iHeartStreet - Technical Due-Diligence Dossier

Prepared for first-pass engineering diligence. Describes the codebase as it exists at /iheartstreet-mobile as of 2026-08-09. No forward-looking claims are presented as shipped unless the code exists; the 3D roadmap section is explicitly labeled as roadmap.

▶1. Stack Summary

LAYER	CHOICE	VERSION
Runtime	React Native (New Architecture enabled)	0.81.4
Framework	Expo SDK (managed workflow, no ejected ios//android/ dirs)	~54.0.12
Language	TypeScript, strict-enough config	~5.9.2
React	React 19	19.1.0
Navigation	React Navigation 7 (native-stack + bottom-tabs)	7.x
Maps	react-native-maps	1.20.1
Backend client	@supabase/supabase-js	2.112.2
Camera	expo-camera (CameraView)	17.0.8
Fonts	@expo-google-fonts (VT323, JetBrains Mono)	0.4.x

Type health: npx tsc --noEmit exits 0 on the current tree. Verified during this audit, not taken from docs.

Codebase size: small and legible. 7 screens, ~14 fx components, ~13 detail components, one ~1,400-line seed dataset, one 200-line data facade. A competent RN engineer can hold the whole app in their head in a day.

Dependency caveat (see §8): package.json carries several dependencies with zero live imports (Firebase messaging, vision-camera, react-hook-form, date-fns, two extra image-picker/geolocation libs). Bloat, not breakage.

▶2. Architecture

NAVIGATION GRAPH

App.tsx builds a two-level graph, fully typed via navigation/types.ts (RootStackParamList, MainTabParamList):

```
NativeStack (headerless)
├── Main → BottomTabs: Home | Art | Music | Food | Map
├── Detail  { category: 'art'|'music'|'food', id }   (card presentation)
└── Capture { id }                                   (fullScreenModal)
```

One Detail screen serves all three categories, branching on the category param. A dev-only AutoTour rig (navigation/devTour.ts) can auto-cycle screens for hands-free demo recording; all its flags are committed false and gated behind __DEV__.

THEME TOKEN SYSTEM + FX LIBRARY

theme/tokens.ts is the single source of visual truth: a T object holding the phosphor palette (paper #070a07, accent #4de26a, amber, red), per-category accent/line pairs (T.category.art|music|food), font names, spacing/radius/duration scales, and a glow() text-shadow helper. It also exports the React Navigation theme and useCRTFonts() (VT323 + JetBrains Mono via expo-font; the app blocks render until loaded).

components/fx/ is the CRT effects library: CRTBoot, Scanlines, GlowText, GlitchTitle, TypeIn, TraceLine, Flicker, BlinkCursor, PressableCRT, PixelHeart, plus useReducedMotion and an sfx sound helper (expo-av). components/fx/README-USAGE.md codifies the contract every screen follows: content wrapped in <CRTBoot>, <Scanlines/> mounted last on top, accents pulled from T.category.* and never hard-coded.

components/detail/ holds the category-page building blocks: PhotoFrame/primitives, ArtGallery, TimeSlider, CommissionSection, TipJarSection, FoodOrder, FeedbackSection, ClaimSection, SocialRow, ShareCard (view-shot + expo-sharing), CRTSheet, TerminalInput.

DATA LAYER: SEED-VS-SUPABASE FACADE

data/records.ts is the app's single data door and the most important file for a diligence read:

- getRecords(category) returns the built-in San Francisco seed (data/seed.ts, ~12 entries per category with full vendor-economy metadata) and, **when Supabase is configured, appends DB rows** to it. DB rows carry category-specific fields in a meta jsonb column; mapper functions (toArtSpot/toMusicSpot/toFoodSpot) coerce them into the exact seed shapes with safe defaults for every field.
- **Graceful degradation is total:** if env vars are absent, lib/supabase.ts exports a null client and supabaseConfigured=false; if any fetch throws, the facade console.warns and serves seed only. The app cannot crash from backend failure — it demotes to a fully functional offline demo. All four data screens (ArtScreen, MusicScreen, FoodScreen, MapScreen) consume only this facade.
- lib/supabase.ts runs anon-only (persistSession: false) — deliberate for the no-auth v1, keeps AsyncStorage out of the loop.

RUNTIME TIMELINE STORE

data/timeline.ts is a ~24-line in-memory pub/sub store: addCapture(recordId, capture), getCaptures(recordId), subscribeCaptures(fn). New field captures append to a wall's timeline for the current session without a refetch; DetailScreen subscribes and re-renders, merging artEntry.history ?? [] with runtime captures before feeding TimeSlider. Session-scoped by design — durable persistence goes through the Supabase upload path in parallel.

▶3. The Capture Pipeline (ALIGN + CAPTURE)

screens/CaptureScreen.tsx is the app's signature instrument: split-screen rephotography for street-art walls. Top half is a reference view of the wall; bottom half is the live camera; the user aligns the wall across the seam and fires.

Reference-layer resolution chain:

1. If EXPO_PUBLIC_GOOGLE_MAPS_API_KEY is set, the reference is a Google **Street View Static API** URL (size=640x480, fov=70, keyed on the record's lat/lng).
2. On image load failure (Image onError — e.g., key missing billing, no coverage), it falls back to the **newest archive capture** (entry.history[last].capture, else entry.imageUrl). The fallback is a one-line state swap; no retry loop, no spinner logic to break.
3. With no key at all, it starts directly on the archive frame. The reticle tag honestly labels the source: ▸ STREET VIEW vs ▸ ARCHIVE — LATEST TAPE.

Simulator mock feed: expo-device's Device.isDevice gates the live half. On simulator (no camera hardware) the "live feed" is the archive image scaled 1.18× and shifted 14px — the alignment interaction stays demoable end-to-end, labeled ▸ LIVE FEED — SIM.

Ghost overlay: a toggle renders the reference image over the live feed at 35% opacity — the actual alignment aid, standard rephotography technique.

On shutter:

1. `takePictureAsync({ quality: 0.7 })` on device (archive URI on simulator; capture errors are swallowed and fall back to archive — see §8).
2. `addCapture()` appends `{ year: <current>, note: 'aligned field capture', capture: uri }` to the runtime timeline — the TimeSlider gains a notch immediately.
3. If Supabase is configured, `uploadCapture()` fires **best-effort in the background** (`.catch(() => {})`): local URI → `fetch` → `ArrayBuffer` → upload to the `captures` storage bucket at `art/<timestamp>.<ext>` → `getPublicUrl` → insert a `records` row pointing at it.
4. A locked-state confirmation types in, then `navigation.replace` back to the Detail screen, where the new capture is already on the slider.

The design choice worth noting for diligence: **local UX never waits on the network.** Persistence is fire-and-forget; the session store is the source of immediacy. The cost is that a failed upload is silent (§8).

▶4. Data Model (`sql/schema.sql`)

One paste-and-run file targeting the Supabase SQL editor. Five tables, all `uuid` PKs via `gen_random_uuid()`:

- **`records`** — the spots. `category` check-constrained to `art|music|food`; plain `lat`/`lng` double-precision columns; category-specific fields (artist/medium/year, genre/schedule, cuisine/menu/hours…) in a `meta` jsonb column; `created_by` is a free-text `@handle` (no auth FK yet); `claimed`/`claimed_by` denormalized flags.
- **`reviews`** — FK → records (cascade), `rating` check 1–5, free-text author handle.
- **`claims`** — FK → records, claimant handle, `proof` text, status check `pending|approved|rejected`.
- **`tips`** — FK → records, `amount_cents > 0`, method check `stripe|venmo|cashapp`.
- **`orders`** — FK → records, line items as jsonb `[{item, price, qty}]`, status check `placed|accepted|ready|picked_up|cancelled`.

Geo posture: PostGIS is deliberately **not** used. "Nearby" is a bounding-box scan backed by a composite `(category, lat, lng)` index — honest and adequate at v1 scale, with the upgrade path (PostGIS + geography column) documented in the file header.

RLS posture — deliberately permissive v1. RLS is *enabled* on all five tables. Policies granted: public `SELECT` and anon `INSERT` on `records` and `reviews` only; `claims`/`tips`/`orders` have RLS on with **no policies** — effectively locked until auth lands. Storage: a public-read `captures` bucket with anon upload policy.

This is a field-test posture and the schema says so in-file, with an explicit hardening TODO: swap inserts `to anon` → `to authenticated`, add owner-scoped UPDATE/DELETE keyed on `auth.uid()`, rate-limit/moderation-scan inserts, and lock claims/tips/orders end-to-end. **What v2 hardening means concretely:** today, anyone with the shipped anon key can insert arbitrary rows into `records`/`reviews` and upload arbitrary files to `captures` (no size cap, no content scan, no rate limit), and *nothing* can be updated or deleted through the API at all (no policies exist). Acceptable for a closed TestFlight cohort; a launch blocker otherwise. The remediation is small (policy swaps + Supabase auth wiring, blueprint phase P1, estimated 15–25 dev-days including accounts and moderation) and is already the documented plan, not an afterthought.

Note also `docs/PRODUCT_BLUEPRINT.md` §3 specifies an eight-drawer model (adds `users`, `commissions`, `media`); the shipped schema implements the five drawers v1 actually needs.

▶5. Design System

Single token file (theme/tokens.ts, §2) — no scattered hex values; screens pull T.colors.* and T.category.*. Category accents are fixed and enforced by convention + the fx README: **art = green #4de26a, music = amber #ffb000, food = red #ff5c57**, each with a matching low-alpha line color. Typography is two-font: VT323 for display, JetBrains Mono for labels/body, with an 11px-uppercase-letterspaced "microlabel" recipe used everywhere.

Accessibility is real work, not a checkbox:

- **Reduced motion:** useReducedMotion (components/fx) subscribes to the OS reduceMotionChanged event. TimeSlider consults it and skips the flash/jitter CRT swap effects entirely, doing an instant image swap instead.
- **VoiceOver:** TimeSlider exposes the scrubber as accessibilityRole="adjustable" with accessibilityValue and increment/decrement accessibilityActions; decorative layers (flash overlay, track line) are hidden via accessibilityElementsHidden / importantForAccessibility="no-hide-descendants". CaptureScreen labels every control in plain English ("Cancel capture", "Toggle ghost overlay" with accessibilityState.selected, "Capture aligned photo") and its image feeds carry descriptive labels.
- **Touch targets:** 44pt minimums appear explicitly in styles (minHeight: 44 on back/ghost buttons, TimeSlider notch cells).
- docs/UX_STANDARDS.md documents remaining a11y debt honestly (contrast audit of graphite-on-paper microlabels; global reduce-motion coverage for CRTBoot/GlitchTitle is listed as TODO — currently only TimeSlider demonstrably honors it).

docs/UX_STANDARDS.md also records *why* each category page is shaped as it is (DoorDash menu-first food pages, Bandsintown next-show music pages, gallery-style art pages) plus an explicit anti-dark-pattern policy. Unusual for a project this size, and it materially de-risks handoff.

▶6. 3D / Splat Roadmap (not in the repo - roadmap only)

Nothing splat-related exists in the codebase today. No code, no deps, no doc mentions. What follows is the proposed direction, stated so diligence can price it as future work, not discount it as vaporware:

- **Single-photo Gaussian splats via apple/ml-sharp.** Apple's open-source SHARP model generates a 3D Gaussian splat from one photograph in seconds — CLI shape is sharp predict -i <image> -o <output>, with a ~2.8 GB model checkpoint. This maps directly onto iHeartStreet's existing asset base: every archive capture is a candidate splat, meaning the back catalog becomes 3D without re-shooting.
- **Native rendering path:** RealityKit 5 ships GaussianSplatResource on iOS 26/27-era SDKs, giving first-party splat rendering without a custom Metal renderer. The proposed **SPLAT CAPTURE** feature extends the ALIGN + CAPTURE instrument: fire the shutter, run/queue SHARP server-side (the 2.8 GB checkpoint rules out on-device inference for now), store the splat next to the photo in the captures bucket, render a swipeable 3D view on the Detail page.
- **Known constraint:** Apple Maps' Flyover-style splats have **no public API** — there is no shortcut to city-scale 3D from Apple's data. Any area-scale ambition means generating and hosting splats ourselves.
- Honest sizing: this is a new server-side inference component (GPU or Apple-silicon queue worker), a storage/format decision (.ply/.splat/.sogs), and an iOS-only rendering surface. It is the roadmap's most differentiating item and its largest unbuilt system.

▶7. Ship Readiness (TestFlight)

Present and working:

- eas.json with development / preview (internal distribution) / production (autoIncrement) build profiles and a submit.production stanza.
- app.json fully configured: bundle id com.ncoded.iheartstreet (iOS + Android), icons/splash assets, portrait, new-arch enabled.
- Secrets are .env-based (EXPO_PUBLIC_SUPABASE_URL/ANON_KEY, optional EXPO_PUBLIC_GOOGLE_MAPS_API_KEY), with .env.example checked in and plain .env in .gitignore. No secrets in source (lib/supabase.ts and config/api.ts read env only). Note EXPO_PUBLIC vars are baked into the JS bundle — fine for the Supabase anon key (that's its design), but the Maps key ships in the binary and should be HTTP-referrer/bundle-restricted in Google Cloud.
- Managed workflow with no generated native dirs → EAS cloud builds need no local Xcode state.

Remaining before TestFlight (all logistics, no engineering unknowns):

1. Apple Developer Program enrollment ($99/yr) and app-store listing assets.
2. eas build --platform ios → eas submit (profiles already defined).
3. Enable **Street View Static API billing** on the Google Cloud project, else the reference layer permanently falls back to archive frames (app degrades gracefully; it just loses the marquee comparison).
4. **Key rotation is pending:** treat the current Supabase anon key and any Maps key in local .env files as exposed-in-dev and rotate/restrict before external distribution.
5. iOS permission-string copy (camera/location Info.plist strings) should be reviewed in app.json before submission — Expo will inject defaults, Apple review prefers explicit ones.

▶8. Honest Technical Debt

1. **v1 RLS permissiveness** (§4). Anon-writable records/reviews and anon-uploadable public storage bucket, no rate limiting, no content moderation, no UPDATE/DELETE path. Documented, deliberate, and the top pre-launch item.
2. **Placeholder imagery.** Seed data generates photos via loremflickr URLs (data/seed.ts); the DB fallback placeholder and schema sample rows use picsum.photos. Fine for demos; real captures must replace these before the app is shown as "populated," and both hosts are third-party dependencies in the render path.
3. **Simulator mock camera.** The capture pipeline's live half is simulated off-device; the real expo-camera path has had less exercise than the demo path. On-device capture errors are silently swallowed (empty catch) and fall back to the archive frame — a real shutter failure looks like success.
4. **Silent background upload.** uploadCapture().catch(() => {}) means a failed persistence write gives the user a success screen. Acceptable trade for demo fluidity; needs a retry queue or at least a status indicator before field testers rely on it.
5. **No automated tests.** Zero test files, no test runner configured, no CI. tsc is the only gate (it passes). The data facade and jsonb mappers in data/records.ts are the highest-value first test targets.
6. **Dead code and stale docs.** An orphaned src/ scaffold (Oct 2025, one legacy AppNavigator.js), a legacy axios REST client for a never-built backend (services/api.ts + config/api.ts, self-labeled legacy, includes a hard-coded LAN IP), a .backup-pre-crt-20260809/ snapshot directory, and COMPLETION_STATUS.md, which describes a long-superseded pre-CRT version of the app (claims 60% complete with placeholder screens — the current tree is far past it). All should be deleted or archived to avoid misleading an incoming engineer.
7. **Unused dependencies.** @react-native-firebase/app/messaging, react-native-vision-camera, react-hook-form, date-fns, react-native-geolocation-service, react-native-image-picker, expo-image-picker, react-native-linear-gradient, react-native-vector-icons have no live imports. Prune before a native build — the Firebase pods in particular will otherwise drag config requirements into EAS builds.
8. **Single-founder bus factor — mitigated by documentation.** One author, no team. The mitigation is real: docs/PRODUCT_BLUEPRINT.md (backend plan, costs, phased roadmap, claims/payments/trust-and-safety policy), docs/UX_STANDARDS.md (design rationale), docs/SUPABASE_SETUP.md (10-step reproducible backend runbook), components/fx/README-USAGE.md (design-system contract), and in-file architecture comments throughout. An experienced RN engineer could take over this codebase from the docs alone.

Bottom line: a small, coherent, type-clean Expo codebase with an unusually distinctive design system, one genuinely novel interaction (rephotography + time slider), a deliberately thin but honest backend, and debt that is documented rather than hidden. The gap between "demo" and "field-testable product" is auth + RLS hardening + upload feedback — weeks, not months. The gap between that and the 3D vision is a new server-side inference system — priced accordingly.