proc. Int. Conf. Logistics Engineering, Management and Computer Science (LEMCS), Atlantis Press, 2015, pp. 1229–1233.
[7] Setyawan, E.B., Yunita, A., and Sekarjatiningrum, S.R., "Development of Automatic Real Time Inventory Monitoring System using RFID Technology in Warehouse," Int. Journal on Informatics Visualization, vol. 6, no. 3, pp. 636–642, Sep. 2022.
[8] Bailkar, S., Bankar, S., Shenoy, K., More, P., and Bedekar, A., "Smart Inventory Optimization using Machine Learning Algorithms," in Proc. 2nd Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT), *IEEE*, 2024, pp. 1395–1400.
[9] Choudhary, V., Patel, K., Niaz, M., Panwala, M., Mehta, A., and Choudhary, K., "Implementation of Next-Gen IoT to Facilitate Strategic Inventory Management System and Achieve Logistics Excellence," in Proc. Int. Conf. Trends in Quantum Computing and Emerging Business Technologies (TQCEBT), *IEEE*, 2024, pp. 1–6.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)