International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 06 June-2026 | Impact Factor: 3.5

Intelligent Inventory Optimisation Using Predictive Analytics

NaveenRajan.A[1], J. Syed Raffi Ahamed[2], A. B. Hajira Be[3]

[1]PG Student, Department of Computer Applications, Karpaga Vinayaga College of Engineering and Technology, Chinna Kolambakkam, Maduranthagam Taluk, Chengalpattu District, Tamil Nadu – 603308, Gmail: naveenarscable@gmail.com

[2]Assistant Professor, Department of Computer Applications, Karpaga Vinayaga College of Engineering and Technology, Chinna Kolambakkam, Maduranthagam Taluk, Chengalpattu District, Tamil Nadu – 603308, Gmail: syed@kveg.in

[3]Associate Professor, Department of Computer Applications, Karpaga Vinayaga College of Engineering and Technology, Chinna Kolambakkam, Maduranthagam Taluk, Chengalpattu District, Tamil Nadu – 603308, Gmail: hajiraab786@gmail.com

How to Cite this Article:

NaveenRajan.A, & Be, A. B. H. (2026). Intelligent Inventory Optimisation Using Predictive Analytics. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.168

https://doi.org/10.55041/ijcope.v2i6.168

Abstract

Effective supply chain management and stock level balancing represent critical existential hurdles for micro, small, and medium-sized enterprises (SMEs). Overstocking ties down substantial working capital in stagnant physical assets and increases vulnerability to product deterioration, while stockouts directly cause immediate revenue leakage, damage customer acquisition metrics, and erode institutional trust. This comprehensive document details the software architecture, engineering patterns, and deployment configurations of an enterprise-grade, cloud-native web application tailored to resolve these challenges: the Intelligent Inventory Optimisation System (IIOS).

Moving beyond traditional static database setups and client-side prototypes, the proposed solution integrates an advanced multi-tier web development architecture centred around a React.js presentation layer, a resilient Node.js REST API layer, and an active predictive analytics engine driven by structured relational persistence models.

By embedding time-series statistical models directly into production web pipelines, the framework converts inventory control from a purely defensive, historical tracking routine into a predictive, automated orchestration workflow. This system empowers small business entities to optimise resource configurations, maximise transactional efficiency, and secure scalable data security across all operational units.

Keywords

Inventory Management; Predictive Analytics; Demand Forecasting; Supply Chain Optimization; Time Series Analysis; Web Application Architecture.

I. INTRODUCTION

In contemporary retail environments, modern commercial transactions generate massive streams of transactional data. For global enterprise corporations, this data is continuously harvested and analysed via expensive, custom machine learning pipelines and extensive data warehouses. However, small businesses frequently remain excluded from these analytical frameworks due to the immense financial cost, architectural complexity, and steep infrastructure learning curves associated with big-data platforms. Consequently, micro-enterprises operate under reactive conditions, relying on legacy spreadsheets or manual human visual checks to make vital purchasing decisions. This architectural gap frequently leads to major capital misallocations and high inventory overhead costs. The core objective of this project is to democratise advanced computational logistics by designing a lightweight, cost-effective, high-performance, web-based platform specifically optimised for small-scale operations. From a web engineering perspective, the system bridges the gap between raw data storage and interactive business intelligence. It must run securely on affordable hosting tiers, provide real-time updates across multiple point-of-sale clients, ensure sub-second response times for billing transactions, and perform background algorithmic forecasts without blocking primary user interface loops. By designing a highly modular framework focused on clean API boundaries, reliable database indexing, and intuitive state-management interfaces, this project proves that modern web technologies can deliver sophisticated predictive operations to everyday small businesses without requiring expensive computational clusters.

II.STRUCTURAL ANALYSIS AND INEFFICIENCIES OF CLIENT-SIDE PROTOTYPES

A structural evaluation of basic frontend-only solutions—such as systems operating exclusively via browser memory abstraction mechanisms like localStorage—highlights severe architecture limitations that make them entirely unviable for production enterprise deployment. While client-side storage architectures provide rapid, zero-setup environments suitable for initial prototypes, they introduce critical vulnerabilities when evaluated against multi-user, multi-device commercial applications:

Data Volatility and Storage Quotas: Web browser architectures impose strict hard memory capacity limits on localStorage engines, typically capped at approximately 5 megabytes per domain. In a real commercial environment, a small retail business processing hundreds of transactions per week across hundreds of stock-keeping units (SKUs) will exhaust this browser memory pool within a few months. Furthermore, browser data lacks transactional safety guarantees; it can be entirely deleted by automated cache-clearing routines, system updates, or unhandled browser user actions, resulting in catastrophic loss of vital business data.

Complete Lack of Concurrency and Multitenancy: Client-side storage models are fundamentally bound to a single local device, application client, and browser profile. In modern business scenarios, multiple operators—such as a front-desk billing teller, a back office warehouse dispatcher, and an off-site general manager—must interact with the exact same inventory tracking state simultaneously. Under a client-side architecture, data updates executed on one point-of-sale terminal remain completely isolated on that hardware node, leading to extreme data fragmentation, inventory sync desynchronization, and operational confusion.

Vulnerability and Absence of Security: Data retained in client-side text forms is stored in clear, unencrypted plaintext within the user's browser environment. Any malicious script, cross-site scripting (XSS) exploit, or unauthorised local operator can easily bypass basic visibility layers to access, alter, or steal complete customer transaction metrics, business margins, and user authentication tables. Production systems require multi-tiered server-side validation rules and secure database systems to protect organisational integrity.

Analytical Compute Constraints: Running complex time-series analysis or multi-month demand prediction models directly within the browser’s single-threaded JavaScript execution context introduces severe performance degradation. Executing large array loops over long historical

data logs blocks the browser's main window loop, causing interface freezing, dropped input packets, and an extremely poor user experience.

III. PROPOSED MULTI-TIER WEB ARCHITECTURE AND TECHNOLOGY STACK

The Intelligent Inventory Optimisation System (IIOS) is engineered around a robust, decoupling pattern consisting of three core interactive conceptual layers. This architectural separation guarantees that UI rendering logic, core business validation workflows, and heavy data mining prediction calculations run independently on optimal hardware allocations.

System Layer Topologies:

Presentation Layer (Frontend App Client): Built as a single-page application (SPA) using React.js and TypeScript. It provides a highly responsive, component-driven user experience. State transitions across the application are managed through a centralised context model or Redux architecture, eliminating unnecessary data re-fetching. Data visualisation layouts use Chart.js abstractions, enabling smooth, interactive browser rendering of sales trendlines, stock alerts, and forecast models.

Application Gateway & Business Logic Layer (Backend API): Built on Node.js and the Express.js framework. This layer acts as the centralised traffic coordinator, exposing a series of stateless RESTful API endpoints. It manages user session states via JSON Web Tokens (JWT), executes thorough data validation routines, manages safe transactional states, and serves sanitised data payloads back to front-end callers.

Data Analytics & Forecast Engine: A micro-module running alongside the core API backend. It handles heavy analytical computations, transforming raw transaction logs into optimised future stock predictions. This engine uses highly optimised array algorithms to calculate moving averages, exponential smoothing coefficients, and linear trend formulas without impacting the live web request pipeline.

Persistence Storage Engine (Database Layer): Powered by an enterprise-grade relational database management system using PostgreSQL. This engine enforces absolute data integrity via strict primary keys, foreign key relations, cascade deletion constraints, and unique compound indices. It handles atomic operations smoothly, preventing race conditions—such as two simultaneous cashiers trying to pull the last available stock-keeping unit from the same warehouse inventory line.

IV. SYSTEM REQUIREMENT

Hardware Requirements:

- Processor: Intel i3 or equivalent
- RAM: 4 GB or higher
- Storage: Minimum 200 MB free space
- Display: 1024 × 768 resolution or higher
- Input Devices: Keyboard and Mouse

Software Requirements:

- Operating System: Windows 10 or above / macOS / Linux
- Browser: Latest version of Google Chrome, Microsoft Edge, or Mozilla Firefox
- Development Tools: Visual Studio Code / Sublime Text / Notepad++
- Technologies Used: HTML, CSS, JavaScript
- Storage: Browser Local Storage for saving user data

V. MATHEMATICAL MODELING AND ALGORITHMIC LAYOUT FOR PREDICTIVE DEMAND FORECASTING

The core computational intelligence of the application derives from the conversion of scattered transaction points into proactive inventory optimisation suggestions. The system handles historical trends and variations by integrating three progressive levels of mathematical forecasting within its codebase:

Model 1: Simple Moving Average (SMA):

The Simple Moving Average algorithm acts as a smoothing filter for products that demonstrate

a highly stable demand profile over time. It functions by calculating an unweighted arithmetic mean of sales metrics across a specific, rolling multi-month observation window (N). Let S_t define the total verified units sold during month t. The forecast for the upcoming target period (F_{t+1}) is formulated mathematically as follows:

$$F_{t+1} = \frac{1}{N} \sum_{i=0}^{N-1} S_{t-i}$$

This equation effectively flattens sudden, isolated transactional spikes, delivering a clean base-level demand estimate for standard non-perishable inventory items.

Model 2: Single Exponential Smoothing (SES):

For inventory classes that are sensitive to immediate shifts in consumer buying habits, the system applies Single Exponential Smoothing. This model balances current real-world observations against historical forecast metrics by introducing a specialised smoothing weight factor (α), bounded strictly between 0 and 1. The mathematical definition is written as:

$$F_{t+1} = \alpha S_t + (1 - \alpha) F_t$$

When α approaches 1.0, the forecast engine minimises historical weight to respond instantly to newly logged sales velocity. When α is set closer to 0.0, the engine prioritises long-term historical stability, minimising the impact of brief marketplace irregularities.

Model 3: Least Squares Linear Regression Trend Tracking:

To support small businesses experiencing clear business scaling or visible contraction trends, the system features a Least Squares Linear Regression engine. This framework treats time blocks as an independent metric axis (X) and unit sales volumes as a dependent axis (Y). It derives an optimal linear equation detailing systemic directionality:

$$Y = mX + b$$

The mathematical calculations for the trend slope line (m) and intercept parameter (b) are explicitly evaluated using the formulas below:

$$m = \frac{N\sum(XY) - \sum X \sum Y}{N\sum(X^2) - (\sum X)^2}$$

$$b = \frac{\sum Y - m\sum X}{N}$$

By evaluating these parameters over a 12-month trailing index window, the system extrapolates the linear equation forward into future operational months, preventing severe inventory shortages during high-growth cycles.

Algorithmic Automation: Safety Stock and Reorder Recalculation Once raw future monthly demand unit values are generated, the analytical backend feeds these estimates into an automated safety replenishment module. The safety threshold (SS) and optimised order trigger configurations directly prevent stock depletion scenarios due to transport delays or unexpected spikes in demand. Let L represent the supplier's average raw lead delivery time in days, D_{avg} represent the average daily demand volume, and σ_D represent the mathematical standard deviation of historical daily sales records. The safety stock target buffer is defined by the safety equation:

$$SS = (Z \times \sigma_D \times \sqrt{L})$$

Where Z represents the standard statistical service factor (e.g., a Z value of 1.65 targets a 95% protection rate against stockouts). The active reorder point (ROP) is calculated continuously as follows:

$$ROP = (D_{avg} \times L) + SS$$

When backend inventory worker updates push an item's current stock level below its calculated ROP value, the application automatically moves that SKU into the dashboard's critical reorder table and triggers automated emails to suppliers, preventing manual entry bottlenecks.

VI. FUNCTIONAL DESIGN AND DATA FLOW MAPPING

The backend application logic is designed to process operations asynchronously, ensuring high

responsiveness across all network connections. The core platform operations handle three primary data flow patterns:

Inventory Management Sub-Module: Provides full CRUD execution logic for products, manages safety thresholds, organises category labels, and structures vendor configuration data profiles.

Point-of-Sale (POS) Transaction Module: Parses incoming transaction data payloads, checks stock balances, updates historical ledger lines, reduces stock counts atomically inside the database, and builds valid transaction records.

Predictive Analytics Processing Worker: Periodically loops over multi-month transaction

timelines, runs the selected forecasting models, and saves the future unit demand numbers into the database cache for instant presentation.

VII.FRONTEND APPLICATION UI AND CLIENT STATE MANAGEMENT

The presentation layer of the IIOS platform translates backend analytics into actionable operational tools. Built with a responsive grid layout using modern web design principles, the dashboard highlights high-priority updates first. It places real-time low-stock alerts and predictive order notifications at the very top of the interface, ensuring critical data is immediately visible. The core client-side components include:

Interactive POS Billing Sub-Panel: Cashiers can search for items by name or scan barcode SKU data rapidly. This action dynamically manages an internal client array of active items, updates total cost calculations, and handles invoicing workflows without causing window layout stutters.

Dynamic Analytical Analytics Dashboard: Pulls time-series data streams and passes the historical arrays directly into Chart.js configuration models. This displays dual lines that track real-world monthly sales histories alongside future predictive demand curves, providing business operators with clear, visual insight into operational trends.

Automated Inventory Status Controller: Renders colour-coded status pills indicating item health. If an item's current stock count falls below its calculated safety stock line (SS), the UI highlights the item in amber. If it falls below the reorder point threshold, it turns red, allowing workers to verify purchasing requirements instantly.

VIII. DEPLOYMENT INFRASTRUCTURE AND DEVOPS AUTOMATION BLUEPRINT

To support small businesses with minimal IT management overhead, the IIOS platform uses a modern, automated container deployment workflow. The complete application infrastructure is managed as code using Docker containers, allowing teams to launch identical developer, staging, and live production environments instantly.

The system deployment pipeline is orchestrated through a structured 12-month implementation timeline, ensuring a smooth transition from legacy manual processes to predictive digital inventory control:

Months 1–3: Core Backend & Database Infrastructure: Setting up the production PostgreSQL schema, fine-tuning indexes for high-volume time-series storage, and building out the foundational Express.js API gateways alongside JWT user authentication matrices.

Months 4–6: UI Components & Point-of-Sale Integration: Building out the front-end interface modules using React and TypeScript, deploying state management engines, and launching the interactive point-of-sale invoicing screen.

Months 7–9: Analytical Engine Optimisation**:** Coding and refining the mathematical forecasting modules (Moving Average and Linear Regression), establishing background processing routines, and calibrating safety stock parameters using historical operational records.

Months 10–12: Validation, System Hardening, and Public Launch: Running exhaustive end-to-end integration stress tests, verifying data accuracy across multiple simultaneous points-of-sale, conducting staff onboarding workshops, and taking the system live into production environments

IX.CONCLUSION

The Intelligent Inventory Optimisation System (IIOS) demonstrates that high-performance predictive log informatics can be delivered effectively to small business environments using modern, accessible web architectures. By replacing fragile browser storage workflows and fragmented desktop accounting entries with a highly scalable, multi-tier web platform, the system eliminates traditional operational tracking bottlenecks. It ensures that data remains durable, transactional states are updated consistently across devices, and sensitive enterprise logs are strictly protected. Most importantly, by building statistical modelling features right into production Node.js and database engines, everyday small retail operations can shift away from reactive workflows. Instead, they can embrace proactive, data-driven management frameworks that maximise working capital efficiency and completely protect their bottom line.

X.FUTURE RESEARCH HORIZONS

Future development horizons will explore several innovative technical paths, including:

Advanced Machine Learning Integrations: Expanding the forecasting module beyond standard linear regressions by incorporating multi-variable recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) network loops via Python data microservices, enabling the engine to adjust predictions based on complex local weather indicators, city event schedules, and regional economic indexes.

Automated Vendor B2B Integration: Building advanced EDI (Electronic Data Interchange) network protocols right into the application backend. This will allow the platform to check vendor pricing sheets in real time and automatically issue electronic purchase orders as soon as local inventory volumes cross established reorder lines.

Distributed Blockchain Verification Networks: Exploring the use of light, consortium-based blockchain entries to cryptographically secure and verify multi-party supply transactions, guaranteeing absolute transparency and untampered ledger logs across independent merchant groups and logistics networks

XI. REFERENCES

[1] T. W. S. Chow and Y. Han, "Demand forecasting and inventory optimisation for small and medium enterprises using predictive analytics," IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5012–5022, Aug. 2020.

[2] M. Ali and A. Al-Radaideh, "A full-stack web application architecture for real-time inventory tracking and supply chain management," in Proceedings of the IEEE International Conference on Software Engineering and Service Science (ICSESS), 2022, pp. 145–149.

[3] J. E. Smith, Predictive Analytics in Operations Management: Foundations and Modern Web Implementations, 2nd ed. New York, NY, USA: IEEE Press, 2023.

[4] L. Zhang, X. Xu, and H. Huang, "Applying exponential smoothing and linear regression models to inventory replenishment in small retail systems," IEEE Access, vol. 9, pp. 112430–112441, 2021.

[5] R. Kumar and S. Sen, "Ensuring database transaction safety and ACID compliance in multi-user web-based point-of-sale systems," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 11, pp. 5432–5445, Nov. 2022.

[6] P. Lopez, "Cloud-native deployment pipelines and DevOps automation for small business information systems," IEEE Software, vol. 39, no. 4, pp. 58–64, Jul./Aug. 2022.

[7] S. R. Garner, "Least squares linear regression techniques for timeseries trend tracking in localised supply chains," IEEE Journal of Selected Topics in Signal Processing, vol. 15, no. 3, pp. 671–683, Apr. 2021.

[8] M. R. Hassan, "Overcoming storage and concurrency limitations of client-side web frameworks in commercial software applications," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1024–1035, Jun. 2022.