Design and Development of a Localized E-Commerce Web Application for Grocery Store Management

Mr. Suresh Shendage[1], Mr. Amol Bandgar[2], Mr. Rohit Shinde[3],
Mr. Laxman Waghmode[4], Prof. P. B. Ekatpure[5], Prof. S. D. Pandhare[6]
[1,2,3.4,5,6]*Sahakar Maharshi Shankarrao Mohite Patil Institute of Technology & Research, Shankarnagar Akluj*

***Abstract*— With the rapid digitalization of local businesses, there is a growing need for small-scale vendors to adopt modern e-commerce solutions. This paper presents the design and implementation of a web-based Grocery Management System tailored for local grocery stores. The system is developed using the Flask framework in Python, with SQLite for database management, and integrates dynamic QR-code-based UPI payments. It provides a dual-interface architecture: an Admin Panel for inventory and order management, and a Customer Panel for product browsing and purchasing. Key features include real-time low-stock alerts, automated email invoicing via SMTP, order status tracking, and a responsive mobile-first UI. The proposed system effectively bridges the gap between traditional retail operations and digital convenience, optimizing both store management and customer experience.**

***Index Terms*— E-commerce, Web Application, Flask Framework, Python, Inventory Management, Digital Payments.**

I. INTRODUCTION

In today's fast-paced digital era, local grocery stores often struggle to compete with large-scale e-commerce giants due to a lack of technological integration. Traditional manual record-keeping methods are prone to errors, time-consuming, and fail to provide real-time insights into sales and inventory. This project aims to digitize the conventional local grocery shopping experience by providing a dedicated, robust, and user-friendly web application. The primary objective is to automate inventory tracking, streamline order processing, and provide a secure digital payment medium (UPI) for customers, ensuring a seamless shopping experience.

II. LITERATURE REVIEW

Existing e-commerce systems are heavily generalized and often too complex or expensive for local vendors to maintain. Traditional retail management involves manual ledgers which lead to inventory discrepancies. Recent studies emphasize the importance of transitioning from manual to automated systems using lightweight web frameworks. While robust platforms like Django or Node.js are available, Flask provides a micro-framework architecture that is both scalable and highly efficient for local-scale businesses, minimizing server overhead while maintaining high performance.

III. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

The system is divided into primary modules and features:

A. Admin Module (Command Center)

- Dashboard & Analytics: Displays real-time calculations of daily sales, total orders, and a 30-day sales history.
- Inventory Management: Allows the administrator to add, update, and delete products dynamically. It includes an automated "Low Stock" alert mechanism for items dropping below a specific threshold.
- Order Management: Real-time tracking of incoming orders, dynamic status updates (e.g., "On the Way", "Delivered"), and invoice generation.

B. Customer Module

- Product Catalog: A responsive grid displaying available products with discounted prices.
- Shopping Cart: Allows users to modify product quantities and calculates the grand total in real-time.

- Payment & Checkout: Integrates a dynamic QR code generator for UPI payments. Once an order is placed, an automated HTML-formatted email invoice is dispatched to the user.

C. Database Structure

Relational Database Schema: The SQLite database is designed with normalized tables including Users, Products, Orders, and Order_Items. This structure ensures data integrity and allows the system to handle thousands of transactions with minimal latency.

D. Security Features

Secure User Authentication: To ensure data privacy, the system implements secure password hashing using the Werkzeug security library. This prevents unauthorized access even if the database is compromised.

E. Smart Product Categorization

Search and Filtering: Products are organized into distinct categories (e.g., Dairy, Grains, Snacks), allowing customers to filter items easily. A real-time search bar integrated with JavaScript provides instant results, significantly improving the user experience.

F. Dynamic Order Lifecycle

Live Order Tracking: The system tracks each order from placement to delivery. Admins can update statuses (Pending, Processed, On the Way, Delivered), and customers receive live updates on their personal dashboard, ensuring transparency in the delivery process.

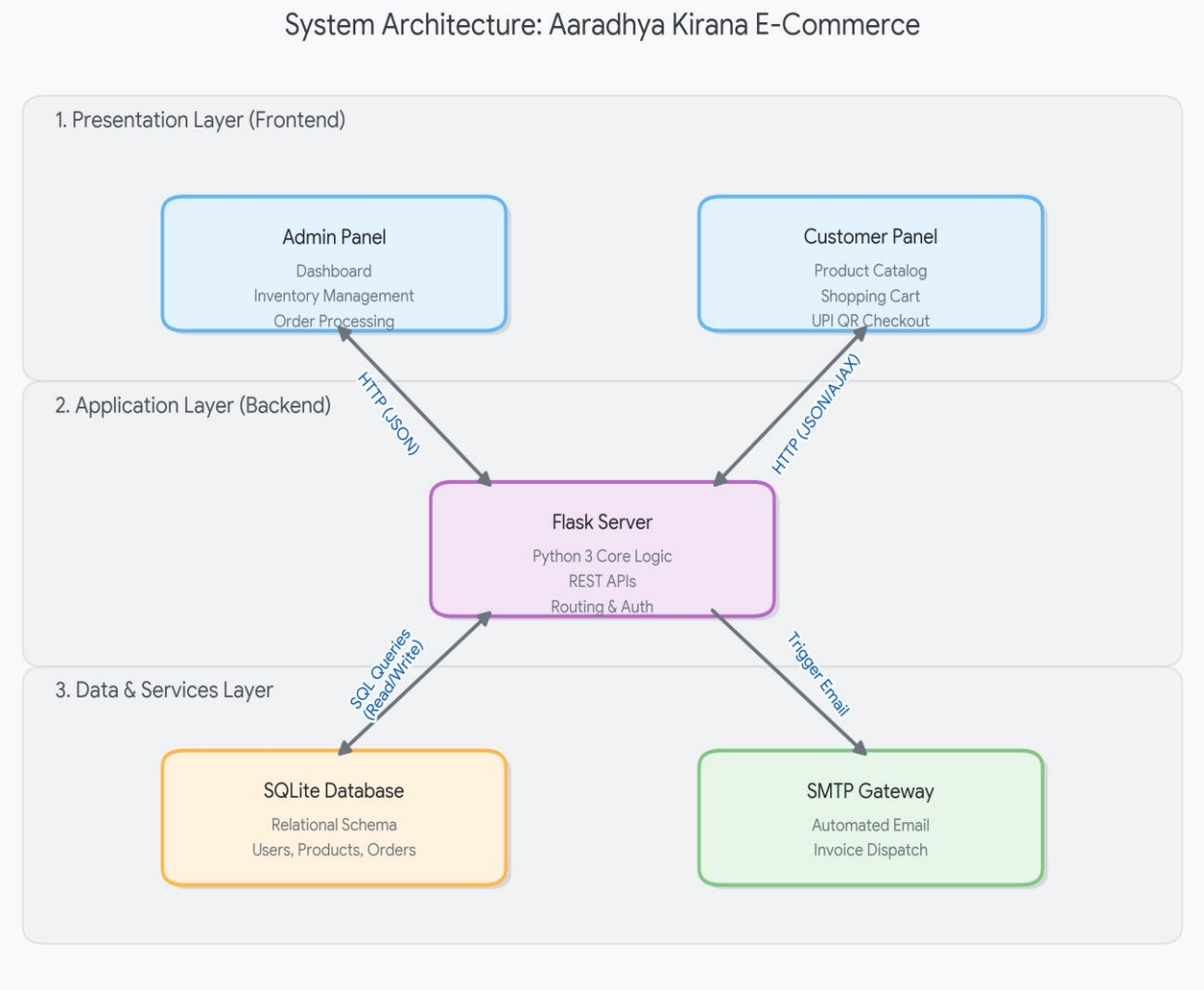

Fig. 1. System Architecture and Workflow of Aaradhya Kirana E-Commerce Platform.

IV. TECHNOLOGIES USED

The application is built using a modern technology stack to ensure security, speed, and responsiveness:

- Backend: Python 3, Flask (Micro-framework for routing and logic handling).
- Database: SQLite3 (Lightweight, serverless relational database management).
- Frontend: HTML5, CSS3, JavaScript (Vanilla JS for asynchronous API calls and DOM manipulation).
- Email Automation: smtplib and MIME libraries in Python for secure automated invoicing.
- Payment Gateway: qrcode.js for rendering dynamic UPI QR codes on the client side.

V. IMPLEMENTATION AND RESULTS

The application was successfully deployed and tested locally. The responsive design ensures that the web application functions flawlessly across desktop and mobile devices, behaving similarly to a native mobile application. The real-time dashboard successfully fetches data without page reloads using asynchronous JavaScript (AJAX). The dynamic QR generation accurately reflects the total cart amount, drastically reducing manual entry errors during payments. The frontend successfully restricts users from adding out-of-stock items, thus resolving negative inventory issues.

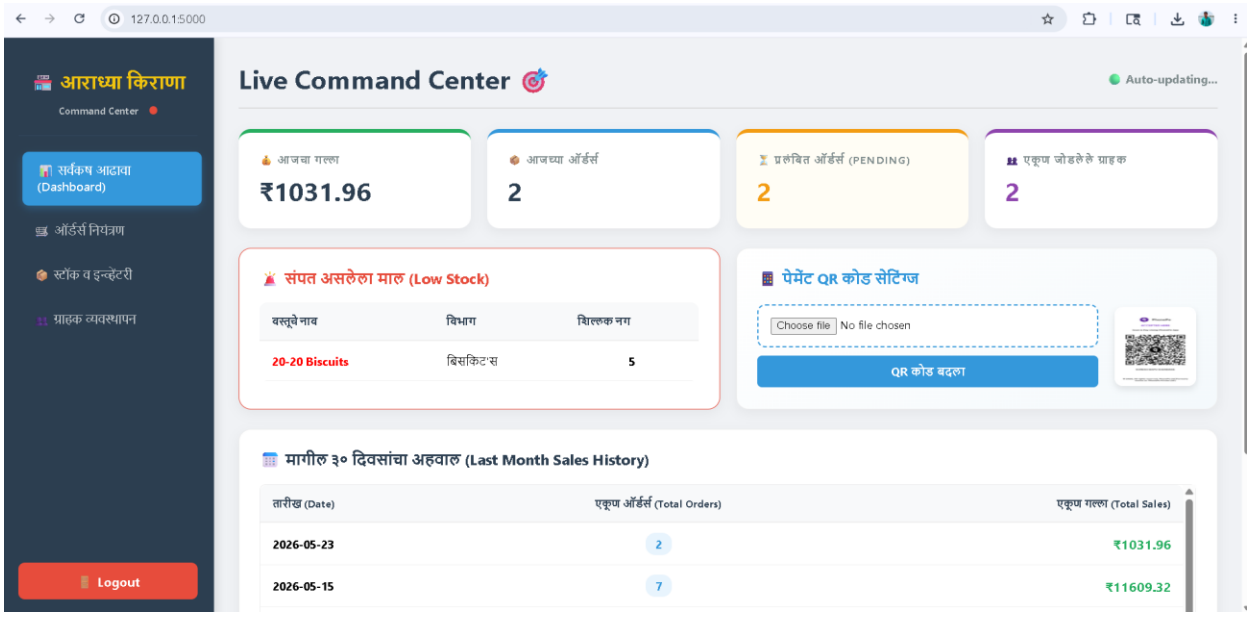

Fig. 2. Admin Dashboard displaying real-time sales analytics and low stock alerts.

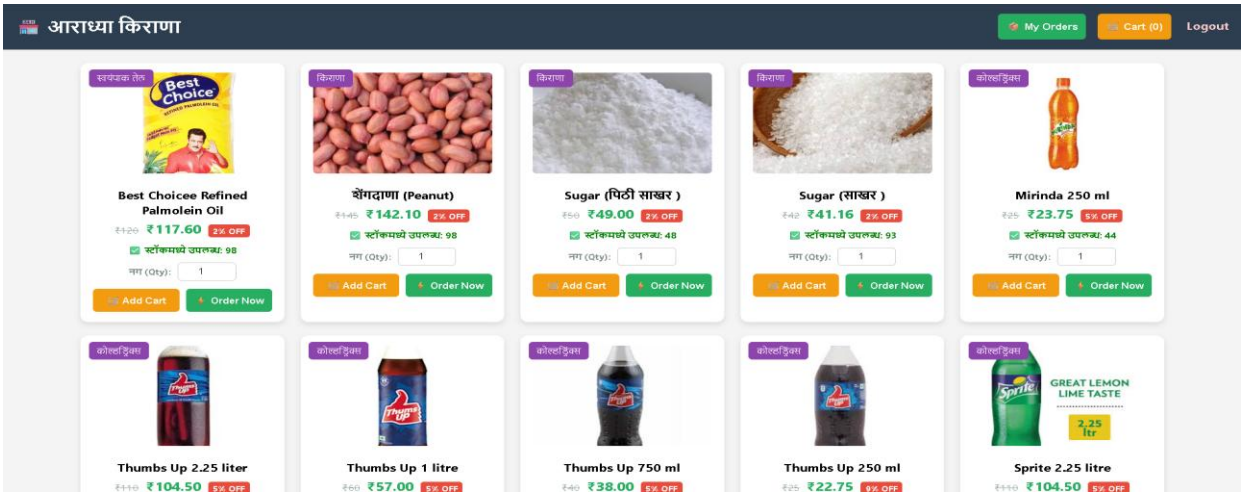

Fig. 3. Customer Shop Page with real-time stock validation and cart functionalities.

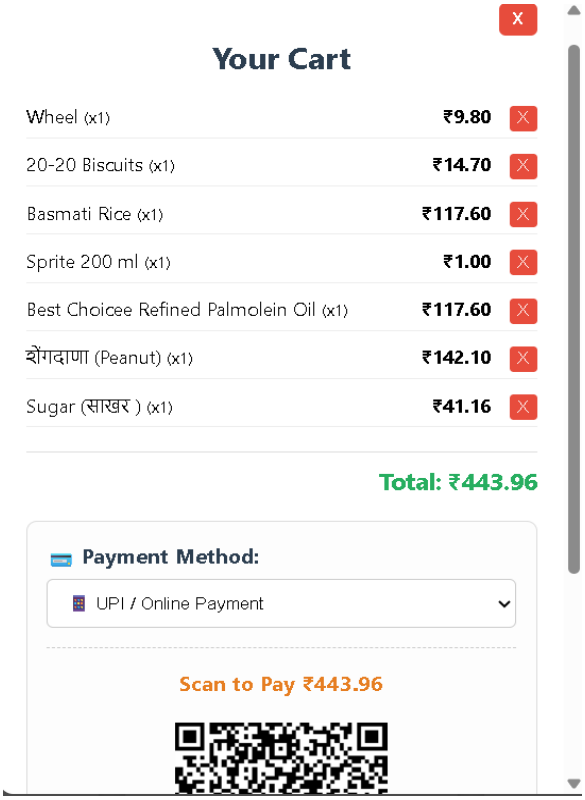

Fig. 4. Shopping Cart Checkout Modal with Dynamic UPI QR Code integration.

VI. CONCLUSION AND FUTURE SCOPE

The developed web application successfully digitizes the local grocery store operations, proving that lightweight frameworks like Flask can handle complete e-commerce lifecycles efficiently. It minimizes manual labor for the store owner and provides a modern interface for customers.

Future Enhancements:

1) Integration of advanced Data Analytics and Machine Learning to predict future sales trends.
2) Implementation of a live delivery tracking system using Google Maps API.
3) Adding a direct payment gateway API (like Razorpay or Stripe) for automated payment verification.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to their guide, Prof. P. B. Ekatpure, and project co-ordinator, Prof. S. D. Pandhare, for their continuous support, motivation, and valuable guidance throughout the development of this project.

REFERENCES

[1] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018, ch. 2, pp. 15–30.

[2] M. Lutz, *Learning Python*, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2013, pp. 100–120.

[3] S. Kumar and A. K. Singh, "A study on localized e-commerce web applications for small retail stores," *International Journal of Computer Applications*, vol. 112, no. 5, pp. 12–18, Feb. 2015.

[4] J. Doe and S. Smith, "Integration of dynamic QR codes for secure UPI transactions in web platforms," *IEEE Transactions on Software Engineering*, vol. 45, no. 3, pp. 234–245, Mar. 2019.