ARIA Hotline Implementation Guide (Redacted)

ARIA HOTLINE

Implementation guide

Engineering reference for the four ARIA systems and the path to production

Document version 1.0 Owner [Client Entity] d/b/a ARIA

Audience Engineering — internal hires, contractors, or the founder six months from now

This copy has been redacted for publication. The client's legal entity name and identifying details have been removed with client permission. Technical content is unmodified.

Date May 7, 2026

This guide describes what exists in the four ARIA systems today, what is mocked vs. real, what patterns must not drift, and what needs to become real before paying customers go live. It is deliberately shorter than a typical 80-page implementation reference because most of the production backend does not yet exist. Writing fiction about a backend that has not been designed creates more harm than help. When the backend decisions are made, this document grows.

How to read this guide

Five sections, in priority order:

- **§1 What exists today.** The four files, their relationships, the data shapes, the design tokens. Start here on day one.
- **§2 The legal-safety substrate.** The patterns the codebase enforces (or pretends to enforce) to keep ARIA defensible. These cannot drift. Read this before changing anything user-facing.
- **§3 The production gap.** What is mocked today and needs to become real. Backend, database, voice integration, AI integration, deployment.
- **§4 Patterns and conventions.** The coding patterns established across the four files. Match these when adding new features.
- **§5 Open architectural decisions.** Decisions deferred from prior sessions that the next developer should not accidentally rediscover.

Conventions used in this document

- **Yellow callouts** = attention required, not an emergency
- **Red callouts** = legal-safety invariant, do not break

- **Green callouts** = confirmed correct as of this writing, useful pattern to copy
- Code shown in monospace font on a light background.

1. What exists today

The ARIA system today is four standalone HTML files. Each is a single-page application: HTML + CSS + vanilla JavaScript in one file, no build step, no backend calls, all data mocked in inline JS arrays. They are demos in the literal sense — buyers can see what the product does without any backend running. They are also accurate to the architectural shape of the eventual real product, which is the load-bearing purpose of this approach: nothing in these files implies a capability the real product cannot deliver.

1.1 The four files

File	Audience	Lines	Purpose
aria-hotline.html Marketing site	Public — buyers, browsers, prospects	3,572	Multi-page SPA: home, platform, use cases, industries, pricing, about, contact. Cookie consent, chat launcher, Cal.com integration.
aria-reporter.html Reporter Portal	Public — reporters submitting reports	4,277	White-labeled intake. Three locales. Anonymous submission, two-token follow-up, evidence upload.
aria-admin.html Client Admin Portal	Customer reviewers — compliance officers, HR, legal	5,458	Case workspace: queue, case detail, SLA tracking, audit log, attestations, reports, exports, categories, routing, severity, templates, users, escalation, notifications, branding, retention, integrations, billing, plan.
aria-ops.html Internal Operations Portal	ARIA staff only — never customer-facing	2,893	Control plane: dashboard, customers, onboarding,

File	Audience	Lines	Purpose
			escalations, billing, quality review, accuracy & hallucination, case integrity, signal monitor, prompt & model, availability, support & issues, audit & access, settings.

Total: ~16,200 lines, ~1.05 MB. All single-file, no dependencies beyond what they fetch from CDN at runtime (Cal.com embed script in marketing, no external scripts in the others).

INVARIANT — the four-file structure Do not split these files into a build system without preserving the single-file deployability. Each file is meant to be openable directly in a browser without any tooling — that is a load-bearing property for buyer demos, security reviews, and procurement walkthroughs. When you add a real backend, the rendered output should still be a single file shipped to the browser per page.

1.2 How the four systems relate to each other

A reporter submits a report through the Reporter Portal. ARIA's intake AI conducts the conversation, produces a recording, transcript, and structured AI summary, and creates a case in the customer's tenant. The customer's reviewer opens the case in the Client Admin Portal, where they confirm or override the AI-suggested category and severity, route the case per their configured rules, and work the case to resolution. ARIA staff watch the system from the Operations Portal — uptime, accuracy, case integrity, customer health — but they are content-blind by default and do not see case content unless the customer explicitly escalates a case for support.

The marketing site is a separate concern. It does not connect to the other three. It exists to convert buyers into customers; once converted, customers move to a provisioned reporter+admin instance.

Critical relationship: tenant isolation

Each customer is a tenant. Each tenant has its own Reporter Portal (white-labeled to the customer's branding) and its own Admin Portal workspace. Tenants do not see each other's data, ever. ARIA staff see tenant boundaries from the Ops Portal but cannot read inside a tenant's case content unless the customer explicitly escalates.

In the demo files this is faked — there is one tenant ("Sunrise Senior Health Group" in the admin portal, multiple tenants in the ops portal). In production, tenant isolation must be enforced at the database layer (row-level security on every tenant-scoped table) and at the application layer (every query must filter by tenant_id; never rely on application logic alone).

1.3 The data model embedded in the demos

The data fixtures in the demos are not arbitrary — they encode the actual data model the production system needs to support. The next developer should treat these as the source of truth for schema design until counsel and product confirm a final ERD.

Core entities

- **Tenant** — a customer organization. Has tier (basic / voice / managed / enterprise), industry, branding configuration, retention rules, routing matrix, and a primary contact. See **TENANTS** in aria-ops.html line 397.
- **User** — a person with login credentials scoped to a tenant. Has role (compliance officer, HR, legal, etc.) and seat type. See **USERS** in aria-admin.html line 1290.
- **Case** — the central entity. One case per report. Has reference number, anonymous flag, category, severity (AI-suggested + reviewer-confirmed), state (one of seven canonical states), timestamps, assignee, audit log, and attached artifacts (recording, transcript, AI summary, evidence files). See **CASES** in aria-admin.html line 919 and aria-ops.html line 494.
- **Reporter** — either anonymous (no record beyond the access tokens) or identified (name, contact info). For anonymous reporters, ARIA does not capture caller-ID, IP, or any identifier that could re-identify them.
- **Reviewer** — the customer's user assigned to work a case. Different from "investigator" — the latter is reserved for external third parties referred by ARIA in the Managed tier. Display strings should always say "reviewer" for the customer's internal staff.
- **Activity / Audit Event** — append-only log entry on a case. Captures who, when, what action, what changed. See **ACTIVITY** in aria-admin.html line 1064 and **AUDIT_EVENTS** line 1184.

Case state machine

Seven canonical case states. The state ID is preserved in the database; the display label can vary by locale and audience.

State ID	Reporter sees	Admin sees	Meaning
received	Received	Received	Report submitted; awaiting triage.
acknowledged	Acknowledged	Acknowledged	A reviewer has accepted the case and started looking at it.
assessment	Acknowledged	Assessment	Reviewer is assessing severity and routing. Reporter sees this collapsed under Acknowledged.

State ID	Reporter sees	Admin sees	Meaning
investigation	Under investigation *should be: Under review*	Investigation	Reviewer is actively working the case. See open item in §5.
resolved	Resolved	Resolved	Case work complete; outcome recorded.
closed	Closed	Closed	Case is final and archived; no further actions.
withdrawn	Withdrawn	Withdrawn	Reporter withdrew the report or the customer determined no action needed.

Note: "investigation" is the schema name for what is now externally called "review." The schema name is intentionally preserved for backward compatibility with existing customer integrations and database migrations. See §5 for the open work item.

Three artifacts per voice case

Every case originating from a voice call produces three distinct artifacts on the case record:

- **Call audio recording —** call_recording_*.wav. Retained per the customer's category-level audioDays retention rule.
- **AI-generated transcript —** transcript_*.txt. Retained per the customer's category-level transcriptDays retention rule.
- **Structured AI summary —** displayed in the reviewer workspace with the AI-suggested category and severity. Reviewer confirms or overrides on every case.

The retention rules are per-category, not per-tier. A customer can configure that "HIPAA · Privacy" audio is retained for 365 days but "General inquiry" audio is retained for 30 days. Production implementation must enforce these via scheduled deletion jobs that run nightly.

Tier matrix

The four tiers are defined in aria-admin.html line 838. Tier inheritance is linear:

// aria-admin.html line 838

var TIERS = [

{ id:"basic", name:"Basic", caseCap:50, seats:3, voiceMins:0, storageGb:5,

```
features:
["web_intake","categories_universal","branding_basic","email_alerts",

          "audit_basic","exports_basic","users_basic"] },
```

{ id:“voice”, name:“Voice”, caseCap:150, seats:10, voiceMins:1500, storageGb:50,

```
inherits:"basic",

features:
["voice_intake","sms_alerts","multi_locale","categories_industry",

          "branding_full","webhooks","templates","reports","tasks"] },
```

{ id:“managed”, name:“Managed”, caseCap:150, seats:10, voiceMins:1500, storageGb:50,

```
inherits:"voice",

features:["external_investigator_access"] },
```

{ id:“enterprise”, name:“Enterprise”, caseCap:-1, seats:-1, voiceMins:5000, storageGb:500,

```
inherits:"managed",

features:
["sso_saml","rbac_custom","pagerduty","ai_insights","risk_register",


"priority_support","routing_full","workflows_custom","sla_custom","sla
ck_teams",

          "attestations","heat_map","exports_full","api_access"] }
```

];

No price field exists in TIERS — pricing is contractual (order form), not displayed in-product. Do not add a price field; the customer-side decision was deliberate. See the changelog edit 13 for context.

> **INVARIANT — Managed inherits Voice; Enterprise inherits Managed** Enterprise customers automatically have external_investigator_access (the Managed feature flag). This is intentional — Enterprise customers can also grant guest investigator access to their tenant. The contractual program-specialist relationship that defines Managed is not a code feature flag; it is in the order form. Do not move external_investigator_access out of Managed without coordinated marketing and contract changes.

1.4 Operations Portal modules

Fourteen modules visible in the Ops Portal sidebar. Each is a discrete view; some are read-only monitoring, others have admin actions.

Module	Purpose	State
Dashboard	Health overview across all customers	Built; pulls from other modules’ aggregate data
Customers	Tenant directory; tier,	Built; uses TENANTS fixture

Module	Purpose	State
	industry, contact, status	
Onboarding	New tenant provisioning workflow	UI built; backend automation does not exist
Escalations	Cases customers escalated to ARIA for support	Built. Critical for content-blind invariant — see §2
Billing & Usage**	Customer billing, usage metrics, invoices	UI built; no Stripe / billing engine wired
Quality Review	Cases flagged by the system or reviewers for ARIA quality review	Built; flag inputs are mocked
Accuracy & Hallucination**	AI classification accuracy vs. customer-reviewer corrections	Built. Hallucination rate = customer-corrected / total. See §2.4 for naming.
Case Integrity	Audit log integrity checks (chain of custody, tamper detection)	Built; checks are simulated
Signal Monitor	Anomaly detection on intake patterns (volume spikes, etc.)	Built; anomaly inputs mocked
Prompt & Model**	AI prompt versions and rollout management	Built; no live prompt-deployment integration
Availability	Per-service uptime, incidents, downtime minutes	Built. Source data for future status.ariahotline.com page
Support & Issues**	Customer support tickets	UI built; not wired to any ticketing backend
Audit & Access**	ARIA staff access log — who accessed what tenant data, when, why	Built. Critical for legal-safety; see §2.5
Settings	Internal ARIA configuration	UI scaffolded

1.5 Brand tokens — preserve these

Color palette, typography, and key spacing tokens used across all four files. Match these when adding new components.

Color palette

Token	Hex	Usage	Notes
–navy	#0B1B32	Primary brand	Headlines, primary CTA, dark surfaces, brand shield
–teal	#07B68F	Accent / signal	Active states, success, AI-affirmative actions, brand

Token	Hex	Usage	Notes
			gradient stop
–gold	#C99A3F (gold) / #E0A83E (gold-soft)	Tier name labels, Enterprise tier	Sparingly used — reserved for premium signals
Brand gradient	#07B68F → #1688C4	Brand shield, avatars, hero accents	Teal → blue, 135° angle — this is the canonical ARIA gradient
–surface	#FFFFFF	Card / page background	White; never tinted
–surface-2	#F4F1EA	Paper / soft section background	Distinguishes alternating sections without harsh contrast

Typography

- **Display / headings:** DM Sans — modern, geometric, clean. Used for all headings, buttons, and high-emphasis text.
- **Body:** DM Sans — same family, lighter weights. The site uses one type family deliberately to keep brand voice consistent.
- **Eyebrow / label / monospace:** JetBrains Mono — for tier names, section eyebrows, code-like labels, and the "engineered" feel.
- **Optional serif accent:** Fraunces — used selectively in some hero treatments. Not universal; pull only when intentional.

All four files use the same brand system. When you build a new feature in one file, copy the existing CSS variables — do not re-derive colors from memory.

2. The legal-safety substrate

ARIA's legal defensibility rests on a small set of invariants. The four files enforce these, more or less, via the patterns described below. The next developer should treat these as load-bearing — breaking any of them invalidates the marketing site's published claims, the contractual representations made to customers, and (in some cases) the regulatory posture for whole industries. None of this can drift.

2.1 AI outputs are advisory; the customer is the decision-maker

ARIA's AI suggests; the customer's reviewer decides. The suggestion is never authoritative. This invariant appears in three places across the codebase:

- **Reporter Portal:** AI suggestions are never shown to reporters at all. Reporters submit; AI processes; the case file is built; reviewers see the suggestions later. Reporters never see "AI thinks this is harassment" — that would imply ARIA made a determination.
- **Admin Portal:** Every AI-suggested field shows the suggestion alongside an override action. The accompanying label says "AI-suggested" explicitly. Confirmation is required — the case cannot move forward in the workflow until the reviewer either confirms or overrides.
- **Marketing site:** Phrasing throughout: "your team confirms or overrides on every case," "your reviewer is the decision-maker." Never "ARIA decides" or "ARIA categorizes."

INVARIANT — never auto-confirm AI Do not add a setting that auto-confirms AI suggestions. Even at Enterprise. Even with a customer requesting it. The legal-safety posture rests on the customer reviewer being the decision-maker on every case. An auto-confirm feature would create regulatory exposure that no contract clause can fully insulate.

2.2 ARIA does not conduct investigations

This is the most consequential disclaimer on the marketing site. The Managed tier provides referrals to external third-party investigators, and gives the customer admin a feature to grant tenant access to those investigators. ARIA does not employ investigators. ARIA does not direct investigations. ARIA is not a party to the customer's engagement with their investigator.

Three implementation consequences:

- **No "investigator" as an ARIA staff role.** ARIA staff are content-blind by default and only access tenant data on explicit customer escalation. They are not investigators. Internal job titles like "Customer Success Specialist" or "Program Specialist" are appropriate; "Investigator" is not.
- **The external_investigator_access feature flag (Managed)** lets a customer admin grant a guest user (the third-party investigator) tenant access. The customer admin grants and revokes; ARIA does not. This is a workflow feature, not an ARIA-runs-investigations feature.
- **The phrase "licensed external investigators" was deliberately removed** from the marketing site at customer direction. Use "external third-party investigators" in any future copy. The IP-licensing clause ("owned by or licensed to [Client Entity]" in the Terms modal) is a different sense of the word and stays.

2.3 Customer is data controller; ARIA is data processor

Each tenant's data belongs to the customer organization. ARIA processes the data under the customer's instructions. This framing matters for GDPR, HIPAA, and most data-protection regimes.

Production implementation requirements that follow from this framing:

- **Tenant isolation enforced at the database layer** via row-level security (PostgreSQL RLS, Supabase policies, etc.). Every tenant-scoped table has a tenant_id column and a policy that filters all queries by the authenticated user's tenant_id. Application logic must never bypass this.
- **Customer-driven retention rules** per category. Scheduled deletion jobs honor the customer's audioDays and transcriptDays settings. ARIA does not retain data beyond the customer's configuration unless legal process requires it.
- **Customer-driven export rights** are contractual and must be operationally executable. The Admin Portal must surface a "Export all" feature for tenant admins. On contract termination, the customer receives a complete export of every case and audit log entry in a documented format.
- **Subprocessor disclosure** is contractual (no longer published publicly per customer direction in edit 11). When you add or change a subprocessor (Twilio, Anthropic, OpenAI, Cloudflare, Supabase, etc.), notify customers per the DPA.

2.4 ARIA staff are content-blind by default

This is the single strongest claim ARIA makes for procurement reviewers and a real architectural constraint. ARIA staff (the people using the Operations Portal) cannot read tenant case content unless the customer explicitly escalates a specific case for support, with documented escalation reason and time-bounded scope.

Production implementation:

- **ARIA staff users are not in any tenant's RBAC.** They have access to tenant-metadata tables (tier, billing, usage) but not to case-content tables (cases, transcripts, evidence) under the default RLS policies.
- **An escalation** creates a temporary access grant: the customer admin clicks "Escalate to ARIA support", picks a reason, and the system creates an access record (tenant_id, ARIA staff user_id, scope, expires_at). RLS policies allow ARIA staff to read case content only when an active escalation exists for that specific case.
- **Every ARIA staff access** is logged in the customer's tenant audit log AND in the Ops Portal Audit & Access module. Bidirectional logging — the customer sees who from ARIA accessed their data; ARIA can audit its own staff.
- **Time bounds default to 7 days** and require customer admin re-authorization to extend. Standing access is not a feature.

INVARIANT — content-blind by default Do not add a "god mode" for ARIA staff. Do not add a setting that lets ARIA staff browse tenant case content without an active escalation. The content-blind invariant is what justifies ARIA's positioning as infrastructure rather than a service provider — a critical distinction for HIPAA, attorney-client privilege, and procurement reviews. Once broken, it is very expensive to re-establish.

2.5 Append-only audit, with integrity verification

Every case has an audit log of activity events: who did what, when, what changed. The log is append-only in the application (no UI to delete or edit entries) and append-only at the database layer (RLS policy blocks UPDATE and DELETE on the audit_events table for non-system roles).

The Ops Portal's Case Integrity module performs nightly integrity checks: hash chain verification, timestamp monotonicity, foreign-key consistency. Any anomaly is flagged for ARIA staff review and surfaces in the customer's tenant audit log if confirmed.

2.6 Marketing claims must match implemented capability

After 18 edit groups of remediation, the marketing site does not claim a single capability that the production system does not implement (or has not committed to implementing). When you add a feature to the product, only then add it to the marketing site. When you remove a feature, remove it from marketing in the same release cycle.

The reverse is also true: when marketing copy changes, the product must reflect that change before the copy goes live. Counsel reviews drift between marketing and product; drift is the most common source of regulatory exposure for SaaS companies in regulated industries.

2.7 Naming hygiene — internal vs. external

Internal terms can be technical. External-facing terms must be neutral. The Ops Portal "Accuracy & Hallucination" module is fine for ARIA staff ("hallucination" is the standard ML term). If any future feature surfaces this metric to customers, rename to "override rate" or "reviewer correction rate." Same number, neutral framing.

Other naming dichotomies to preserve:

- **Internal "investigation" state ID** → external "Under review" display string (when the §5 fix is applied)
- **Internal "investigators" variable name** → external "Reviewers" display label
- **Internal hallucination rate** → external override rate (if surfaced)
- **Internal severity score** → external "AI-suggested severity" label

3. The production gap

The four HTML files describe what the product does. They do not implement it. This section is honest about what is mocked, what needs to become real, and the recommended path of least resistance to first paying customer.

3.1 What is mocked

Mocked	What it pretends to do today
Voice intake	The marketing site demo plays a scripted call

Mocked	What it pretends to do today
	transcript. The reporter portal shows a 'voice line: 1-800-…' page but the number is illustrative. Real voice intake requires Twilio + Vapi (or equivalent) + Anthropic Claude for the conversation handling.
AI processing	All AI-suggested category, severity, and summary outputs are hard-coded in the demo fixtures. Real implementation: Anthropic API for conversation, transcription, and classification. OpenAI API may be used as an alternative or fallback for severity scoring.
Case persistence	All case data is in-memory JS arrays (CASES, ACTIVITY, MATTERS, etc.). No database. Submitting a report through the Reporter Portal demo does not actually create a case anywhere persistent.
Authentication	Hard-coded TENANT and user objects. No real login. The Admin Portal renders as if 'Maria Reyes, Compliance Officer' is logged in — that's a fixture, not a session.
Tenant isolation	One tenant in the demo. Production requires PostgreSQL row-level security on every tenant-scoped table.
File storage	Recordings, transcripts, evidence files are referenced as filenames (call_recording_apr26.wav, transcript_apr26.txt) but no actual file storage backend. Production: S3 / Cloudflare R2 / similar with per-tenant prefix.
Email / SMS delivery	Notifications module describes channels (email, SMS, Slack, PagerDuty) but no actual delivery. Production: Resend or SendGrid for email, Twilio for SMS, Slack/PagerDuty webhooks.
Billing	Plan/billing UI shows mocked invoice records. Production: Stripe Billing.
Status page	Internal Availability module exists but no public status.ariahotline.com. Production: build from existing Availability data.
Cal.com booking	Embed integration is wired but the CAL_LINK constant is a placeholder. Sign up at cal.com and replace the slug. See

Mocked	What it pretends to do today
	marketing site changelog edit 17.
Live chat	Form-fallback mode currently. Provider integration hook (Crisp / Intercom / Front) commented in the JS. See changelog edit 16.

3.2 Recommended production stack

Stack chosen for fastest path to first paying customer with the least architectural risk. Other stacks are valid; this one is recommended because each component is well-documented, the integrations between them are well-understood, and the operational surface is small enough for a solo founder to manage.

Backend

- **Vercel** for hosting. Next.js application; serverless functions. Free tier is generous enough for early customers; scales smoothly. Edge runtime for the marketing site, Node runtime for the API.
- **Supabase** for database (Postgres) and authentication. Built-in row-level security. Native support for tenant isolation. Has its own object storage (alternative to R2/S3). Realtime subscriptions for live admin-portal updates.
- **Cloudflare R2** for recordings, transcripts, and evidence files. S3-compatible API. Cheaper than S3. No egress fees — important when ARIA staff or customer reviewers stream recordings.

Voice + AI

- **Twilio** for the voice line (DID acquisition, telephony). Already a subprocessor.
- **Vapi or equivalent voice-AI orchestration** to handle the call flow (greet, listen, route to AI agent, hang up). Vapi is the leading option but evaluate alternatives — this layer is moving fast.
- **Anthropic Claude** for the conversation AI (the agent the reporter talks to), the transcript clean-up, and the structured AI summary. Existing subprocessor.
- **OpenAI** as fallback or specifically for category and severity classification. Existing subprocessor. Optional — Claude can do this too.
- **Deepgram** for speech-to-text if Vapi's built-in STT is insufficient for the language quality bar. Existing subprocessor.
- **ElevenLabs** for text-to-speech (the AI agent's voice). Existing subprocessor.

Infrastructure &** operations**

- **Resend** for transactional email (notifications, exports, password resets). Existing subprocessor.

- **Stripe Billing** for invoicing, subscription management, and payment collection. Customer pricing is contractual; Stripe is the operational layer.
- **Sentry** for error tracking. Required for production observability.
- **PostHog or Plausible** for product analytics. Gated behind the Functional cookie consent toggle on the marketing site.
- **Atlassian Statuspage or Better Stack** for the public status page. Pulls from the Ops Portal Availability module.

3.3 Migration path — from HTML demos to production

The four HTML files do not need to be thrown away when production is built. Each becomes a Next.js page tree:

- **aria-hotline.html** → Next.js app at ariahotline.com. The single-file SPA structure becomes Next pages with shared layout. Preserve the brand tokens and component styles verbatim.
- **aria-reporter.html** → Next.js app at reporter.{customer}.ariahotline.com (per-tenant subdomain). Each tenant's branding overrides the defaults; the structure is identical.
- **aria-admin.html** → Next.js app at admin.{customer}.ariahotline.com. Behind authentication. RLS-scoped queries replace the in-memory CASES, ACTIVITY, etc. arrays.
- **aria-ops.html** → Next.js app at ops.ariahotline.com. ARIA staff only. SSO required (Google Workspace or similar).

Migration order — recommended

- **1. Marketing site first.** No backend dependencies. Migrating it to Next.js gives you a real deployment pipeline, a CI/CD setup, and a place to put the eventual API. Cookie consent and Cal.com integration carry over directly.
- **2. Reporter Portal + intake API.** The reporter-facing flow is the highest-stakes surface (consequences of bugs are reporter-facing) but it has the smallest surface area. Build the intake flow against a real Twilio + Vapi + Claude pipeline. Provision one demo tenant. Make the demo real.
- **3. Admin Portal core (cases + workspace).** The case queue, case detail, audit log, AI suggestion confirmation flow. Get to feature parity with the demo.
- **4. Ops Portal core (customers + escalations + audit &** access). **Bare minimum for ARIA staff to provision new customers and respond to escalations.
- **5. Everything else.** Reports, exports, attestations, severity model editor, retention rules, billing — these are real features but secondary to the core case-flow. Build in priority order driven by customer demand.

3.4 Production readiness checklist

Items that must be true before the first paying customer goes live. These map to obligations counsel and procurement reviewers will check.

- Tenant isolation enforced via RLS on every tenant-scoped table
- ARIA staff content-blind by default; escalation flow operational
- Audit log append-only at DB layer; integrity verification scheduled
- Per-category retention rules enforced via scheduled deletion jobs
- Customer data export executable on demand by tenant admin
- Subpoena response policy operative — counsel-ratified, customer-facing exhibit
- DPA template (and BAA where applicable) ratified by counsel
- Anonymous reporting verified — caller-ID stripped, no IP logging, two-token follow-up working
- Imminent-harm flow tested — emergency advisory shown, customer team notified, ARIA does not auto-act
- Public status page live at status.ariahotline.com
- SOC 2 Type I in progress (Type II takes longer; many buyers will accept Type I + roadmap)
- HIPAA compliance documentation if any healthcare customer is in pilot
- Sentry monitoring on; PagerDuty (or equivalent) on for after-hours incidents
- Backup and disaster recovery — tested, documented, RTO and RPO defined

4. Patterns and conventions

The four files share a small set of patterns. Match these when adding new features. Inconsistency between the four files creates friction for the next developer.

4.1 Single-file SPA structure

Every file follows the same skeleton:

<!DOCTYPE html>

ARIA · …

4.2 Click delegation pattern

Instead of attaching individual click handlers, use event delegation on document with data-attribute selectors. This is what the marketing site uses for navigation, Cal.com, cookie consent, and chat:

```
document.addEventListener("click", function(e) {
// Cal.com booking
var calTrigger = e.target.closest("[data-cal-link]");
if (calTrigger) {
var slug = calTrigger.getAttribute("data-cal-link");
if (openCalModal(slug)) { e.preventDefault(); return; }
// fall through to data-nav if Cal unavailable
}
// Page navigation
var navTarget = e.target.closest("[data-nav]");
if (navTarget) {
e.preventDefault();
showPage(navTarget.dataset.nav);
return;
}
// ... other delegated handlers
});
```

4.3 Tier-feature gating (Admin Portal)

Pattern from aria-admin.html. Every gated feature checks the current tier's feature set:

```
function tierFeatureSet(tier) {
var idx = tierIndex(tier && tier.id ? tier.id : tier);
var set = {};
for (var i = 0; i <= idx; i++) {
(TIERS[i].features || []).forEach(function(f) { set[f] = true; });
```

```
}
return set;
}
function featureEnabled(featureId) {
return !!tierFeatureSet(currentTier())[featureId];
}
// Usage in render
if (featureEnabled("ai_insights")) {
html += renderAIInsightsPanel();
} else {
// Show locked tier-upgrade modal
html += renderTierLock("ai_insights");
}
```

*Production should preserve this pattern at the API layer too. Every tenant-scoped endpoint checks the tenant**'s tier features before responding. Don't** trust client-side gating alone — clients can be modified.*

4.4 Localization (Reporter Portal)

All user-facing strings live in a localization map indexed by locale:

```
// aria-reporter.html line 2240+
var STRINGS = {
brandSub: { "en-US": "Reporting", "es-US": "Reporte", "fr-CA": "Signalement" },
checkStatus: { "en-US": "Check status", "es-US": "Verificar estado", "fr-CA": "Vérifier le statut" },
// …
};
function t(key) {
return (STRINGS[key] && STRINGS[key][currentLang]) || STRINGS[key]["en-US"] || key;
}
```

Three production locales: en-US (English), es-US (US Spanish), fr-CA (Canadian French). Adding a locale = adding entries to every key in STRINGS. Do not add user-facing strings outside this map; do not concatenate translated strings.

4.5 No build step, no dependencies (today)

The current files have no npm dependencies, no build step, no bundler. They are openable directly in any browser. This is intentional and load-bearing for buyer demos and security reviews — counsel reviewers can read the source. When you migrate to Next.js, preserve as much of this as possible: minimize external dependencies, keep the bundle size small, and ensure the rendered output is still inspectable.

4.6 Inline SVG over icon fonts

All icons are inline SVG (Lucide-derived). Never use icon fonts (Font Awesome, etc.) — they bring CDN dependencies, accessibility issues, and brand inconsistency. The SVG approach lets the icons inherit color via currentColor and scales without bitmap artifacts.

4.7 Accessibility patterns

- Semantic HTML — buttons are , links are , dialogs have role="dialog"
- aria-label on every interactive element without visible text
- Focus-visible styles distinct from hover styles
- Color contrast meets WCAG 2.1 AA (verify with Stark or similar before deploys)
- Escape closes modals; backdrop click closes modals
- Form validation errors announced via aria-live regions

4.8 JS coding style

- ES5 syntax — var, function, no arrow functions, no destructuring. Compatible with old browsers without transpilation. (Migrate to ES6+ when Next.js arrives — Babel handles compat.)
- No external libraries. No jQuery. No React in the demos. Plain DOM API.
- Functions are organized in render-function clusters per feature: renderDashboard, renderCases, renderAuditLog, etc.
- State lives in a single state object; renderApp() is idempotent.

PATTERN — verify JS validity before any commit Every change to a single-file HTML must pass node –check on the extracted script blocks. The pattern I've used throughout this engagement: extract

---

5. Open architectural decisions

Decisions deferred from prior sessions. Each is documented so the next developer does not accidentally rediscover it. Some are small (terminology cleanup); some are launch-blockers (regulatory positioning).

5.1 Reporter Portal terminology cleanup

Display strings translate to “Reviewer” in three locales but the underlying CSS classes (.message.investigator), schema fields (caseInvestigator, m.from === ‘investigator’), and state translation (stateInvestigation displayed as “Under investigation”) still use the older terminology. This contradicts the marketing FAQ that says ARIA does not conduct investigations. Three-line edit in the translation table resolves it (rename to “Under review” in all three locales). Schema state ID stays as “investigation” for backward compat. Documented as a low-risk cleanup task. Estimated effort: 1 hour.

5.2 Admin Portal Investigators page label drift

The page header at aria-admin.html line 2821 still reads “Investigators” while the sidebar nav label says “Reviewers.” Same terminology drift as 5.1, different file. Estimated effort: 30 minutes.

5.3 Public status page

The Ops Portal Availability module has the data; the public surface does not exist. Procurement reviewers ask for status.ariahotline.com routinely. Recommended: stand up Atlassian Statuspage or Better Stack pulling from the Availability module data once production is live. Estimated effort: 1 day.

5.4 Hallucination rate naming if surfaced externally

If any future feature surfaces the AI accuracy metric to customers (a customer-facing accuracy report, for example), do not use the word “hallucination” externally. Use “override rate” or “reviewer correction rate.” Same number, neutral framing. The internal Ops Portal can keep “hallucination.”

5.5 Counsel-blocking decisions (regulatory positioning)

Items that require counsel ratification before going live with the relevant industry vertical:

- **PSO status under PSQIA.** Whether ARIA operates as a Patient Safety Organization for healthcare customers, or merely processes data on their behalf. Determines liability and disclosure obligations.
- **42 CFR Part 2 compliance** for behavioral health and SUD customers. Substance use disorder records have heightened protections; the architecture supports it but counsel ratification needed before behavioral health pilots.
- **FDA SaMD classification** of the AI severity output. If the AI’s severity suggestion is interpreted as clinical decision support in a healthcare context, FDA’s Software as a Medical Device framework may apply. Likely not — but counsel review is needed.
- **DPA, BAA, and QSOA template language** operative versions ratified by counsel. Marketing references “customer DPA” and “BAA where applicable” — these have to be real, signable documents.

- **Subpoena response policy operative version.** The marketing-site policy modal was removed at customer direction (changelog edit 11). The contractual exhibit needs to exist and be ratified.
- **Twilio caller-ID stripping verification.** Anonymous reporting depends on caller-ID being stripped before storage. Verify in Twilio configuration; document in security questionnaire.

5.6 Brand consolidation question (client entity vs ARIA)

The legal entity is [Client Entity] d/b/a ARIA. Marketing-side [Client Entity] mentions are confined to legal sections (copyright, trademarks, data-controller framing, contact). ARIA is the public-facing brand throughout body copy. The structure works as long as the d/b/a is preserved on contracts and trademark filings. Do not move toward incorporating ARIA as a separate entity without coordinated counsel and trademark advice — that is a much bigger restructure than it sounds.

5.7 Multi-event-type Cal.com expansion

The Cal.com integration today wires every booking button to a single event type ("demo"). The data-cal-link attribute pattern supports per-button override (e.g., data-cal-link="username/deep-dive" for a 60-minute architecture review). When you add a second event type in Cal.com, individual buttons can point to it without changing the global CAL_LINK constant. Documented inline in the JS.

5.8 Live chat provider selection

Chat launcher today runs in form-fallback mode (mailto:). Provider integration hook is documented in the JS for Crisp / Intercom / Front / HelpScout. Crisp's free tier is the most defensible "real chat without paying" option for ARIA's stage. Two-line wire-in once you sign up.

Closing

The ARIA codebase is in good shape. The four files are internally consistent, the legal-safety substrate is enforced via real architectural patterns (not just disclaimers), and the design system is mature enough to expand without re-deriving every choice. The hardest parts of the product story — the legal-defensibility language, the AI-advisory framing, the content-blind invariant, the tenant-isolation model — are already decided and documented.

What's left is implementation. Backend, database, voice integration, AI integration, deployment pipeline, observability. None of these are conceptually hard; they are mechanical work against a clear specification. The order of operations in §3.3 is the recommended path.

When in doubt, read §2 again. The legal-safety invariants are the part that breaks expensively if it drifts. Everything else is fixable.

End of guide.