Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

Design and Development of Emphramark: A Web-Based Stock Management System for Kirana Stores in Local India

[1]Shashwat Inaganti, [2]Pawan Kumar
[1]Student, [2]Assistant Professor
[1,2]Amity University Chhattisgarh
[1]shashwatinagantil@gmail.com, [2]pkumar@rpr.amity.edu

ABSTRACT

This research paper presents the design and development of Emphramark a web-based stock management system intended for kirana stores and small local retailers in India. The system addresses a common operational problem: many small stores still depend on handwritten registers, memory-based tracking, or loosely maintained spreadsheets, making it difficult to maintain accurate stock visibility, identify low-stock items, and update quantities during daily business. Emphramark provides a browser-accessible application that combines secure authentication, dashboard-driven inventory overview, manual item creation, stock-in and stock-out operations, low-stock alerting, profile settings, and barcode-assisted quick product entry.

The implementation uses React and TypeScript for the frontend, Vite for development and bundling, Tailwind CSS and shadcn-ui for user interface construction, and Supabase for authentication, relational database storage, row-level security, and edge-function execution. A distinctive feature of the system is its quick scan workflow, which uses camera-based barcode scanning, product lookup through Open Food Facts where available, and an AI-backed edge function to suggest category and unit values. The project demonstrates how modern web technologies can be combined to create a focused, secure, and context-aware inventory solution for small retail environments.

Keywords: Kirana Store, React, TypeScript, AI Categorization, Low-Stock Alerts, Web Application

I. INTRODUCTION

Inventory management is a basic but critical activity in retail businesses because it connects purchasing, storage, replenishment, pricing, and sales readiness. In organized retail, this activity is usually supported by dedicated software systems and specialized staff. In local kirana stores, however, inventory management is frequently handled through memory, handwritten notebooks, informal verbal communication, or delayed manual updates. These methods may appear manageable at a very small scale, but they become unreliable as the number of products increases, demand fluctuates, and product turnover accelerates.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

Kirana stores occupy an important place in the Indian retail ecosystem. They are local, accessible, trusted, and deeply connected to neighborhood life. Their operational model depends on high item variety, limited shelf space, rapid daily transactions, and quick customer service. Store owners must track packaged goods, household items, personal care products, beverages, grains, spices, and other fast-moving stock categories while simultaneously handling procurement, billing, and customer interaction.

The main design challenge is not simply to build another inventory system. A useful system for kirana stores must be lightweight, fast to learn, usable from common devices, and structured around essential daily actions. Emphramark was developed with this context in mind. It is a web-based stock manager with secure authentication, a dashboard inventory overview, low-stock alerts, stock update actions, store profile settings, and a quick scan mechanism that reduces product entry effort.

A. Objectives of the Study

- Design a stock management system aligned with kirana store operations.
- Provide secure user authentication and protected access to shop-specific data.
- Enable manual product creation with commercial and stock-related fields.
- Support stock-in and stock-out operations with transaction meaning.
- Generate low-stock visibility from stored quantity and threshold values.
- Reduce product-entry effort through barcode scanning and AI-backed suggestions.
- Maintain profile information such as shop name and default threshold.
- Document the project in an academic research-paper format suitable for submission.

B. Scope of the Work

The current system focuses on a single-store authenticated stock management workflow. It includes login and signup, dashboard overview, item creation, stock updates, low-stock alerts, quick scan-assisted product insertion, and settings management. The system does not yet replace billing, supplier management, full accounting, or comprehensive analytics. Its strength lies in being appropriately scoped around stock visibility, convenience, and secure item-level management.

II. LITERATURE REVIEW

Small-business inventory systems generally evolve through three stages. The first stage is manual recording, where notebooks or loose registers are used to track goods. The second stage is spreadsheet-supported recording, where item lists and quantities are maintained digitally but most update logic remains manual. The third stage is a dedicated digital stock application, where

authentication, structured storage, validation, and real-time visibility are integrated into a single interface.

Emphramark belongs to the third category, but it does not attempt to imitate full enterprise resource planning systems. Instead, it adopts a lightweight architecture suitable for local stores. This distinction matters because system quality should be evaluated relative to intended users. A large feature set is not automatically better if the target user only needs a reliable subset of functions.

Modern web development frameworks make focused solutions increasingly achievable. React supports component-oriented user interface construction through reusable pages, dialogs, forms, tables, and protected routes. TypeScript improves maintainability by adding static typing to application logic. Backend-as-a-service platforms such as Supabase provide authentication, relational storage, and policy enforcement without requiring a custom backend server for every function.

Barcode-assisted product entry is also relevant to retail systems because it reduces typing effort and speeds product identification. However, barcode scanning alone does not guarantee a complete record. Emphramark therefore supplements barcode reading with external product lookup and AI-assisted category and unit suggestion. This layered method combines deterministic input capture with assistive intelligence while keeping the user in control of final values.

III. PROBLEM STATEMENT

The core problem addressed by Emphramark is the difficulty of maintaining timely and reliable stock records in local kirana stores using manual or semi-manual methods. When product entries are not centralized digitally, it becomes difficult to identify low-stock items, update quantity changes consistently, and preserve price or unit information across time.

- No real-time stock visibility: Stock levels are often known only after manual checking.
- Manual record errors: Handwritten registers and memory-based systems can miss updates or contain inconsistent values.
- High product-entry effort: If entering products takes too long, store owners may postpone updates.
- Weak data isolation: Store data should remain associated with the correct authenticated user.
- Lack of structured transaction history: Quantity changes are more useful when the system records the event that caused the change.

Therefore, the research question is: how can a lightweight, user-friendly, and secure stock management system be designed for kirana stores using modern web technologies while minimizing the burden of data entry?

IV. PROPOSED METHODOLOGY / MODEL

A. System Architecture / Design

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

The system follows a client-service architecture. The client side is a React application responsible for user interaction, route transitions, forms, dialog flows, local UI state, and quick scan workflow. Backend services are delegated to Supabase, which handles authentication, relational database persistence, row-level security, and edge-function execution. External services support barcode lookup and AI-assisted categorization.

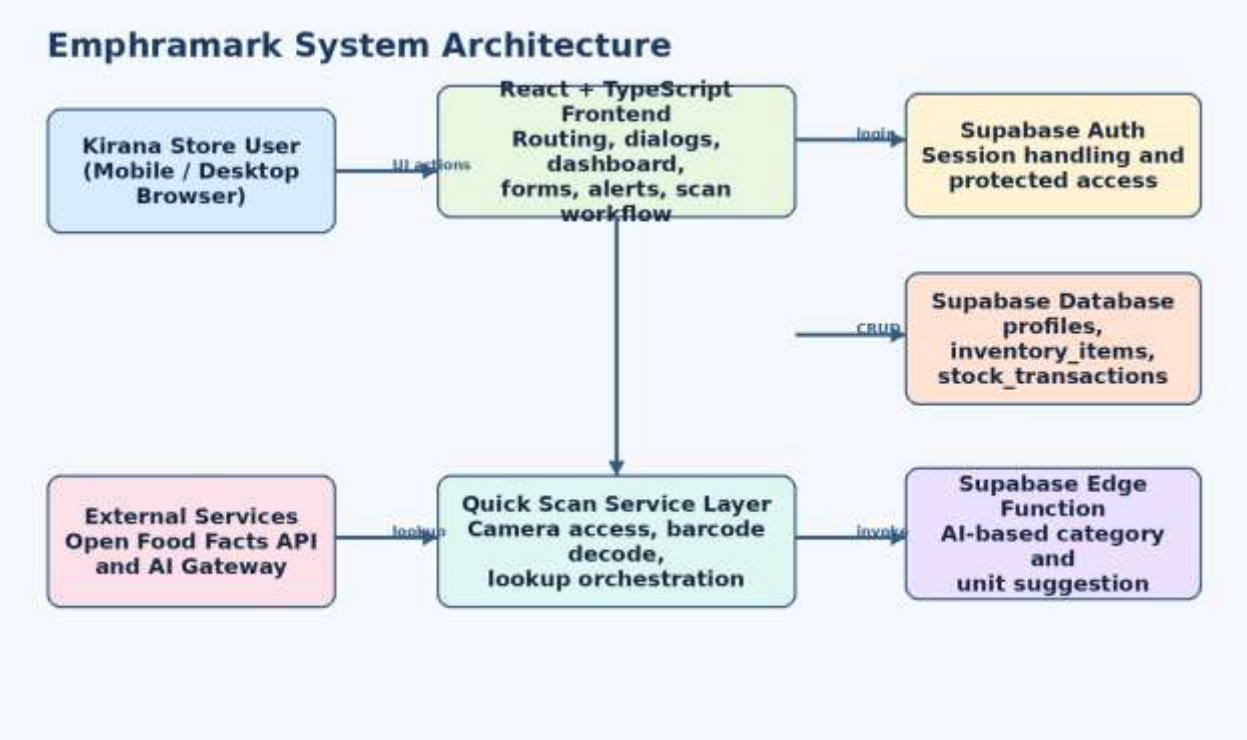

Fig. 1: Emphramark high-level system architecture

B. Authentication and Access Flow

The authentication flow begins when the user opens the application. If an existing Supabase session is available, the user is allowed to access protected routes. If no session exists, the user is redirected to the login page. Successful login creates a session, after which the protected dashboard becomes available.

C. Stock Update and Alert Flow

The stock update flow allows the user to select an item, choose stock-in or stock-out, enter a quantity and notes, validate the requested action, update the inventory quantity, and insert a stock transaction record. Low-stock status is recalculated from current quantity and threshold values after updates.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

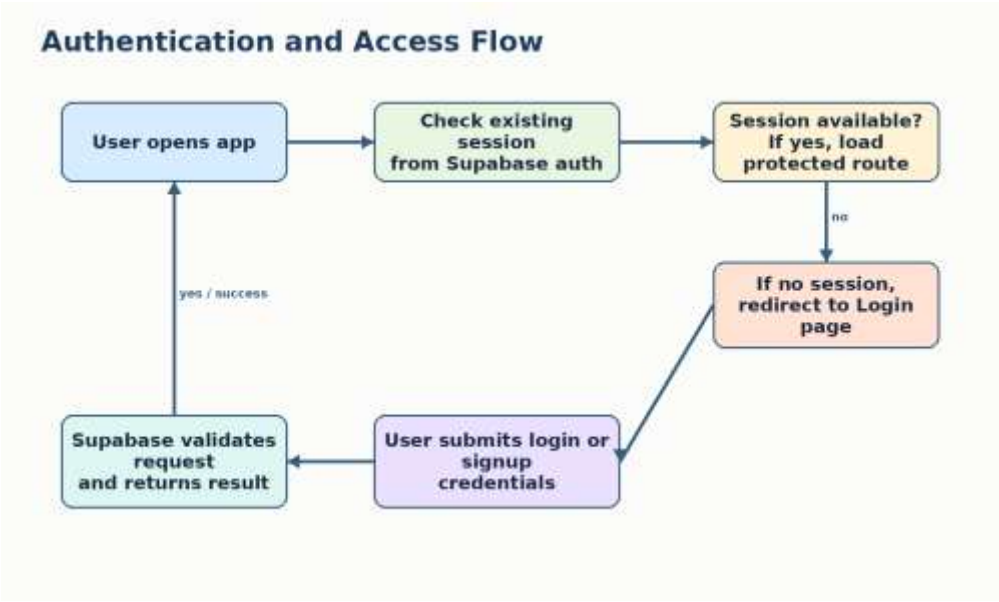

Fig. 2: Authentication and protected route flow

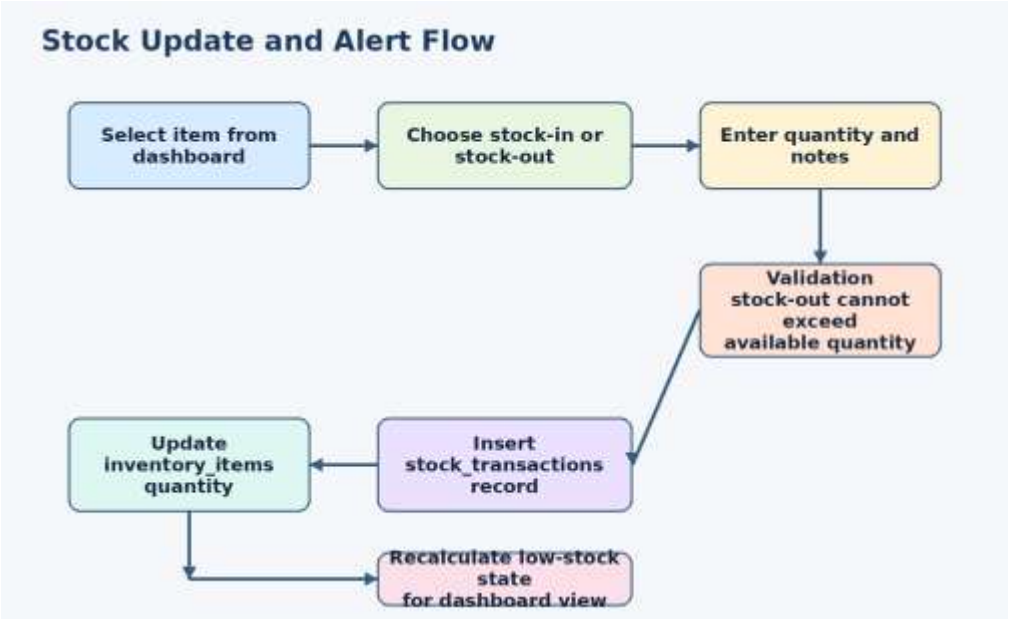

Fig. 3: Stock update and low-stock alert flow

D. Quick Scan Product Entry Flow

The quick scan flow combines camera access, barcode decoding, product lookup, AI-assisted categorization, user review, and final save. The user remains in control and can correct suggested values before inserting the item into inventory.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

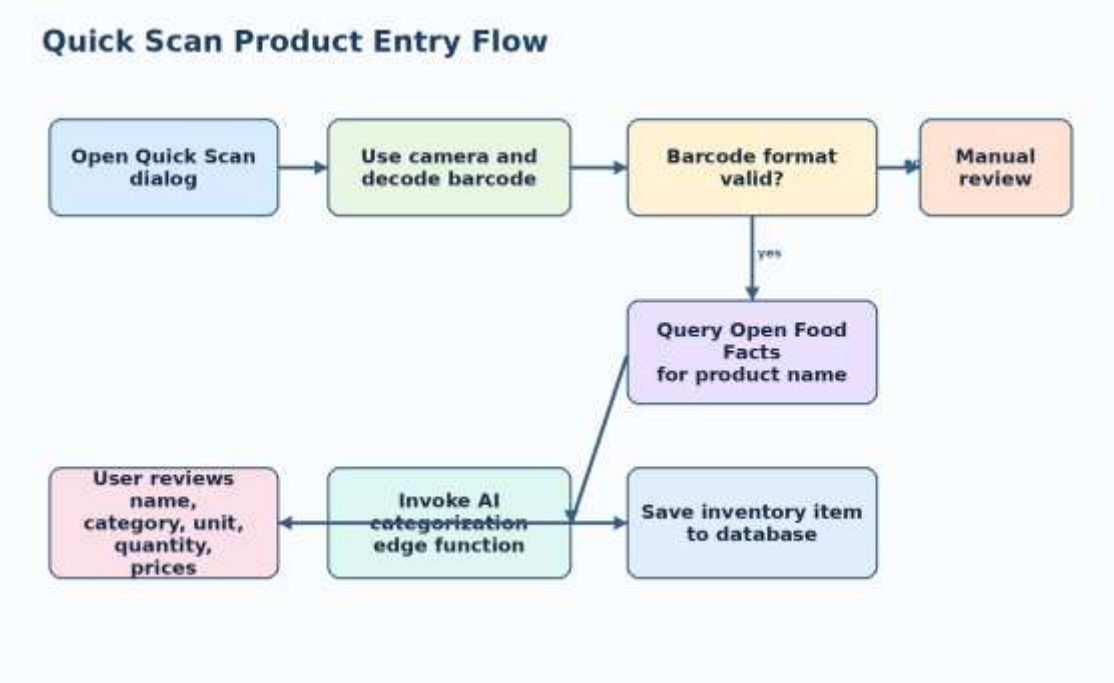

Fig. 4: Quick scan product entry flow

E. Database Schema Design

The database design is intentionally compact. It models store-user profile data, live inventory records, and stock transaction history. Row-level security policies restrict access based on the authenticated user identifier, protecting data ownership at the database layer.

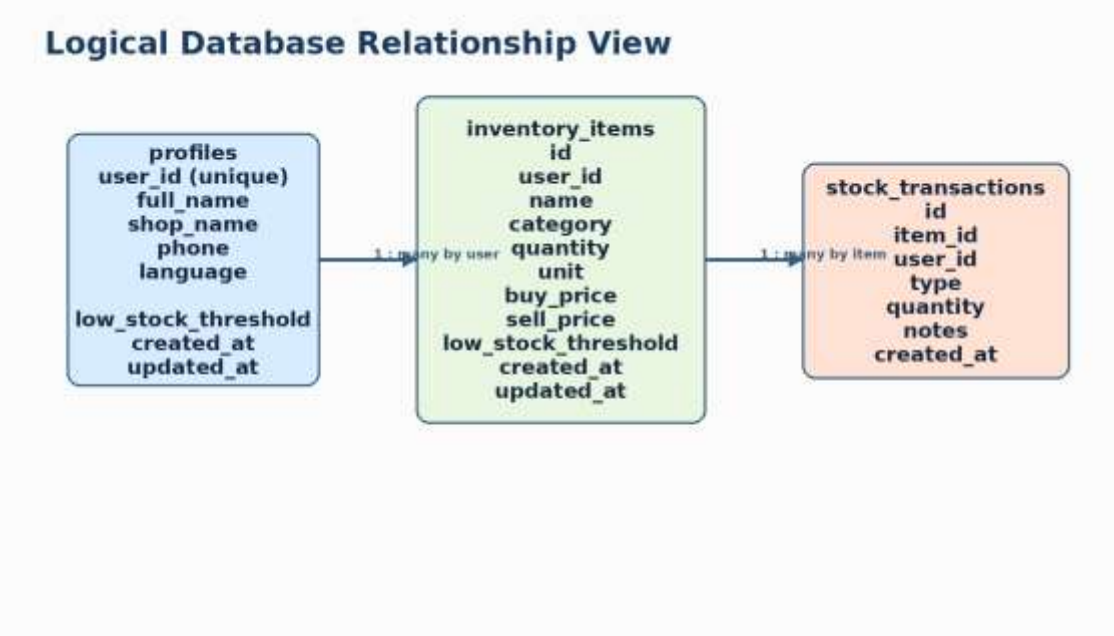

Fig. 5: Logical database relationship view

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

TABLE I - CORE DATABASE ENTITIES

Entity	Main Attributes	Purpose
profiles	user_id, full_name, shop_name, phone, language, low_stock_threshold	Stores user and store identity information.
inventory_items	name, category, quantity, unit, buy_price, sell_price, low_stock_threshold	Stores live inventory records shown on the dashboard.
stock_transactions	item_id, user_id, type, quantity, notes, created_at	Stores quantity-change history for stock operations.

F. Algorithms and Techniques Used

TABLE II - KEY ALGORITHMS AND TECHNIQUES

Technique	Location	Purpose
Protected Route Guard	React Router / Auth Context	Redirects unauthenticated users away from dashboard and settings pages.
Row-Level Security	Supabase PostgreSQL	Restricts user access to rows owned by the authenticated account.
Low-Stock Thresholding	Dashboard Module	Marks items as low stock when quantity is less than or equal to threshold.
Stock-Out Validation	Stock Action Dialog	Prevents removal of more stock than the current available quantity.
Barcode Decode	Quick Scan Module	Reads barcode content through the device camera.
AI Category Suggestion	Supabase Edge Function	Suggests category and unit values from product name information.
Graceful Degradation	Quick Scan Workflow	Allows manual review when lookup or AI assistance fails.

V. IMPLEMENTATION

A. Tools and Technologies

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

TABLE III - SOFTWARE REQUIREMENTS AND TECHNOLOGY STACK

Layer	Technology	Role in the System
Frontend	React 18 + TypeScript	Builds reusable pages, dialogs, protected routes, and typed logic.
Build Tool	Vite	Supports local development and production bundling.
Styling	Tailwind CSS + shadcn-ui	Provides utility styling and reusable UI primitives.
Routing	React Router	Maps navigation between public and protected pages.
State and Data	React Context and React Query	Supports auth state propagation and data-query coordination.
Backend	Supabase	Handles authentication, database storage, policies, and edge functions.
Barcode Scan	html5-qrcode	Decodes barcode content using the device camera.
Product Lookup	Open Food Facts API	Returns product-name data when barcode information exists.
AI Assistance	Supabase Edge Function / AI Gateway	Suggests category and unit values from product names.

B. Module Implementation Details

Authentication Module: The authentication module is built around an auth context provider and Supabase session services. It retrieves the current session on application load and subscribes to auth-state changes, ensuring the interface stays synchronized with login and logout actions.

Protected Route Module: The protected route checks loading state and authenticated user state. If the user is not logged in, the route redirects to the login page. If authentication is valid, the requested protected component is rendered normally.

Dashboard Module: The dashboard fetches inventory items from Supabase, stores them in local state, and derives summary metrics such as total items, low-stock count, total stock quantity, and category count. It also supports searching by item name or category.

Add Item Module: The add-item dialog collects item name, category, quantity, unit, buying price, selling price, and threshold values. Numeric fields are parsed before insertion so that the database receives values in the intended format.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

Stock Action Module: The stock action dialog supports both stock-in and stock-out. It inserts a stock transaction record and updates the latest quantity. For stock-out operations, it prevents removal beyond available quantity.

Quick Scan Module: The quick scan module coordinates camera access, barcode decoding, product lookup, AI suggestions, manual review, and final inventory insertion. It includes fallback paths so that the user can complete product entry even when external services fail.

Settings Module: The settings page retrieves and updates store profile information, including full name, shop name, phone number, and default low-stock threshold. The email field is displayed as account identity but is not treated as arbitrary editable profile data.

TABLE IV - MODULE-LEVEL FUNCTIONAL MAPPING

Module	Primary Inputs	Primary Outputs / Effects
Authentication	Email, password, full name	User session, signup confirmation, login state.
Protected Route	User and loading state	Spinner, redirect, or protected page rendering.
Dashboard	Inventory items, search text	Summary cards, low-stock alerts, item table.
Add Item	Product form fields	New inventory record inserted into database.
Stock Action	Selected item, quantity, notes, action type	Transaction row and updated quantity.
Quick Scan	Camera barcode value and user review fields	Suggested item details and final saved record.
Settings	Profile fields	Updated user and store information.

VI. USER INTERFACE AND SCREEN EXPLANATION

The user interface is designed around clarity and quick action. The dashboard is the central workspace and combines high-level metrics, low-stock alerts, search, and product rows. Dialogs are used for item creation, stock updates, and scan actions so that the user remains in the dashboard context while completing operational tasks.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

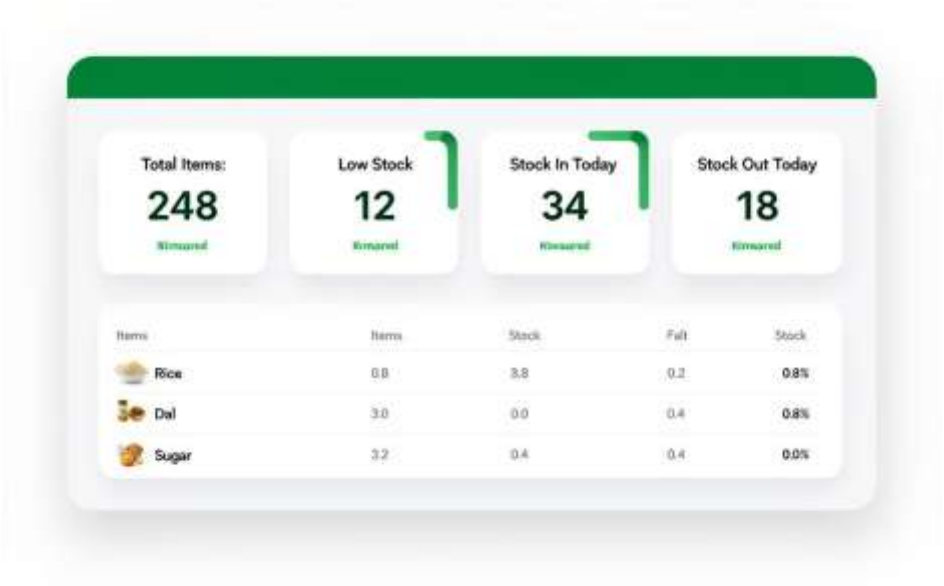

Fig. 6: Dashboard mock-up of Emphramark

A. Landing Page

The landing page introduces Emphramark as a lightweight inventory tool for kirana stores. It highlights barcode scanning, AI categorization, low-stock alerts, and a small-retail use case. This page communicates the intended audience and practical purpose of the system.

B. Login and Signup Pages

The login and signup pages use card-based forms with clear input fields and direct call-to-action buttons. Error messages are translated into user-friendly wording where possible, reducing confusion for non-technical users.

C. Dashboard Page

The dashboard displays summary cards for total items, low stock, total quantity, and category count. It provides a searchable table of items and row-level actions for stock changes. This design gives store owners both overview and action in the same screen.

D. Add Item and Stock Action Dialogs

The add-item dialog captures a complete product record in one interaction. The stock action dialog shows current stock and supports stock-in or stock-out with validation. This reduces the chance of accidental negative stock values.

E. Quick Scan Dialog

The quick scan dialog is the most interaction-rich part of the interface. It includes a scan area, lookup steps, AI categorization, and a review form. Despite the technical complexity behind the feature, the visible user flow remains simple: scan, review, and save.

VII. RESULTS AND DISCUSSION

The system improves daily inventory handling by converting informal stock tracking into structured digital records. Store owners can add items, update quantities, and identify low-stock

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

products without maintaining separate registers. The dashboard provides quick visibility, while the quick scan workflow reduces product-entry effort for packaged goods.

A key result is that the design balances automation and human control. Barcode scanning and AI suggestions assist the user, but final confirmation remains manual. This is suitable for kirana stores because product data from external sources may be incomplete, and local naming conventions may differ from public databases.

TABLE V - EXPECTED SYSTEM BENEFITS

Area	Manual Method Limitation	Emphramark Improvement
Stock Visibility	Stock is known only after manual checking.	Dashboard shows current item records and low-stock conditions.
Data Entry	Product entry is slow and repetitive.	Quick scan and AI suggestions reduce typing effort.
Security	Registers and spreadsheets are not user-isolated.	Supabase authentication and RLS protect user-specific data.
Stock Changes	Quantity changes may not be traceable.	Stock transactions preserve change events.
Usability	Complex systems can be unsuitable for small stores.	Focused dialogs and dashboard actions match daily workflows.

VIII. TESTING AND VALIDATION

Testing should cover user authentication, protected route behavior, dashboard loading, item creation, stock update validation, quick scan fallback, profile update, and database policy correctness. During thesis preparation, runtime installation was constrained by package-registry access in the environment, so evaluation distinguishes code-backed confirmation from full runtime execution. The system structure, route definitions, module logic, SQL schema, and edge function behavior support the intended features.

TABLE VI - FUNCTIONAL TEST SCENARIOS AND OBSERVATIONS

Test Area	Expected Behaviour	Observation
Signup Flow	Accept valid details and create account.	Implemented through Supabase auth and user metadata.
Login Flow	Allow valid user and reject invalid credentials.	Implemented with user-friendly error handling.

Protected Access	Redirect unauthenticated user to login.	Confirmed from protected route logic.
Dashboard Fetch	Load items for authenticated user.	Confirmed from Supabase query design.
Manual Item Add	Insert product into inventory_items.	Confirmed from add-item insert logic.
Stock-Out Validation	Block removal beyond current stock.	Confirmed from explicit validation guard.
Quick Scan Fallback	Allow manual review if lookup fails.	Confirmed from quick scan state flow.
Profile Update	Persist store settings.	Confirmed from settings update query.
RLS Policies	Allow access only to user-owned rows.	Confirmed from row-level security design.
Transaction Record	Record stock-in and stock-out changes.	Supported through stock_transactions table.

IX. SECURITY, VALIDATION AND ERROR HANDLING

Security begins with authentication, continues through protected routes, and is enforced at the database layer through row-level security policies. This layered approach is stronger than relying only on frontend route checks. Profiles, inventory items, and stock transactions are associated with the authenticated user, preventing one account from accessing another account's records.

Validation is present in several workflows. Signup checks password quality, login failures are translated into clearer messages, numeric fields are parsed before storage, stock-out operations are blocked if the requested quantity exceeds available stock, and barcode values are checked before product lookup. The quick scan workflow also demonstrates graceful degradation by allowing manual review if camera access, lookup, or AI assistance fails.

X. DEPLOYMENT AND MAINTENANCE

The project can be deployed using a standard frontend hosting workflow, while Supabase provides hosted authentication, database, and edge-function services. Deployment should verify environment variables, database migrations, authentication redirect configuration, edge-function secrets, and active row-level security policies. Post-deployment acceptance testing should include signup, login, item creation, stock-in, stock-out, low-stock behavior, profile update, and quick scan testing on a camera-enabled device.

Maintenance involves dependency updates, periodic review of Supabase policies, monitoring external API compatibility, and improving workflows based on real user feedback. Future versions

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 383-395

should also support item editing, transaction-history views, export tools, multilingual interface refinement, and improved analytics.

XI. CONCLUSION

Emphramark demonstrates a focused and relevant software engineering solution to a real small-retail problem. By targeting kirana stores and prioritizing practical stock visibility, it avoids the common mistake of building an oversized enterprise system for a modest use case. The project combines secure authentication, structured inventory storage, low-stock alert generation, stock transaction records, and convenience-oriented input workflows in a coherent architecture.

The selected technology stack is appropriate for the scale of the project. React, TypeScript, Tailwind CSS, and Supabase provide a modern foundation, while row-level security reflects strong attention to data protection. The quick scan and AI-assisted categorization path adds innovation without removing the manual control small retailers still need. Overall, the system is academically defensible, practically relevant, and extensible for future development.

XII. FUTURE SCOPE

- Complete item edit workflow and item history screen.
- Transaction history dashboard for stock movement auditing.
- Supplier management and purchase planning support.
- Advanced analytics for low-stock trends and category performance.
- Multilingual interface support for wider local adoption.
- Export and reporting features for store owners.
- Offline-friendly behavior for unstable network conditions.
- POS and billing integration in future versions.
- Role-based access for owners, helpers, and managers.
- Mobile app or progressive web app support.

REFERENCES

[1]React Documentation, “React – A JavaScript library for building user interfaces,” Meta Platforms, Inc

[2] Vite Documentation, “Vite – Next Generation Frontend Tooling,”

[3] Supabase Documentation, “Supabase: Open Source Firebase Alternative,”

[4] Supabase Database Guide, “Database Overview,”

[5] html5-qrcode GitHub Repository, “HTML5 QR Code Scanner,”

[6] Open Food Facts API Documentation, “Open Food Facts – Product Data API,”

[8] Inaganti, S., and Kumar, P., “Design and Development of Emphramark: A Web-Based Stock Management System for Kirana Stores in Local India,