2XG

Enterprise Resource Planning

System Architecture Flow Document

Comprehensive architecture blueprint covering current modules, new enhancements, and implementation roadmap for the 2XG ERP system.

Version 2.0 | February 2026

Prepared for: 2XG Growth

Stack: React + Express + Supabase (PostgreSQL)

Deployment: Coolify (OVH Self-Hosted)

Table of Contents

1. Executive Summary

The 2XG ERP system is a production-deployed monorepo application serving as the central business management platform for 2XG Growth. This document outlines the architecture for 6 new/enhanced modules that extend the current system to support **serial-level inventory tracking**, **advanced warehouse bin management**, **goods receipt with discrepancy handling**, **enhanced sales CRM with KYC**, **scheduled stock audits with auto-correction**, and an **Issue Bin lifecycle workflow**.

> **Key Principle:** All 28 existing APIs and 48 existing pages remain untouched. New modules are added alongside existing ones, following the established Route → Controller → Service → Supabase pattern.

2. Current System Overview

2.1 Existing Modules (28 API Routes, 48 Pages, 55 Components)

Core Business Modules

- Items / Inventory Management
- Purchase Orders + Bills
- Sales Orders + Invoices
- Payments Made / Received
- Vendor Credits
- Expenses + Categories
- Transfer Orders
- Delivery Challans

Supporting Modules

- Customer Management
- Vendor Management
- Bin Locations (basic)
- Stock Count (basic)
- POS Sessions
- Brands & Manufacturers
- Reports & AI Insights
- CRM / CARE / Logistics

2.2 Tech Stack

LAYER	TECHNOLOGY	DETAILS
Frontend	React 18 + Vite + TypeScript	Tailwind CSS, React Router, Axios, Lucide Icons
Backend	Express.js + TypeScript	28 route files, JWT auth, Helmet, Morgan, CORS
Database	PostgreSQL (Supabase)	Self-hosted, PostgREST gateway, Service Role Key (bypasses RLS)
Auth	Custom JWT	bcrypt passwords, 7-day expiry, web + mobile user support
Deployment	Coolify (OVH Server)	Docker-based PaaS, auto-deploy on push to main
URLs	Production	Frontend: erp.2xg.in | Backend: api.erp.2xg.in

2.3 Current Data Flow

```
User Action (React Component)
    ↓
Frontend Service (axios + JWT header)
    ↓
Backend Route → Auth Middleware → Controller → Service
    ↓
Supabase Client (PostgREST API)
    ↓
PostgreSQL Database
```

```
    ↓
Response: { success: boolean, data?: T, error?: string }
```

3. Gap Analysis: Current vs Required

FEATURE	STATUS	CURRENT STATE	REQUIRED ENHANCEMENT
Items / Inventory	EXISTS	Quantity-based stock tracking (current_stock)	Per-unit serial tracking, assembly state
Bin Locations	ENHANCE	Basic bin CRUD with location_code, capacity	Bin categories (Electric/Gear/Kids/Issue/Box), state (Assembled/Saleable)
Transfer Orders	ENHANCE	Manual transfer between locations	Auto-generation from stock count discrepancy
Stock Count	ENHANCE	Basic counting interface	Scheduled (Mon/Wed/Fri), physical vs accounting, auto-correction
Purchase Orders	ENHANCE	Full PO CRUD with GST	Goods receipt linking, partial receipt status, shortage handling
Delivery Challans	ENHANCE	Manual DC creation	Auto-DC from reverse pick-up / self pick-up sales
Customers	ENHANCE	Basic CRUD with name, email, GSTIN	Full KYC, B2B/B2C type, pin code, alt mobile
Sales Orders	ENHANCE	Standard sales order flow	Exchange flow, accessory list, delivery type, salesperson assignment
Vendor Credits	ENHANCE	Manual credit note creation	Auto-trigger from goods receipt discrepancy
Serial Tracking	NEW	Does not exist	Per-unit serial numbers, full lifecycle history, auto-yield
Goods Receipt (GRN)	NEW	Does not exist	Receipt against PO, inspection, discrepancy (shortage/damage/mismatch)
Issue Bin Workflow	NEW	Does not exist	Damaged/returned item lifecycle: repair, replace, scrap, return-to-vendor

4. Module 1: Enhanced Warehouse & Bin Management

> **Type:** Enhancement of existing module | **Affects:** bin_locations table, binLocations.service.ts, BinLocationsPage.tsx

4.1 Warehouse Hierarchy

```
WAREHOUSE HIERARCHY

Location (Godown / Store)
  └─ Bin Category
       └─ Electric Bin     ← E-bikes, electric scooters
       └─ Gear Bin         ← Geared cycles/bikes
       └─ Non-Gear Bin     ← Single-speed cycles
       └─ Kids Bin         ← Children's cycles
       └─ Issue Bin        ← Damaged / returned items
       ├─ Box Bin          ← Inbound unprocessed stock

  Each Bin tracks:
    • Assembly State: Assembled / Unassembled
    • Saleability:    Saleable / Non-Saleable
    • Serial Items:   Linked serial numbers
```

4.2 Database Changes (ALTER bin_locations)

COLUMN	TYPE	VALUES	PURPOSE
location_type	TEXT	godown, store	Physical location classification
bin_category	TEXT	electric, gear, non_gear, kids, issue, box	Product category this bin holds
assembly_state	TEXT	assembled, unassembled, na	Assembly tracking for bin contents
saleability	TEXT	saleable, non_saleable	Whether items in this bin can be sold

4.3 Internal Transfer Logic

```
STOCK MOVEMENT THROUGH BINS

  Goods Receipt (Inbound)
      ↓
  Box Bin (Unassembled, Non-Saleable)
      ↓ [Assembly Process]
  Category Bin (Electric/Gear/Non-Gear/Kids)
  (Assembled, Saleable)              ← Auto-Yield triggers here
      ↓
  Available for Sale
      ↓ [If returned/damaged]
  Issue Bin (Non-Saleable)
      ↓
  Resolution: Repair / Replace / Scrap / Return to Vendor
```

5. Module 2: Serial Tracking & Auto-Yield

Type: New module | **New Tables:** serial_numbers, serial_history | **New API:** /api/serial-tracking

5.1 Serial Lifecycle Flow

```
SERIAL NUMBER LIFECYCLE

  Item (SKU: GEAR-001)
    ├─ Serial #030
    │     └─ Status: active / issued / returned / replaced
    │     └─ Current Bin: Gear Bin (Store)
    │     ├─ History Timeline:
    │          └─ 2026-01-10  Inbound (I/B) → Box Bin
    │          └─ 2026-01-12  Assembled     → Gear Bin (Store)
    │          └─ 2026-01-20  SOLD          → Invoice #INV-00045
    │          └─ 2026-02-01  RETURNED      → Issue Bin
    │          └─ 2026-02-02  Credit Note   #CN-0012 issued
    │          ├─ 2026-02-05  Replacement   #030-R → Gear Bin
    │
    ├─ Serial #028
          ├─ Status: active (in stock, saleable)
```

5.2 New Tables

serial_numbers

COLUMN	TYPE
id	UUID (PK)
item_id	UUID → items
serial_no	TEXT UNIQUE
status	TEXT (active/issued/returned/replaced/scrapped)
current_bin_id	UUID → bin_locations
assembly_state	TEXT
saleability	TEXT
created_at	TIMESTAMP

serial_history

COLUMN	TYPE
id	UUID (PK)
serial_id	UUID → serial_numbers
event_type	TEXT
from_bin_id	UUID (nullable)
to_bin_id	UUID (nullable)
reference_type	TEXT (PO/SO/GRN/CN/TO)
reference_id	UUID
notes	TEXT
created_at	TIMESTAMP

5.3 Auto-Yield Logic

Auto-Yield Rule: When a serial item is moved from **Box Bin** to any **Category Bin** (Electric/Gear/Non-Gear/Kids), the system automatically sets `assembly_state = 'assembled'` and `saleability = 'saleable'`. This represents the physical assembly process being completed.

5.4 Event Types

EVENT	TRIGGER	EFFECT ON SERIAL
`inbound`	Goods Receipt completed	Serial created, placed in Box Bin
`assembled`	Transfer from Box Bin → Category Bin	Auto-yield: Assembled + Saleable
`transferred`	Transfer Order between bins	Updates current_bin_id
`sold`	Invoice/Sales Order completed	Status = sold, linked to invoice
`returned`	Customer return	Moves to Issue Bin
`issued_cn`	Credit Note generated	CN reference linked
`replaced`	Replacement unit received	New serial created, old marked replaced
`scrapped`	Item write-off	Status = scrapped, Non-Saleable

6. Module 3: Sales CRM + KYC Enhancement

> **Type:** Enhancement of existing modules | **Affects:** customers table, sales_orders table, customers.service.ts, sales-orders.service.ts

6.1 Sales CRM KYC Flow

```
SALES CRM KYC CAPTURE (at point of sale)

  Customer Details
    └─ Name, Phone, Email, Pin Code
    └─ Customer Type: B2B / B2C
    |      └─ B2B → GSTIN required, GST Tax Invoice
    |      ├─ B2C → GSTIN optional, Retail Invoice
    └─ Alt Mobile No
    └─ Accessory List (items sold with main product)
    ├─ Delivery Details:
          └─ Delivery Date
          └─ Self Pick-up?
          |      ├─ YES → Auto-generate Delivery Challan (D.C)
          └─ Reverse Pick-up?
          |      ├─ YES → Auto-generate D.C + assign delivery driver
          ├─ Exchange?
                ├─ YES → Salesperson enters exchange value
                       → Old item → Issue Bin
                       → Exchange value deducted from invoice
```

6.2 Database Changes

ALTER customers table

COLUMN	TYPE
customer_type	TEXT (b2b/b2c)
pin_code	TEXT
alt_phone	TEXT
kyc_completed	BOOLEAN

ALTER sales_orders table

COLUMN	TYPE
delivery_type	TEXT (self_pickup/delivery/reverse_pickup)
is_exchange	BOOLEAN
exchange_value	DECIMAL
exchange_item_desc	TEXT
salesperson_id	UUID
auto_dc_generated	BOOLEAN

NEW TABLE: sale_accessories

COLUMN	TYPE	PURPOSE
id	UUID (PK)	Primary key
sales_order_id	UUID → sales_orders	Parent sales order
item_id	UUID → items	Accessory item
item_name	TEXT	Denormalized name

quantity	INTEGER	Qty sold with main product

6.3 Auto-Trigger Logic

When delivery_type = 'reverse_pickup' or 'self_pickup':
→ sales-orders.service.ts auto-calls deliveryChallansService.create() and links the DC to the Sales Order.

When is_exchange = true:
→ Exchange value is deducted from total_amount on the generated invoice.
→ The exchanged old item is recorded into the Issue Bin with source_type = 'sales_order'.

7. Module 4: Stock Ordering & Discrepancy Flow (GRN)

> **Type:** New module | **New Tables:** goods_receipts, goods_receipt_items | **New API:** /api/goods-receipts

7.1 Goods Receipt Flow

```
STOCK ORDERING → GOODS RECEIPT → DISCREPANCY HANDLING

  PO Created (15 pcs ordered from Vendor)
      ↓
  Expected Lead Time: 10 days
      ↓
  Goods Receipt (GRN) created against PO
      └─ Received: 10 pcs (physically arrived)
      └─ Shortage: 5 pcs  (not delivered)
      └─ Damaged:  1 pc   (arrived damaged)
      ├─ Mismatch: 3 pcs  (wrong item/spec)

  SYSTEM AUTO-ACTIONS:
      └─ 6 Good pcs     → Box Bin (Inbound) + create serial numbers
      └─ 1 Damaged pc   → Issue Bin + serial marked 'issued'
      └─ 3 Mismatch pcs → Issue Bin + serial marked 'issued'
      └─ Auto Credit Note generated for 4 pcs (damage + mismatch)
      └─ PO Status updated to 'partial_received'
      ├─ Option: Re-order shortage OR adjust PO

  PO STATUS FLOW:
  draft → sent → partial_received → received → closed
                        ↓
                  discrepancy_flagged
```

7.2 New Tables

goods_receipts

COLUMN	TYPE
id	UUID (PK)
purchase_order_id	UUID → purchase_orders
grn_number	TEXT (auto-generated)
receipt_date	DATE
received_by	UUID → users
status	TEXT
notes	TEXT
created_at	TIMESTAMP

goods_receipt_items

COLUMN	TYPE
id	UUID (PK)
goods_receipt_id	UUID → goods_receipts
po_item_id	UUID → purchase_order_items
item_id	UUID → items
ordered_qty	INTEGER
received_qty	INTEGER
damaged_qty	INTEGER
mismatch_qty	INTEGER
shortage_qty	INTEGER (auto-calc)
destination_bin_id	UUID → bin_locations
issue_bin_id	UUID → bin_locations

7.3 Completion Trigger Logic

```
ON GOODS RECEIPT COMPLETION:

  1. Good items → items.current_stock += received_qty
  2. Good items → Create serial_numbers (status: active, bin: Box Bin)
  3. Good items → Create serial_history entries (event: inbound)
  4. Damaged/Mismatch → Move to Issue Bin (create issue_bin_entries)
  5. IF damaged + mismatch > 0 → Auto-create Vendor Credit (CN)
  6. IF shortage > 0 → PO status = 'partial_received'
  7. IF all items received → PO status = 'received'
```

8. Module 5: Enhanced Stock Count + Auto-Correction

> **Type:** Enhancement of existing module | **Affects:** Stock count tables, stockCount.service.ts, StockCountPage.tsx

8.1 Stock Count Audit Flow

```
STOCK COUNT / AUDIT WORKFLOW

  Schedule: Monday / Wednesday / Friday (configurable)

  1. Create Stock Count
     ├─ Select: Specific bin OR full warehouse
     ├─ System snapshots current accounting_stock

  2. Physical Count
     ├─ Staff enters actual qty per item
     ├─ If serial-tracked: scan/enter serial numbers

  3. System Compare:
     └─ Accounting Stock (DB current_stock)  vs  Physical Stock (counted)

  4. Discrepancy Actions:
     └─ Surplus (+3 pcs)
     │      ├─ Auto-generate Transfer Order Draft
     │          (from "Unknown Source" to correct bin)
     └─ Shortage (-2 pcs)
     │      └─ Auto-generate Transfer Order Draft (if likely misplaced)
     │      ├─ OR Adjustment Entry (if confirmed loss)
     ├─ Match (0)
           ├─ Mark as verified ✓

  5. Manager Approval: Reviews auto-generated drafts
  6. Execute: Approved drafts update stock

  STATUS FLOW:
  scheduled → in_progress → pending_review → approved → closed
```

8.2 Database Enhancements

NEW COLUMN (ON STOCK COUNT TABLES)	TYPE	PURPOSE
scheduled_date	DATE	Planned count date
schedule_day	TEXT	monday / wednesday / friday
bin_id	UUID	Which bin is being counted
accounting_stock	INTEGER	System qty snapshot at count time
physical_stock	INTEGER	Actual counted qty
discrepancy_qty	INTEGER	physical - accounting (auto-calculated)
auto_action	TEXT	transfer_order_draft / adjustment / none
auto_reference_id	UUID	FK to auto-generated TO or adjustment

9. Module 6: Issue Bin Workflow

> **Type:** New module | **New Table:** issue_bin_entries | **New API:** /api/issue-bin

9.1 Issue Bin Lifecycle

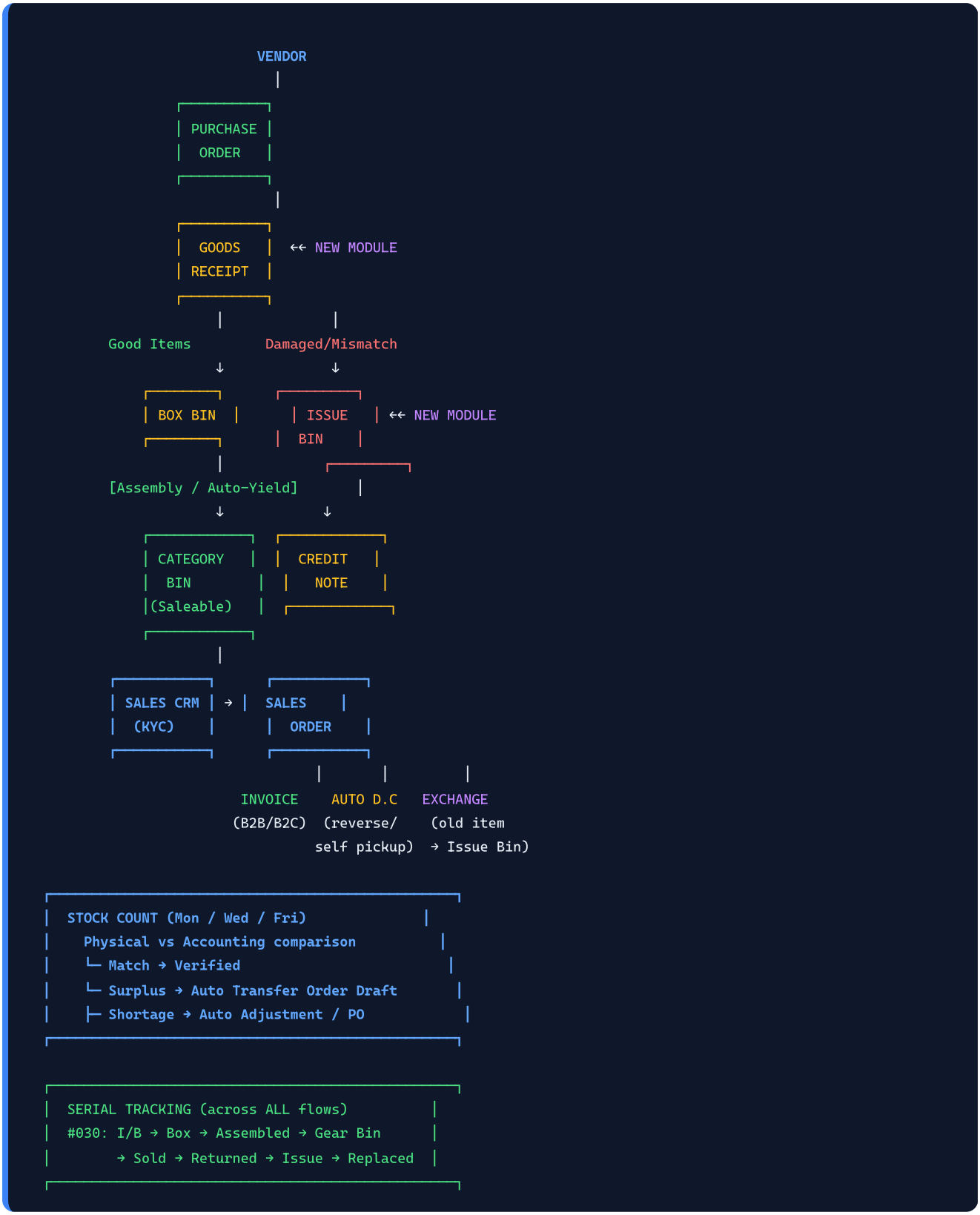
```
ISSUE BIN LIFECYCLE

  Item enters Issue Bin when:
    └─ Customer Return       (from Sales Order)
    └─ Damaged on Receipt    (from GRN inspection)
    └─ Mismatch on Receipt   (from GRN - wrong item/spec)
    ├─ Failed Quality Check  (from Stock Count)

  Item in Issue Bin:
    └─ Status: pending_review
    └─ Linked to: Credit Note / Return Order / GRN
    ├─ Resolution Options:

          └─ "Repairable"
          |     ├─ Send to assembly → back to Category Bin (Saleable)
          |
          └─ "Replace"
          |     ├─ Wait for vendor replacement
          |           └─ On arrival: New unit → correct Bin (Saleable)
          |           ├─ Old unit → Scrap or return to vendor
          |
          └─ "Scrap"
          |     ├─ Mark Non-Saleable, write-off from inventory
          |
          ├─ "Return to Vendor"
                ├─ Link to Vendor Credit, schedule return pickup

  IMPORTANT: Issue Bin items are NEVER available for sale
```

9.2 issue_bin_entries Table

COLUMN	TYPE	PURPOSE
id	UUID (PK)	Primary key
serial_id	UUID → serial_numbers	If serial tracked (nullable)
item_id	UUID → items	Which item
quantity	INTEGER	Qty in issue bin (if not serial tracked)
bin_id	UUID → bin_locations	Which Issue Bin
reason	TEXT	customer_return / damaged_receipt / mismatch / quality_fail
source_type	TEXT	sales_order / goods_receipt / stock_count
source_id	UUID	FK to source document
resolution	TEXT	pending / repairable / replace / scrap / return_to_vendor
credit_note_id	UUID	FK to vendor_credits (if CN issued)
replacement_serial_id	UUID	FK to replacement serial number

resolved_at	TIMESTAMP	When issue was resolved
resolved_by	UUID → users	Who resolved it
notes	TEXT	Resolution notes
created_at	TIMESTAMP	When item entered Issue Bin

10. Complete Data Flow (All Modules Connected)



11. Database Schema Changes Summary

11.1 New Tables (4)

TABLE	PURPOSE	KEY RELATIONSHIPS
`serial_numbers`	Per-unit tracking with status and location	items, bin_locations
`serial_history`	Complete audit trail for each serial	serial_numbers, bin_locations
`goods_receipts + goods_receipt_items`	Receipt verification against POs	purchase_orders, purchase_order_items, items
`issue_bin_entries`	Damaged/returned item lifecycle	serial_numbers, items, bin_locations, vendor_credits
`sale_accessories`	Accessories sold with main product	sales_orders, items

11.2 Altered Tables (4)

TABLE	NEW COLUMNS
`bin_locations`	location_type, bin_category, assembly_state, saleability
`customers`	customer_type, pin_code, alt_phone, kyc_completed
`sales_orders`	delivery_type, is_exchange, exchange_value, exchange_item_desc, salesperson_id, auto_dc_generated
`stock_counts / stock_count_items`	scheduled_date, schedule_day, bin_id, accounting_stock, physical_stock, discrepancy_qty, auto_action, auto_reference_id

11.3 Updated PO Status Values

CURRENT STATUSES	NEW STATUSES ADDED
draft, sent, received, closed	`partial_received`, `discrepancy_flagged`

> **IMPORTANT:** After ALL DDL changes, run `NOTIFY pgrst, 'reload schema';` to refresh the PostgREST cache. All new columns must use snake_case naming. Never rename existing columns.

12. New API Endpoints

ENDPOINT **/api/serial-tracking**	METHOD	PURPOSE
`/`	GET	List all serials (filter by item, bin, status)
`/:id`	GET	Serial detail + full history timeline
`/`	POST	Register new serial number
`/:id/move`	PUT	Move serial between bins (triggers auto-yield)
`/:id/history`	GET	Serial event timeline

/api/goods-receipts		
/	GET	List all GRNs (filter by PO, date, status)
/:id	GET	GRN detail with items and discrepancies
/	POST	Create GRN against a Purchase Order
/:id	PUT	Update GRN (inspection results)
/:id/complete	POST	Complete receipt (triggers all auto-actions)
/api/issue-bin		
/	GET	List all issue bin entries (filter by reason, resolution)
/:id	GET	Issue entry detail with linked documents
/	POST	Add item to issue bin
/:id/resolve	PUT	Resolve issue (repair/replace/scrap/return)

13. File Changes Summary

13.1 New Files to Create (9 backend + 10 frontend = 19 files)

MODULE	BACKEND FILES	FRONTEND FILES
Serial Tracking	services/serial-tracking.service.ts controllers/serial-tracking.controller.ts routes/serial-tracking.routes.ts	services/serial-tracking.service.ts pages/SerialTrackingPage.tsx pages/SerialDetailPage.tsx components/serial-tracking/SerialHistoryTimeline.tsx
Goods Receipt	services/goods-receipt.service.ts controllers/goods-receipt.controller.ts routes/goods-receipt.routes.ts	services/goods-receipt.service.ts pages/GoodsReceiptPage.tsx pages/GoodsReceiptDetailPage.tsx components/goods-receipt/NewGoodsReceiptForm.tsx
Issue Bin	services/issue-bin.service.ts controllers/issue-bin.controller.ts routes/issue-bin.routes.ts	services/issue-bin.service.ts pages/IssueBinPage.tsx components/issue-bin/ResolveIssueModal.tsx

13.2 Existing Files to Modify (9 files)

FILE	CHANGES REQUIRED
backend/src/**server.ts**	Register 3 new route prefixes: /api/serial-tracking, /api/goods-receipts, /api/issue-bin
frontend/src/**App.tsx**	Add ~12 new routes for Serial, GRN, and Issue Bin pages
frontend/src/components/layout/**Sidebar.tsx**	Add navigation items: Serial Tracking, Goods Receipt, Issue Bin
backend/src/services/**binLocations.service.ts**	Add filters by bin_category, assembly_state, saleability
backend/src/services/**customers.service.ts**	Handle new KYC fields (customer_type, pin_code, alt_phone)
backend/src/services/**sales-orders.service.ts**	Add exchange logic, delivery_type, auto-DC trigger
backend/src/services/**purchase-orders.service.ts**	Add partial_received / discrepancy_flagged statuses, GRN linking
backend/src/services/**items.service.ts**	Link serial numbers to items, assembly state awareness
frontend/src/services/**stockCount.service.ts**	Add scheduling, auto-correction, discrepancy APIs

14. Implementation Roadmap

Each phase can be deployed independently without breaking production. Dependencies flow downward.

Phase 1
Foundation

Database Foundation (No dependencies on other new modules)

- **1a.** ALTER `bin_locations` table — add location_type, bin_category, assembly_state, saleability columns
- **1b.** CREATE `serial_numbers` + `serial_history` tables
- **1c.** ALTER `customers` table — add customer_type, pin_code, alt_phone, kyc_completed columns
- **1d.** Run `NOTIFY pgrst, 'reload schema';`

Phase 2
Core Flows

Core New Workflows (Depends on Phase 1 tables)

- **2a.** Build Goods Receipt (GRN) module — CREATE tables, backend service/controller/routes, frontend pages
- **2b.** Build Issue Bin workflow — CREATE table, backend service/controller/routes, frontend pages + ResolveIssueModal
- **2c.** Build Serial Tracking module — backend service/controller/routes, frontend pages + SerialHistoryTimeline
- **2d.** Implement Auto-Yield logic in serial tracking service (Box Bin → Category Bin = assembled + saleable)

Phase 3
Business Logic

Business Logic Integration (Depends on Phase 2 modules)

- **3a.** Sales CRM KYC flow — enhance customer forms with B2B/B2C, accessory list
- **3b.** GRN discrepancy handling — auto Credit Note, auto Issue Bin entry, PO status updates
- **3c.** Exchange flow in Sales Orders — salesperson valuation, old item → Issue Bin, invoice adjustment
- **3d.** Auto-DC generation — trigger DC from reverse/self pickup sales orders

Phase 4
Automation

Scheduled Automation (Depends on Phase 3 integrations)

- **4a.** Stock Count scheduling — Mon/Wed/Fri auto-creation, physical vs accounting comparison
- **4b.** Stock Count auto-correction — auto Transfer Order Draft / Adjustment Entry generation
- **4c.** Serial History Timeline UI — visual lifecycle display on item detail pages
- **4d.** Manager approval workflow for auto-generated drafts (TO, adjustments)

14.1 Phase Dependency Graph

```
IMPLEMENTATION DEPENDENCY FLOW

  Phase 1 (Foundation)
    └─ 1a. Bin table columns
    └─ 1b. Serial tables
    └─ 1c. Customer KYC columns
    ├─ 1d. PostgREST reload
         |
         ↓
  Phase 2 (Core Workflows) — depends on Phase 1 tables
    └─ 2a. GRN module        (needs bins)
    └─ 2b. Issue Bin module   (needs bins + serials)
    └─ 2c. Serial Tracking    (needs serial tables)
    ├─ 2d. Auto-Yield logic   (needs bins + serials)
         |
         ↓
  Phase 3 (Business Logic) — depends on Phase 2 modules
    └─ 3a. Sales CRM KYC     (needs customer columns)
    └─ 3b. GRN discrepancy   (needs GRN + Issue Bin)
    └─ 3c. Exchange flow     (needs Issue Bin + Sales Orders)
    ├─ 3d. Auto-DC           (needs Sales Orders + DC service)
```

```
        │
        ↓
Phase 4 (Automation) — depends on Phase 3 integrations
  └─ 4a. Stock Count scheduling
  └─ 4b. Auto-correction
  └─ 4c. Serial Timeline UI
  ├─ 4d. Approval workflows
```

Deployment Note: Each phase is independently deployable. Push to `main` branch after each phase passes `npm run build` in both `/backend` and `/frontend`. Coolify auto-deploys on push.

2XG

Enterprise Resource Planning

Architecture Flow Document v2.0

Modules: 6 New/Enhanced

New Tables: 5 | Altered Tables: 4

New API Endpoints: 14 | New Files: 19

Modified Files: 9

Prepared for 2XG Growth | February 2026
Stack: React + Express + Supabase (PostgreSQL)
Deployed via Coolify on OVH Self-Hosted Infrastructure