

Street Vendor Aggregator — Design Documents

1. Architecture Overview

A modern, scalable web architecture using:

- Next.js + TypeScript for frontend
- Supabase for auth, PostgreSQL DB, and storage
- Optional Google Maps/Mapbox for location services

Data Flow:

- User signs up → profile created
- Vendor role assigned → vendor creates listing
- Customers browse vendors → vendor details loaded with location

Security:

- Full Row Level Security (RLS) on all tables
- Users can access only their own data

2. System Design

API Endpoints:

- GET /vendors
- GET /vendors/:id
- POST /vendors
- PUT /vendors/:id
- POST /reviews

Caching:

- CDN caching for vendor images, static pages

3. Wireframes (Textual)

Landing Page:

- Logo, hero section, Find Vendors button, Register as Vendor button

Signup/Login:

- Email, password, name, phone fields

Vendor Registration:

- Upload photo, shop name, category, description, area, location coordinates

Vendor List:

- Search bar, category filters, vendor cards

Vendor Detail:

- Photo, description, map preview, contact vendor

4. Database Design

Tables:

profiles(id, full_name, phone, email)

user_roles(user_id, role)

vendors(id, user_id, shop_name, category, description, photo_url)

vendor_locations(id, vendor_id, latitude, longitude, area_name)

reviews(id, vendor_id, user_id, rating, comment)

Indexes:

- vendors(category)
- vendor_locations(area_name)
- vendor_locations(latitude, longitude)

5. RLS Policies

profiles:

- Users can insert/select/update only their row

user_roles:

- Users can insert/select only their roles

vendors:

- Anyone can select
- Only owners can insert/update

vendor_locations:

- Anyone can select active locations

- Auth users can insert

6. Handoff Notes

- Convert wireframes to Figma if needed
- Ensure Supabase RLS is applied exactly
- Seed DB with sample vendor entries