MAKING YOURSELF **CLEAR**

SECURITY & ARCHITECTURE

Making Yourself Clear — Security & Architecture White Paper

> *Technical companion to* `compliance-one-pager.md`*, intended for the engineer reviewing Making Yourself Clear on behalf of a security or compliance team. Last reviewed:* ***2026-07-08****.*

This document describes Making Yourself Clear's architecture, threat model, and controls in enough detail to support a serious security review. It is meant to be read alongside the one-pager (which summarises the high-level posture) and the data-flow document (`data-flow.md`) (which walks through the transcript pipeline step-by-step with a diagram).

---

1. Scope of this document

Making Yourself Clear is an AI-powered leadership communication coaching platform. Customers upload meeting transcripts; the platform analyses them via large language models and returns structured coaching feedback. The platform is offered as a multi-tenant SaaS (Tier 0), with optional per-organization BYO LLM endpoints (Tier 1), and a self-hosted deployment (Tier 2, generally available).

The product is operated by Executive Sound Board, a small focused team (currently solo-founded). This document describes controls implemented in code and operational practice. Where a control is planned but not yet implemented, it is marked clearly.

---

2. Architecture overview

2.1 Components

Making Yourself Clear is composed of four runtime components, all hosted on Railway (built on Google Cloud Platform), in the US West region.

Component	Stack	Responsibility
Frontend	Next.js 15 (App Router), React 19, TypeScript, Tailwind CSS	UI for coachees, coaches, org_admins, vendor_admins
Backend API	FastAPI (Python), authlib (OAuth), bcrypt (password hashing), itsdangerous (signed sessions)	Authentication, request handling, route gating, audit logging
Worker	Celery (Python), with Redis broker	Background analysis: parses transcripts, runs PII redaction, calls the LLM provider, persists structured results
Database	Railway-managed Postgres	All persistent data: auth, organizations, transcripts, runs, baseline packs, experiments, audit log, system configs

Supporting services:

Service	Purpose
Redis	Celery message broker
Sentry	Error tracking across all five Sentry runtimes (backend, worker, Next.js client / server / edge)
Better Stack	Centralised log forwarding + uptime monitoring
SendGrid	Transactional email (verification, reset, invite)

2.2 Tenancy model

Making Yourself Clear is multi-tenant at the application layer. Each domain row (transcript, run, baseline pack, experiment, attempt event) carries an `organization_id` column. Cross-organization access is blocked at the query layer via `WHERE organization_id = %s` filters on every list endpoint and `_record_in_org` checks on every detail endpoint. Cross-organization probe attempts return `404 Not Found` rather than `403 Forbidden` to prevent ID enumeration.

Individual coachees who are not members of an organization have `organization_id = NULL` on their domain rows; these rows are visible only to the coachee, their assigned coach (if any), and Making Yourself Clear vendor_admins.

2.3 Roles and permissions

Five roles, enforced on every API call:

Role	Sees	Can do
vendor_admin	All data across all organizations	Manage organizations, manage users (cross-tenant), view all audit logs, configure deployment-level defaults, edit vendor IP (prompts, taxonomy, scoring rubric)
org_admin	All data within their organization	Invite / remove members, configure per-org settings (PII redaction overrides, BYO LLM endpoint, Enterprise SSO), see all data within the org. **Never assignable via OIDC** — created via invite, peer-promotion, or first-run wizard only
team_lead	Members of teams they lead (Phase B, 2026-05-29)	Invite / manage users in teams they lead (coachee / coach / team_lead role changes). No settings access. Suitable for line managers who shouldn't be able to rotate the LLM endpoint or change SSO config. Team scoping is enforced at the user-listing, role-change, and audit-log endpoints
coach	Only their assigned coachees	View transcripts, runs, experiments; cannot delete. Can belong to zero, one, or many organizations (multi-org affiliation via the `coach_organizations` link table)
coachee	Only their own data	Upload transcripts, view analyses, accept / park / abandon experiments, delete their own data

Roles are stored in `users_auth.role` with a CHECK constraint. Role transitions are audit-logged. Role-elevation paths (promote-to-coach, invite-accept for elevated roles) are gated on the user having an OAuth credential — email-and-password-only accounts cannot be promoted to elevated roles (coach, team_lead, org_admin, vendor_admin), because elevated roles require IdP-inherited MFA.

OIDC clamp: The org_admin role is deliberately OIDC-unassignable. An org_admin can edit the OIDC config itself, so allowing OIDC claims to promote to org_admin would create a circular trust path (a compromised IdP entry could persistently elevate someone). The `admin_group` OIDC config field maps to `team_lead` instead. To create an org_admin, an existing org_admin (or vendor_admin) must invite or promote them manually inside Making Yourself Clear.

---

3. Threat model

Making Yourself Clear's controls are designed against the following adversaries, in decreasing order of how thoroughly we defend against them:

Threat	Defended?	Notes
Casual web attacker (drive-by SQL injection, XSS, etc.)	✅	Parameterised queries everywhere, React's auto-escaping, CSP planned
Compromised customer account (stolen password / token)	✅ for OAuth-protected roles; ⚠️ for password-only coachees	OAuth roles inherit MFA from IdP. Coachee password-only accounts pending application-level MFA.
Database leak (backup theft, accidental dump)	✅	All customer content (transcripts + the full conversation surface) and all at-rest secrets are application-layer AES-256-GCM encrypted under per-organization keys; on SaaS the root key is held in AWS KMS (adds decrypt audit + key revocation). Cloud-managed volume encryption applies beneath.
Cross-tenant probe (org A trying to read org B)	✅	Row-level scoping + `_record_in_org` checks + 404-on-cross-org
LLM provider data retention	⚠️ standard OpenAI API retention (up to 30 days, abuse-monitoring) applies — no ZDR agreement in effect yet	OpenAI does not train on API data; `store=False` disables OpenAI-side conversation storage. A ZDR agreement is applied-for/pending. BYO LLM routes under the customer's own provider contract.
Sub-processor compromise	⚠️	We monitor sub-processor security postures; cannot fully defend against a downstream breach. Mitigated via short retention + minimal data sent.
Insider threat (rogue Making Yourself Clear employee with DB access)	⚠️ partial	Audit log records admin actions; read-only access not yet audited at app layer; planned for SOC 2
Full application compromise (RCE on the worker)	❌	The worker by definition holds plaintext during analysis. Tier 2 self-host is the only defense for customers needing zero trust in the vendor
Customer-side compromise (their endpoint or browser)	❌	Customer responsibility

The "not defended" rows are honest — they are limits of the default SaaS tier, addressed (where the customer needs them addressed) by Tier 2 self-hosting.

4. Authentication and session management

4.1 Identity providers

Making Yourself Clear supports three authentication methods:

- **Google OAuth 2.0** (via authlib).
- **Microsoft OAuth 2.0** (via authlib).
- **Email + password.** Passwords are hashed with bcrypt (cost factor 12). Email verification is mandatory before login; a fresh signed token is sent via SendGrid on signup.

The OAuth handshake uses authlib's standard PKCE flow. OAuth state is signed with HMAC and includes a CSRF token + the post-login redirect target.

4.2 Sessions

Sessions are signed cookies (itsdangerous, HMAC-SHA256). Cookie attributes:

- `HttpOnly`
- `Secure` (HTTPS only)
- `SameSite=Lax`
- Idle + absolute-cap TTL, DB-configurable (deployment-wide, per-org overridable); defaults ≈ 4h idle / 24h absolute

The cookie payload is the session ID; the session itself lives in the `sessions` Postgres table with `created_at`, `expires_at`, and a foreign key to `users_auth`.

4.3 MFA posture

Users authenticating via Google or Microsoft inherit MFA from the IdP — if the IdP enforces it, Making Yourself Clear gets it for free.

Email + password coachees do not yet have application-level MFA. The asymmetry is intentional:

- **Elevated roles (coach, team_lead, org_admin, vendor_admin)** require an OAuth credential, enforced at promotion time. So all elevated-role users have MFA via their IdP.
- **Coachees** can sign up with email + password and use the platform without MFA. Application-level MFA (TOTP) is on the roadmap; until then, coachees handle higher-risk content (transcript text) but cannot affect organizational settings or other users.

4.4 Invite tokens

When an org_admin invites a member, an invite token is generated (HMAC-signed, 7-day TTL). Tokens are single-use and recorded in the `invite_tokens` table with `used_at` set on acceptance. Email-bound invites (sent via SendGrid) require the recipient to be logged in as the bound email

address before acceptance is allowed. The invite-accept flow re-validates the token against the database on every request to defend against replay.

An invite may carry a **set of roles** (multi-role users, 2026-06-04). At creation, the inviting actor must be authorized to grant **every** role in the set — each member is checked against the same capability matrix used by the role-change endpoint (a team_lead, for example, cannot mint an org_admin). `vendor_admin` can never be combined with another role. On acceptance the invitee is granted the full set additively; if the set contains any elevated role (coach / team_lead / org_admin) the OAuth-credential (MFA) requirement applies before the grant. Multi-role invites are email-bound only — the shareable anyone-with-link path remains single-role.

---

5. Authorization

5.1 Permission checks

Every protected endpoint resolves the current user via the session cookie, then either:

1. **Calls `get_current_user()` dependency** (FastAPI), which loads the user from the session and the database.
2. **Asserts role and (where applicable) organization scope** via helpers like `_require_vendor_admin()` and `_require_org_access(user, org_id)`.

`_require_org_access` allows vendor_admin to access any organization or org_admin to access only their own; anything else returns 403.

5.2 Cross-organization defense

Direct-object-reference attacks (e.g. an org_admin in Org A trying to read `/api/admin/tenant/orgs/B/...`) are blocked by `_require_org_access`. Cross-org probes against specific user IDs use `_get_coachee_in_org` and return 404 (not 403) to prevent enumeration of valid IDs across orgs.

5.3 Coachee-level isolation

Within an organization, a coach sees only the coachees assigned to them (via `users_auth.coach_id`). A coachee sees only their own data. The query layer enforces this — no application-layer post-filter.

5.4 Team-level scoping (Phase B, 2026-05-29)

Phase B adds a `teams` data abstraction. A `team_lead`'s reach is the union of users across teams they lead — tighter than the prior org-wide scope.

- **User listing** (`GET /api/admin/users`): for `team_lead` actors, results filter to users in teams the actor leads (plus the actor themselves), via `teams_db.list_user_ids_in_teams_led_by`.

- **Role-change endpoint** (`PATCH /api/admin/users/{id}/role`): a `team_lead` actor can only change roles for users in a team they lead, and only within `{team_lead, coach, coachee}` (no admin promotions).
- **Audit log** (`GET /api/admin/system/audit`): for `team_lead` actors, results filter to rows whose actor or target user is in a team they lead, within the actor's organization.

Stale `team_memberships` rows (e.g. after a user is demoted out of a team-eligible role) do not bleed permissions: `teams_db.list_teams_led_by_user` and its callers join on `users_auth.role = 'team_lead'`, so the role check is the gate. Cleanup of memberships on role change is for UI consistency, not permission-bleed prevention. Cleanup fires only on transitions to non-team-eligible roles (e.g. `team_lead → coach`); `team_lead ↔ coachee` preserves memberships because both roles can legitimately participate in teams.

5.5 Multi-org coach affiliation (Phase 4 #1 + Phase B consumption)

A coach can belong to zero, one, or many organizations. The `coach_organizations` link table (shipped in Phase 4 #1, May 2026) is the source of truth for which orgs a coach is affiliated with. Phase B work consumes this table consistently across the role-change, invite, and member-listing paths. A coach's `users_auth.organization_id` is the "primary" org (the one they were originally invited into); the link table holds all current affiliations.

5.6 vendor_admin promotion paths (Members-polish, 2026-05-30)

vendor_admin can be created via three audited paths:

1. **Email invite** — the vendor_admin invitee receives a /invite link, signs in via OAuth (MFA-required), and accepts. The accept handler clears `organization_id` (vendor_admin is cross-org). Email-only — there is **no shareable-link path for vendor_admin** because anyone-with-link → vendor_admin is unsafe.
2. **PATCH role endpoint** — an existing vendor_admin can promote any user via `PATCH /api/admin/users/{id}/role` with `new_role='vendor_admin'`. Audited.
3. **First-run wizard** — bootstraps the very first admin during install (the wizard creates an `org_admin` bound to a wizard-created org; promotion to vendor_admin happens after via path 2).

---

6. Encryption

6.1 In transit

- **Client ↔ Making Yourself Clear:** TLS 1.2+ enforced by Railway's edge.
- **Making Yourself Clear ↔ LLM providers:** HTTPS via the OpenAI / Anthropic SDKs, certificate-validated.

- **Making Yourself Clear ↔ Sub-processors (SendGrid, Sentry, Better Stack):** HTTPS via their SDKs.
- **Internal service-to-service** (backend ↔ worker via Redis, backend ↔ Postgres): currently within Railway's private network; not TLS-encrypted on the private network as of this writing. Planned upgrade.

6.2 At rest

- **Customer content — application-layer AES-256-GCM.** Raw and redacted transcript text, the reversible token mapping, analysis output, and baseline summaries are encrypted on write with AES-256-GCM under a **per-organization data-encryption key (DEK)**, at the single DataStore write chokepoint (`backend/core/postgres_datastore.py`, `backend/core/org_data_keys.py`). Decrypt-on-read is always-on and gated by the same authorization checks; a stray plaintext row (e.g. an old-backup restore) still reads. Encrypt-on-write is unconditional — there is no runtime toggle to disable it.
- **At-rest secrets — application-layer AES-256-GCM.** BYO LLM API keys, OIDC client secrets, the vendor OAuth / LLM / Stripe / GitHub-App secrets, the box content key, and the box private key are encrypted with AES-256-GCM (versioned, `kid`-tagged `sec:v1:` envelope, HKDF-derived key) under the deployment root key `APP_ENCRYPTION_KEY` (`backend/auth/llm_crypto.py`).
- **KMS-rooted root key (SaaS).** On the vendor SaaS the `APP_ENCRYPTION_KEY` root is held in **AWS KMS** (`APP_ENCRYPTION_KEK_PROVIDER=aws-kms`, with an enforced encryption context), so every DEK / secret unwrap goes through a KMS Decrypt — adding decrypt **auditability** and **key revocation / crypto-shred**. Deleting an organization's DEK renders that organization's content unrecoverable.
- **Underlying storage + backups:** cloud-managed AES-256 volume encryption (Railway managed Postgres on GCP) applies beneath the application-layer encryption; backups inherit it.
- **Log storage** (Better Stack): Per Better Stack's published security posture.

6.3 What is NOT yet supported: customer-managed keys (CMEK / BYOK)

Transcript content and secrets are application-layer AES-256-GCM encrypted (§6.2), and on SaaS the root key is held by the vendor in AWS KMS. What is **not** yet offered is **customer-managed** encryption keys (CMEK / BYOK) — a key the *customer* holds, audits, and can revoke. That is on the roadmap for customers who require it. Until then, a customer who needs to hold the keys themselves uses **Tier 2 self-host**, where the box encrypts under its own `APP_ENCRYPTION_KEY` on the customer's own infrastructure.

7. PII redaction pipeline

7.1 Architecture

The redaction layer (`backend/core/pii_redactor.py`) uses Microsoft Presidio as the entity-detection engine, augmented with custom recognizers tailored to corporate communication contexts (employee IDs, internal credential formats, etc.).

7.2 Allowlist (speakers preserved)

Before invoking Presidio's analyzer, Making Yourself Clear constructs an allowlist from the transcript's `speaker_labels` . Each speaker label is added to the allowlist as-is and decomposed into its individual word tokens. This protects "Sarah Chen" as well as the standalone "Sarah" and "Chen" later in the transcript.

The rationale: the LLM produces coaching feedback that references speakers by name ("Sarah, you interrupted Daniel twice"). Redacting speakers would damage coaching quality without meaningfully improving privacy — the coachee's own name is already in their account.

7.3 Aggressiveness levels

Level	Entities detected (in addition to high-risk identifiers)
Conservative	Names, emails, phones, addresses, government IDs, credit cards, employee IDs, DOBs, credentials, IPs, URLs, **locations**, **nationalities / religions / political groups**
Standard (default)	All of Conservative *except* generic locations and NRP categories
Permissive	High-risk identifiers only (government IDs, credit cards, credentials)

Aggressiveness is configurable at the vendor level and overridable per-organization.

7.4 Storage after redaction

For every analysis:

- The **redacted text** is stored on the `transcripts.redacted_text` column — this is the exact bytes the LLM sees.
- A **reversible token mapping** (`<PERSON_1>` → `"Mark Smith"`) is stored on `transcripts.redaction_mapping_json` if the org has the reversible-mapping setting on. Used to display original names back to the coachee in the in-app "View what was sent" panel. **Not used to substitute back into the AI output** (the AI sees and references only redacted tokens).
- A **per-entity audit log** (entity type, token, turn ID, character range) is appended.

When reversible-mapping is off (configurable per-org), the mapping is discarded after the LLM call — the entity-counts audit is still recorded.

7.5 Customer-visible verification

Coachees and coaches can open a "View what was sent to the LLM" panel on any analysis to see the exact redacted text. This is a deliberate trust-building affordance: customers can verify, not just take our word.

---

8. LLM provider posture

8.1 Default routing

Making Yourself Clear's LLM router (`backend/core/llm_client.py`) selects a provider based on the configured model name:

- Models starting with `claude-` route to Anthropic.
- All other models route to OpenAI.

The model name is set in the vendor system config (`system_configs['model_name']`) by the vendor_admin. Production today defaults to OpenAI; Anthropic is enabled for customers who configure a Claude model via BYO LLM (and supported in the codebase for vendor-level routing if needed).

8.2 Retention controls

Provider	Retention control
OpenAI (default)	`store=False` on every request, which disables OpenAI-side storage of the conversation. OpenAI does not train on API data (standard API terms). Standard API retention of **up to 30 days** (abuse-monitoring) still applies. A Zero-Data-Retention agreement has been **applied for and is pending** — not yet in effect, so we do not claim zero retention today.
Anthropic (Claude models only)	Not the production default. Anthropic's standard commercial API policy is no training on API data and up-to-30-day operational retention. Customers wanting stronger controls should use BYO LLM with their own enterprise agreement.
BYO LLM	Governed by the customer's contract with their LLM provider. Making Yourself Clear does not retain a copy of the LLM request or response on the LLM provider's side; only the structured response is stored back in Making Yourself Clear's Postgres database.

8.3 BYO LLM implementation details

When an org_admin configures a BYO LLM endpoint:

1. `provider`, `base_url`, `model` are stored as plaintext in the `organizations` table (these are not secrets).

2. `api_key` is encrypted with AES-256-GCM (under the KMS-rooted `APP_ENCRYPTION_KEY`) before storage; the ciphertext goes into `organizations.llm_api_key_ciphertext`.
3. The CHECK constraint `organizations_llm_all_or_nothing_check` ensures the four columns are either all NULL (inherit vendor LLM) or all non-NULL.
4. At analysis time, the worker resolves the coachee's organization, decrypts the API key momentarily, and passes the (api_key, base_url, model) tuple to the LLM router. The plaintext lives in process memory only for the duration of the LLM call.
5. A "Test connection" endpoint lets the org_admin verify the configuration before saving by firing a minimal LLM call through the proposed endpoint.

8.4 What we don't send to LLMs

- Personally identifiable information beyond what survives redaction (speaker names by design; everything else removed by Presidio).
- Account credentials, session tokens, or other secrets.
- Other coachees' data (LLM context is per-call, per-coachee).
- Internal system information beyond the structured transcript and a static prompt template.

---

9. Logging, monitoring, observability

9.1 Application logs

Backend and worker logs are forwarded from Railway to Better Stack (EU data region). Retention is 3 days on the current plan. Standard log fields: timestamp, level, module, message. PII is not deliberately logged; redaction operates on transcript content before any LLM call, and request bodies are not logged.

9.2 Error tracking

Sentry is wired across five runtimes (backend FastAPI, Celery worker, Next.js client / server / edge). Privacy posture is intentionally aggressive:

- `sendDefaultPii=False` — no PII attached to error events
- `max_request_body_size="never"` — never send request bodies
- `include_local_variables=False` — no stack-frame locals in error reports
- Custom `beforeSend` scrubs IP-bearing headers, cookies, and the `Authorization` header

Source maps are uploaded to Sentry for sourcemap resolution; releases are tagged by git SHA.

9.3 Admin audit log

`admin_audit_log` is an append-only Postgres table recording every administrative action:

- `actor_user_id`, `actor_email`
- `action` (string, e.g. `update_org_redaction_settings`, `set_org_llm_settings`)
- `target_type`, `target_id` (e.g. `organization`, `42`)
- `details` (JSONB diff; sensitive values like API keys never appear in plaintext or ciphertext — only `"set"` / `"rotated"` / `"cleared"`)
- `created_at`, `ip_address`, `user_agent`

The audit log can be queried by vendor_admin via the admin UI.

Read-only access to user data (browsing the admin UI without taking an action) is not yet audited at the application layer. This is on the SOC 2 roadmap.

9.4 Uptime monitoring

Better Stack pings `/health` every three minutes from four global regions. Alerting via email to the on-call founder.

---

10. Vulnerability management

10.1 Dependency monitoring

Dependabot is configured for all four package ecosystems used in the repo (pip, npm, docker, github-actions), with grouped weekly PRs (patch and minor grouped per-ecosystem; major bumps individual). Triage of open alerts is part of the routine release cadence.

10.2 Secret scanning

GitHub Secret Scanning + Push Protection are enabled on the repository. Accidentally committed credentials are detected on push; new ones are prevented from landing.

10.3 Static analysis

`npx tsc --noEmit` and `npx next build` are part of the standard verification flow for every change touching the frontend. The backend test suite (1,600+ tests) runs on every PR.

10.4 Independent penetration testing

Planned. Engaging a recognised firm (NCC Group, Trail of Bits, Bishop Fox, or equivalent) for a focused web-application engagement. Reports will be shared under NDA with prospective customers.

10.5 Vulnerability disclosure

Security researchers can report vulnerabilities to security@makingyourselfclear.com. We commit to acknowledging reports within 3 business days and to coordinated disclosure on a reasonable

timeline.

11. Sub-processor evaluation criteria

When evaluating a new sub-processor, Making Yourself Clear looks for:

1. **Security posture** — SOC 2 Type II or equivalent attestation, documented encryption practices, incident response track record.
2. **Data minimisation** — does this sub-processor actually need customer data, or can we use it for ops-only data?
3. **Data residency** — alignment with our region commitments where applicable.
4. **Contractual posture** — DPA available, sub-processor sub-processors disclosed.
5. **Operational maturity** — uptime track record, change management, customer notification practices.

The current sub-processor list is documented in `compliance-one-pager.md`. Material changes are notified per the DPA's sub-processor clause.

12. Incident response

12.1 Detection

- Sentry alerts on unhandled exceptions
- Better Stack uptime alerts on health-check failures
- Better Stack log-based alerts on suspicious patterns (planned: brute-force, anomalous error rates)

12.2 Response procedure

1. On-call founder is alerted via email.
2. Triage in Sentry / Better Stack to scope the incident.
3. If a personal-data breach is confirmed, customer notification is initiated within 72 hours of confirmation (GDPR Article 33 standard).
4. Post-incident review documents root cause, customer impact, remediation, and prevention.

12.3 Notification

- Customers are notified of confirmed personal-data breaches within 72 hours.
- Sub-processor material changes are notified per the DPA's notification clause.

- Service disruptions are communicated via the status page (planned) and direct email for major incidents.

12.4 Business continuity

- Postgres backups are taken daily by Railway's managed Postgres, with point-in-time recovery.
- Restore procedures are exercised at major releases.
- In the event of vendor failure: customers can export their data on request; data is returned in a machine-readable format on subscription termination.

---

13. Roadmap and known limitations

Making Yourself Clear is an actively developed product. The following are tracked as known limitations or planned improvements:

Planned

- **SOC 2 Type II.** Engaging an attestation firm; Type I within 6 months of engagement, Type II to follow.
- **Independent penetration test.** Scheduling in 2026.
- **Application-level MFA for email/password coachees** (TOTP).
- **Customer-managed encryption keys (CMEK / BYOK)** — a customer-held, customer-revocable key over the at-rest content (distinct from the vendor-held KMS root that is already live, and from the existing BYO LLM API-key encryption).
- **SIEM** alongside the SOC 2 trajectory.
- **Read-access audit logging** at the application layer.
- **Status page** for service health and incident communication.

Known limitations (acknowledged tradeoffs)

- The default SaaS tier does not provide cryptographic isolation from the vendor. Tier 2 self-host is the answer for customers who need it.
- The platform is not intended for PHI; no BAA is offered.
- Data residency is currently US West only; non-US regions are not yet offered.

Not on the roadmap (explicitly out of scope)

- Fully local LLM ("Tier 3"). Open-weight models do not yet match frontier-model coaching quality, and Making Yourself Clear's value proposition depends on coaching quality.
- ISO 27001. SOC 2 Type II is the priority; ISO 27001 may follow if customer demand requires it.

---

14. References

The technical claims in this document can be verified by inspecting the codebase:

- **Authentication and session:** `backend/api/auth.py`, `backend/auth/`
- **Authorization helpers:** `backend/api/routes_admin_org.py` (`_require_vendor_admin`, `_require_org_access`)
- **Multi-tenant scoping:** `backend/auth/organizations_db.py` (`migrate_user_data_to_organization`, `list_coachees_for_organization`)
- **PII redaction:** `backend/core/pii_redactor.py`, `backend/core/workers.py` (redaction invocation)
- **LLM routing:** `backend/core/llm_client.py`, `backend/core/openai_client.py`, `backend/core/anthropic_client.py`
- **BYO LLM encryption:** `backend/auth/llm_crypto.py`, `backend/auth/organizations_db.py` (`resolve_llm_credentials_for_user_record`)
- **Audit logging:** `backend/api/audit.py`, `backend/alembic/versions/` (`admin_audit_log` table)
- **Observability config:** `backend/observability.py`, `frontend/sentry.*.config.ts`

Contact

Questions about anything in this document, or about how a specific control would work for your use case:

- **Security questions:** security@makingyourselfclear.com
- **Architecture / technical review:** chris.iskander@makingyourselfclear.com
- **Procurement / DPA:** chris.iskander@makingyourselfclear.com

---

Document control

- Version: **1.1** (2026-07-08) — reflects the current architecture: application-layer AES-256-GCM at-rest encryption over content + secrets (per-org DEK, KMS-rooted root key, crypto-shred), the current LLM-provider posture, and Tier 2 self-host (generally available).
- Owner: Chris Iskander, Executive Coach and Founder, Executive Sound Board
- Last reviewed: 2026-07-08
- Next scheduled review: Annual review, or on material change to architecture or controls