# HomeListingAI System Audit (Engineering Deep Dive)

Date: 2026-02-21
Scope: Backend/API wiring, funnel execution logic, lead-to-funnel flow, email analytics flow, voice call pipeline, route/page bloat risks, and reliability gaps.
Requested by: Product owner ("do not remove anything now; produce actionable audit + recommendations").

---

## 1) Executive Summary

The app has strong feature depth, but core reliability is being reduced by **architecture drift** and **duplicate wiring paths**.

Top 3 issues to solve first:

1. **Funnel system is split across multiple engines/data models** (inconsistent execution + analytics behavior).
2. **Server route duplication and monolith growth** (hard to reason about, easy to regress).
3. **Schema/code mismatch in production logs** (scheduler + funnel analytics warnings show DB drift).

Good news: the core plumbing for lead import, enrollment, and voice call initiation exists and can be stabilized quickly with focused cleanup.

---

## 2) Current State (What Exists Today)

### 2.1 Core code footprint

- Backend monolith file: `backend/server.cjs` (~14,978 LOC).
- Frontend app shell: `src/App.tsx` (~1,618 LOC).
- Route handlers in `server.cjs`: ~244 (`get/post/put/delete/patch`).
- Funnel-related engines/services found:
  - `backend/services/FunnelEngine.js`
  - `backend/services/funnelExecutionService.js`
  - Additional ad-hoc funnel trigger logic directly in `backend/server.cjs`

### 2.2 Lead → Funnel → First-touch flow (as currently wired)

There are **multiple entry points** for lead creation/import, each with different funnel logic:

- `/api/admin/leads/import` inserts leads, then inserts `funnel_enrollments` and relies on scheduler loop.
- `/api/admin/leads` creates lead and calls `funnelEngine.assignFunnel(...)` (different execution model).
- `/api/leads/public` also calls funnel assignment logic.

This means behavior differs depending on which endpoint creates the lead.

### 2.3 Voice (Telnyx ↔ Hume)

Voice path is implemented and reachable. Recent issue root cause was config mismatch (`c6cd...` vs `6cd6...`).

---

# 3) Key Findings (Prioritized)

# P0 (High Risk / immediate)

### P0-1: Funnel architecture is fragmented (multiple models)

Evidence: - `backend/services/FunnelEngine.js` uses `lead_funnel_progress` + `funnel_steps`. - `backend/services/funnelExecutionService.js` uses `funnel_enrollments` + `funnels.steps` JSON. - `/api/funnels/:userId/:funnelType` writes to `funnel_steps` and explicitly marks `funnels.steps` as deprecated (`backend/server.cjs` around line 770).

Impact: - The funnel a user edits is not guaranteed to be the funnel the scheduler executes. - Metrics and progression can look inconsistent across pages/endpoints.

---

### P0-2: Duplicate route definitions in the same server file

Evidence in `backend/server.cjs`: - `/api/track/email/open/:messageId` appears at lines ~1981, ~4852, ~9974. - `/api/track/email/click/:messageId` appears at lines ~2028, ~4917, ~10011. - `/api/leads/stats` appears twice (~8129 and ~8382). - `/api/email/settings/:userId` PATCH appears twice (~13878 and ~13997). - `/api/listings` POST appears twice (~5829 and ~12901).

Impact: - Last registered handler wins, earlier handlers become misleading/dead. - Hotfixes can appear “applied” but hit different handler in runtime.

---

### P0-3: Missing module reference in active code path

Evidence: - `backend/server.cjs` around line ~4289 calls `require('./services/funnelService')`, but that file does not exist in `backend/services`.

Impact: - Runtime failures when that branch executes. - Hidden breakage risk during chatbot/lead auto-enroll branch.

---

## P1 (Important reliability)

### P1-1: Schema drift errors are visible in runtime logs

Observed errors: - `column email_tracking_events.status does not exist` - `column agents.metadata does not exist`

Likely sources: - Funnel analytics fallback query selecting `status` from `email_tracking_events` (`backend/server.cjs` ~650). - Trial scheduler selecting/updating `agents.metadata` (`backend/services/schedulerService.js` ~84 onward).

Impact: - Analytics not updating cleanly. - Scheduler features silently degraded.

---

### P1-2: `/api/funnels/assign` likely incorrect argument contract

Evidence: - `funnelService.enrollLead(user.id, leadId, funnelType)` called in `backend/server.cjs` (~1278). - `funnelExecutionService.enrollLead(...)` expects a **funnel ID** and queries `funnels` by `id`.

Impact: - Enrollment route can fail or behave unpredictably if passed a funnel key/type instead of UUID.

---

### P1-3: Backend depends on `VITE_` env variables

Evidence examples: - Mixed usage of `VITE_*` vars in backend runtime logic.

Impact: - Environment confusion between frontend build-time and backend runtime. - Harder production ops/debugging.

---

## P2 (Performance/maintainability)

### P2-1: App shell and route model includes legacy/dev leftovers

Evidence: - `src/types.ts` `View` union contains duplicates (`'payments'`, `'checkout'`) and many dev/test routes.

Impact: - Increased cognitive overhead. - Harder to enforce stable routing contract.

---

### P2-2: Repo contains tracked artifacts/log/history files

Tracked files include: - `server_history.txt` - `logs/server.out` - `logs/vite.out` - `debug_leads_request.log` (and backend variant)

Impact: - Repository noise and accidental merge conflicts. - Larger clone/build contexts and harder code review.

---

## 4) End-to-End Wiring Check (Your Example)

Target path: **New lead → assigned funnel → first notification → analytics update**

Current status:

1. **Lead ingest**: Implemented via multiple endpoints (works, but inconsistent path selection).
2. **Funnel assignment/enrollment**: Implemented, but split between two execution models.
3. **First notification**: Implemented (email/sms/call step execution exists), timing differs by endpoint/scheduler path.
4. **Analytics update**: Implemented in concept (mailgun + tracking + fallback query), but schema mismatch currently causes partial failure/warnings.

Conclusion: - Wiring exists but is **not fully coherent** yet across all entry points. It needs consolidation into one canonical flow.

---

## 5) Recommended Plan (No destructive changes first)

## Phase A — Stabilize Production Logic (1–2 days)

1. Define one canonical funnel execution engine (recommend `funnel_enrollments` + `funnelExecutionService`).
2. Add adapter layer so all lead-creation paths call same enrollment function.
3. Remove route collisions by introducing a centralized route registry (without deleting old code yet, just stop mounting duplicates).
4. Add startup validation checks for required tables/columns and fail-fast warnings.

Deliverable: deterministic lead→funnel behavior with clear logs.

---

## Phase B — Schema Contract Alignment (1 day)

1. Add migration(s) for missing production columns used by code (`email_tracking_events.status`, `agents.metadata`) OR update code to current schema.
2. Create DB contract document for analytics/tracking tables.
3. Add `/api/health/schema` endpoint to assert required columns/indexes.

Deliverable: analytics and scheduler stop throwing schema errors.

---

## Phase C — Performance + Maintainability (2–4 days)

1. Split `backend/server.cjs` into domain routers (`voice`, `funnels`, `leads`, `analytics`, `admin`).
2. Keep compatibility wrapper routes, then deprecate safely after traffic check.
3. Move repo artifacts/log outputs to ignored runtime directories and stop tracking them.
4. Trim legacy route/view unions and test-only pages behind feature flags.

Deliverable: lower regression risk and faster iteration velocity.

---

## 6) Concrete Suggestions (What I'd do next)

1. **Single source of truth for funnel steps**:
   - Persist and execute from one schema only (recommend `funnel_steps` + `funnel_enrollments` with explicit joins), or fully commit to JSON `funnels.steps` and remove step table writes.
2. **Canonical "Lead Created" pipeline function**:
   - Create one service method: `createLeadAndEnroll({source, userId, lead, funnelKey})` and call it from every endpoint.
3. **Analytics event normalization**:
   - Ensure all email events pass through one function that maps status/open/click/reply consistently.
4. **Voice safety checks**:
   - Validate Hume config ID existence at startup and before dialing.
   - Add log marker per call with stage transitions and outcome code.
5. **Operational guardrails**:
   - Add an internal admin page that shows: queue depth, due enrollments, failed steps, last 50 voice call outcomes.

---

## 7) What I Did During This Audit

- Verified call bot table wiring/API behavior after migration.
- Confirmed live bot config ID can be updated and retrieved.
- Isolated previous no-audio root cause to invalid Hume config ID and env duplication pattern.

- Audited route and funnel wiring for duplication/contract mismatches.

---

## 8) Immediate Action Checklist (Practical)

1. Keep only one value for `VOICE_PHASE1_FOLLOWUP_CONFIG_ID` and use the verified valid ID.
2. Confirm `VOICE_PUBLIC_BASE_URL` and `PUBLIC_URL` point to Render backend URL.
3. Prioritize Phase A item #1 (choose one funnel engine and align all lead entry points).
4. Fix schema drift (Phase B) to restore analytics/scheduler confidence.
5. Freeze feature additions for 48 hours while this stabilization lands.

---

## 9) Final Recommendation

Do **not** migrate providers yet (Hume → Retell) until this architecture cleanup is done. The current blockers are mostly **wiring and schema consistency**, not provider capability. Once flow is unified, you can evaluate providers with objective A/B metrics instead of debugging infrastructure noise.