Product Requirements Document (PRD)
Daily Sales & Inventory Tracker with AI-Driven Insights
Last Updated: May 16, 2025

---

1. Executive Summary

A Progressive Web App (PWA) for small-to-mid retailers to track daily sales and turnover, manage multi-location inventory, and harness AI-driven insights—built on zero-dollar-upfront foundations (Supabase, Vercel, Clerk) with enterprise-grade scalability.

---

2. Objectives & Success Metrics

Objective	Metric	Target
Accurate stock tracking	Stock discrepancy rate	< 1% of SKUs
Fast restock decisions	Time from alert → PO creation	< 10 minutes
AI insight adoption	% of users acting on AI suggestion	≥ 40% within 24 hrs
Free-tier cost containment	Cloud spend before revenue	$0 until 1,000 users
User adoption	Active shops (≥ 3 daily transactions)	100 in 3 months

---

3. Functional Requirements

3.1 Sales & Inventory

Atomic Sales Transaction
Ensures stock consistency:

```
WITH sale AS (
  INSERT INTO sales (product_id, qty_sold, price_sold, location_id, user_id)
  VALUES (...)
  RETURNING product_id, qty_sold
)
UPDATE products SET stock = stock - sale.qty_sold
```

FROM sale WHERE products.id = sale.product_id;

-
- **Multi-Location Transfers**
    - Create transfer orders with origin/destination
    - Audit logs of each move

3.2 Purchase Orders

- **PO Creation & Receipt**
    - PO header: supplier, date, expected delivery
    - PO items: product, qty, cost
    - “Receive” flow increases inventory_by_location

3.3 AI-Driven Insights

Feature	Implementation	User Benefit
Demand Forecasting	Prophet in Supabase Edge Function	“Order 80 units of SKU123 by May 20”
Anomaly Detection	PyCaret clustering	“Sales of SKU456 down 60% vs last week”
Prescriptive Actions	GPT-4 + business logic	“Transfer 25 units from B → A to avoid stockout”
Automated Summaries	Edge Function → email via Resend API	Daily “Top 3 Opportunities” email at 7 AM

3.4 Reporting & Compliance

- **Dashboards**: turn-over, profit, live inventory
- **Exports**: CSV/download history
- **Audit Logs**: immutable record of all stock/Purchase Order/AI events

3.5 Authentication & Authorization

- Clerk.dev for SSO, passwordless, roles (staff vs. admin)
- Post-login → redirect to dashboard
- RLS policies in Supabase for data isolation

3.6 UX Flows & Screens

Screen Name	Route	Purpose
Login	`/auth/login`	Email/password or Google SSO
Signup	`/auth/signup`	New account creation
Forgot Password	`/auth/forgot`	Passwordless magic link
Dashboard	`/dashboard`	Key metrics, AI insights widget
Products List	`/products`	CRUD, search, mobile card view
Product Detail / Edit	`/products/[id]`	View stock history, price, supplier
Sales Entry	`/sales/new`	Add one/multiple sales, typeahead select
Purchase Orders	`/po`	List & create POs
Inventory Transfers	`/transfers/new`	Create inter-location transfer
Reports & Exports	`/reports`	CSV exports, history filters
Settings	`/settings`	Team, roles, integrations

---

4. Non-Functional Requirements

- **Performance**: < 200 ms page load in 90th percentile
- **Scalability**: support 1,000 daily users on free tiers, 10,000+ on paid
- **Reliability**: 99.9% uptime, daily backups
- **Security**: GDPR compliance, TLS everywhere, RLS policies, secret management via Vault/AWS Secrets Manager
- **Offline**: PWA caching for products & queue sales offline → sync
- **Zero-Dollar Billing**:
    - **Auth**: Clerk free up to 1k users
    - **DB**: Supabase free tier (500 MB, row limits)
    - **Functions**: Supabase Edge free tier
    - **Hosting**: Vercel/Netlify free static
    - **Queue & Storage**: Firebase free tier

---

5. Technical Architecture

```
graph LR
  A[Next.js PWA (App Router)]
    -->|HTTPS| B[Cloudflare CDN & WAF]
    --> C[Vercel Edge Network]
  A --> D[Supabase Edge Functions]
  D --> E[(PostgreSQL)]
  D --> F[Redis Cache]
  E --> G[Prophet & PyCaret Models]
  G --> H[OpenAI API]
  H --> A
  subgraph Observability
    I[Prometheus] --> J[Grafana]
    K[Sentry]
  end
  A & D --> I & K
```

---

6. Project Directory Skeleton

Path	Description
`/app/`	Next.js App-Router pages & layouts
`src/components/auth/`	Auth UI (login/signup/forgot)
`src/components/ui/`	Shared UI primitives (buttons, forms, calendar, carousel)
`src/features/`	Domain features: `sales/`, `products/`, `dashboard/`
`src/db/schema/`	Drizzle table definitions
`migrations/`	SQL migration files generated by Drizzle
`styles/`	`globals.css` with Tailwind directives & theme
`postcss.config.js`	Loads `@tailwindcss/postcss` + `autoprefixer`
`tailwind.config.js`	Content paths, theme extensions, plugins
`next.config.js`	Next.js settings (ESLint skip, rewrites, etc.)
`drizzle.config.js`	Drizzle-kit config for migrations
`terraform/`	IaC for Supabase, Redis, Cloudflare
`.github/workflows/`	CI pipeline config
`Dockerfile` & `docker-compose.yml`	Local dev Docker compose & prod Dockerfile
`helm/`	Kubernetes Helm charts for production

---

7. Data Model

Tables

- `products` (id, sku, name, cost_price, sell_price, stock)

- `locations` (id, name)
- `inventory_by_location` (product_id, location_id, stock)
- `sales` (id, product_id, qty_sold, price_sold, location_id, user_id, timestamp)
- `suppliers` (id, name, contact_info)
- `purchase_orders` (id, supplier_id, date, status)
- `po_items` (po_id, product_id, qty, cost)
- `price_history` (product_id, old_price, new_price, changed_at)
- `audit_logs` (id, entity, action, user_id, timestamp, details)
- `recommendations` (id, sku, type, message, generated_at)

---

8. API Contracts

Base URL: `https://api.example.com/v1`

Create Sale

POST `/sales`

{ "product_id": 123, "qty_sold": 5, "price_sold": 9.99, "location_id": 1 }

200

{ "id": 456, "remaining_stock": 45 }

Get Products

GET `/products?limit=50&page=1`
200

{ "data": [{ "id":123,"sku":"SKU123","stock":50 }], "pagination": { "page":1,"pages":2 } }

Forecast Insights

GET `/insights/forecast?sku=SKU123`
200

{ "forecast": [{ "date":"2025-05-17", "predicted": 80 }, …] }

(Full API spec with error formats and auth headers in separate doc.)

---

9. Tooling & Integration

Category	Tool	Purpose & Justification	Integration
Frontend	Next.js 15 (App Router)	SSR/SSG, App Router for layouts & nested routes	Vercel build & preview deployments
Styling	Tailwind v4 + Shadcn/UI	Utility-first, prebuilt accessible components	PostCSS plugin + config globs
Auth	Clerk.dev	Passwordless, SSO, role management	Next.js middleware & RLS policies
DB & ORM	Supabase (free) + Drizzle ORM	Hosted Postgres + type-safe migrations	Drizzle CLI + `drizzle.config.js`
Cache & Sync	Redis	Cache AI forecasts, CDN caching	Terraform provision + Edge Funcs
AI/ML	Prophet, PyCaret, OpenAI GPT-4	Forecast, anomalies, GPT insights	Edge Functions + Redis cache layer
Hosting & CDN	Vercel free tier + Cloudflare	Global edge, static hosting, WAF	GitHub → Vercel integration
IaC	Terraform	Reproducible infra: Supabase, Redis, Cloudflare	CI-driven apply via GitHub Actions
Containerization	Docker, Kubernetes, Helm	Local dev parity + future multi-region production roll-out	Docker Compose + Helm charts

Observability	Prometheus, Grafana, Sentry	Metrics, dashboards, error & performance monitoring	Helm deployments + Sentry DSN in env
CI/CD	GitHub Actions → Vercel	Automated builds, tests, preview URLs, auto-deploy	Workflow YAML in `.github/workflows`

---

10. Roadmap & Milestones (14 Days)

Phase	Days	Deliverables
Kickoff & Infra	1–2	Supabase + RLS toggle; Drizzle migrations; Terraform scripts
Auth & UI Shell	3–4	Clerk integration; App Router layouts; global styles
Core CRUD MVP	5–7	Product & Sales screens + API routes; inventory math
Dashboard & AI	8–10	Dashboard charts; Prophet forecasting; anomaly feed
PO & Transfers	11–12	PO creation/receipt; transfer workflows; audit logs
Testing & CI/CD	13	Unit & e2e tests; GitHub Actions; ESLint/prettier hooks
Polish & Deploy	14	PWA offline; caching; Sentry; final Vercel deployment

---

11. Exit Criteria & Sign-Off

Requirement	Pass	Fail
Stock Consistency	Zero negative stock bugs	Any negative stock occurs
Forecast Accuracy	±10% error vs actual	> 20% error
Page Performance	≤ 200 ms interactive paint	> 500 ms
Free-Tier Cost	$0 until 1k users	Any charge before revenue
AI Insight Engagement	≥ 40% action rate	< 20% action rate

Prepared by: Product & Engineering Leads
Approved by: _____ (Product Manager) / _____ (Engineering Manager)