Teknis Project

Struktur Halaman (Sitemap)

Kita bagi berdasarkan fungsi: **Mobile untuk Operasional Cepat**, **Web untuk Manajemen & Analisa**.

Mobile App (Flutter) - Fokus: Kasir & Input Lapangan

1. **Authentication (Gerbang Masuk)**
    - **Login Page:**
        - Input: Username & Password.
        - **Biometric Toggle:** "Aktifkan Fingerprint/FaceID" (muncul setelah login sukses pertama kali).
    - **Splash Screen:** Logo branding & indikator sinkronisasi data (memastikan data lokal terupdate).
2. **Home Page (Dashboard Operasional)**

    Halaman utama yang muncul pertama kali. Desain bersih dan langsung ke tujuan.

    - **Top Card (Status Keuangan):**
        - **Pendapatan Hari Ini:** Ditampilkan besar (misal: "Rp 1.500.000").
        - **Profit Hari Ini:** *Hidden by default* (icon mata untuk intip), menjaga privasi.
    - **Main Action Buttons (Floating/Fixed Bottom):**
        - Menggantikan tombol "Buy/Sell" lama dengan konsep Arus Barang:
        - **[IN] (Tombol Hijau/Biru):** Pintasan cepat ke halaman Pembelian/Barang Masuk.
        - **[OUT] (Tombol Merah/Oranye):** Pintasan cepat ke halaman Penjualan/Barang Keluar.
    - **Recent Transaction Feed (List):**
        - Menampilkan gabungan transaksi IN dan OUT dalam satu *timeline*.

- Visual: Icon Panah Hijau (Masuk) vs Panah Merah (Keluar).
- Info: Jam transaksi, Nama Toko/Konsumen, Nominal.

3. **IN Page (Pembelian / Stock In)**

   Konsep UI: 100% Chat Interface (Seperti WhatsApp/Telegram).

   - **Input Methods:**
     1. **Tombol Kamera:** User foto struk belanja → Kirim ke Chat → AI memproses OCR → Balas dengan "Draft Transaksi JSON".
     2. **Input Text Field:** User ketik manual (Contoh: *"Beli 50kg alpukat @10rb dan 2 bal keripik @50rb"*).
   - **AI Interaction Flow:**
     - *User:* Kirim Gambar/Teks.
     - *System:* Menampilkan *bubble* balasan berupa ringkasan item yang dideteksi.
     - *Action:* User klik "Simpan" (jika benar) atau "Edit" (jika AI salah baca).

4. **OUT Page (Penjualan / Stock Out)**

   Konsep UI: Hybrid (Memberi kebebasan pada User).

   - **Tab 1: Chat Mode (Default)**
     - Sama seperti halaman IN, tapi untuk penjualan.
     - Contoh: *"Jual 2kg alpukat ke Bu Ani"* → Sistem auto-cek stok dan hitung total.
   - **Tab 2: Form Mode (Manual)**
     - UI Klasik untuk user yang ingin input terstruktur.
     - **Input Pelanggan:** Dropdown/Search Kontak.
     - **Input Produk:** Search Produk → Pilih Satuan Dinamis (Kg/Pcs/Bal) → Input Qty.
     - **Cart:** Menampilkan list barang yang akan di-checkout.
     - **Payment:** Input nominal bayar & pilihan metode (Cash/Transfer).

5. **Stock Page (Inventory)**

Halaman ini **wajib ada** untuk pengecekan gudang tanpa harus buka laptop.

- **Stock List:**
  - Daftar semua produk dengan stok saat ini (Real-time).
  - Filter: Kategori (Buah, Snack, Bahan Baku).
  - Search Bar: Cari nama barang cepat.
  - Indikator: Warna merah jika stok di bawah batas aman (*Low Stock Alert*).
- **Stock Opname (Fitur Audit):**
  - Scanner Barcode (jika ada) atau pilih item manual.
  - Input: "Stok Fisik" (Apa yang ada di tangan).
  - Sistem menghitung selisih otomatis (Stok Sistem vs Stok Fisik).

6. **Reports (Lite)**
   - Ringkasan Omset & Profit Harian/Mingguan.
   - Top 5 Produk Terlaris hari ini.
7. **Product**

   Solusinya adalah konsep **"Lazy Creation" (Pembuatan Inline)**. Berikut logikanya:

   1. **Jalur Utama (Master Data):** User sengaja buka menu "Produk" → "Tambah Produk" untuk memasukkan barang baru yang belum pernah dibeli, tapi akan dijual (atau stok awal).
   2. **Jalur Pengadaan (Procurement Flow):**
      - User scan struk/input manual di halaman `Buy`.
      - Backend/Frontend mengecek: *"Apakah nama barang ini ada di DB?"*
      - **Jika ADA:** Langsung ambil `product_id` nya, update stok & *average cost* otomatis.
      - **Jika TIDAK ADA (Barang Baru):** Sistem **harus** memunculkan popup/modal kecil: *"Produk baru terdeteksi. Konfirmasi detailnya (Satuan Dasar, Nama, Kategori) sebelum lanjut."*
      - Setelah user konfirmasi, sistem melakukan 2 hal dalam 1 kali klik "Simpan":

1. `INSERT` ke tabel `products` (Create Master).
2. `INSERT` ke tabel `transactions` & `transaction_items` (Create Transaction).

Kesimpulan: Data produk bisa masuk dari kedua pintu tersebut. Namun, pintu "Pengadaan" membutuhkan fitur *interceptor* (pengecekan) sebelum transaksi disimpan.

Yang ada di navigation bottom: Home, List Transaction, List Product, List Contact

Web App (Next.js) - Fokus: Admin & Owner

- **Dashboard Utama:**
  - **Grafik Utama:** Tren Pendapatan & Profit (Line Chart).
  - **Inventory Health:** Pie chart komposisi stok & peringatan stok menipis.
  - **Audit Log Stream:** Melihat aktivitas user (siapa yang edit stok, siapa yang login)
- **Master Data (Produk & Kontak):**
  - **Produk:**
    - CRUD Produk Lengkap.
    - **Conversion Rules Setting:** Di sini tempat mengatur "1 Karung = 30 Kg", "1 Bal = 20 Pcs" (JSONB Editor).
    - Set Harga Modal Awal & Harga Jual.
  - **Kontak:**
    - Database Supplier & Customer.
    - Riwayat transaksi per kontak.
- **Inventory Management:**
  - **Kartu Stok (Ledger):** Tabel mutasi super detail. Setiap barang masuk/keluar tercatat di sini (siapa, kapan, kenapa). Solusi untuk masalah "Stok Hilang".
  - **Approval Opname:** Menyetujui hasil *Stock Opname* dari mobile app (untuk mencegah kecurangan pegawai).
- **Financial Reports**

    - **Laba Rugi (P&L):** Menggunakan metode **Average Cost** agar perhitungan profit akurat meski harga beli naik-turun.
    - **Laporan Arus Stok:** Analisis barang paling cepat laku (*Fast Moving*).
    - Export Data (Excel/PDF).
- **Settings:**
    - **User Management:** Tambah/Edit akun pegawai & hak akses.
    - **AI Configuration:** Menambah template parsing baru jika ada format struk toko baru yang unik

---

Fitur-Fitur Kunci (Feature List)

1. **Smart Input (AI Core):**
    - **Struk-to-Data:** Mengubah image struk menjadi JSON transaksi.
    - **Text-to-Data:** Mengubah teks chat orderan menjadi JSON transaksi.
    - **Unit Memory:** Sistem "mengingat" konversi satuan (misal user pernah set 1 Karung = 50kg, besoknya auto-suggest).
2. **Dynamic Inventory System:**
    - **Multi-Unit Conversion:** Beli satuan besar (Bal/Karung), Jual satuan kecil (Pcs/Kg).
    - **Average Costing:** Menghitung ulang HPP setiap kali ada pembelian baru dengan harga berbeda.
3. **Dynamic Sales Form:**
    - UI input penjualan berubah sesuai tipe produk (Berat/Varian/Paket).
4. **Audit Trail (Stock Ledger):**
    - Mencatat *siapa* yang mengubah stok, *kapan*, dan *mengapa* (Penjualan, Pembelian, atau Barang Busuk).

Struktur Database (SQL Schema)

Ini adalah bagian terpenting. Copy code di bawah ini dan jalankan di **SQL Editor** pada dashboard Supabase atau PostgreSQL client kamu.

Schema ini sudah mendukung: **JSONB**, **Average Cost**, dan **Stock Ledger**.

SQL

```
-- ============================================================
=
-- 1. CONFIGURATION & ENUMS
-- ============================================================
=
-- Enable UUID extension for secure IDs
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Enums to standardize data types
CREATE TYPE trx_type AS ENUM ('IN', 'OUT'); -- IN (Beli/Mas
uk), OUT (Jual/Keluar)
CREATE TYPE contact_type AS ENUM ('SUPPLIER', 'CUSTOMER',
'BOTH');
CREATE TYPE user_role AS ENUM ('OWNER', 'ADMIN', 'STAFF');

-- ============================================================
=
-- 2. USER & AUTHENTICATION
-- ============================================================
=
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    username VARCHAR(100) UNIQUE NOT NULL,
    full_name VARCHAR(100),
    role user_role DEFAULT 'STAFF',

    -- Security
    password_hash TEXT NOT NULL,
    biometric_token TEXT, -- Token for mobile faceID/finger
print
    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    last_login_at TIMESTAMPTZ
);
```

```
-- ============================================================
=
-- 3. MASTER DATA (PRODUCTS & CONTACTS)
-- ============================================================
=
-- Unified Contact Table (Supplier + Customer)
CREATE TABLE contacts (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255) NOT NULL,
    type contact_type DEFAULT 'CUSTOMER',
    phone VARCHAR(50),
    address TEXT,
    notes TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- THE CORE PRODUCT TABLE
CREATE TABLE products (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    sku VARCHAR(50) UNIQUE,
    name VARCHAR(255) NOT NULL,

    -- Base Unit: The smallest unit used for tracking (e.
g., 'kg', 'pcs', 'ml')
    base_unit VARCHAR(20) NOT NULL DEFAULT 'pcs',

    -- Current Stock Level (Always in Base Unit)
    current_stock DECIMAL(15, 3) DEFAULT 0,

    -- Financial Logic (Average Cost Method)
    average_cost DECIMAL(15, 2) DEFAULT 0, -- Rata-rata mod
al per base_unit
    latest_selling_price DECIMAL(15, 2) DEFAULT 0,

    -- Smart Conversion Rules (JSONB)
    -- Example: {"karung": 30, "bal": 20, "pack": 0.5}
    -- Means: 1 Karung = 30 base_units, 1 Pack = 0.5 base_u
nits
```

```
    conversion_rules JSONB DEFAULT '{}'::jsonb,

    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ========================================================
=
-- 4. TRANSACTIONS (IN & OUT)
-- ========================================================
=
-- Transaction Header (Gabungan Beli & Jual)
CREATE TABLE transactions (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),

    -- Determines if it's Buy (IN) or Sell (OUT)
    type trx_type NOT NULL,

    contact_id UUID REFERENCES contacts(id),
    transaction_date TIMESTAMPTZ DEFAULT NOW(),
    invoice_number VARCHAR(50) UNIQUE,

    -- Financials
    total_amount DECIMAL(15, 2) DEFAULT 0,
    payment_method VARCHAR(50), -- CASH, TRANSFER, QRIS

    -- Input Source Tracking
    input_source VARCHAR(50) DEFAULT 'MANUAL', -- 'CHAT_A
I', 'OCR_CAM', 'FORM'
    evidence_url TEXT, -- URL to receipt image

    created_by UUID REFERENCES users(id),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Transaction Details (Items)
CREATE TABLE transaction_items (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    transaction_id UUID REFERENCES transactions(id) ON DELE
```

```
TE CASCADE,
    product_id UUID REFERENCES products(id),

    -- User Input Data (What the user typed/selected)
    input_qty DECIMAL(12, 2) NOT NULL, -- e.g., 2
    input_unit VARCHAR(20) NOT NULL,   -- e.g., 'Karung'
    input_price DECIMAL(15, 2) NOT NULL, -- Price per 'Karu
ng'

    -- System Conversion (What stored in DB)
    conversion_rate DECIMAL(12, 2) NOT NULL DEFAULT 1, --
e.g., 30 (if 1 Karung = 30kg)

    -- Calculated Base Qty (2 * 30 = 60kg)
    base_qty DECIMAL(15, 3) GENERATED ALWAYS AS (input_qty
* conversion_rate) STORED,

    -- Financial Snapshot (CRITICAL FOR PROFIT REPORT)
    -- Stores the COGS (HPP) at the moment of sale
    cost_price_at_moment DECIMAL(15, 2) DEFAULT 0,

    subtotal DECIMAL(15, 2) GENERATED ALWAYS AS (input_qty
* input_price) STORED,

    notes TEXT
);

-- ========================================================
=
-- 5. AUDIT & INVENTORY CONTROL
-- ========================================================
=
-- Stock Ledger (Kartu Stok) - The Truth Source
CREATE TABLE stock_ledger (
    id BIGSERIAL PRIMARY KEY,
    product_id UUID REFERENCES products(id),
    transaction_id UUID REFERENCES transactions(id),
```

```
    date TIMESTAMPTZ DEFAULT NOW(),
    type trx_type NOT NULL, -- IN / OUT / ADJUST

    qty_change DECIMAL(15, 3) NOT NULL, -- (+/- amount)
    stock_after DECIMAL(15, 3) NOT NULL, -- Running balance

    notes TEXT,
    created_by UUID REFERENCES users(id)
);

-- Indexes for Performance
CREATE INDEX idx_products_search ON products(name);
CREATE INDEX idx_trans_date ON transactions(transaction_dat
e);
CREATE INDEX idx_trans_type ON transactions(type);
CREATE INDEX idx_ledger_prod ON stock_ledger(product_id);

-- ========================================================
=
-- 6. AUTOMATION TRIGGERS (THE BRAIN)
-- ========================================================
=
-- Function to Auto-Update Stock & Average Cost
CREATE OR REPLACE FUNCTION process_stock_movement() RETURNS
TRIGGER AS $$
DECLARE
    curr_avg_cost DECIMAL(15, 2);
    curr_stock DECIMAL(15, 3);
    new_avg_cost DECIMAL(15, 2);
    trx_type trx_type;
BEGIN
    -- Get Transaction Type
    SELECT type INTO trx_type FROM transactions WHERE id =
NEW.transaction_id;

    -- Get Current Product State
    SELECT average_cost, current_stock INTO curr_avg_cost,
curr_stock
```

```
    FROM products WHERE id = NEW.product_id;

    -- LOGIC FOR IN (PURCHASE/RESTOCK)
    IF trx_type = 'IN' THEN
        -- Calculate New Average Cost (Weighted Average)
        IF (curr_stock + NEW.base_qty) > 0 THEN
            new_avg_cost := ((curr_stock * curr_avg_cost) +
NEW.subtotal) / (curr_stock + NEW.base_qty);
        ELSE
            new_avg_cost := NEW.input_price / NULLIF(NEW.co
nversion_rate, 0);
        END IF;

        -- Update Product Master
        UPDATE products
        SET current_stock = current_stock + NEW.base_qty,
            average_cost = COALESCE(new_avg_cost, 0),
            updated_at = NOW()
        WHERE id = NEW.product_id;

        -- Write to Ledger
        INSERT INTO stock_ledger (product_id, transaction_i
d, type, qty_change, stock_after, notes)
        VALUES (NEW.product_id, NEW.transaction_id, 'IN', N
EW.base_qty, curr_stock + NEW.base_qty, 'Purchase/IN');

    -- LOGIC FOR OUT (SALES)
    ELSIF trx_type = 'OUT' THEN
        -- Update Product Master
        UPDATE products
        SET current_stock = current_stock - NEW.base_qty,
            updated_at = NOW()
        WHERE id = NEW.product_id;

        -- Snapshot Cost Price (Locking the HPP for Profit
Calc)
        NEW.cost_price_at_moment := curr_avg_cost;
```

```
        -- Write to Ledger
        INSERT INTO stock_ledger (product_id, transaction_i
d, type, qty_change, stock_after, notes)
        VALUES (NEW.product_id, NEW.transaction_id, 'OUT',
-NEW.base_qty, curr_stock - NEW.base_qty, 'Sale/OUT');
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Attach Trigger
CREATE TRIGGER trg_stock_update
BEFORE INSERT ON transaction_items
FOR EACH ROW EXECUTE FUNCTION process_stock_movement();
```

Workflow Teknis: Logic Backend (FastAPI)

Fase 1: AI Processing (Pre-Transaction)

Fase ini terjadi saat User mengirim foto struk atau chat, SEBELUM data disimpan ke database.

1. **Receive Input:** API menerima data berupa `image` (Struk) atau `text` (Chat) dari Flutter.
2. **AI Parsing (Gemini/LLM):**
    - Backend mengirim prompt ke AI: *"Ekstrak data ini menjadi JSON berisi: nama_produk, qty, satuan_input, harga_total"*.
    - **Smart Matching:** Backend mencocokkan `nama_produk` dari AI dengan tabel `products` di database (Fuzzy Search).
    - **Unit Detection:** Jika AI mendeteksi "Bal", Backend mengecek di `products.conversion_rules`: Apakah 1 Bal sudah terdefinisi? Jika belum, tandai untuk minta input user.
3. **Return Draft:** Backend mengembalikan JSON "Draft" ke Flutter untuk dikonfirmasi User (Benar/Edit).

Fase 2: Proses "IN" (Pembelian / Restock)

Terjadi saat User menekan tombol "Simpan" pada halaman IN.

1. **Receive Payload:** Menerima JSON final dari Flutter (berisi produk, input_qty, harga_beli, dan supplier).
2. **Conversion Logic:**
   - Backend menghitung `base_qty` = `input_qty` * `conversion_rate` (Misal: 2 Bal * 20 = 40 Pcs).
3. **Execution (Insert ke DB):**
   - **Insert Header:** Masukkan data ke tabel `transactions` dengan `type = 'IN'`.
   - **Insert Items:** Masukkan detail barang ke `transaction_items`.
4. **Automation (via DB Trigger):**
   - *Logic:* Database otomatis mendeteksi barang masuk.
   - Math: Menghitung Average Cost Baru:

$$HargaBaru = \frac{(StokLama \times HppLama) + (QtyMasuk \times HargaBeli)}{StokLama + QtyMasuk}$$

   - *Update:* Kolom `current_stock` bertambah & `average_cost` terupdate di tabel `products`.
   - *Audit:* Insert otomatis satu baris ke `stock_ledger` dengan status `IN`.

Fase 3: Proses "OUT" (Penjualan)

Terjadi saat User menekan tombol "Deal/Bayar" pada halaman OUT (Chat/Form).

1. **Stock Validation (Pre-Check):**
   - Backend mengecek: Apakah `current_stock` >= `qty_yang_diminta`?
   - *Jika Kurang:* Return Error "Stok tidak cukup" ke Flutter.
   - *Jika Cukup:* Lanjut.
2. **Execution (Insert ke DB):**

- **Insert Header:** Masukkan data ke tabel `transactions` dengan `type = 'OUT'`.
- **Insert Items:** Masukkan detail barang ke `transaction_items`.

3. **Automation (via DB Trigger):**
   - *Logic:* Database mendeteksi barang keluar.
   - *Update:* Kolom `current_stock` berkurang di tabel `products`.
   - **PROFIT LOCKING (Crucial):** Database otomatis menyalin nilai `products.average_cost` (HPP saat ini) ke kolom `transaction_items.cost_price_at_moment`.
     - *Tujuan:* Agar laporan profit transaksi ini **abadi** dan tidak berubah-ubah meskipun harga modal barang naik di masa depan.
   - *Audit:* Insert otomatis satu baris ke `stock_ledger` dengan status `OUT`.

Fase 4: Stock Opname (Audit Gudang)

Terjadi saat User melakukan cek fisik di Halaman Stok.

1. **Receive Input:** API menerima `product_id` dan `fisik_qty` (Jumlah nyata di gudang).
2. **Calculate Difference:**
   - Backend menghitung selisih: `Delta = Fisik_Qty - System_Qty`.
3. **Execution:**
   - **Insert Header:** Masukkan ke `transactions` dengan `type = 'ADJUSTMENT'`.
   - **Update Stock:** Trigger akan mengupdate stok agar sesuai dengan fisik.
   - **Audit Log:** `stock_ledger` mencatat penyesuaian ini (Misal: "Koreksi Stok: -2 Pcs karena rusak").

Workflow ini dirancang agar **FastAPI fokus pada Validasi & Logika Bisnis**, sementara **PostgreSQL menangani Integritas Data & Matematika Keuangan**. Ini membuat aplikasimu cepat, aman, dan minim bug perhitungan.

Menggunakan static const String baseUrl = 'http://10.0.2.2:8000';