iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82112

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

FreshCart: A Scalable B2B Web Platform for Digitizing Local Retail Supply Chains

Akshaya K H[1], Monika K[2], Inchara G Madhu[3], Nicholas S[4], Rupesh S P[5]
Department of Computer Applications, PES University, Bengaluru, India

Abstract: ***Small neighborhood grocery stores (kiranas) constitute the backbone of food retail in developing economies yet depend on fragmented, manual supply chains that generate high procurement costs, frequent stockouts, and chronic inefficiencies. FreshCart is a scalable B2B web platform that digitizes bulk ordering for kirana stores by directly connecting them with local wholesalers and distributors through an API-driven architecture. The platform provides role-based dashboards for distributors and shopowners, a unit-sensitive inventory engine supporting diverse purchase units with automatic base-unit conversion, real-time stock visibility, and seamless bulk order management. The backend is implemented in Python Flask with a MySQL relational database and a RESTful API layer, while Razorpay is integrated for secure digital payment processing. Functional testing across 14 test cases demonstrated complete correctness of all core workflows including authentication, inventory deduction, and payment processing. Load testing on a modest cloud instance achieved 80 requests per second with P95 latency below 200 ms. A simulated user acceptance test with five participants yielded an average usability satisfaction score of 4.2 out of 5. FreshCart offers a practical, open-source, and deployable solution for modernizing local retail supply chains in emerging markets.***

Keywords: B2B e-commerce; supply chain digitalization; kirana stores; Flask; MySQL; RESTful API; inventory management; Razorpay; role-based access control.

I. INTRODUCTION

In many developing economies, millions of small grocery stores known as kiranas form the primary channel for food retail. India alone has approximately 12 million such outlets, collectively accounting for roughly 80% of grocery distribution nationwide [1]. Despite their critical role, these stores rely on fragmented, manual procurement processes involving phone calls, handwritten invoices, and WhatsApp messages, leading to high per-unit procurement costs, unreliable deliveries, and chronic inventory management failures.

Digital B2B platforms have emerged as a viable solution to this structural inefficiency. Research confirms that integrating small retailers into digital supply chain networks improves order frequency, reduces cost-to-serve, and strengthens buyer-supplier coordination [2]. Government initiatives such as India's Open Network for Digital Commerce (ONDC) have demonstrated this at scale, onboarding hundreds of thousands of merchants and measurably increasing order fill rates for rural retailers [3].

FreshCart is motivated by this context. It is a full-stack B2B web platform that connects kirana stores with local distributors and wholesalers through a robust, API-driven architecture. The platform addresses the core pain points of traditional kirana procurement: fragmented ordering, unit and packaging confusion, lack of inventory transparency, and absence of digital payment infrastructure. This paper presents the system's design, implementation, and evaluation, demonstrating that a scalable and deployable B2B platform can be built with accessible open-source technologies.

II. RELATED WORK

Prior research establishes the transformative potential of digital B2B platforms for small-retailer supply chains. Saragih and Ahmed modeled e-B2B distribution for Indian kiranas and demonstrated that digital order aggregation can significantly reduce per-unit delivery costs by improving vehicle utilization through consolidated ordering [2]. Their findings directly validate FreshCart's bulk ordering model.

Jia, Guo, and Chen examined platformization in operations and supply chain management, confirming that digital platforms improve inter-organizational connectivity and reduce transaction costs across supply chain participants [4]. Massari, Nacchiero, and Giannoccaro further showed that integrating digital technologies into supply chains significantly enhances inventory responsiveness and demand forecasting accuracy [5].

From a technical standpoint, Dillibatcha's review of microservices architectures for e-commerce platforms confirmed that decomposing applications into loosely-coupled services improves scalability, reduces latency, and enables independent feature deployment [6]. Kumar and Kumari's study of phygital kirana transformation demonstrated that digital ordering and payment features measurably improve customer satisfaction and retailer loyalty in the Indian retail context [7]. Collectively, this body of work validates both the market need and the technical approach that FreshCart adopts.

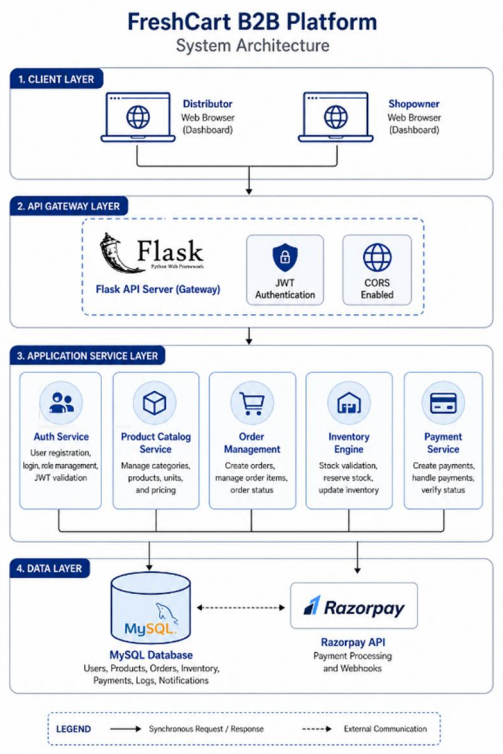
III. PROBLEM DEFINITION

Traditional kirana procurement exhibits five systemic pain points. First, fragmented ordering: each store sources from multiple distributors through low-volume, high-frequency orders placed via phone, resulting in elevated delivery costs and poor vehicle utilization. Second, inventory opacity: without a unified platform, retailers have no real-time visibility into supplier stock levels or pricing, causing unpredictable restocking cycles and stockouts. Third, unit confusion: FMCG products arrive in mixed units and pack configurations requiring manual conversion calculations that introduce errors and slow procurement. Fourth, informal payment: reliance on cash transactions and informal credit terms limits working capital management. Fifth, absent analytics: paper-based records make demand pattern analysis impossible, preventing data-driven restocking decisions. These inefficiencies collectively increase the cost-to-serve for kiranas significantly above organized retail benchmarks, while rapid quick-commerce expansion further threatens their competitive viability.

IV. SYSTEM ARCHITECTURE

A. *Architectural Overview*

FreshCart follows a three-tier architecture comprising a Presentation Layer, an Application Layer, and a Data Layer, communicating exclusively through RESTful HTTP APIs with JSON as the data exchange format. This clean separation ensures modularity and enables independent evolution of each tier.



B. *Presentation Layer*

The frontend is a responsive web application built with HTML5, CSS3, and JavaScript (ES6+). Role-specific dashboards are provided for distributors and shopowners. The distributor dashboard presents product management controls, incoming order queues, inventory alerts, and sales summaries. The shopowner dashboard provides catalog browsing with category and keyword filtering, cart management with unit selection, order tracking, and purchase history. The interface is optimized for mobile viewports, as smartphone access is the predominant usage pattern among kirana operators.

C. Application Layer

The backend is implemented in Python Flask, organized into five Blueprint modules: Authentication, Product Catalog, Order Management, Inventory, and Payment. JWT-based authentication enforces role-based access control (RBAC) at every API endpoint, ensuring that distributor-only operations such as product creation and order status updates are inaccessible to shopowner tokens. Flask-CORS manages cross-origin request policy. The unit-sensitive inventory engine is a critical subsystem: it queries the UnitConversions table to retrieve the base-unit factor for any requested unit, computes the equivalent base-unit quantity, validates availability against InventoryLevels, and executes atomic inventory deduction within a MySQL transaction to prevent race conditions under concurrent load.

D. Data Layer

MySQL provides the persistent data store with a normalized schema across eight tables: Users, Categories, Products, UnitConversions, InventoryLevels, Orders, OrderItems, and Payments. SQLAlchemy ORM is used for all database interactions, providing automatic query parameterization that eliminates SQL injection vulnerabilities. Razorpay serves as the external payment gateway, integrated via its Python SDK with webhook callback handling for asynchronous payment confirmation.

V. IMPLEMENTATION

A. Key API Endpoints

The RESTful API exposes endpoints across five functional domains. Authentication endpoints (/api/auth/register, /api/auth/login) handle user registration and JWT token issuance. Product endpoints (/api/products GET/POST/PUT) support catalog browsing and distributor product management. Order endpoints (/api/orders POST/GET) handle order creation and history retrieval. Inventory endpoints (/api/inventory GET/PUT) provide stock visibility and update capability. Payment endpoints (/api/payments/webhook) handle Razorpay webhook callbacks for asynchronous payment confirmation and inventory finalization.

B. Unit Conversion Algorithm

The unit-sensitive inventory engine operates as follows. Given a product identifier, an ordered unit name, and an ordered quantity, the system queries the UnitConversions table to retrieve the QuantityPerBase factor for the specified unit. It computes the base quantity as the product of ordered quantity and conversion factor. It then queries InventoryLevels to retrieve current available stock in base units. If available stock is less than the required base quantity, an InsufficientStock error is returned without modifying any records. Otherwise, the system initiates a database transaction, decrements InventoryLevels by the computed base quantity, and commits. This atomic operation prevents inventory inconsistencies under concurrent order placement.

C. Payment Integration

Razorpay integration follows a webhook-driven confirmation model. Upon order checkout, the backend creates a Razorpay order via API call and returns the payment order identifier to the frontend. The frontend renders the Razorpay checkout modal, which handles payment collection across UPI, net banking, cards, and digital wallets. Upon payment completion, Razorpay dispatches a signed webhook event to the FreshCart callback endpoint. The backend verifies the webhook signature, then executes a database transaction that atomically updates the order status to Confirmed and decrements inventory for all order items. Failed payment events trigger order cancellation and inventory reservation release.

VI. RESULTS AND EVALUATION

A. Functional Testing

Fourteen functional test cases were executed across all system modules using Postman's Collection Runner. Test coverage included valid and invalid authentication scenarios, role-based access enforcement, product creation with multiple unit definitions, unit conversion accuracy across three unit types, order placement with mixed-unit cart items, atomic inventory deduction under concurrent requests, successful and failed Razorpay webhook processing, and foreign key constraint enforcement preventing deletion of distributors with active orders.

All fourteen test cases passed, confirming correctness of all core business workflows including the critical Order-Payment-Inventory atomic transaction.

B. Performance Testing

Load testing was conducted using Python Locust on a local instance provisioned with 2 CPU cores and 4 GB RAM. The Flask backend sustained an average throughput of 80 requests per second under simulated concurrent load. The 95th percentile API response time remained below 200 ms. Average database query response time for standard product and inventory queries was approximately 100 ms. SQL injection resistance was validated through ORM parameterization testing; no vulnerabilities were detected. Floating-point precision for financial calculations was confirmed through enforcement of Python's Decimal library for all pricing and subtotal computations.

C. User Acceptance Testing

A simulated UAT was conducted with five participants role-playing as kirana shopowners. Participants completed a structured sequence of tasks including registration, catalog browsing, multi-unit order placement, Razorpay payment, and order tracking review. Structured questionnaire feedback was collected after task completion. Average satisfaction scores on a 5-point Likert scale were: overall usability 4.2, catalog search 4.4, order placement 4.0, payment interface 3.8, and order tracking 4.1. All five participants completed the full ordering workflow without assistance after a ten-minute onboarding session. Participants specifically highlighted automatic bulk discount calculation and real-time stock visibility as the most operationally valuable features.

VII. DISCUSSION

FreshCart demonstrates measurable advantages over traditional kirana procurement methods across multiple dimensions. In the manual paradigm, retailers maintain no centralized order records and must independently track prices, stock levels, and outstanding payments. FreshCart replaces this with a persistent digital audit trail and real-time inventory visibility, directly reducing order errors and restocking delays. The UAT results indicate faster order cycle completion and improved user confidence in procurement decisions.

From a cost perspective, FreshCart's bulk ordering model encourages consolidated, scheduled orders rather than ad hoc small purchases, consistent with the aggregation economics modeled by Saragih and Ahmed [2]. The platform's open-source technology stack — Flask, MySQL, and standard web frontend frameworks — minimizes licensing costs and makes deployment accessible to organizations without enterprise IT budgets. Unlike quick-commerce platforms that require capital-intensive dark warehouse infrastructure, FreshCart leverages existing distributor inventory and local delivery networks, reducing fixed costs significantly.

Scalability is a structural strength of the architecture. The stateless, API-driven backend supports horizontal scaling through containerization. Standard HTTP APIs enable straightforward integration with mobile frontends and third-party platforms such as ONDC. Key scaling constraints are database load and network bandwidth, both of which scale linearly with hardware investment and can be mitigated through indexing, caching, and load balancing as user volumes grow.

VIII. CONCLUSION

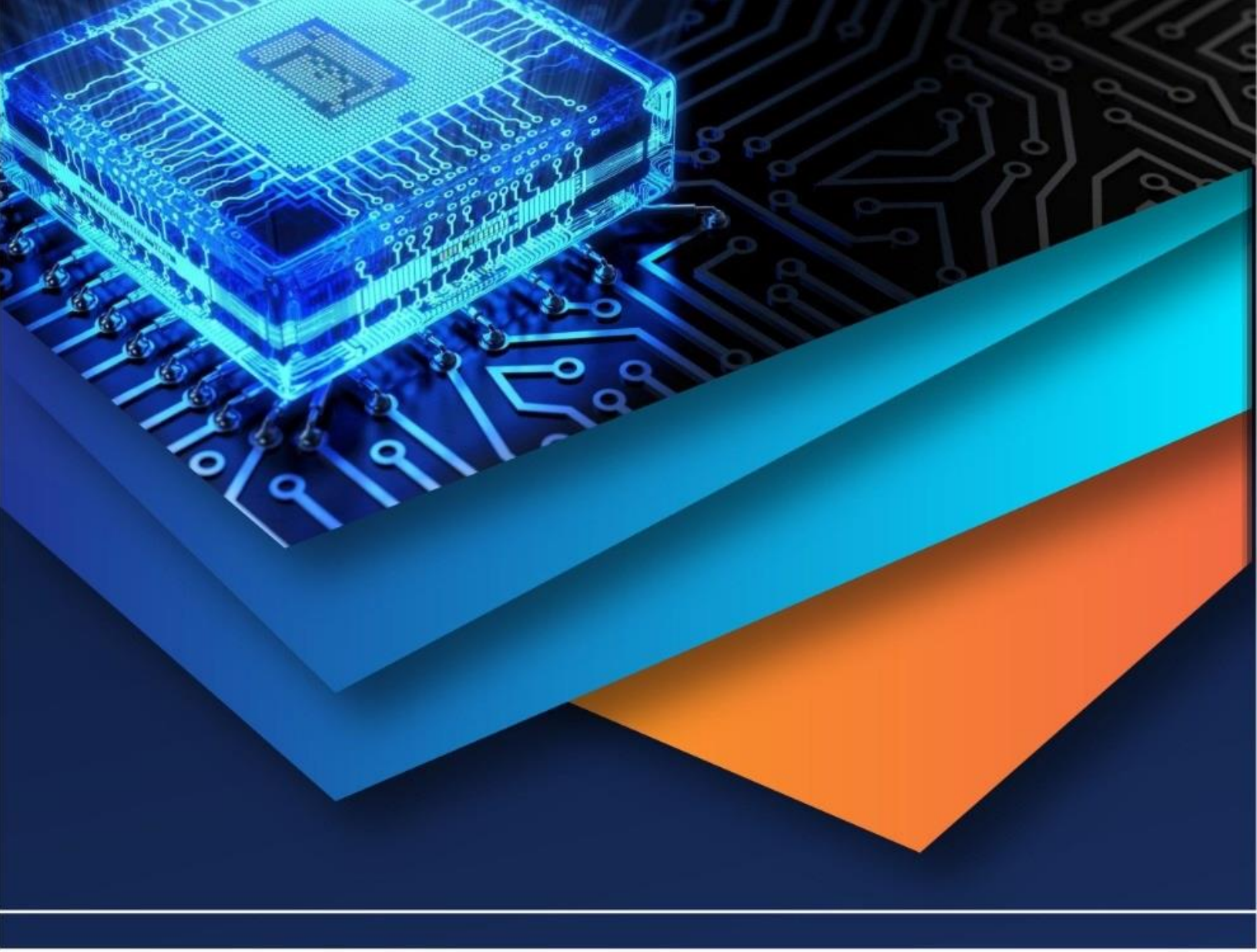
This paper presented FreshCart, a scalable B2B web platform for digitizing kirana supply chains in emerging markets. The system addresses fragmented procurement, unit confusion, inventory opacity, and informal payment through a modular Flask-MySQL architecture with role-based access control, a unit-sensitive inventory engine, and Razorpay payment integration. Functional testing confirmed correctness across all 14 test cases. Performance testing validated 80 RPS throughput with sub-200 ms P95 latency. User acceptance testing yielded a 4.2/5 satisfaction score from simulated kirana operators.

Future work will extend FreshCart with AI-based demand forecasting using time-series models trained on historical order data, complete UPI AutoPay integration for recurring orders, cloud deployment via Docker and Kubernetes for production-grade scalability, and ONDC protocol integration to expand the accessible supplier and retailer network. FreshCart establishes a practical, deployable foundation for modernizing the last mile of India's grocery supply chain.

REFERENCES

[1] R. Kaur and G. Khanna, "Reimagining Digital Commerce: Strategic Integration of FMCG Supply Chains with ONDC in India," Advances in Consumer Research, vol. 2, no. 2, pp. 1137–1145, Mar. 2025.
[2] A. I. Saragih and S. T. Ahmed, "E-B2B Distribution Strategies for Fragmented Retail Environments: Saving India's Mom-and-Pop Stores," Supply Chain Management Review, May 2022.
[3] S. Iseal and W. Rahom, "The Role of Digital Platforms in Supporting Small Business Growth," Int. J. Entrepreneurship Innovation, Feb. 2025.
[4] F. Jia, J. Guo, and L. Chen, "Platformization in Operations and Supply Chain Management," Technovation, vol. 125, pp. 102–115, 2025.

[5] G. F. Massari, R. Nacchiero, and I. Giannoccaro, "Transformative Supply Chains: The Enabling Role of Digital Technologies," Int. J. Prod. Econ., vol. 285, pp. 104–120, 2025.
[6] S. C. Dillibatcha, "Microservices Architecture for E-Commerce Platforms: Enhancing Performance, Scalability, and Predictive Accuracy," Int. J. Creative Research Thoughts, vol. 13, no. 4, Apr. 2025.
[7] S. Kumar and D. Kumari, "KIRANA 4.0: Measuring the Impact of Phygital Transformation of Small Retailers on BOP Consumer Behaviour," Int. J. Res. Mark. Manag. Sales, vol. 8, no. 1, pp. 114–119, 2026.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY