

# Marketplace Platform Overview for Prospective Buyers

## Executive Summary

Your potential acquisition is a modern marketplace platform built with Supabase (Postgres + Auth + Storage) and a Vite/React frontend. It enables sellers to onboard quickly, list products, manage POS operations, chat with buyers, and monitor analytics. The system enforces ownership, supports offline-first POS workflows, and offers business-type customization plus location-aware features.

## Stack & Architecture

- Frontend:** React 18 via Vite, TypeScript, Tailwind CSS with custom theme tokens, Radix UI derivatives, Sonner notifications, next-themes for light/dark modes, React Router for navigation, and React Query for data synchronization.
- Backend:** Supabase manages authentication, Row-Level Security policies, scheduled analytics, storage buckets for images, and Edge functions (e.g., Gemini-based `analyze-product`).
- Data Model Highlights:**
  - `profiles`: contains owners, contact info, geolocation, display settings.
  - `products`: supports categories/subcategories, variants, status, and location metadata.
  - `orders`, `order_items`, `payments`: POS-ready tables with enforced ownership, triggers, snapshots, and receipt printing.
  - Supporting tables:** `categories`, `locations`, `analytics_daily`, `product_views`, `profile_views`, `pos_sessions`, `pos_payment_methods`, `variants`, `receipt_snapshots`.
- Monitoring & Observability:** Custom `src/lib/monitoring` tracks errors & Supabase connectivity; errors routed to Sonner toasts + log outputs.

## Functional Blocks (What It Does)

- Authenticated Marketplace**
  - Sellers register via Supabase Auth; profiles created automatically on signup.
  - Profile pages display vendor info, ratings, contact, and optional location (with district detection when coordinates fall in configured ranges).
  - Category and location filters route buyers to relevant offerings.
- Product Management**
  - Business type (set in Settings or /sell) filters category/variant suggestions in the ProductForm.
  - AI assistant (Gemini Vision) analyzes product images to propose descriptions, categories, subcategories, and variants.
  - Sellers upload photos (Supabase Storage), set prices, and select locations/categories; subcategories can be reused from other listings.
- Point of Sale (POS)**
  - POS interface supports multi-order tabs, cart editing via numpad, discounts, customer selection, and payments with change calculation.
  - Orders insert into Supabase, payments stored, receipts printed/snapshotted, and offline transactions queued for sync with `offlineSync` utilities.
- Chat & Analytics**
  - Conversations created automatically when others view a seller's product; messages logged via `chat_conversations/chat_messages` tables.
  - Daily rollups (`analytics_daily`) refreshed via Postgres function for views, orders, revenue. Product/profile views deduped per UTC day.
- Location Awareness**
  - Settings page requests geolocation permission; stores lat/lon and labels district (Abdi Asis for provided ranges). Profile optionally displays district to buyers.
  - Seller location visibility toggles per profile; coordinates stored in Supabase for future mapping or locality filtering.
- Design System**
  - Global theme variables for colors, shadows, and typography; theme toggle (light/dark/system) uses next-themes.
  - All pages share consistent radial gradients, cards, and responsive layouts (desktop/mobile). Custom components: `StatsCard`, `CategoryFilters`, `ImageLightbox`, `ThemeToggle`.

## Detailed Workflow (How Buyers/Owners Use the Platform)

- Seller Onboarding & Profile Setup**
  - Sign up via `/auth`; Supabase trigger creates profile with username.
  - Visit Settings: select business type (e.g., Electronics), configure receipts, enable Display Location, and tap "Use My Current Location" (browser prompts allow location capture). The app stores lat/lon and automatically labels "abdi asis" when within the configured coordinate range.
  - Update profile (company, phone, avatar) and set full name (cooldown enforces once per 7 days).
- Product Listing**
  - Use `/sell` to select business type; this writes keywords to `localStorage`.
  - Access `/products/new/edit`: categories/variants filtered by keywords, AI helper suggests description/category/subcategory/variants based on uploaded photo, location select uses Supabase locations, and variant chips help create variants records.
  - Save product: data inserted into `products`; variants inserted with price derived from base price.
- Customer Browsing**
  - Visit `/` or `/category`: search, filter by location, subcategory, sort order. Product grid uses skeleton loaders until data arrives.
  - Click product to view detail: image lightbox, seller badge, ratings/reviews, and chat with seller. Each view writes to `product_views` (day-deduped) and triggers a chat message saying "👁️ viewed your product" (seller sees it, viewer sees their conversation only).
- POS Operations**
  - Merchants log into `/pos`: add products via multi-order tabs, adjust quantities/discounts via cart and numpad, open payment modal to capture multiple payments, get change due, and confirm order.
  - Orders insert into Postgres `orders`, `order_items`, `payments` tables (with `created_by`). Receipt data saved to `receipt_snapshots`. Offline mode stores actions locally and syncs when the device reconnects.
- Analytics & Reporting**
  - Daily rollups populated by `refresh_daily_rollups`. Sellers can view aggregated items per day via Dashboard and Receipts pages (SQL queries rely on Seller ID from `products/order_items`). Product views, profile views, cart adds, receipts, and revenues all aggregated for reporting.

## Operational Considerations

- Deployment:** Use Supabase CLI (`supabase db push`) to apply migrations. Ensure `supabase/migrations/config.toml` references correct project. Frontend builds with `npm run build` (chunks currently large; consider manual chunking if needed).
- Scaling:** Supabase handles auth/RLS and storage; POS uses offline sync for poor connectivity. Analytics stored in `analytics_daily` keeps query costs low.
- Security:** RLS policies ensure sellers only access their profiles, products, orders, and payments (via helpers like `is_order_owner`). Sensitive receipts/variants scoped to owners.
- Extensibility:** Business type flow, variant templates, geolocation detection, lightbox component, stats cards, and theme tokens ease future UI/UX upgrades.

## Next Steps for Buyers

- Walk through Settings → select business type, configure receipt, enable location, and try location capture.
- Create a product via `/products/new/edit`, applying AI suggestions and variant chips; inspect Supabase tables for new entries.
- Run POS flow: add items, apply discounts, capture payments, print receipt, verify `orders/payments/receipt_snapshots` tables.
- Test chat: log in as seller, view product as another user, verify chat message creation.
- Review analytics: run `refresh_daily_rollups`, inspect `analytics_daily` to ensure rollups align with activity.
- Use `/category` and `/` filters to confirm search, subcategory chips, and location filters show appropriate products.

