MyJara E-Marketplace - Complete Technical Documentation

> ***Version***: *1.0* | ***Last Updated***: *January 2026*
> ***Platform***: *Next.js 16 + Supabase + Vercel*

Table of Contents

1. Executive Summary
2. Technology Stack
3. Project Structure
4. Database Schema
5. Authentication & Authorization
6. User Flows
7. Core Features
8. Component Library
9. Server Actions & API
10. Deployment & Configuration

---

1. Executive Summary

MyJara is a multi-tenant e-marketplace platform built for the Nigerian market, specifically focused on the **Abuja Federal Capital Territory**. The platform operates as a "store-of-stores" model where:

- **Brands/Wholesalers** create independent storefronts within the unified platform
- **Retailers** discover products across all stores via unified search
- **Jara (Bonus Products)** is the core value proposition — not discounts

Key Differentiators

Feature	Description
Jara System	"Buy X, Get Y Free" — bonus products instead of price discounts
Multi-Tenant	Each brand has isolated data with Row Level Security
Abuja-Focused	Location search restricted to 6 Abuja LGAs and districts
Market Day Integration	Links products to physical market schedules
Google Maps Navigation	One-click directions from user to store location

---

2. Technology Stack

```
graph TB
    subgraph Frontend
        NEXT[Next.js 16<br/>App Router]
        TW[Tailwind CSS v4]
        SHADCN[shadcn/ui Components]
        REACT[React 19]
    end

    subgraph Backend
        SUPA[Supabase<br/>PostgreSQL + Auth]
        RLS[Row Level Security]
        REALTIME[Supabase Realtime]
    end

    subgraph External
        VERCEL[Vercel<br/>Hosting]
        FLUTTER[Flutterwave<br/>Payments]
        RESEND[Resend<br/>Emails]
        GMAPS[Google Maps<br/>Navigation]
    end

    NEXT --> SUPA
    NEXT --> VERCEL
    SUPA --> FLUTTER
    SUPA --> RESEND
```

Versions

Package	Version	Purpose
next	16.1.1	React framework with App Router
react	19.2.3	UI library
tailwindcss	4.x	CSS-first styling
@supabase/ssr	latest	Server-side Supabase client
lucide-react	latest	Icon library

---

3. Project Structure

```
myjara/
├── public/                      # Static assets
│   ├── favicon.ico             # Site favicon
│   ├── logo.png                # Brand logo
│   └── site.webmanifest        # PWA manifest
│
├── src/
│   ├── app/                    # Next.js App Router
│   │   ├── (admin)/            # Admin dashboard routes
│   │   ├── (auth)/             # Authentication routes
│   │   ├── (brand)/            # Brand/seller dashboard
│   │   ├── (marketplace)/      # Customer-facing pages
│   │   ├── actions/            # Server Actions
│   │   ├── api/                # API routes
│   │   ├── layout.tsx          # Root layout
│   │   ├── page.tsx            # Homepage
│   │   └── globals.css         # Global styles + Tailwind
│   │
│   ├── components/
│   │   ├── admin/              # Admin-specific components
│   │   ├── chat/               # Real-time chat components
│   │   ├── marketplace/        # Product cards, search, filters
│   │   ├── shared/             # Header, Footer, Phone Dialpad
│   │   └── ui/                 # shadcn/ui primitives
│   │
│   ├── context/
│   │   └── cart-context.tsx    # Shopping cart state
│   │
│   ├── lib/
│   │   ├── constants.ts        # Abuja locations, markets, plans
│   │   ├── resend.ts           # Email templates
│   │   ├── supabase/           # Supabase client setup
│   │   └── utils.ts            # Utility functions
│   │
│   ├── types/
│   │   └── database.ts         # TypeScript database types
│   │
│   └── middleware.ts           # Auth session handling
│
├── supabase/
│   └── migrations/             # SQL migration files
│
├── next.config.mjs             # Next.js configuration
├── package.json                # Dependencies
└── README.md                   # Architecture documentation
```

Route Groups Explained

Route Group	Purpose	Access
`(admin)`	Platform admin dashboard	Platform admins only
`(auth)`	Login, registration, verification	Public
`(brand)`	Seller/brand dashboard	Store owners
`(marketplace)`	Product browsing, checkout	Public + customers

---

4. Database Schema

The database consists of **12 core tables** with Row Level Security (RLS) for multi-tenant isolation.

Entity Relationship

```
erDiagram
    users ||--o{ orders : places
    users ||--o{ stores : owns
    users ||--o{ chat_messages : sends

    stores ||--o{ products : contains
    stores ||--o{ orders : receives
    stores ||--o{ store_logistics : defines
    stores ||--o{ store_domains : has

    products ||--o{ product_images : has
    products ||--o{ order_items : "appears in"

    orders ||--|{ order_items : contains
    orders ||--o{ transactions : "paid via"
    chat_conversations ||--o{ chat_messages : contains
```

Table Descriptions

Table	Purpose	Key Fields
users	All platform users	`id`, `email`, `phone`, `role`, `full_name`
stores	Brand storefronts	`id`, `owner_id`, `name`, `slug`, `status`
products	Product catalog	`id`, `store_id`, `price`, `jara_buy_quantity`, `jara_get_quantity`
product_images	Product photos	`id`, `product_id`, `url`, `is_primary`
categories	Product categories	`id`, `parent_id`, `name`, `slug`
store_logistics	Delivery/pickup options	`id`, `store_id`, `type`, `city`, `delivery_fee`
orders	Customer orders	`id`, `user_id`, `store_id`, `total`, `status`
order_items	Order line items	`id`, `order_id`, `product_id`, `quantity`, `jara_quantity`
transactions	Payment records	`id`, `order_id`, `flutterwave_tx_id`, `status`
chat_conversations	Customer-brand chats	`id`, `user_id`, `store_id`
chat_messages	Chat messages	`id`, `conversation_id`, `message`, `sender_type`
store_domains	Custom domain mappings	`id`, `store_id`, `domain`, `is_verified`

User Roles

```
type UserRole = 'customer' | 'brand_admin' | 'platform_admin'
```

Role	Permissions
customer	Browse, purchase, chat with brands
brand_admin	Manage own store, products, orders
platform_admin	Full access, verify stores, manage categories

---

5. Authentication & Authorization

Authentication Flow

```
sequenceDiagram
    participant User
    participant App
    participant Supabase
    participant Database

    User->>App: Enter phone number
    App->>Supabase: Sign up with email/password
    Supabase->>Database: Create auth.users entry
    Database->>Database: Trigger creates public.users
    Supabase-->>App: Return session
    App-->>User: Redirect to dashboard
```

Implementation Files

File	Purpose
middleware.ts	Session refresh on every request
server.ts	Server-side Supabase client
client.ts	Browser-side Supabase client

Row Level Security (RLS)

All tables have RLS policies enforcing:

- Users can only read/write their own data
- Brand admins can only access their store's records
- Platform admins bypass RLS via Service Role Key

---

6. User Flows

6.1 Customer Flow

```
flowchart LR
    A[Homepage] --> B[Search/Explore]
    B --> C[Product Detail]
    C --> D[Add to Cart]
    D --> E[Checkout]
    E --> F[Payment]
    F --> G[Order Confirmation]
```

6.2 Seller Registration Flow

```
flowchart TD
    A[Sell on MyJara] --> B{Wholesaler or Retailer?}
    B -->|Wholesaler| C[/register/brand]
    B -->|Retailer| D[/register/retailer-type]
    D --> E{Physical/Online/Market Day}
    E --> F[/register/retailer]
    C --> G[Phone Dialpad Entry]
    F --> G
    G --> H[Multi-Step Form]
    H --> I[KYC Verification]
    I --> J[Pending Approval]
```

6.3 Brand Dashboard Flow

Key pages in `/dashboard` :

- **Overview** — Sales stats, pending orders
- **Products** — Add/edit products with Jara offers
- **Orders** — Process and fulfill orders
- **Chat** — Customer conversations

- **Settings** — Store profile, logo, description
- **Logistics** — Delivery zones and pickup points

---

7. Core Features

7.1 Jara (Bonus Product) System

The **Jara** system is the core value proposition:

```
interface Product {
    price: number
    jara_buy_quantity: number  // Buy this many...
    jara_get_quantity: number  // ...get this many FREE
}
```

Example: Buy 5, Get 1 Free

- `jara_buy_quantity: 5`
- `jara_get_quantity: 1`

Display: `🎁 Buy 5, Get 1 FREE`

7.2 Abuja Location System

Location search is restricted to **6 Abuja LGAs**:

LGA	Key Districts
AMAC (Municipal)	Garki, Wuse, Maitama, Gwarinpa, Asokoro, Jabi
Bwari	Kubwa, Dawaki, Dutse-Alhaji
Gwagwalada	Gwagwalada Central, Zuba
Kuje	Kuje Central, Kwali
Abaji	Abaji Central
Kwali	Kwali Central

7.3 Market Day Integration

Links products to physical Abuja markets:

Market	Days
Wuse Market	Daily
Garki International	Daily
Karmo Market	Tue, Fri
Kuje Market	Wed, Sat
Gwarinpa Farmers Market	Fri, Sun

7.4 Google Maps Navigation

The LocationButton component:

1. Requests user's GPS location (with permission)
2. Opens Google Maps with directions
3. Falls back gracefully if permission denied

---

8. Component Library

Shared Components

Component	File	Purpose
Header	header.tsx	Navigation, auth state, cart
Footer	footer.tsx	Links, social, legal
PhoneDialpad	phone-dialpad.tsx	Full-page phone entry

Marketplace Components

Component	File	Purpose
ProductCard	product-card.tsx	Product grid item with Jara badge
SearchBar	search-bar.tsx	Search with location filter
FilterSidebar	client-filter-sidebar.tsx	Price, category, Jara filters
CheckoutForm	checkout-form.tsx	Multi-step checkout
LocationButton	location-button.tsx	Google Maps directions

UI Primitives (shadcn/ui)

Located in `src/components/ui/` :

- Button, Card, Badge
- Input, Select, Checkbox
- Dialog, Sheet, Toast
- Form controls

---

9. Server Actions & API

Server Actions

Located in src/app/actions/:

Action	Purpose
`admin-auth.ts`	Admin login with secret key
`admin.ts`	Dashboard data fetching (bypasses RLS)
`analytics.ts`	Store statistics and metrics
`chat.ts`	Real-time chat operations
`logistics.ts`	Delivery option management
`subscription.ts`	Retailer subscription plans

`verification.ts`	Store approval/rejection

Database Functions

The `search_products` RPC function enables efficient cross-store search:

```sql
CREATE FUNCTION search_products(
    search_query TEXT,
    filter_city TEXT,
    filter_category_id UUID,
    filter_min_price DECIMAL,
    filter_max_price DECIMAL,
    filter_min_jara INTEGER
) RETURNS TABLE (...)
```

---

10. Deployment & Configuration

Environment Variables

Required in `.env.local`:

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbG...
SUPABASE_SERVICE_ROLE_KEY=eyJhbG...

# Payments
FLUTTERWAVE_PUBLIC_KEY=FLWPUBK_TEST-...
FLUTTERWAVE_SECRET_KEY=FLWSECK_TEST-...

# Email
RESEND_API_KEY=re_xxx...

# Admin
ADMIN_SECRET_KEY=your-secret-key
```

Next.js Configuration

next.config.mjs:

```js
const nextConfig = {
    images: {
        remotePatterns: [
            { protocol: "https", hostname: "**" }
        ]
    }
}
```

Deployment Checklist

1. ✅ Push to GitHub repository
2. ✅ Connect to Vercel
3. ✅ Set environment variables in Vercel
4. ✅ Run SQL migrations in Supabase

5. ✅ Verify RLS policies are active
6. ✅ Test payment webhooks

Build Commands

```
npm install          # Install dependencies
npm run dev          # Development server
npm run build        # Production build
npm run start        # Start production server
```

Appendix: File Quick Reference

Key Files to Know

Purpose	File
Homepage	src/app/page.tsx
Root Layout	src/app/layout.tsx
Global Styles	src/app/globals.css
Database Types	src/types/database.ts
Constants	src/lib/constants.ts
Search Page	src/app/(marketplace)/search/page.tsx
Product Card	src/components/marketplace/product-card.tsx
Header	src/components/shared/header.tsx

Document generated by Antigravity AI Assistant