International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Inventra: A Web-Based Inventory Management System using Frontend Technologies

Rethika V
Department of Computer Science and Information Technology
VELS Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India
Sri dharani D
Department of Computer Science and Information Technology
VELS Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India
Research Supervisor: **Dr. T.R. Nisha Dayana**

Assistant Professor, Department of Computer Science and Information Technology, VELS Institute of Science, Technology & Advanced Studies (VISTAS), Chennai

https://doi.org/10.55041/ijsmt.v2i4.646

Cite this Article: V, R. & D, S. D. (2026). Inventra: A Web-Based Inventory Management System using Frontend Technologies. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.646

Abstract—Inventory management is a critical business function ensuring that the right products are available at the right time. Traditional manual methods are labour-intensive, error-prone, and inefficient, particularly in medium to large-scale environments. This paper presents Inventra—a modern, web-based inventory management system developed entirely using HTML5, CSS3, and JavaScript (ES6+). The system provides a responsive, dashboard-driven interface enabling businesses to manage products, track stock movements, monitor low-stock alerts, generate analytical reports, and manage user roles from any device through a standard web browser, without server-side installation. Six core modules are implemented: Product Management, Stock In/Out Transactions, Alert Management, Report Generation and Analytics, User and Role Management, and an interactive Dashboard Overview. Chart.js is leveraged for dynamic data visualisation, Material Symbols for iconography, and a CSS variable system enforces a cohesive design language throughout. The system demonstrates a practical, accessible, and scalable approach to modernising inventory operations through pure frontend technologies.

Keywords—*inventory management, web application, HTML5, CSS3, JavaScript, Chart.js, frontend technologies, responsive design, role-based access control, stock tracking*

I. INTRODUCTION

Inventory management is a fundamental business process involving the systematic ordering, storing, tracking, and controlling of goods within an organisation. It ensures that the right quantity of products is available at the right time, maintaining a balance between supply and demand. Whether in retail, healthcare, manufacturing, or e-commerce, maintaining accurate inventory records is essential for smooth day-to-day functioning.

Traditional manual methods—paper registers or basic spreadsheets—are no longer adequate for modern business needs due to inherent limitations in accuracy, speed, and scalability. Existing legacy desktop software requires local installation, licensing, and dedicated IT support, offering no mobile accessibility or modern UI. Spreadsheet-based systems suffer from formula errors, version conflicts, and lack of real-time visibility.

This paper proposes Inventra, a web-based inventory management system built entirely using modern frontend technologies: HTML5, CSS3, and JavaScript. The system delivers a fully functional, responsive user interface that works on any device with a standard web browser, making it accessible and easy to deploy without complex server infrastructure. It addresses six key functional domains: product management, stock transaction recording, automated low-stock alerting, analytical reporting, user and role management, and an interactive dashboard for operational overview.

The remainder of this paper is structured as follows: Section II reviews related literature. Section III describes the system analysis, including limitations of existing systems. Section IV covers the system architecture and design. Section V details the implementation and key code components. Section VI presents results and discussion. Section VII concludes and outlines future enhancements.

II. LITERATURE REVIEW

Prior research in inventory management spans machine learning, IoT, and computer vision domains. Yin et al. [1] proposed a CNN-based inventory defect detection algorithm, demonstrating high accuracy for classifying stock conditions in real time—an approach that influenced Inventra's alert severity classification design. Kumari et al. [2] combined computer vision with sensor technologies for real-time stock level detection and alerts, a concept reflected in Inventra's Critical/Warning alert system.

Ravi et al. [3] implemented an IoT-based inventory status monitoring and alert system using storage sensors and centralised data transmission, which informed Inventra's real-time alert dashboard design. Liang et al. [4] designed an intelligent management system integrating sensors, data analytics, and predictive algorithms, directly influencing Inventra's analytics and reporting module.

Zhang et al. [5] applied visual inspection techniques for quality classification in assembly processes, which shaped Inventra's product status classification interface. Thakur et al. [6] integrated image processing with data analytics to extract inventory insights—directly informing the Chart.js-based analytics visualisations used in this system. Tasin et al. [7] developed automated out-of-stock detection using threshold-based imaging, informing the Out of Stock badge and critical alert row design.

Qing et al. [8] demonstrated deep learning-based automated inventory detection with high accuracy, whose automation philosophy guided Inventra's automated alert triggering. Andrijasevic et al. [9] used RNNs on sequential inventory state data for real-time analysis, informing the Stock Movement line chart and transaction history table. Vishnani et al. [10] applied computer vision for automatic inventory item detection and mapping, informing the product catalogue and SKU-based identification system.

While existing solutions predominantly rely on machine learning, IoT hardware, or proprietary backends, Inventra implements the core principles of proactive monitoring, role-based access, and data-driven decision support through a pure frontend JavaScript approach—eliminating infrastructure barriers while maintaining functional depth.

III. SYSTEM ANALYSIS

A. Existing System Limitations

Manual inventory systems, whether paper-based or spreadsheet-driven, exhibit critical limitations: they are time-consuming, requiring significant staff effort per transaction; susceptible to human data-entry and calculation errors; at risk of data loss with no reliable backup; incapable of real-time stock updates; difficult to report from; and vulnerable to unauthorised tampering. Legacy desktop software further compounds these issues with platform lock-in, mobile inaccessibility, and high maintenance overhead.

B. Proposed System: Inventra

Inventra addresses these limitations through a single-tier, client-side web architecture. All application logic, presentation, and interaction are handled within the browser through HTML, CSS, and JavaScript files—no server-side processing, database, or backend API is required. The system is structured around seven modules:

- Login & Authentication Interface — role selection, form validation, animated transitions.
- Dashboard Overview — KPI summary cards and dynamic Chart.js charts for stock movement and category distribution.
- Product Management — searchable, filterable product catalogue with SKU tracking, status badges, and action controls.
- Stock In/Out Transaction Module — transaction recording form with history table and user attribution.
- Alert Management Module — priority-sorted Critical/Warning alert table with reorder automation.
- Reports & Analytics Module — date-range-filtered sales vs. purchases charts, inventory health doughnut chart, and category breakdown.
- User & Role Management Module — team directory with Administrator and Staff Member role-permission cards.

IV. SYSTEM DESIGN

A. System Architecture

Inventra follows a single-tier client-side architecture comprising four logical layers: (1) Presentation Layer — HTML5 pages for each module (index.html, dashboard.html, product.html, stock.html, alert.html, reports.html, users.html); (2) Style Layer — CSS3 files (style.css, login.css) providing responsive layout, animations, and a CSS custom-properties design token system; (3) Logic Layer — a JavaScript controller (script.js) implementing all interactive behaviours, Chart.js initialisation, sidebar navigation, form handlers, and table interactions; (4) CDN Layer — Chart.js v4, Google Fonts (Inter), Material Symbols Outlined, and UI Avatars API loaded via CDN.

B. Technology Stack

TABLE I. TECHNOLOGY STACK

Category	Technology	Purpose
Markup	HTML5	Semantic structure & page layout
Styling	CSS3 (Flexbox, Grid, Variables)	Responsive layout, animations, design tokens
Scripting	JavaScript ES6+	Interactivity, event handling, chart logic
Charting	Chart.js v4 (CDN)	Line, Bar, Combo, and Doughnut charts
Iconography	Google Material Symbols (CDN)	Navigation & action icons
Typography	Inter via Google Fonts (CDN)	Screen-optimised typeface
Avatars	UI Avatars API	Placeholder user profile images

C. CSS Design Token System

A CSS custom-properties (variables) system defines application-wide design tokens, including the primary olive-green colour palette (--primary-color: #606c38), danger burnt-orange (--danger: #BC6C25), warning mustard (--warning: #DDA15E), and success dark-green (--success: #283618), as well as spacing, border-radius, and shadow definitions. This ensures visual consistency across all seven pages and enables future theme customisation with minimal code changes.

D. Database Design (Frontend Representation)

As Inventra is a pure frontend application, it does not use a live database. Data is represented as static structured HTML simulating real database records. Three primary logical entities are modelled: Products (fields: Name, SKU ID, Category, Quantity, Price, Status), Transactions (fields: Product, Type, Quantity delta, Date, Updated By), and Users (fields: Name, Email, Role, Status, Last Login). This design supports easy future migration to a backend database (MySQL, MongoDB) without architectural redesign.

V. SYSTEM IMPLEMENTATION

A. Page Flow and Navigation

The application follows a linear page-flow model. The entry point index.html (Login Page) authenticates the user and redirects to dashboard.html. From the dashboard, all module pages (product.html, stock.html, alert.html, reports.html, users.html) are accessible via the persistent sidebar navigation, with the active page highlighted through a CSS active class. Logout from any page returns the user to index.html. Navigation is implemented through standard HTML anchor links—no client-side routing framework is required.

B. JavaScript Controller (script.js)

The script.js file is the single JavaScript controller for the entire application. It initialises on DOMContentLoaded, checks whether Chart.js is available, and calls initCharts() and setupInteractions(). The initCharts() function renders four Chart.js visualisations: a gradient-filled Line Chart for monthly Stock Movement on the dashboard; a multi-colour Bar Chart for Category Distribution; a combo Bar/Line Chart for Sales vs. Purchases in the reports module; and a Doughnut Chart for Inventory Health distribution. The setupInteractions() function implements sidebar navigation active-state toggling, button micro-animations (scale transform on mousedown), table-row click highlights, form submission handling with loading-state animations, and the mobile sidebar slide-in/overlay toggle.

C. Alert Management Logic

The Alert Management module implements a visual severity classification system distinguishing Critical (red tint, near-zero stock) and Warning (amber tint, below reorder threshold) states through CSS background-colour tinting on table rows. Each row includes a quantity progress bar implemented in CSS, showing current stock as a percentage of minimum required. JavaScript event listeners on reorder buttons animate through Processing → Ordered state transitions using CSS class manipulation and setTimeout scheduling, providing immediate visual feedback without any server round-trip.

D. Responsive Design Implementation

Responsive layout is implemented through CSS media queries at four breakpoints. At widths above 1200px, the full four-column KPI card grid and sidebar are displayed. Below 1200px, cards collapse to a two-column grid. Below 992px, the sidebar converts to a slide-out drawer triggered by a mobile menu button. Below 768px, all grids collapse to single-column layout with a compact header. The two-column stock transaction grid similarly collapses to a single column on mobile devices.

VI. RESULTS AND DISCUSSION

A. Module-Level Outcomes

The Inventra system was successfully developed and tested across multiple browsers and screen sizes. The Login Page renders a split-panel layout with animated illustration and correctly validates required fields before redirecting. The Dashboard correctly displays four KPI cards (Total Products: 1,250; Low Stock Items: 32; Out of Stock: 12; Total Transactions: 8,430) alongside Chart.js visualisations that render with smooth on-load animations.

The Product Management module displays five product entries with correctly colour-coded status badges (green for In Stock, orange for Low Stock, red for Out of Stock), bulk checkbox selection, and a sticky table header for long lists. The Stock Transaction module correctly renders the Stock In/Out radio toggle with colour coding, and the form

submission handler delivers animated Processing → Success confirmation. The Alert module correctly renders severity-differentiated rows, progress bars reflecting stock levels (0% for out-of-stock, 10% for near-critical items), and animated Reorder buttons cycling through Processing → Ordered states.

The Reports module renders all four Chart.js visualisations correctly with responsive container adaptation. The Users module displays role permission cards with check/cancel icons and the team directory with correct avatar, role badge, status, and last-login styling.

B. Responsive Design Testing

TABLE II. RESPONSIVE DESIGN BREAKPOINT TESTING

Screen Size	Breakpoint	Sidebar	Layout Behaviour
Desktop (≥1440px)	Default	Fully visible	4-col KPI cards, 2-col grid
Laptop (1200px)	<1200px	Visible	2-col cards, 1-col grid
Tablet (992px)	<992px	Slide-out drawer	Mobile menu button appears
Mobile (≤768px)	<768px	Hidden (overlay)	Single column, compact header

C. Browser Compatibility Testing

TABLE III. BROWSER COMPATIBILITY RESULTS

Browser	Result	Notes
Google Chrome 120+	PASS	All features render correctly
Mozilla Firefox 121+	PASS	All features render correctly
Microsoft Edge 120+	PASS	All features render correctly
Safari 17+	PASS	Minor font rendering difference; fully functional

D. Performance and Accessibility

Because Inventra relies entirely on client-side technologies with CDN-delivered dependencies, it requires no server round-trips for UI interactions. Page load times are primarily dependent on CDN resource delivery (Chart.js ~60KB gzipped, Material Symbols, Google Fonts Inter). The application passes WCAG 2.1 AA contrast requirements for the olive-green and warm-earth colour palette, and all interactive elements are keyboard-navigable via standard tab-focus behaviour.

VII. CONCLUSION AND FUTURE WORK

This paper presented Inventra, a web-based inventory management system developed using HTML5, CSS3, and JavaScript. The system successfully implements all stated objectives: a multi-module responsive interface, role-based access distinction, interactive Chart.js dashboards, automated low-stock alerting, stock transaction recording, analytics reporting, and user management—all operable from any modern web browser without server installation or database configuration.

Inventra demonstrates that a pure frontend approach can deliver substantial functional depth for inventory management use cases, reducing the infrastructure barrier for small and medium enterprises. By eliminating backend

complexity while providing real-time visual feedback, the system enhances operational efficiency and supports informed restocking decisions.

Future enhancements are planned in three phases. Short-term: LocalStorage/IndexedDB for persistent browser-based data, PDF/Excel export via jsPDF and SheetJS, and CSS dark mode toggle. Mid-term: Progressive Web App (PWA) support with service workers for offline access and push notifications for critical alerts, barcode/QR scanning via QuaggaJS or ZXing, and multi-language internationalisation (i18n). Long-term: full Node.js + Express backend integration with MySQL or MongoDB for live persistent storage, JWT-based authentication, and WebSocket-driven real-time stock alert broadcasting to all connected users.

ACKNOWLEDGMENT

The author gratefully acknowledges the guidance and constant support of Dr. T.R. Nisha Dayana, Professor, Department of Computer Science and Information Technology, VISTAS, Chennai, throughout the duration of this project. Sincere thanks are extended to Dr. R. Parameswari, Professor & Head of the Department, and to the management of VELS Institute of Science, Technology & Advanced Studies (VISTAS) for providing the opportunity and resources to carry out this work.

REFERENCES

[1] X. Yin, X. Li, Q. Gao, and H. Li, "Inventory defect detection algorithm based on convolutional neural network," Journal of Intelligent & Fuzzy Systems, vol. 36, no. 6, pp. 6071–6079, 2019.

[2] A. Kumari, P. Ashrani, M. Basha, and B. S. A. Kumar, "An approach for identifying and tracking inventory items using computer vision and sensor technologies," in Proc. IEEE ICMLA, 2023.

[3] M. Ravi, K. M. Parvesh Ahmed, and P. Sengottaiyan, "IoT based inventory status monitoring and alert system," in Proc. IEEE ICIRD, 2023.

[4] Y. Liang, L. Chen, and B. Xu, "Design of intelligent management system for inventory control," in Proc. Int. Conf. Smart Systems and Energy Management, 2022.

[5] X. Zhang, Z. Wang, H. Yu, M. Liu, and B. Xing, "Research on visual inspection technology in automatic assembly," IEEE Access, vol. 10, pp. 12340–12351, 2022.

[6] K. Thakur, A. Adhya, and C. Bajpai, "Inventory management using image processing and data analytics," in Proc. IEEE ICCMC, 2021, pp. 516–519.

[7] H. S. Tasin, M. S. Sarkar, and M. A. Rahman, "Design of automated detection system for out-of-stock items," in Proc. IEEE ICCCNT, 2021.

[8] L. Qing, K. Yang, W. Tan, and J. Li, "Automated detection of inventory using deep learning," IEEE Transactions on Industrial Electronics, vol. 68, no. 5, pp. 4321–4330, 2020.

[9] U. Andrijasevic, J. Kocic, and V. Nesic, "Inventory status detection using recurrent neural networks," in Proc. IEEE ICET, 2020.

[10] V. Vishnani, A. Adhya, and C. Bajpai, "Inventory detection using image processing," in Proc. IEEE ICARCV, 2020.

[11] Chart.js Documentation. [Online]. Available: https://www.chartjs.org/docs/latest/ [Accessed: April 2026].

[12] MDN Web Docs, HTML5, CSS3, JavaScript Reference. [Online]. Available: https://developer.mozilla.org/ [Accessed: April 2026].

[13] Google Material Symbols. [Online]. Available: https://fonts.google.com/icons [Accessed: April 2026].