Flux

Product Requirements Document (PRD)

Baseline Release: Phase 0 (Walking Skeleton) + Phase 1 (Trimmed MVP)

Version: 1.0
Date: 31 January 2026

This PRD defines *who Flux is for*, *what problems it solves*, and *what we will build first* (Phase 0 and Phase 1). It is aligned with the Project Charter, Trimmed MVP, Baseline SRS, Core ERD, Latest Scope, Roadmap, and Requirements Catalogue.

Document control

Item	Value
Product	Flux - Offline-first POS & Inventory PWA
Baseline release	Phase 0 + Phase 1
Primary users	Cashier, Store Manager, Super Admin
Backend	Supabase (PostgreSQL) with Row Level Security (RLS)
Offline storage	IndexedDB (client-side queue + cached masters)

1. Executive summary

Flux is an **offline-first POS & Inventory system** built as an **installable Progressive Web App (PWA)**. It targets small-to-medium retail businesses that need fast billing, reliable stock control, and multi-branch visibility, even when the internet is unstable. The baseline release is split into two parts: **Phase 0 (Walking Skeleton)** to prove the architecture end-to-end, and **Phase 1 (Trimmed MVP)** as the first sellable release.

Key promise of the product

- Sell items **while offline** (no internet) and **do not lose transactions** if the browser closes.
- When connectivity returns, **sync automatically** to Supabase and keep stock correct.
- Enforce **branch isolation** using Supabase **Row Level Security (RLS)** scoped by `location_id`.
- Support the **minimum real-world retail flows** first (Sales, GRN, Transfers, Batches/Expiry, Basic reports).

2. Problem & opportunity

Retail shops often run on paper, spreadsheets, or disconnected apps. This creates issues such as:

- Slow billing and manual mistakes during peak hours.
- Stock-outs or overstock because stock is not visible per branch.
- No single source of truth for sales and stock movements.
- Internet outages stop the counter work, which is unacceptable for retail.

Flux solves this by using a PWA that can run like a native app, keep working offline, and sync to a central database when online.

3. Users, roles, and what they need

Flux baseline supports three core roles:

Role	Main goals	Typical tasks in Phase 0/1
Cashier	Fast billing with low errors	Create sales, scan/search items, apply simple discounts, take paymen
Store Manager	Keep stock accurate and controlled	Maintain items/suppliers/customers, create GRNs, do stock transfers,
Super Admin	Configure and oversee multiple branches	Create locations, create users/roles, cross-branch visibility and control

Note: Accounting-heavy roles (e.g., Accountant) and finance workflows are **deferred** to later phases.

4. Goals and success criteria

### 4.1 Product goals (Phase 0 + Phase 1)

- Offline-first POS: cashier can complete a sale offline and keep a durable queue until sync.
- Accurate stock: sales/GRN/transfers update stock via a ledger (stock moves) + snapshot (stock balances).
- Branch isolation: every master and transaction row is scoped by `location_id` and protected by RLS.
- Usable MVP: enough screens and controls for a real shop to run daily operations.

### 4.2 Measurable success criteria (baseline)

- PWA installability: app installs and launches in standalone mode; loads to core pages offline.
- Offline durability: create a sale offline, close/reopen browser, sale is still pending and can sync later.
- Sync correctness: after reconnection, pending items sync in FIFO order; no duplicates on retry (UUID-based).
- Security: a user from Location A cannot read/write Location B data via UI or API (RLS tests).
- Performance: target Lighthouse >= 90 for PWA/performance/accessibility on a representative profile.

5. Scope and priorities

This PRD uses a simple priority model: **Phase 0** proves the technology end-to-end; **Phase 1** is the first sellable release; **Post-MVP** items are intentionally pushed out to avoid early complexity.

Capability	Phase 0 (Walking Skeleton)	Phase 1 (Trimmed MVP)	Later (Phase 2+)
PWA installability + service worker	MUST	Improve + harden	Advanced caching, notifications
Supabase Auth + RLS	MUST	Extend roles & policies	MFA, deep audit UI
Core masters (minimum)	Category, Unit, Item (dummy)	Locations, Categories, Units, Items, Suppliers, Customers	Manufacturers, extra masters (banks, le
POS Sales	Sell 1 item, cash only; offline queue + sync	Full POS: barcode scan/search, qty, discounts, cash/card/credit, receipt print	Temp invoices, post-adjustments, prom
Inventory model	Stock moves + stock balances (basic)	Batch/expiry (lots), stock adjustments, negative stock policy	Cycle counts, reconciliation, serial track
Purchasing	N/A	PO + GRN (draft->post), batch/expiry on GRN	GRN post adjustments/cancellation, pu
Multi-location transfers	N/A	Transfer out + receive (two-step)	Advanced transit and logistics
Reporting	Simple sales list	Sales summary, stock balance/valuation, expiry list, outstanding credit	Full reporting suite + accounting statem

### 5.1 Explicit out-of-scope for baseline (Phase 2+)

- Barcode printing (scanning/search is allowed; printing is deferred).
- Inventory reconciliation / cycle counts / stock takes.
- Full accounting and financial statements (GL, Trial Balance, P&L;, Balance Sheet).
- Invoice post-adjustments, complex promotions/loyalty.
- GRN post-adjustments and posted GRN cancellation.
- Cheque modules, bank reconciliation, period closing (day/shift/month/year).
- Manufacturing/BOM and production flows.
- Bulk import/export and advanced admin tools beyond essentials.

6. What we are building (baseline epics)

The baseline release is organised into these product epics:

- **E1 - PWA foundation**: installable app, cached shell, offline status indicators.
- **E2 - Authentication & security**: Supabase login, roles, and RLS by `location_id`.
- **E3 - Master data**: locations, categories, units, items, suppliers, customers (with credit fields).
- **E4 - Inventory core**: stock moves ledger + stock balances snapshot; batch/expiry lots; negative stock policy.
- **E5 - POS sales**: draft->post, barcode scan/search, discounts, payment types, receipt print, offline queue + sync.
- **E6 - Purchasing**: PO + GRN (draft->post), stock-in on posting, batch/expiry capture.
- **E7 - Stock transfers**: two-step transfer out + receive, with batch-aware movement.
- **E8 - Basic reporting**: sales summary, stock balances/valuation, expiry report, credit summary.

7. Key workflows (simple real-life scenarios)

### 7.1 Offline sale at the counter

- Cashier opens POS and searches/scans an item.
- Adds quantity, applies a simple discount if needed, selects payment (cash/card/credit).
- If offline, the sale is saved to IndexedDB as **Pending Sync** and a receipt can still be printed.
- When internet returns, the app syncs the sale and stock is reduced on the server.

### 7.2 Receiving goods (GRN) with expiry tracking

- Store Manager creates a GRN, selects a supplier, adds item lines and costs.
- For batch-tracked items, enters batch number and expiry date (required).
- Posts the GRN: stock-in moves are created and stock balances increase.

### 7.3 Transfer stock between branches

- Source branch creates a Transfer Out, selects destination, and posts it.
- Destination branch receives and posts the Transfer In (can support partial receiving if designed).
- Ledger records the moves; balances update per location.

8. Non-functional requirements (baseline)

These requirements are product-defining (they are not optional).

Area	**Baseline requirements**
Offline reliability	Core workflows must function without internet. Offline-created records must be durable and retry-safe (no d
Security	RLS must be enabled and correct for all exposed tables. UI must follow role permissions; no cross-branch l
Performance	Fast item search from local cache; PWA Lighthouse target >= 90 on a representative profile.
Usability	Touch-friendly POS. Clear Online/Offline and Unsynced Count indicators. Simple error messages and retry
Maintainability	TypeScript end-to-end types; consistent API contract; incremental schema/SRS updates per release.

9. Data and architecture constraints (must-follow rules)

Flux baseline is built around these hard constraints:

- **UUID primary keys** for all tables (offline-safe identity).
- **location_id on master and transaction tables** to enforce branch isolation via RLS.
- **Audit fields** (`created_at`, `updated_at`, `created_by`) on all tables.
- Inventory uses **stock_moves (ledger)** + **stock_balances (snapshot)**; balances must have uniqueness constraints to avoid duplicates.
- **Offline sync flags are client-only**: the database does not store 'synced' columns; IndexedDB tracks pending/synced/failed.
- Document numbers (invoice/grn/transfer) must be **unique per location**.

10. Major risks and how we reduce them

- **Offline sync duplication**: use UUID identities, server-side unique constraints, and idempotent sync logic.
- **RLS mistakes**: define policies early, test with multi-user and multi-location datasets.
- **Service worker update issues**: version the service worker and use a safe update flow to prevent blank screens.
- **Scope creep**: keep Phase 1 limited to the Trimmed MVP; move extra screens to Phase 2+.
- **Printing differences**: test early with common thermal printers and provide simple templates.

11. Source references

This PRD is aligned with the following project source documents:

- Flux - Project Charter New (PDF)
- MVP Definition for Flux (New) (PDF)
- Flux Baseline SRS (PDF)
- Flux Baseline SRS - Requirements Catalogue (PDF)
- Database Architecture Design (Core ERD) (PDF)
- Flux Latest Scope (UI/Fields) (PDF)
- Roadmap for Flux Development (PDF)
- Documentations Required in Each Release (DOCX)

End of document.