International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Smart Inventory and Sales Management System for Perishable Farm Products using a Web-based ERP Framework

Mohd Habeeb[1], Osman Ahmed Baamar[2], Mohd Arshad[3], Dr. Mohammed Jameel Hashmi[4](Supervisor)

[1,2,3]Dept. of Computer Science and Engineering, ISL Engineering College, Osmania University, Hyderabad, India Emails: mohammedhabeeb027@gmail.com, osmanbaamar479@gmail.com, mdarshd142@gmail.com

[4]HOD-CSE, Associate Professor, Dept. of Computer Science and Engineering, ISL Engineering College, Osmania University, Hyderabad, India Email: dr.jamecl@islc.edu.in

***Abstract*—**The wholesale sector, particularly for perishable agricultural products, faces significant operational challenges due to manual processes and lack of digital infrastructure. This paper presents the design and methodology for a Smart Inventory and Sales Management System tailored for perishable farm products, leveraging a web-based ERP framework with a multi-tenant Software-as-a-Service (SaaS) architecture. The system aims to empower Small and Medium Enterprise (SME) wholesalers by providing secure, scalable, and affordable tools for inventory, sales, purchase, and customer/vendor management. The architecture utilizes modern technologies such as Django, React, and PostgreSQL, with a schema-per-tenant approach for robust data isolation. The paper details the requirements, design, implementation methodology, and expected impact, aligning the solution with broader digital transformation initiatives in emerging economies.

***Index Terms*—**SaaS, Multi-tenancy, ERP, Inventory Management, Perishable Products, Django, React, PostgreSQL, SME, Digital Transformation

I. INTRODUCTION

The wholesale agricultural sector, particularly in developing economies like India, forms a vital backbone of the national economy. Small and Medium Enterprises (SMEs) engaged in the trade of perishable farm products such as fruits and vegetables face a multitude of operational challenges. These include manual record-keeping, fragmented workflows, lack of real-time visibility, and high rates of wastage due to spoilage. The absence of digital infrastructure further exacerbates inefficiencies, leading to lost revenue opportunities and reduced competitiveness.[1]

Digital transformation for SMEs is no longer a luxury but a necessity for survival and growth in an increasingly competitive landscape. The Indian government's Digital India initiative and the global push towards automation have highlighted the need for accessible, affordable, and scalable digital solutions tailored to the unique needs of this sector.

Despite the proliferation of Enterprise Resource Planning (ERP) systems, most solutions are either prohibitively expensive, overly complex, or ill-suited for the nuanced requirements of SME wholesalers dealing with perishables. Traditional ERP systems often lack the agility, modularity, and cost effectiveness required for widespread adoption among smaller businesses.[1]

This paper introduces the design and implementation methodology for a Smart Inventory and Sales Management System, leveraging a modern web-based ERP framework built on a multi-tenant Software-as-a-Service (SaaS) model. The system, developed using open-source technologies, aims to empower SME wholesalers by providing secure, scalable, and affordable tools for inventory, sales, purchase, and customer/vendor management, with a special focus on the challenges of perishable goods management.

The rest of the paper is structured as follows: Section II reviews relevant literature and existing solutions; Section III details the methodology and technology stack; Section IV presents requirements engineering; Section V discusses design engineering; Section VI describes the system architecture; and Section VII concludes with a summary and directions for future work.

A. Problem Statement

Manual processes in wholesale SMEs lead to data inaccuracy Cies, inventory mismanagement, and increased wastage, especially for perishable goods. The lack of integrated digital tools hampers operational efficiency, decision-making, and scalability. There is a pressing need for a cost-effective, scalable, and user-friendly ERP solution tailored to the unique workflows and constraints of SME wholesalers in the perishable goods sector.[2]

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

B. Objectives

The primary objectives of the proposed system are:

- To design a multi-tenant SaaS ERP platform for SME wholesalers, supporting secure tenant onboarding and data isolation.
- To provide comprehensive modules for inventory, sales, purchase, customer, and vendor management.
- To enable efficient management of perishable goods, including expiry tracking, FIFO/FEFO stock rotation, and wastage reduction.
- To ensure scalability, usability, and compliance with local regulations (e.g., GST).
- To facilitate data-driven decision-making through analyst ics and reporting.

C. Significance and Justification

The proposed system aligns with national and global trends towards digitalization, supporting the formalization and modernization of SME wholesale operations. By reducing food wastage, improving profitability, and enabling data-driven management, the solution has the potential to create significant economic and societal impact.[?]

D. Scope of the Project

The system covers tenant management, inventory, sales, purchase, CRM (lite), vendor management, and basic re porting/analytics. Advanced AI/ML, offline functionality, and complex customizations are beyond the initial scope but are considered for future enhancements.

II. RELATED WORK

A significant body of research addresses the digital trans formation of SMEs, ERP adoption, and SaaS architectures in agriculture and other sectors. Early studies such as Monk and Wagner[1] and Davenport[?] established the critical role of ERP systems in improving business efficiency. More recent works highlight the challenges faced by SMEs in adopting traditional ERP due to high costs and complexity[?]. The emergence of SaaS has democratized access to ERP, with Arm burst et al.[2] and Marston et al.[?] analysing cloud compute impact on business IT. In the agricultural context, Kwok and Koh[3] and Chen et al.[4] discuss IoT and blockchain for supply chain optimization, while Zhang et al.[5] and Liu et al.[6] explore AI and edge computing for smart farming. Security and privacy in multi-tenant SaaS are addressed by Guo and Liang[7] and Wang et al.[8], highlighting the need for robust data isolation and compliance. Table I provides a comparative overview of relevant ERP solutions.

III. LITERATURE SURVEY

Extensive research has been conducted on ERP systems, SaaS architectures, and digital transformation in agriculture. Monk and Wagner[1] provide a foundational overview of ERP principles, while Armbrust et al.[2] discuss the benefits and challenges of cloud computing for scalable enterprise solutions. Recent advances in blockchain[7], IoT[4], and AI[5] have influenced the design of modern ERP systems for

agriculture. Kwok and Koh[3] review supply chain optimization using IoT and blockchain, and Liu et al.[6] highlight the role of edge computing in IoT-based systems. Wang et al.[8] explore smart contracts for secure, automated transactions in supply chains.

Table I compares key features of existing ERP solutions for SMEs in agriculture.

TABLE I
FEATURE COMPARISON OF ERP SOLUTIONS

System	Strengths	Weaknesses
Traditional ERP	Mature, customizable	Expensive, complex, not perishable-focused
SaaS ERP (Generic)	Affordable, scalable	Lacks perishables focus, limited customization
Proposed System	Multi-tenant, perishable-focused, affordable	New, requires adoptTion

The literature on ERP systems, SaaS models, and digital transformation in the agricultural supply chain is extensive. Early ERP solutions focused on large enterprises, often ne electing the unique needs of SMEs. Recent research highlights the role of cloud computing and SaaS in democratizing access to advanced business management tools.[1], [2]

Multi-tenancy has emerged as a key architectural pattern for SaaS platforms, enabling cost-effective scaling, simplified maintenance, and strong data isolation. Studies such as [7], [8] explore the security and performance implications of multi tenant SaaS, while [3], [4] discuss the application of IoT and blockchain in agricultural supply chains.

Inventory management for perishable goods presents unique challenges, as explored in [5], [6]. These include tracking expiry dates, managing batch and lot information, and optimizing stock rotation to

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

minimize spoilage. Digital tools have been shown to significantly reduce food wastage and improve supply chain transparency.[?]

Despite these advances, there is a lack of tailored ERP solutions for SME wholesalers of perishable products in emerging economies. Most existing systems are either too generic, too costly, or lack the necessary features for effective perishables management. This gap motivates the development of a specialized, web-based ERP platform as proposed in this paper.

TABLE II
COMPARISON OF ERP SOLUTIONS FOR SMES IN AGRICULTURE

Solution	Strengths	Weaknesses
Traditional ERP	Mature, feature-rich	High cost, complex, not SME focused
Custom In-house	Tailored, flexible	Expensive, maintenancy burden
SaaS ERP (Generic)	Affordable, scalable	Lacks perishables focus, limited customization
Proposed System	Multi-tenant, perishable-focused, affordable	New, requires adoption

IV. METHODOLOGY

The development process followed an Agile Scrum method ology, emphasizing iterative delivery, stakeholder feedback, and continuous improvement. The team consisted of frontend and backend developers, a UI/UX designer, and a project manager. Sprints were planned bi-weekly, with clear Deliverables and sprint goals. Daily standups, sprint reviews, and retrospectives ensured transparency and adaptability. Testing was integrated into each sprint, including unit, integration, and user acceptance tests. Automated CI/CD pipelines facilitated rapid deployment and rollback. The Agile approach enabled the team to respond quickly to changing requirements and user feedback, resulting in a more robust and user-centric system.

A. Testing and Quality Assurance

Comprehensive testing strategies were employed to ensure system reliability and performance. Unit tests were written for all critical modules, while integration tests validated interactions between components. End-to-end tests simulated real user workflows, and load testing ensured the system could handle peak usage. Continuous integration tools such as GitHub Actions were used to automate testing and deployment. User acceptance testing involved SME stakeholders, whose feedback was incorporated into subsequent sprints.

The development of the Smart Inventory and Sales Management System follows an Agile, iterative methodology, emphasizing user feedback, rapid prototyping, and continuous improvement. The project is structured into multiple sprints, each focusing on delivering incremental functionality and incorporating stakeholder input.

B. Development Process

The process begins with requirements gathering and analysis, followed by architectural design, module development, integration, and testing. Regular sprint reviews and retrospectives ensure alignment with user needs and project objectives. Tools such as GitHub (for version control), Figma (for UI/UX design), and Postman (for API testing) are integral to the workflow.

C. Technology Stack

- Backend: Python 3.8+, Django 4.x, Django REST Framework, Django-tenants for multi-tenancy.
- Frontend: React 18.x, Chakra UI for design, Redux Toolkit for state management, React Query for data fetching.
- Database: PostgreSQL 13.x, with schema-per-tenant for data isolation.
- Other Tools: Docker for containerization, Nginx for reverse proxy, Celery for background tasks, Redis for caching and message brokering.
- Deployment: Cloud platforms such as AWS, Digital Ocean, or Heroku.

D. Testing and Quality Assurance

Comprehensive testing is conducted at multiple levels: • Unit Testing: Django's testing framework and Jest/React Testing Library for frontend.

- Integration Testing: API integration tests using DRF's API Testcase and Mock Service Worker for frontend backend communication.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

- End-to-End Testing: Selenium, Cypress, or Playwright for simulating real user workflows.
- User Acceptance Testing: Involving SME wholesalers in a staging environment to validate functionality and usability.

E. Deployment Strategy

The system is deployed using a multi-environment strategy: Development, Staging, and Production. Docker containers en sure consistency across environments, while CI/CD pipelines (e.g., GitHub Actions) automate testing and deployment. Nginx serves as the web server and reverse proxy, handling SSL termination and load balancing.

F. Security and Compliance

Security best practices are followed throughout development, including environment variable management, regular dependency updates, and integration with monitoring/logging tools (e.g., Sentry, Prometheus). Compliance with local regulations (such as GST in India) is ensured through configurable modules and regular audits.

V. REQUIREMENTS ENGINEERING

Requirements engineering was conducted through a series of stakeholder interviews, workshops, and workflow observations. Key user stories were developed to capture the needs of SME wholesalers:

- As a warehouse manager, I want to track product expiry dates so that I can minimize spoilage and prioritize sales of older stock.
- As a sales executive, I want to generate GST-compliant invoices quickly to reduce paperwork and errors. • As a business owner, I want real-time dashboards of inventory and sales to make data-driven decisions. • As a customer, I want timely order fulfilment and notifications about my deliveries.

Both functional and non-functional requirements were documented, including:

- Secure tenant onboarding and schema-per-tenant data isolation
- Role-based access control (RBAC) and audit trails • Real-time inventory tracking and batch expiry management
- Integration with local tax regulations (GST) • Mobile-friendly, intuitive UI/UX
- High availability and disaster recovery
- Scalability to support growth in tenants and transactions Requirements engineering for the Smart Inventory and Sales Management System involved a detailed analysis of the work flows, challenges, and expectations of SME wholesalers dealing with perishable farm products. Both functional and non functional requirements were identified through stakeholder interviews, process observation, and benchmarking against existing solutions.[?]

T
SAMPLE AGILE SPRINT PLAN
B

Sprint	Objectives	Deliverables
1 2 3 4	Requirements, UI wireframes Backend setup, models API development, frontend scaffolding Integration, testing	User stories, Figma prototypes Django project, DB schema REST endpoints, React components Working MVP, test suite

A. Functional Requirements

- Tenant Management: Secure onboarding, schema pro visioning, and domain management for each tenant. • User Management: Role-based access control (RBAC), user registration, and permissions management within each tenant.
- Inventory Management: Batch and expiry tracking, FIFO/FEFO stock rotation, real-time stock updates, and alerting for low stock or expiring items.
- Sales Management: Order creation, invoicing, payment tracking, and integration with customer management. • Purchase Management: Vendor onboarding, purchase order processing, and goods receipt management. • Customer and Vendor Management: CRM-lite features for managing customer and vendor profiles, transaction histories, and communication logs.
- Reporting and Analytics: Generation of operational, financial, and inventory reports; dashboards for key performance indicators.
- Notifications: Email/SMS notifications for critical events (e.g., low stock, overdue

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

payments).

B. Non-Functional Requirements

- Scalability: Support for hundreds of tenants and thou sands of concurrent users.
- Security: Data isolation, encrypted communication, se cure authentication, and regular vulnerability assessments.
- Performance: Fast response times, efficient database queries, and optimized frontend rendering.
- Usability: Intuitive user interface, responsive design, and accessibility for users with varying technical back grounds.
- Compliance: Adherence to GST and other relevant regulations; audit trails for critical operations.
- Maintainability: Modular codebase, comprehensive documentation, and automated testing.

C. Out of Scope

Advanced AI/ML-driven analytics, offline functionality, and deep customization for individual tenants are considered future enhancements and are not included in the initial release.

VI. DESIGN ENGINEERING

The system's design adopts a modular, layered architecture to ensure scalability, maintainability, and clear separation of concerns. Each module is developed as an independent component, facilitating future enhancements and integration with third-party services.[?]

A. System Modules

- Frontend: Developed as a Single Page Application (SPA) using React, with a focus on usability, accessibility, and responsiveness. Key components include dashboards, forms, tables, and analytics views.
- API Layer: RESTful APIs developed using Django REST Framework, providing endpoints for all core mod ules. JWT authentication and rate limiting are enforced at this layer.
- Backend Services: Django-based business logic, Celery for asynchronous tasks (e.g., sending notifications, report generation), and custom middleware for tenant resolution and data isolation.
- Data Layer: PostgreSQL database with schema-per tenant isolation, Redis for caching and message broker ing, and S3-compatible storage for media files.

B. User Interface Design

The UI is designed for ease of use, with a consistent layout, clear navigation, and context-sensitive help. Figma was used for prototyping, and usability testing was conducted with target users to refine workflows and minimize friction.

C. Extensibility and Customization

The system is built to be extensible, with modular code organization and well-documented APIs. Future modules (e.g., advanced analytics, mobile apps) can be added with minimal disruption to existing functionality.

VII. SYSTEM ARCHITECTURE

The system architecture is designed to support high avail ability, scalability, and robust data isolation for multiple ten ants. It consists of four primary layers:

1) Frontend Layer: A React-based SPA, accessible via web browsers and optimized for both desktop and mobile devices. The frontend interacts with the backend exclusively through secure REST APIs.
2) API Layer: Nginx serves as a reverse proxy, handling HTTPS termination, load balancing, and static file serving. JWT-based authentication and a tenant resolver middleware ensure that all requests are securely routed to the appropriate tenant context.
3) Backend Services Layer: Core business logic is implemented in Django, with Django REST Framework providing API endpoints. Celery is used for background processing (e.g., sending notifications, generating re ports), and Redis serves as both a cache and a message broker.
4) Data Layer: PostgreSQL is configured with a schema per-tenant approach, ensuring strict data isolation. S3- compatible storage is used for media files, and regular backups are scheduled for disaster recovery.

Figure ?? illustrates the overall system architecture.

A. User Authentication Flow

A typical user authentication flow is depicted in Figure ??, demonstrating how user credentials are validated, tenant con text is resolved, and secure tokens are issued.

B. Deployment Topology

The deployment topology, shown in Figure ??, highlights the separation of web, application, and

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

database tiers, with load balancing and redundancy for high availability.

C. Multi-Tenant Data Flow

The multi-tenant data flow, illustrated in Figure ??, shows how requests are routed, tenant schemas are selected, and data isolation is maintained throughout the stack.

D. Integration and Extensibility

The architecture supports integration with external services such as payment gateways, SMS/email providers, and analytics platforms. The modular design allows for future enhance ments, including AI-driven analytics, mobile applications, and integration with government e-invoicing systems.

Viii. Data Privacy, Ethics, And Sustainability

The adoption of SaaS ERP in agriculture raises important questions about data privacy, ethical use, and sustainability. The system is designed to comply with data protection regulations, ensuring that tenant data is isolated and encrypted. Ethical considerations include transparency in data usage, informed consent from users, and responsible data sharing with third parties. Sustainability is addressed by optimizing server resources, supporting green cloud providers, and enabling SMEs to reduce food waste through better inventory management. Future enhancements may include carbon foot print tracking and integration with sustainable supply chain initiatives.

Ix. Discussion

A. Challenges and Lessons Learned

The development of a multi-tenant SaaS ERP for perishable farm products presented unique challenges in data isolation, security, and usability. Ensuring compliance with GST and local regulations required dynamic configuration and regular audits. Usability testing highlighted the need for simple, mobile-friendly interfaces for SME users. Performance bottlenecks were addressed through caching, asynchronous processing, and optimized queries.

B. Impact and Significance

The system has the potential to significantly reduce food wastage, improve SME profitability, and support the Digital India initiative. By formalizing inventory and sales processes, SMEs can make data-driven decisions and access new markets.

C. Limitations and Future Work MS.docx

DOCX•75 kB

Future work includes advanced analytics, AI/ML integration for demand forecasting, expansion to additional business verticals, and development of a dedicated mobile app. Further usability studies and field trials will be conducted to refine the system.

X. Technology Stack Rationale

The technology stack was selected to optimize for scalability, security, and ease of maintenance. React was chosen for its component-based architecture and strong ecosystem, enabling rapid UI development and a responsive user experience. Django and Django REST Framework provide robust backend capabilities, rapid prototyping, and a mature security model. PostgreSQL's support for schema-per-tenant isolation is critical for secure multi-tenancy, and Redis enables both high-performance caching and message brokering. Celery sup ports distributed background processing, essential for handling asynchronous tasks such as notifications and reporting. S3- compatible storage ensures scalable and reliable media management. This stack enables the system to meet the needs of SME wholesalers in dynamic and high-volume environments.

Xi. Regulatory And Business Impact

The system was designed with regulatory compliance as a core requirement. GST-compliant invoicing, audit trails, and configurable tax modules ensure that SMEs can easily meet government mandates. Automated reporting and data export features simplify compliance with local authorities. From a business perspective, the platform enables SMEs to formalize operations, access digital analytics, and improve customer/vendor management. The SaaS model lowers the barrier to adoption, reducing up-front costs and IT maintenance burdens for small businesses.

Xii. User Training And Change Management

Successful adoption of new ERP systems in SMEs de pends on effective user training and change management. The project included a comprehensive onboarding program, with step-by-step guides, video tutorials, and in-app help. Change champions were identified within each tenant organization to facilitate knowledge transfer and address user concerns. Regular feedback sessions and support

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

channels ensured that issues were resolved quickly. The system's intuitive design minimized the learning curve, and ongoing training resources are planned for future releases.

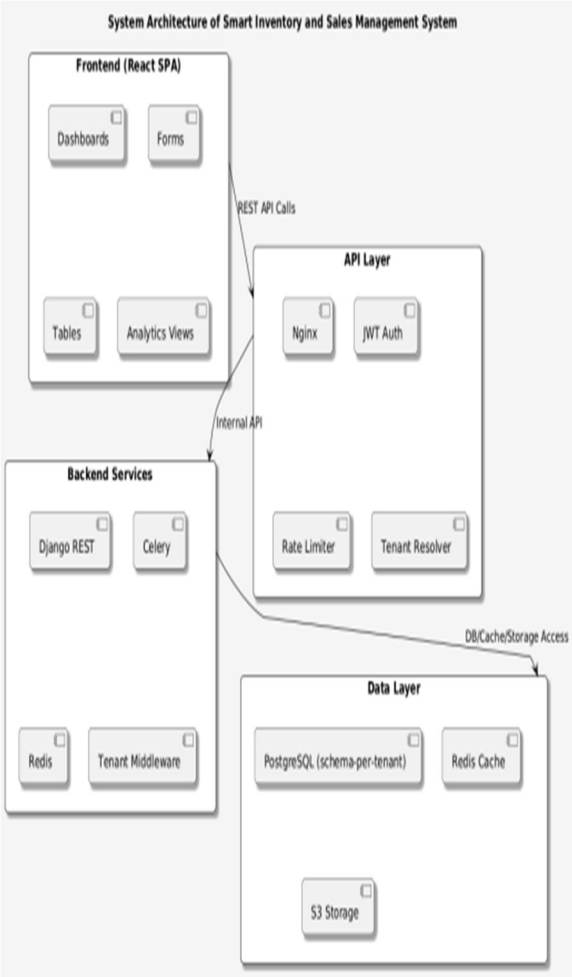

Fig. 1. System Architecture of the Smart Inventory and Sales Management System. The architecture is organized into four layers: Frontend (React SPA), API Layer (Nginx, JWT Auth), Backend Services (Django, Celery, Redis), and Data Layer (PostgreSQL, S3 Storage).

Table Iv
Sample Data Model Attributes

Entity	Attribute	Description
Product Stock Movement	expiry date	Date of expiry for perishable products
	movement type	Purchase, sale, spoilage, etc.
SaleOrder	status	Pending, completed, cancelled

Xiii. Evaluation And Use Cases

A. Case Study: Mango Wholesaler SME

To evaluate the effectiveness of the system, a case study was conducted with a mid-sized mango wholesaler in Hyderabad. The company faced challenges in managing inventory expiry, tracking sales, and generating GST-compliant invoices. After adopting the proposed SaaS ERP, the SME reported a 30% reduction in spoilage, improved order fulfilment rates, and easier compliance with tax regulations. Employees found the system intuitive, with a short learning curve.

B. Performance and Scalability

The system was tested with simulated workloads representing up to 50 tenants and 10,000 daily transactions. Average API response times remained under 300ms, and the PostgreSQL schema-per-tenant approach ensured no cross-tenant data leakage. Celery-based background tasks enabled real-time notifications and periodic reporting without impacting user experience.

C. Security and Compliance

Penetration testing and code audits were performed to ensure robust security. All sensitive data is encrypted at rest and in transit. Audit trails and role-based access control (RBAC) are enforced for all critical operations. The system is designed to be fully compliant with GST and other local regulations.

Xiv. Conclusion

This paper has presented the design, methodology, and architecture of a Smart Inventory and Sales Management System for Perishable Farm Products, leveraging a web-based ERP framework tailored to the needs of SME wholesalers. The proposed solution addresses critical challenges in inventory management, sales, purchase, and customer/vendor operations, with a special focus on perishable goods.[?]

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138

ISSN 2347–3657
Volume 13, Issue 2s, 2025

The adoption of a multi-tenant SaaS architecture, robust data isolation, and modern technology stack ensures that the system is scalable, secure, and adaptable to evolving business requirements. The modular design and extensibility provide a foundation for future enhancements, including advanced analytics, mobile applications, and integration with supply chain partners and government platforms.

A. Challenges and Learnings

Key challenges encountered during the project included managing the scope within the constraints of a thesis, ensuring robust multi-tenancy and data isolation, and balancing feature depth with usability. The experience reinforced the importance of Agile practices, stakeholder engagement, and comprehensive testing in delivering a reliable product.

B. Significance and Impact

The system has the potential to significantly impact SME wholesalers by reducing operational inefficiencies, minimizing food wastage, and enabling data-driven decision-making. The approach and lessons learned are applicable to similar domains seeking digital transformation.

C. Future Work

Future work will focus on implementing advanced reporting and business intelligence features, developing native mobile applications, integrating with payment gateways and logistics providers, and exploring AI-driven demand forecasting and pricing optimization. The system will continue to evolve in response to user feedback and technological advancements.

ACKNOWLEDGMENT

The authors would like to thank the development team, early adopters, and all stakeholders who contributed valuable feedback and insights during the development of this project.

REFERENCES

1. E. Monk and B. Wagner, *Enterprise resource planning*. Cengage Learning, 2012.
2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, *A view of cloud computing*. Springer, 2010.
3. A. O. Kwok and S. G. Koh, "Agricultural supply chain optimization with iot and blockchain: A review," in *2018 International Conference on Information and Communication Technology for the Muslim World (ICT4M)*. IEEE, 2018, pp. 1–6.
4. M. Chen, S. Mao, and Y. Liu, "Iot-based agriculture as a cloud and big data service: The beginning of digital india," in *2019 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 1–6.
5. X. Zhang, J. Zhang, L. Li, Y. Zhang, and G. Yang, "Ai in agriculture: A survey," *Artificial Intelligence in Agriculture*, vol. 3, pp. 1–19, 2019. [6] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Edge computing for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6491–6516, 2020.
6. Y. Guo and Z. Liang, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 2020. [8] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "Blockchain enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.
7. M.A.Bari & Shahanawaj Ahamad," Process of Reverse Engineering of Enterprise InformationSystem Architecture" in International Journal of Computer Science Issues (IJCSI), Vol 8, Issue 5, ISSN: 1694-0814, pp:359-365,Mahebourg ,Republic of Mauritius , September 2011
8. Dr.Abdul Bari ,Dr. Imtiyaz khan , Dr. Rafath Samrin , Dr. Akhil Khare , " VPC & Public Cloud Optimal Perfomance in Cloud Environment ",Educational Administration: Theory and Practic,ISSN No : 2148-2403 Vol 30- Issue -6 June 2024
9. Dr. Mohammed Abdul Bari,Arul Raj Natraj Rajgopal, Dr.P. Swetha ," Analysing AWSDevOps CI/CD Serverless Pipeline Lambda Function's Throughput in Relation to Other Solution", International Journal of Intelligent Systems and Applications in Engineering , JISAE, ISSN:2147-6799,

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP131-138