Subhakamana Store Management System - Full System Report

Project: Subhakamana Store Management System

Business: Subhakamana Store, Nepali fashion and clothing retail

Date: 2026-07-17

Prepared for: GitHub repository, project demonstration, academic/project submission, and developer handoff

Current status: Prototype-ready system with production-readiness backend foundation and QA workflows

1. Executive Summary

The Subhakamana Store Management System is a prototype-ready retail platform for a Nepali clothing store that sells kurtas, sarees, dupattas, blouses, festival wear, and related fashion products.

The project now includes three user-facing operating areas:

- Admin ERP - a store management system for products, stock, orders, payments, delivery, returns, reports, and business controls.
- Cashier POS - a physical-store counter workflow optimized for quick billing and safe cashier operation.
- Customer Website - a branded e-commerce storefront for browsing products, cart, checkout, and customer trust information.

The system also includes production-readiness modules for PostgreSQL, Prisma, backend APIs, authentication, sessions, RBAC, Product and Inventory repositories, CI workflows, Playwright QA, local runner scripts, reports, manuals, UML diagrams, and a complete shareable prototype package.

This project is strong enough for demonstration, review, academic presentation, and developer onboarding. It is not yet a fully hosted production business application. The remaining work is mainly live PostgreSQL validation, backend deployment, server-side business enforcement, real payment/courier integration, object storage, monitoring, and backup configuration.

2. Project Purpose

The purpose of the project is to show how Subhakamana Store can move from manual or scattered store work into a structured retail management system.

The system is designed to help the store:

- Add and manage clothing products.
- Track stock by product, SKU, barcode, size, color, and variant.
- Sell products in-store through POS billing.
- Support online product browsing and checkout.
- Manage orders, payments, delivery, and returns.

- Keep product-focused stock movement records.
- Prepare for a real backend, database, hosting, and production deployment.

3. System Modules

3.1 Admin ERP

The Admin ERP is the main management system. It includes:

- Dashboard with daily attention items.
- Nepali calendar planning section.
- Product list and inventory view.
- Add Product workflow with SKU, barcode, category, pricing, stock, variants, supplier, images, and website settings.
- Stock receiving and purchase tracking.
- Customer orders and delivery by bill number.
- NCM delivery display and tracking status area.
- Payments center for cash, COD, eSewa, Khalti, bank transfer, and QR payment records.
- Returns, exchanges, refunds, stocktake, cashier shifts, and manager checks.
- Reports, numerical analysis, audit logs, stock movements, payment records, and CSV export.
- Settings and production setup notes.

3.2 Cashier POS

The POS workflow has been hardened for physical store use. It now includes:

- Secure role/PIN based demo login.
- Open-shift requirement before completing a sale.
- Barcode/SKU scanning in keyboard-mode scanner style.
- Exact variant display in product selection and cart.
- Live stock verification before checkout.
- Transaction-style checkout preflight.
- Rollback recovery if checkout fails.
- Idempotency key per completed sale.
- Duplicate-click protection.
- Bargained selling price per line.
- Separate item discount and bill discount.
- Role-based discount limits.
- Manager/owner PIN approval for large discounts.
- Member/customer phone lookup.

- Simple member creation from POS phone lookup.
- Multiple payment methods.
- Split payments.
- Digital payment verification requirement.
- Receipt generation and reprinting.
- Thermal receipt-style print view.
- Printer failure recovery by keeping receipt visible.
- Park and resume sale.
- Sale void with manager approval and stock restoration.
- Wrong-payment correction with manager approval.
- Return by bill and partial-return quantity limit.

3.3 Customer Website

The customer website is a branded fashion storefront. It includes:

- Subhakamana Store branding and warm Nepali fashion styling.
- Product cards with images, names, category, price, stock badges, and actions.
- Search, category filtering, and product browsing.
- Cart and checkout flow.
- Delivery or in-store pickup choice.
- Payment options placed during checkout.
- Trust sections for cash on delivery, exchange support, fast delivery, quality checking, and customer support.
- Footer with store links, social/contact links, and privacy policy.

3.4 Production Readiness Package

The project includes a production-readiness folder with:

- Target system architecture.
- PostgreSQL database schema.
- REST API contract.
- Security and RBAC plan.
- Testing and deployment plan.
- Migration plan from localStorage to backend APIs.
- Module 1 database implementation.
- Module 2 API foundation implementation.
- PostgreSQL validation and CI reports.
- POS cashier hardening implementation status.
- Remaining production work and validation status.

4. Backend and Database Foundation

4.1 Module 1 - PostgreSQL and Prisma

Module 1 provides the database foundation:

- Prisma schema.
- PostgreSQL-compatible schema design.
- Hardened seed script.
- Role, permission, and administrator seed planning.
- Environment example.
- Database verification script.

4.2 Module 2 - Backend API Foundation

Module 2 provides backend source and tests for:

- Authentication logic.
- Password hashing with bcrypt.
- Session creation, expiry, validation, and revocation.
- RBAC permissions and middleware.
- Product repository.
- Inventory repository.
- Product List API behavior.
- Add Product API behavior.
- Inventory View behavior.
- Receive Stock behavior.
- Duplicate SKU and barcode rejection.
- Inventory movement creation.
- PostgreSQL integration test gate.

4.3 Strict API Mode

The ERP has a strict API mode prepared for:

- Product List through backend API.
- Add Product through backend API.
- Inventory View through backend API.
- Receive Stock through backend API.

In API mode, these workflows must not silently fall back to localStorage. Backend failure is shown visibly to the operator.

5. QA and Validation

5.1 Static and Local Validation Completed

The following checks have been performed in the available environment:

- ERP JavaScript syntax validation passed.
- Customer website JavaScript syntax validation passed.
- Package link validation passed.
- Prototype zip contents verified.
- Backend non-listener tests passed: 24 passed, 0 failed, 0 skipped.
- Package copy matches the main ERP file.
- POS hardening markers verified inside the final zip.

5.2 Playwright QA

Playwright QA has been added with Chromium project configuration.

The tests cover:

- ERP opens and demo login reaches dashboard.
- Main ERP pages do not create document-level horizontal scroll.
- Customer website opens and includes the Privacy Policy link.
- Start page links to Admin ERP, Physical POS, and Customer Website.

Run locally:

```
pnpm install
./RUN_PLAYWRIGHT_QA.command
```

5.3 PostgreSQL CI

The GitHub Actions workflow `PostgreSQL Validation` is included. It is designed to:

- Start temporary PostgreSQL service databases.
- Validate CI secrets.
- Generate Prisma Client.
- Apply migrations to empty development and test databases.
- Run the hardened seed.
- Verify roles, permissions, mappings, and administrator.
- Run gated PostgreSQL integration tests.
- Validate authentication, sessions, Product creation, Inventory creation, duplicate SKU/barcode handling, Receive Stock, rollback behavior, concurrent updates, and persistence.

This workflow must be run in GitHub Actions or a real non-production database environment before production claims are made.

6. GitHub Readiness

The project is now GitHub-ready with:

- Root `README.md`.
- `package.json` for Playwright QA.
- `playwright.config.js`.
- Playwright test suite.
- GitHub Actions workflow for Playwright QA.
- GitHub Actions workflow for PostgreSQL Validation.
- Backend offline test runner.
- PostgreSQL local validation runner.
- GitHub update script.
- Full prototype package and zip.

Important scripts:

- `RUN_PLAYWRIGHT_QA.command`
- `RUN_BACKEND_OFFLINE_TESTS.command`
- `RUN_POSTGRESQL_VALIDATION_LOCAL.command`
- `outputs/UPDATE_GITHUB_WITH_LATEST_PROTOTYPE_CHANGES.command`

7. Current Limitations

The current system is not yet fully production-live. Important remaining limitations:

- Demo mode still uses browser localStorage.
- Real backend deployment is not complete.
- PostgreSQL CI must still be run in GitHub Actions.
- Payment gateway verification is simulated.
- NCM delivery integration is represented as a demo/status display.
- Printer integration uses browser print behavior, not a dedicated print agent.
- Server-side POS checkout, void, discount, return, and payment correction enforcement must still be implemented.
- Image upload/storage is local/demo only.
- Production monitoring and backups are not active.

8. Remaining Production Work

Highest-priority remaining work:

- 1. Push the latest GitHub-ready project.
- 2. Run PostgreSQL Validation in GitHub Actions.
- 3. Run Playwright QA in GitHub Actions or on the user Mac.
- 4. Confirm ERP strict API mode against a real backend server.
- 5. Implement server-side POS checkout.
- 6. Implement website checkout and online order APIs.
- 7. Implement payment verification APIs.
- 8. Implement delivery/courier production API integration.
- 9. Implement server-side returns, exchanges, refunds, voids, and payment corrections.
- 10. Add object/image storage.
- 11. Add production hosting, secrets, SSL, monitoring, and backups.

9. Project Standing

Estimated standing:

- Prototype/demo package: 95% complete.
- Local operator demonstration: 90% complete.
- Backend foundation package: 65% prepared.
- Fully hosted production business system: about 50% complete.

The remaining work is mostly production infrastructure, live validation, and server-side enforcement rather than UI design.

10. Final Conclusion

Subhakamana Store Management System is now a polished, GitHub-ready prototype package with a strong ERP, POS, customer website, reports, documentation, backend foundation, QA setup, and production migration path.

The project is ready to share with other developers for review and continued production conversion. The next decisive step is to push the latest package to GitHub and run the included PostgreSQL Validation and Playwright QA workflows.