YourBarrio Website Report (Non-Developer Friendly)

Table of Contents

Executive Summary

YourBarrio is a Next.js App Router web app backed by Supabase for auth, database (Postgres + RLS), realtime messaging, and storage. It supports three visible personas: Public visitors, Customers, and Businesses. There is no admin UI in this repo, but there are admin-only API routes (e.g., geocoding). Content for some banners comes from Strapi (CMS). Maps and address lookups use Mapbox or Google Geocoding if configured.

The app uses both server-side and client-side data fetching. Some pages read directly from Supabase in the browser; others go through API routes. This mixed model makes auth cookies and RLS policies the most sensitive parts of the system.

Architecture & Main Entry Points

App Router Structure (Core Shells)

- Root layout: app/layout.js
- Public shell: app/(public)/layout.js
- Customer shell (role-guarded): app/(customer)/layout.js
- Business shell (role-guarded): app/(business)/layout.js
- Account shell (role-guarded): app/account/layout.js
- Orders shell (customer header): app/orders/layout.js

Auth & Session Controls

- Middleware gatekeeping: middleware.js
- Server role enforcement: lib/auth/server.js
- Client auth state: components/AuthProvider.jsx
- Client fallback redirect: components/auth/AuthRedirectGuard.jsx
- Cookie logic: lib/authCookies.js
- Supabase server client (cookies): lib/supabaseServer.js
- Supabase browser client: lib/supabaseClient.js
- Supabase service role (server-only): lib/supabase/server.js
- Public (no-cookie) server client: lib/supabasePublicServer.js

API / Route Handlers (Server)

All route handlers live in app/api/*. Key ones:

- Auth: app/api/auth/*, app/api/logout/route.js
- Listings: app/api/home-listings/route.js, app/api/search/route.js
- Cart / Orders: app/api/cart/route.js, app/api/orders/route.js, app/api/business/orders/route.js
- Messaging: app/api/customer/*, app/api/business/*
- Business analytics: app/api/business/dashboard/route.js, app/api/business/views/route.js
- Admin geocode: app/api/admin/geocode-missing/route.js

Shared UI / Components
- Global headers/nav: components/nav/GlobalHeader.jsx, components/navbars/*
- Auth modals: components/modals/*
- Business profile system: components/business/profile/*
- Public business profile: components/publicBusinessProfile/*
- Cart system: components/cart/CartProvider.jsx
- Messaging UI: components/messages/*

Configuration / Environment
- Next config: next.config.mjs
- Tailwind: tailwind.config.js
- Env values: .env.local (not inspected for secrets)
- Strapi integration: lib/strapi.ts, components/banners/StrapiBannersServer.tsx

Website Map (by Persona)

Where to update: Route files live in app/.../page.js(x) and shared UI in compone
nts/...

Public (No Login Required)

Route | Purpose | Who can access | Data read/write | Code location
/ | Marketing landing | Anyone | Reads Strapi banner; no DB write | app/(public)
/(marketing)/page.js, components/HomeBanner.tsx, components/marketing/HomePageCl
ient.jsx
/about | Marketing about | Anyone | No DB | app/(public)/(marketing)/about/page.
js
/privacy | Privacy policy | Anyone | No DB | app/(public)/(marketing)/privacy/pa
ge.js
/terms | Terms | Anyone | No DB | app/(public)/(marketing)/terms/page.js
/business | Business marketing | Anyone | No DB | app/(public)/business/page.js
/business/about | Business about | Anyone | No DB | app/(public)/business/about/
page.js
/business/login | Business login landing | Anyone | No DB | app/(public)/busines
s/login/page.js
/business-auth/login | Business login | Anyone | Supabase Auth | app/(public)/bu
siness-auth/login/page.js, components/business-auth/BusinessLoginClient.jsx
/business-auth/register | Business registration | Anyone | Supabase Auth + users
 table | app/(public)/business-auth/register/page.js, components/business-auth/B
usinessRegisterClient.jsx
/listings | Public listings | Anyone | Reads listings | app/(public)/listings/pa
ge.js
/listings/[id] | Listing detail | Anyone (extra features for logged-in) | Reads
listings, users; writes saved_listings, conversations, messages, carts | app/(pu
blic)/listings/[id]/page.js
/b/[id] | Public business profile | Anyone | Reads users, listings, business_rev
iews, business_gallery_photos, business_announcements; writes business_views | a
pp/(public)/(marketing)/b/[id]/page.jsx
/profile | Basic profile editor | Logged-in users | Writes users.full_name | app
/(public)/profile/page.js
/oauth/callback | OAuth bridge | Auth callback | No DB | app/(public)/oauth/call
back/route.js

Customer (Logged-in “customer” role)

Route | Purpose | Who can access | Data read/write | Code location
/customer/home | Customer home feed | Customers | Reads via /api/home-listings |
 app/(customer)/customer/home/page.js, app/api/home-listings/route.js
/customer/nearby | Nearby businesses map | Customers | Reads /api/public-busines

ses, /api/reverse-geocode | app/(customer)/customer/nearby/page.js, app/api/public-businesses/route.js
/customer/listings/[id] | Listing detail (customer shell) | Customers | Same as public listing | app/(customer)/customer/listings/[id]/page.js
/customer/saved | Saved listings | Customers | Reads saved_listings + listings | app/(customer)/customer/saved/page.js, app/(customer)/customer/saved/CustomerSavedClient.jsx
/customer/messages | Inbox | Customers | Reads conversations | app/(customer)/customer/messages/page.js
/customer/messages/[conversationId] | Message thread | Customers | Reads/writes messages; realtime | app/(customer)/customer/messages/[conversationId]/page.js
/customer/settings | Account settings | Customers | Updates users + avatar storage | app/(customer)/customer/settings/page.js
/customer/onboarding | Customer onboarding | Customers | Placeholder | app/(customer)/customer/onboarding/page.js
/customer/about | Customer info page | Customers | Static | app/(customer)/customer/about/page.js
/customer/b/[id] | Business profile in customer shell | Customers | Same as /b/[id] | app/(customer)/customer/b/[id]/page.js

Customer Operations (Outside /customer, still customer persona)

Route | Purpose | Who can access | Data read/write | Code location
/cart | Cart management | Any logged-in user | Reads/writes carts, cart_items | app/cart/page.js, app/api/cart/route.js
/checkout | Checkout (submit order) | Any logged-in user | Creates orders, order_items; updates carts | app/checkout/page.js, app/api/orders/route.js
/orders/[order_number] | Order receipt | Customers | Reads orders, order_items | app/orders/[order_number]/page.js
/account/orders | Active orders | Customers | Reads orders | app/account/orders/page.js
/account/purchase-history | Completed orders | Customers | Reads orders | app/account/purchase-history/page.js

Business (Logged-in “business” role)

Route | Purpose | Who can access | Data read/write | Code location
/business/dashboard | Business analytics | Businesses | Reads orders, business_views, listings | app/(business)/business/dashboard/page.tsx, app/api/business/dashboard/route.js
/business/listings | Listings list | Businesses | Reads listings | app/(business)/business/listings/page.js, app/api/business/listings/route.js
/business/listings/new | Create listing | Businesses | Writes listings; uploads to listing-photos | app/(business)/business/listings/new/page.jsx
/business/listings/[id]/edit | Edit listing | Businesses | Updates listings; uploads to listing-photos | app/(business)/business/listings/[id]/edit/page.js
/business/orders | Order management | Businesses | Reads/updates orders; writes notifications, messages; updates inventory | app/(business)/business/orders/page.js, app/api/business/orders/route.js
/business/messages | Inbox | Businesses | Reads conversations | app/(business)/business/messages/page.js
/business/messages/[conversationId] | Message thread | Businesses | Reads/writes messages; realtime | app/(business)/business/messages/[conversationId]/page.js
/business/profile | Public profile editor | Businesses | Updates users; uses business_gallery_photos, business_reviews, business_announcements | app/(business)/business/profile/page.js, components/business/profile/*
/business/settings | Account settings | Businesses | Updates users; uploads to business-photos | app/(business)/business/settings/page.js
/business/onboarding | Business onboarding | Businesses | Updates users + mapbox lookup | app/(business)/business/onboarding/page.js
/business/preview | Preview public profile | Businesses | Reads profile + public profile data | app/(business)/business/preview/page.js

Admin / Debug / Internal

Route | Purpose | Access | Data | Code location
/debug/crashlog | Crash logging view | Internal use | Crash logs | app/debug/crashlog/page.js
/api/admin/geocode-missing | Admin geocode batch | Token-protected | Updates users/businesses coords | app/api/admin/geocode-missing/route.js

Authentication & Authorization

How Login / Logout Works (Plain English)
- Customer login happens in modals (components/modals/CustomerLoginModal.jsx, components/modals/CustomerSignupModal.jsx) and uses Supabase Auth (email/password or OAuth).
- Business login happens on /business-auth/login and /business-auth/register and uses Supabase Auth (components/business-auth/*).
- OAuth callbacks go to /oauth/callback -> /api/auth/callback, which exchanges the code, fetches the user role, then redirects to the correct dashboard (lib/auth/redirects.js).
- Logout clears Supabase cookies and local storage via /api/auth/logout and /api/logout.

Where Sessions Live
- Supabase sessions are stored in cookies named like sb-<project>-auth-token plus standard Supabase cookies.
- Some auth flow state is stored in localStorage and sessionStorage (e.g., business auth redirect handoff, auth guard diagnostics).

Where Session State Is Checked
- Server side: lib/auth/server.js (used in layouts).
- Middleware: middleware.js (checks for auth cookies on /customer/*, /business/*, /api/*).
- Client side: components/AuthProvider.jsx + components/auth/AuthRedirectGuard.jsx.

Role Checks (Customer vs Business)
- Primary role source: public.users.role.
- Fallback: user.app_metadata.role from Supabase.
- Server layout enforcement: requireRole("customer") or requireRole("business").

Common pitfall: A user can be authenticated but still get redirected if their users.role is missing or mismatched. The server guard in lib/auth/server.js will send them to the opposite dashboard.

Known Failure Modes (Observed in Code)
- Navbar changes but page doesn’t redirect: client auth state updates but route not reloaded. Fallback guard does router.replace and then hard redirect.
  Files: components/auth/AuthRedirectGuard.jsx, components/AuthProvider.jsx
- Login “stuck” until refresh: cookie persistence or Safari storage issues; there are refresh token backoffs.
  Files: components/modals/CustomerLoginModal.jsx, components/modals/CustomerSignupModal.jsx, lib/supabaseClient.js
- Redirect loops after token refresh: middleware and client guards both redirect if auth cookies are invalid or refresh_token_already_used.
  Files: middleware.js, lib/supabase/middleware.js, app/api/auth/refresh/route.js

High risk change: Editing middleware.js, lib/auth/server.js, or components/AuthProvider.jsx can lock out users or cause redirect loops.

Data Model & Supabase

Key Tables (from migrations)
- users: all users (customer or business) + profile details
- listings: products/services posted by businesses
- saved_listings: customer saves
- businesses: legacy/secondary business table
- conversations, messages: messaging
- business_views: profile analytics
- carts, cart_items: shopping cart
- orders, order_items: customer orders
- vendor_members: users linked to vendors
- notifications: order status + events
- inventory_jobs: inventory sync queue

RLS (Row Level Security) Summary — Plain English
- Listings: anyone can read; only owning business can insert/update/delete.
- Saved listings: only logged-in customer can insert/read their saves.
- Conversations/messages: only participants can read; customers can start; participants can send; recipients can mark read.
- Business views: anyone can insert; only business can read its own views.
- Carts/cart items: only logged-in customer can read/write.
- Orders/order items: customers can read/create own orders; vendors can read/update if member.
- Vendor members: owners can self-assign; members can read memberships.
- Notifications: only recipient can read/update.

RLS policies live in supabase/migrations/*.sql.

Common pitfall: There is no DELETE policy for saved_listings in migrations, but the UI deletes saved listings. This can cause delete failures when RLS is enforced.

RLS Debug Checklist (Non-Technical)
1. Confirm the user is logged in (auth cookie exists).
2. Check the user role (customer vs business).
3. Verify the row is tied to the same user_id or business_id.
4. If an insert/update fails, verify a policy exists for that action.
5. If a server API works but the client doesn't, it's likely RLS (service role bypasses RLS).
6. If the table is missing in migrations, it may not exist in production.
7. Check Supabase logs for permission denied or RLS errors.
8. For storage errors, confirm bucket permissions and file size limits.
9. If data appears in one place but not another, confirm the query isn't filtering by role or city.

Storage & Uploads

Buckets Observed in Code
- listing-photos: listing images
- business-photos: business avatars/cover images
- business-gallery: business gallery images
- avatars: customer profile photos

Upload Flow
- Uploads are direct from the browser using Supabase Storage.
- Files are saved to public buckets and read via public URLs.

- No signed URL flow is used.

Timeouts / Size Limits
- Listing photo upload has a 20s timeout.
- uploadPublicImage enforces 8MB max + image type whitelist.

Common pitfall: Upload succeeded but image doesn’t show usually means the stored URL is a bucket path, not a full public URL.

Operational Flows (Step-by-Step)

1) Create Listing (Business)
User action: Business clicks “Create listing”
Page: /business/listings/new
DB changes: Upload photos to listing-photos, insert row into listings
Policies: Listings insert allowed only for business user
Files to change:
- app/(business)/business/listings/new/page.jsx
- supabase/migrations/20251205204210_init.sql
- lib/storageUpload.js

2) Publish Listing
There is no separate publish flag in this repo; publish = insert listing.

3) Customer Browses Listings
User action: Visits /customer/home or /listings
Page: /customer/home (uses /api/home-listings)
DB changes: Reads listings
Files to change:
- app/(customer)/customer/home/CustomerHomeClient.jsx
- app/api/home-listings/route.js
- app/(public)/listings/page.js

4) Save Listing
User action: Clicks “Save” on listing
Page: /listings/[id] or /customer/listings/[id]
DB changes: Insert/delete saved_listings
Policies: Insert allowed for own user; DELETE policy not in migrations
Files to change:
- app/(public)/listings/[id]/page.js
- app/(customer)/customer/saved/CustomerSavedClient.jsx
- supabase/migrations/20251205204210_init.sql

5) Leave a Review
User action: Review form on business profile
Page: /b/[id] or /customer/b/[id]
DB changes: Insert/update/delete business_reviews
Policies: Not defined in migrations (unknown)
Files to change:
- components/publicBusinessProfile/BusinessReviewsPanel.jsx
- components/business/profile/ReviewsPanel.jsx
- app/(public)/(marketing)/b/[id]/page.jsx

6) Create Order (Delivery / Pickup)
User action: Add to cart -> checkout
Pages: /cart -> /checkout -> /orders/[order_number]
DB changes: carts/cart_items update; orders + order_items insert; carts.status s
et to submitted; notifications/messages created
Policies: Customers can create orders; vendors can read/update orders
Files to change:

- app/api/cart/route.js
- components/cart/CartProvider.jsx
- app/checkout/page.js
- app/api/orders/route.js
- app/api/business/orders/route.js

7) Messaging Flow
User action: “Message business” from listing or order update
Pages: /customer/messages/*, /business/messages/*
DB changes: RPC get_or_create_conversation; insert messages; trigger updates con
versations counters
Policies: Only participants can read/send
Files to change:
- lib/messages.ts
- app/(customer)/customer/messages/[conversationId]/page.js
- app/(business)/business/messages/[conversationId]/page.js
- supabase/migrations/20251207090000_add_messaging.sql

Where to Make Updates

Where to update: Use the file paths below directly. These are safe entry points
for non-dev changes (copy/labels) vs risky changes (auth/RLS).

Marketing / Copy (Safe)
- Homepage hero text and CTA: components/marketing/HomePageClient.jsx
- Homepage banner (CMS): components/HomeBanner.tsx + Strapi content
- Business marketing page: app/(public)/business/page.js
- Public about/terms/privacy: app/(public)/(marketing)/about/page.js, app/(publi
c)/(marketing)/terms/page.js, app/(public)/(marketing)/privacy/page.js

Navigation / Links
- Public/global header: components/nav/GlobalHeader.jsx
- Business navbar: components/navbars/BusinessNavbar.jsx
- Customer navbar: components/navbars/CustomerNavbar.jsx

Listings & Fields
- Listing form fields (create): app/(business)/business/listings/new/page.jsx
- Listing form fields (edit): app/(business)/business/listings/[id]/edit/page.js
- Listing display cards: app/(public)/listings/page.js, components/publicBusines
sProfile/BusinessListingsGrid.jsx

Categories
- Category list: lib/businessCategories.js
- Category filters: app/(customer)/customer/nearby/NearbyBusinessesClient.jsx, a
pp/(public)/listings/page.js

Reviews (Display Name or Rules)
- Display name format: components/publicBusinessProfile/BusinessReviewsPanel.jsx
 (function formatReviewer)
- Business reply handling: components/business/profile/ReviewsPanel.jsx

Orders & Status Labels
- Status labels/descriptions: lib/orders.js
- Business order workflow: app/api/business/orders/route.js
- Badge rendering: components/orders/OrderStatusBadge.jsx
- DB enum values: supabase/migrations/20260109120000_add_cart_and_orders.sql, su
pabase/migrations/20260109140000_add_vendor_members_and_notifications.sql

Business Onboarding Copy / Fields
- Business onboarding form & copy: app/(business)/business/onboarding/page.js

- Address lookup behavior: same file + NEXT_PUBLIC_MAPBOX_TOKEN env var

High risk change: Anything in supabase/migrations/*.sql, middleware.js, or lib/auth/server.js can break login or block access.

Admin/Operator Playbook

Troubleshooting Without Coding
- Login issues: Check cookies + refresh token errors in browser console.
- Data missing: Check if the user is logged in and has correct role.
- Saved listings not deleting: Likely missing RLS delete policy.
- Messages not appearing: Ensure conversation exists and RLS allows access.

Where to Check Logs
- Browser console: Auth and messaging diagnostics; look for [AUTH_DIAG], [MSG_DIAG].
- Server logs: Next.js server output; watch for API errors.
- Supabase logs: Auth logs, database logs (RLS denies), storage logs.

RLS vs Frontend Bug Signals
- RLS issue: API returns 401/403; data visible with service role but not client.
- Frontend issue: No network request fired, or response is OK but UI doesn't update.
- Session issue: Auth cookies missing or refresh_token_already_used.

Glossary (Plain English)
- Session: proof that a user is logged in (stored in cookies).
- RLS: Supabase rules that decide who can read/write rows.
- Bucket: a folder in Supabase Storage for images/files.
- Route: a page URL in the app (e.g., /customer/home).
- Route Handler: backend API endpoint under /api/*.

Open Questions / Assumptions
1. Reviews, galleries, announcements tables (business_reviews, business_gallery_photos, business_announcements) are referenced in code but not present in supabase/migrations. Are these in production?
2. Several user profile columns are referenced (cover_photo_url, hours_json, social_links_json, publish flags) but not in migrations. Confirm schema source of truth.
3. The IDE shows 20260111120000_allow_notification_deletes.sql but it is not in this repo. Was it deleted or moved?
4. Public routes like /cart, /checkout, /profile are not guarded by middleware. Is this intentional?

10-Item Checklist
1. Understand the role guard flow (middleware.js, lib/auth/server.js).
2. Review Supabase RLS policies in supabase/migrations/*.sql.
3. Confirm users.role values for customer vs business.
4. Verify storage buckets exist (listing-photos, business-photos, business-gallery, avatars).
5. Check the listing creation flow (app/(business)/business/listings/new/page.jsx).
6. Confirm order flow (app/api/cart/route.js -> app/api/orders/route.js).
7. Confirm business order updates and inventory decrement (app/api/business/orders/route.js).
8. Verify messaging RPC and policies (supabase/migrations/20251207090000_add_mes

saging.sql).
9. Keep Strapi banner integration working (lib/strapi.ts, components/HomeBanner.tsx).
10. Document any missing tables/columns that exist in production but not in migrations.