Flux — Baseline Software Requirements Specification (SRS)

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

Primary Source Documents: Project Charter , MVP Definition , Database Architecture (Core ERD) , Flux Latest Scope

---

1. Scope & MVP Boundaries

Objective

Flux (Phase 0+1) SHALL deliver a **sellable, offline-first POS & Inventory system** for multi-branch retail, built as an **installable PWA** with **secure branch isolation** and **reliable offline sales + stock operations**, syncing automatically when internet returns.

In-Scope (Phase 0 + Phase 1 modules)

Based on the **Trimmed MVP feature filter**, the following modules are **IN SCOPE**:

1. **PWA Foundation (Phase 0)**
   - Installable PWA, service worker caching, offline-ready shell.
2. **Authentication & Access Control**
   - Supabase Auth-based login and role-based access.
3. **Core Masters**
   - Categories, Sub-categories, Items, Customers, Suppliers, Units, Stock Locations (per branch).
4. **Inventory (with Batches/Expiry)**
   - Batch-tracked and non-batch items; expiry dates on batches; stock balance views.
5. **POS / Sales (Offline-capable)**
   - Create sales (draft → post), payments (cash/card/credit), receipt print, offline queue + sync.
6. **Purchasing — GRN (Goods Received Note)**
   - Record supplier GRNs, capture batches/expiry, post to update stock.
7. **Stock Transfers (Inter-branch / inter-location)**
   - Transfer out from source, receive at destination (2-step).
8. **Basic Reporting (Trimmed)**
   - Essential operational reports (e.g., sales totals, stock balances, expiry list, outstanding credit summary).

Out-of-Scope (Explicit Exclusions — Deferred to Phase 2+)

The following are **explicitly excluded** from this SRS baseline (even if present in older full-scope menus), per the **Trimmed MVP filter**:

- **Barcode printing** (barcode scanning/search is allowed; printing is deferred).
- **Serial number tracking**.
- **Cycle counts / stock takes**.
- **Financial statements / full accounting module** (GL, P&L, Balance Sheet).
- **GRN post-adjustments / posted GRN cancellation**.
- **Invoice post-adjustments / complex sales corrections**.
- **Advanced promotions/loyalty/multi-offer pricing engine**.
- **Manufacturing / production / BOM**.
- **Bulk import/export and "advanced admin tools" beyond essentials**.

---

2. Actors, Roles & Permissions

Define Actors

Flux SHALL support the following actors:

1. **Super Admin**
   - Full system access including cross-branch configuration and visibility.
2. **Store Manager**
   - Branch/location operations: purchasing (GRN), transfers, inventory operations, and user management within the branch.
3. **Cashier**
   - POS sales execution and customer lookup; restricted from sensitive inventory actions.

Permissions Matrix (Baseline)

Flux SHALL enforce permissions using **both**:

1. **Role-based application rules**, and
2. **Supabase Row-Level Security (RLS)** for branch/location isolation.

Capability / Module	Super Admin	Store Manager	Cashier
System configuration (branches/locations/settings)	✅	❌	❌
User management (create/disable/reset)	✅ (all)	✅ (branch only)	❌
Masters CRUD (items/customers/suppliers/units/categories)	✅ (all)	✅ (branch only)	⚠️ Read-only (except customer search/select)
POS sales: draft/post	✅	✅	✅
POS returns (simple)	✅	✅	✅ (policy-controlled)
Purchase GRN: draft/post	✅	✅	❌
Stock transfers: create/receive	✅	✅	❌
Stock adjustments	✅	✅ (limited)	❌
Reporting (basic)	✅ (all)	✅ (branch only)	✅ (sales-only)

RLS Constraint (Branch Isolation): All master and transaction data SHALL be scoped by `location_id` (or equivalent) and enforced by Supabase RLS so users only see permitted branch/location rows.

---

3. Offline Logic & Sync Rules (Critical Section)

Offline First

Flux SHALL operate as **offline-first**, meaning:

- Users SHALL be able to **continue POS sales** when offline.
- The app SHALL not lose work if the browser closes while offline (data persistence required).
- When internet returns, Flux SHALL sync automatically.

Data Storage

1. **Local Storage (Client) — IndexedDB**
   - IndexedDB SHALL store:
     - Product catalog (masters needed for offline selling)
     - Current user session context (limited, non-sensitive)
     - Unsynced transactional records (sales/GRN/transfers) with metadata needed for syncing.

2. **Remote Storage (Server) — Supabase (PostgreSQL)**
    - Supabase/PostgreSQL SHALL be the long-term system of record for synced data.

Sync Strategy (Client Queue + Last-Write-Wins)

Client-Side Queue

Flux SHALL implement a client-side **sync queue** where each queued record includes at minimum:

- local_id (UUID)
- entity_type (e.g., sales_invoice, grn, stock_transfer)
- payload (the record + child lines)
- location_id
- created_at, updated_at (client timestamps)
- sync_status (pending, synced, failed)
- retry_count, last_error

Offline Sync Flags MUST live on the client

- The system SHALL NOT store synced flags or "offline sync state" in the database schema.
- Sync status SHALL exist **only on the client** (IndexedDB), per the database architecture decision.

Sync Process

Flux SHALL sync using this baseline flow:

1. Detect connectivity.
2. Pull server deltas for masters/needed references (as applicable).
3. Push queued transactions in FIFO order (oldest first).
4. On successful server commit, mark local record as synced.
5. On error, mark failed, store error, retry with backoff.

Conflict Policy (Initial)

- Flux SHALL use **Last-Write-Wins (LWW)** as the initial conflict policy, using updated_at timestamps.

Sync Indicators (UI Requirements)

Flux SHALL provide visible indicators:

- **Online/Offline** status indicator.
- **Unsynced Count** badge (number of pending items).
- Per-transaction status: Draft, Pending Sync, Synced, Failed.

---

4. Core Workflows (Functional Requirements)

4.1 POS Sale (The "Hot" Path)

Goal: Allow fast selling (including offline), correct stock movement, and receipt printing.

Flow (Draft → Post):

1. Cashier SHALL start a **new sale** (creates a local draft invoice with UUID).
2. Cashier MAY select a **Customer**:
    - Optional for cash/card "walk-in" sales

   - Mandatory for **credit** sales.
3. Cashier SHALL **scan or search item** (barcode/search) and add line(s).
   - Barcodes SHALL be unique per location to avoid duplicates during sync.
4. If the item is **batch-tracked**, Cashier SHALL select a batch (expiry-aware).
5. Cashier SHALL be able to edit:
   - Quantity (supports decimals where item/unit allows)
   - Discount (line or invoice-level, per implementation)
6. Cashier SHALL choose payment type:
   - Cash / Card / Credit.
7. On **Post**:
   - System SHALL create a sales invoice + lines + payments record (queued offline if needed).
   - System SHALL generate inventory movement (stock-out) and update stock snapshots upon sync.
8. System SHALL print a receipt (browser print / connected printer workflow).

Validation — Negative Stock Policy

- System SHALL prevent stock from silently going negative.
- System SHALL support a configurable policy:
  - **WARN** (allow post but warn user), or
  - **BLOCK** (do not allow post if it causes negative stock).

    (Policy stored as a branch/location setting; default SHOULD be WARN for early rollout.)

4.2 Purchasing (GRN)

Goal: Receive stock from suppliers with batch/expiry capture and posting to inventory.

Flow (Draft → Post):

1. Store Manager SHALL create a **GRN** and select a Supplier.
2. Add item lines with quantity and unit cost.
3. For batch-tracked items, user SHALL assign:
   - Batch number (lot)
   - Expiry date (required for batch-tracked items).
4. GRN MAY remain **Draft** and be edited/deleted while draft.
5. On **Post**:
   - System SHALL add stock (stock-in moves) and update stock balances.
6. Posted GRNs SHALL NOT be cancelled or post-adjusted in this baseline (deferred).

4.3 Stock Transfer

Goal: Move stock between locations with clear source/destination responsibility.

Flow (Two-step):

1. Source Location (Store Manager) SHALL create a **Transfer Out**:
   - Select destination location
   - Add item lines (batch-aware if applicable).
2. System SHALL mark transfer status as "Sent/Out" on posting (implementation status names may vary).
3. Destination Location SHALL **Receive**:
   - Confirm received quantities (full/partial per rules)
   - Post receive to finalize stock-in at destination.

Inventory Effect

- System SHALL record stock movements for transfers and maintain batch traceability where `from_batch_id` / `to_batch_id` is used.

4.4 Masters Management

Flux SHALL support CRUD for essential masters. At minimum:

Category

- Fields SHALL include: Category No, Category Name, Short Code.

Sub Category

- Fields SHALL include: Sub Category No, Sub Category Name, Short Code, Category Name (link).

Supplier

- Fields SHALL include: Supplier No, Supplier Name, Contact Person, Address, Telephone No, Fax No, Mobile No, Email, Control Account.

Customer

- Fields SHALL include: Customer No, Customer Name, Contact Person, Address, Telephone No, Fax No, Mobile No, Email, Control Account.
- Customer credit terms (Credit Days, Credit Limit) SHALL be supported for Trimmed MVP credit sales.

Items (Core rules)

- Items SHALL include per-location identity and barcode uniqueness to prevent duplicates during sync.
- Items SHALL support batch-tracked or non-batch behavior (see Section 5).

---

5. Data Rules & Integrity

Inventory Model (Ledger vs Snapshot)

Flux SHALL use a dual model:

1. **Stock Moves (Ledger)** — immutable movement records (in/out/adjust/transfer)
2. **Stock Balances (Snapshot)** — current on-hand quantity by item/batch/location
   This structure SHALL be the basis for reporting and sync-safe recalculation.

Batch/Expiry (Lots)

- Items SHALL be either:
  - **Non-Batch**: stock tracked without lots
  - **Batch-Tracked**: stock tracked by batch with expiry date.
- For batch-tracked items:
  - GRN posting SHALL require expiry date.
  - POS SHALL require selecting a batch (or system-assisted batch selection).

Unit Handling (Decimals)

- Quantity fields SHALL support decimals (e.g., 1.5 kg) using numeric precision appropriate for retail.

Uniqueness & Anti-Duplicate Rules (Sync Safety)

Flux SHALL enforce uniqueness to prevent sync duplicates:

- **Barcode uniqueness per location**.
- **Invoice number uniqueness per location** (or equivalent invoice identity).

- Where server-generated numbers are used (invoice/GRN/transfer), the system SHOULD prefer server-assigned sequences after sync, while keeping UUIDs as the true stable identity.

---

6. Non-Functional Requirements

Performance

- Flux SHALL target **Lighthouse score > 90** for the PWA baseline.
- Offline item search SHOULD feel instant; target **<100ms** search response on a typical branch catalog (implementation benchmark).

Security

- Flux SHALL use **Supabase Auth** for authentication.
- Flux SHALL enforce **Row-Level Security (RLS)** for branch/location isolation using `location_id` scoping.

Reliability

- Offline-created transactions SHALL persist in IndexedDB and SHALL NOT be lost on refresh/tab close/browser restart.
- Sync SHALL be retry-safe and shall surface failures clearly to the user (failed queue items, reason, and next retry).
