Engineering Specification Document (Revised)

Project: seda.fm

Document Type: ENGINEERING SPECIFICATION

Version: 2

Created by: ENGINEERING_LEAD

Created: Sat Sep 06 2025

Executive summary This document translates the architecture, UX spec and PRD into an actionable engineering implementation plan for the seda.fm Recommendation Engine and its UX surfaces (Rooms, Playlists, Artists), provider integrations, embeddings, real time presence, notifications, privacy/export/delete and EU residency controls.

It includes: implementation strategy, API contracts, detailed data model (with residency/audit), processing flows (batch & realtime), EmbeddingsService design with spend caps & EU routing, real time + Socket.IO patterns, provider integration rules, background jobs, CI/CD and QA/sandbox, observability, security and RLS guidance, acceptance gates, and risks with mitigations.

Scope and priorities MVP features (deliver first)

- Minimal onboarding (account + ≥3 genres) and first meaningful feed <60s
- Feed modules: Recommended Rooms/Playlists/Artists with friend avatars, Join/Follow CTAs
- Provider linking: Spotify (persist refresh token + background sync opt in), Apple (session-only import)
- EmbeddingsService (OpenAI primary) with per-user region routing and spend-cap enforcement
- Vector store: pgvector on Supabase (partitioned by entity_type; HNSW)
- Real-time presence & toasts via Socket.IO + Redis adapter (coalescing)
- Push policy: opt in, pre-permission flows, caps
- Privacy: Export / Delete imports with re-auth, typed confirmation, immediate exclusion + 30 day physical purge (with backup key handling)
- QA sandbox: sandbox-us + sandbox-eu, Mock Provider Service, Mock EmbeddingsService, Event Replayer

Notes on additions from Q&A:

- Implement canonical users.data_residency field and residency_changes audit with strict precedence for EU routing.
- Provide per-user opt in for temporary non EU fallbacks and require two approver org overrides for system-wide fallbacks.
- Provide admin/test APIs and CLI to simulate embedding spend thresholds (80%/100%) and exercise override flows in sandbox.
- Spotify refresh_token persistence allowed but background sync default = OFF; immediate deletion on unlink required.

Implementation strategy High-level principles

- Single source of truth: Supabase Postgres + pgvector for MVP. Add users.data_residency as canonical residency key and users.data_residency_source and residency_changes audit table.
- Abstractions first: EmbeddingsService and VectorStore adapter interfaces to decouple providers and enable shadow/dual-write later.
- Fail-safe behavior: cache-first, circuit-breakers, coalescing, queued fallbacks for degraded infra.
- Observability & gating: instrument everything; gate rollouts via LaunchDarkly; programmatic spend controls and sandbox controls for QA.
- Security & compliance-first: KMS-encrypted tokens, RLS policies, EU project isolation for EU users.

Teams & responsibilities

- Platform/Infra: provision EU runner, LaunchDarkly, Supabase projects, EU sandbox, KMS, CI/CD (Railway).
- Backend Engineering: NestJS services, EmbeddingsService adapter, Queue workers (BullMQ), Socket.IO server, API endpoints, admin/test APIs and CLI.
- Mobile/Web: React client, Socket.IO client patterns, pre-permission flows, accessibility.
- SRE/Ops: cost controls, overrides, on-call for embedding limits & critical incidents.
- Data Science: ranker weighting, A/B-tests, evaluation metrics.

Tech stack mapping

- Backend: Node.js + NestJS (Railway)
- DB: Supabase Postgres (pgvector), separate projects for prod-us and prod-eu
- Auth: Supabase Auth
- Real-time: Socket.IO + Redis adapter (Upstash)

- Jobs: BullMQ + Upstash Redis
- Embeddings provider: OpenAI primary; Cohere standby; EU self-hosted SentenceTransformers runner
- Push: Firebase FCM
- Feature flags & experiments: LaunchDarkly (prod), Supabase flags (dev)
- Observability: OpenTelemetry traces, Prometheus-style metrics, Sentry, PostHog
- Storage: Supabase Storage
- CI/CD: GitHub Actions Railway deployment

Data model (canonical, implementation-ready) Use snake_case, UUID PKs, JSONB for flexible props.

Key tables (column types summary)

- users
    - id UUID PK
    - auth_user_id TEXT UNIQUE (supabase sub)
    - created_at TIMESTAMPTZ DEFAULT now()
    - dob DATE
    - region TEXT -- onboarding-selected, not authoritative
    - data_residency TEXT -- ENUM('eu','non-eu') **canonical routing key**
    - data_residency_source TEXT -- enum ('billing','user_declared','kyc','subscription','ip_suggested','admin_override')
    - data_residency_changed_by UUID NULL
    - data_residency_changed_at TIMESTAMPTZ NULL
    - role TEXT -- enum ('user','artist','admin','super_admin')
    - preferences JSONB -- { genres: [...], locales... }
    - privacy_settings JSONB -- { presence_default: 'friends'|'hidden', push_opt_in: false, teen: true/false, allow_non_eu_fallback: false, consent_timestamp: ts }
    - deleted_at TIMESTAMPTZ NULL
    - last_seen_at TIMESTAMPTZ
- residency_changes (audit)
    - id UUID PK
    - user_id UUID
    - old_value TEXT
    - new_value TEXT

    - source TEXT
    - actor_id UUID NULL
    - reason TEXT NULL
    - created_at TIMESTAMPTZ

- linked_accounts
    - id UUID PK
    - user_id UUID FK
    - provider TEXT -- 'spotify','apple'
    - external_user_id TEXT
    - scopes JSONB
    - refresh_token_encrypted BYTEA NULL -- persisted only for Spotify if user opted into background sync
    - background_sync_enabled BOOLEAN DEFAULT FALSE
    - last_synced_at TIMESTAMPTZ NULL
    - sync_status TEXT -- 'ok'|'needs_reauth'|'paused'
    - created_at TIMESTAMPTZ

- artists / playlists / rooms / user_profiles / recommendations / embeddings_audit / fact_events / privacy_jobs
    - Preserve original schema, adding:
        - embedding_meta JSONB enriched with region, override_id, provider/model/dim/version, computed_at
        - embeddings_audit must record deleted_at and job_id and owner_user_id when applicable

Indexes and partitions

- Vector indices: per-table pgvector HNSW indexes.
- Partition by entity_type (artists_vectors, playlists_vectors, rooms_vectors).
- EU-residency: separate Supabase project for EU users. Do not cross-write.

APIs & contracts (BFF / public) Implement REST + WebSocket channels. Use idempotency keys for mutating ops.

Core endpoints (summary; include admin/test endpoints added per Q&A)

- GET /feed?user_id=…&surface=home

    - Returns cached top-50, freshness flag, embedding_meta provenance.
- POST /join-room { room_id, client_id, idempotency_key }
    - Optimistic join semantics; returns ack with event_id.
- POST /integrations/spotify/link -- OAuth handshake (server side). Persist refresh_token_encrypted only after explicit consent for background sync. Provide sync toggle in settings; default OFF.
- POST /integrations/apple/import -- accepts musicUserToken (transient) and enqueues import job; returns 202 if async.
- POST /integrations/spotify/unlink -- delete token, immediate exclusion, enqueue purge.
- POST /privacy/export-data { user_id } -> returns job_id
- POST /privacy/delete-imports { user_id, reauth_token, typed_confirmation } -> returns job_id + immediate_exclusion flag
- GET /privacy/jobs/{job_id}
- GET /admin/spend-status (internal)
- POST /admin/override-requests -- create override request (scope, reason, duration, cost_estimate)
- POST /admin/override/approve -- approver signs
- POST /admin/override/execute -- executes override after approvals
- POST /admin/test/embeddings/simulate-spend -- sandbox only; simulate 80/100% triggers
- GET /admin/embeddings/actions -- list enforcement actions
- POST /admin/override-embeddings { override_id, duration, approvers[] } -- two-approver flow
- POST /admin/embeddings/actions/pause-jobs, /resume-jobs, /switch-model

Real-time channels

- Socket.IO authenticated per user. Channels:
    - user:{user_id} -- user-specific notifications/toasts
    - room:{room_id} -- room presence & chat events
- Events:
    - presence.join/leave; presence.snapshot
    - toast.notify { small payload }
    - privacy:job_update
- Enforce client-side coalescing: clients re-render at 500–1000ms windows.

Residency & routing: canonical rules (integrated)

- Canonical field: users.data_residency (enum: 'eu' | 'non-eu') is authoritative for routing, storage, and embedding provider selection.
- Precedence (highest lowest):

    1. Billing/contract address (verified) — enterprise binding.
    2. Explicit user-declared residency (onboarding/settings) — requires explicit confirmation.
    3. Verified identity/KYC country.
    4. Persistent account locale/phone-country/subscription metadata.
    5. IP geolocation (ip_suggested) — advisory only, used to pre-fill UI, not to set canonical value.

- Changes:

    - User-initiated changes require re-auth and typed confirmation; record source='user_declared' and log residency_changes.
    - Admin overrides require two-approver flow (SRE/Engineering + Product/Legal/Finance), must be timeboxed, and recorded as source='admin_override'.
    - For enterprise customers, billing/contract jurisdiction overrides other signals and can be enforced via contract_residency flag with legal sign-off.

- Enforcement:

    - All embedding and vector-write code must consult users.data_residency at write time and route to corresponding region project (EU EU Supabase) and provider.
    - Implement pre-write validators (service layer), and DB guards/triggers or RLS checks (defense-in-depth) to reject cross-region writes.
    - On residency change, enqueue propagation job: re-route pending jobs, invalidate caches and recommendations, and schedule re-embeddings if needed.

EmbeddingsService: design & enforcement (expanded) Responsibilities

- Batching, provider adapter, rate-limiter, circuit-breaker, EU routing, cost metering, shadow-write capabilities, model switching, failover sequence, and admin/test hooks.

Provider adapters and fallback sequence (decided)

- Primary production model: OpenAI text-embedding-3-small (1536d).
- Pre-approved automated fallback sequence (order of preference):

1. Same-dim, lower-cost hosted SKU from primary provider (if available and compatible).
2. Cohere adapter producing identical dimensionality (1536d), if available and validated.
3. EU self-hosted runner configured to produce the same dimensionality for EU users.
4. Self-hosted lower-dim model (e.g., SentenceTransformers miniLM 384d/512d) ONLY as controlled temporary fallback if compatibility strategy is in place (shadow lower-dim catalog vectors, projection layer, or non-ANN fallback).

- Runtime checks:
    - EmbeddingsService must validate embedding_meta.dim and refuse to mix dims in ANN searches. If dim mismatch is detected, use non-ANN ranking signals (CF/social/popularity) or trigger a projection/shadow strategy.
- Shadow writes:
    - Allowed at controlled % (default 1–2%). Shadow writes must respect EU residency constraints (do not shadow EU users to non-EU providers).
- Batching & windows:
    - batch_size = 128, window = 200ms default; configurable per provider and region.

Circuit-breaker & cost metering

- Trip conditions: provider error_rate > 2% OR P95 latency > 1s for >60s.
- Open duration: 60s; half-open probes at low traffic.
- Enforcement at spend thresholds:
    - 80% projected run-rate:
        - throttle background jobs (reduce concurrency, increase batching)
        - route on-demand user vectors to cheaper same-dim provider or cheaper model if validated
        - emit admin alerts and show degraded banner
    - 100%:
        - stop non-essential jobs (catalog re-embeds, shadow writes)
        - allow only critical on-demand computes via cheap model or queue with user-visible ETA
        - require two-approver override to resume full operations
- Admin/test APIs:

    - expose POST /admin/test/embeddings/simulate-spend in sandbox to validate enforcement behavior
    - expose GET /admin/spend-status and /admin/embeddings/actions for observability

EU routing and emergency fallback behavior

- Per-user routing uses users.data_residency to decide provider and vector storage region.
- Fallback policy on EU provider outage:
    - Default: DO NOT route EU users to non-EU processing.
    - Exceptions:
        - Per-user opt in: user has pre-consented (privacy_settings.allow_non_eu_fallback = true). If opt in true and EU path is down, route that user's processing non EU temporarily (with strong constraints).
        - Org-level emergency override: two-approver override enables system-wide non EU fallback for specified duration.
    - Safe-guards for temporary non EU processing:
        - Ephemeral artifacts preferred; do not persist vectors non-EU unless necessary.
        - If persisted, tag embedding_meta.region = 'non-eu-temp' and embedding_meta.override_id, with automatic purge TTL (recommended delete within 24 hours after EU recovery).
        - Log every fallback use (override_id, user_id, job_id) for audit.
        - Notify affected users with concise in-app message and provide audit details in Settings Privacy Processing Details (override_id, approvers, expiry, purge ETA).
        - On EU recovery, reprocess affected users on EU path, replace artifacts, purge temp non-EU artifacts, and record reconciliation.

Vector store & ANN

- Use pgvector HNSW per table. Initial params: M=16, ef_construction=200, ef_search default=64 (tune).
- Index per entity type to reduce search domain.
- Query strategy: filter by genre/metadata first, ANN search top-K, then re-rank with features.
- Maintain lower-dim compatibility plan if fallback to lower-dim models is used: either maintain shadow catalog vectors at lower-dim or use a learned projection layer. Do not mix dims in ANN search.

Ranking pipeline

- Candidate generation:
  - ANN from taste_embedding -> entity vectors (per-genre filter)
  - CF offline features (ALS) and popularity/trending
  - Social boost (friend presence, recent follows)
- Re-ranker: weighted linear model (start) combining embedding_similarity, collaborative_score, friend_presence_count, recency, popularity, diversity_penalty
- Diversity: enforce at least 30% novel recommendations; use cluster-based de-duplication.

Real-time presence & toasts Server-side aggregation

- Aggregate presence events per-room over 250–500ms windows before broadcasting.
- Friend-driven notifications target friends-of-actor only.

Client constraints

- Coalesce socket events, re-render at 500–1000ms.
- Toast caps: max 3 per session; 10min cooldown.
- Avatar group: show up to 4 avatars; overflow +N; respect privacy flags.

Optimistic join semantics

- Client POST /join-room -> optimistic UI; server acknowledges via socket with authoritative state.
- Idempotency via idempotency_key.

Provider integrations Spotify (expanded)

- Persist refresh_token_encrypted in linked_accounts only with explicit user consent for background sync. Default background_sync_enabled = OFF.
- Background sync cadence:
  - Default nightly summary sync (e.g., 03:00 local, staggered).
  - Hourly incremental only if user explicitly opts in and global quotas allow.
- Token security:
  - Envelope encryption with KMS; stored in linked_accounts.refresh_token_encrypted (BYTEA).
  - Decrypt only in server-side workers authorized via service role. Do not log raw tokens.
  - Rotate refresh tokens atomically if provider returns new refresh_token on refresh.

- Failure handling:
    - On invalid_grant or repeated failures, mark sync_status='needs_reauth' and notify user to re-authenticate.
- Unlink flow:
    - On unlink: delete refresh_token_encrypted immediately, exclude imported artifacts from ranking, enqueue privacy_job for physical purge (30-day rule).
- Privacy:
    - Only store summary signals (top N artists/playlists/genres) per PRD. No raw play-by-play storage beyond allowed windows.

Apple Music

- Session-only imports; do NOT persist musicUserTokens. Persist developer token encrypted; rotate as required.

Privacy, export & delete flows (expanded backup treatment) Delete imports flow

- Re-auth requirement: POST /auth/reauth returns short-lived reauth_token.
- POST /privacy/delete-imports { reauth_token, typed_confirmation } -> job_id and immediate_exclusion = true.
- Immediate online behavior:
    - Tombstone rows (deleted_at), set immediate_exclusion flags, invalidate caches and MVs, and stop serving imported artifacts.
- Background purge job:
    - Physically delete vectors, imported rows, recommendations, and storage objects within 30 days.
    - Delete entries from embeddings_audit (mark deleted_at) and record job_id.
- Backups, PITR, replicas:
    - Ensure physical unrecoverability within 30 days. Options:
        - Option A (recommended): Configure backups/PITR retention <= 30 days so physical purge coincides with backup expiry.
        - Option B (crypto-erase): Use per-user envelope keys for sensitive artifacts; destroy the user envelope key to render backups unrecoverable; record KMS key-deletion proof.

        - Option C (exception): documented, timeboxed legal exception with DPO approval (rare).
    - Replica treatment: purge rows on replicas or rebuild replicas from snapshots excluding deleted users, or apply crypto-delete.
- Auditability: persist privacy_jobs, embedding_audit.deleted_at, KMS key deletion logs, and provide a verification artifact for DSARs confirming live data absence.
- Runbook for privacy_job worker: tombstone -> delete rows & storage -> delete or rotate crypto keys (if used) -> update privacy_jobs status -> emit privacy:job_completed.

Re-auth & typed confirmation UI

- Require re-auth (password or biometric) + typed confirmation to perform destructive deletes or residency changes that reduce protections.
- Provide job progress via Socket.IO events privacy:job_update and privacy:job_completed.

Admin & override workflows (detailed)

- Overrides must be auditable and enforce two-approver workflow for critical actions (non-EU fallback, spend overrides, resume paused jobs).
- Internal/CLI acceptable for MVP with protected APIs; admin console to follow ASAP.

Override request lifecycle (API & CLI)

- POST /admin/override-requests { requester, reason, scope, affected_cohorts, estimated_cost, expiry_minutes }
    - Creates override_id and status REQUESTED.
- Approvers call POST /admin/override/approve { override_id, approver_id, role, comment }.
    - Require two distinct approvers, at least one technical (SRE/Engineering) and one business/legal (Product/Legal/Finance). For residency/legal-risk overrides include Legal as approver.
- Executor (SRE) runs POST /admin/override/execute { override_id }.
    - Execution toggles feature flags, routes, or jobs as specified. Execution recorded with executed_by and executed_at.
- Automatic enforcement:
    - Override expires at expiry_time; system auto-reverts and logs revert events.
    - If safety gates (latency/cost) trip, auto-revert and page on-call and approvers.

Minimum fields stored:

- override_id, requested_by, requested_at, reason, scope, affected_cohorts, duration/expiry, approvers_required, approvals[], executed_by, executed_at, status, cost_projection, rollback_plan, audit_log.

CLI (MVP)

- Provide embedctl wrapper to call admin APIs for SRE:
    - embedctl override create|approve|execute|revert
    - embedctl spend status|simulate
    - embedctl pause-jobs|resume-jobs|switch-model

CI/CD, feature flags & rollout

- GitHub Actions Railway deployment.
- Feature flags in LaunchDarkly for user-impacting features (embeddings model choice, vector store reads, toast variants).
- Canary ramp: internal -> 1% -> 5% -> 25% -> 50% -> 100% with automated operational gates.

Testing strategy & QA sandbox (detailed)

- Unit tests (Jest), integration tests (Supabase sandbox), E2E (Cypress), contract tests (OpenAPI), load tests (k6).
- Sandbox suite:
    - sandbox-us + sandbox-eu Supabase projects, seeded profiles (Smoke, Medium, Large).
    - Mock Provider Service with failure/latency knobs.
    - Mock EmbeddingsService + EU self-hosted runner.
    - Event Replayer (server injection & headless Socket.IO clients).
    - Admin/test endpoints for spend simulation: POST /admin/test/embeddings/simulate-spend.
- QA capabilities (provided):
    - Simulate 80% and 100% embedding spend to verify enforcement actions (throttle/switch/stop).
    - Execute two-approver override workflow end-to-end via API/CLI.
    - Inject provider latencies, 429s, 5xx to validate circuit-breaker and fallback behavior.

    - Event Replayer supports sustained 1k–5k events/sec and bursts to 10k+, with larger runs scheduled and approved.

Observability & telemetry (expanded) Emit events (fact_events)

- rec_impression, rec_click, join_room, listen_30s, track_skip, rec_dismiss, provider_linked, provider_unlinked, export_requested, delete_imports_requested, embedding_use (with model/provider/region/override_id), spend_cap_event (threshold, action), override.requested/approved/executed/reverted, privacy:override_impacted_user.

Metrics and alerts

- Metrics: time_to_first_feed, time_to_enrich, join_ack_latency, embedding_latency (P50/P95/P99), rec_api_latency, socket_fanout_latency, queue_depths, embedding_cost_runrate.
- Alerts:
    - Embedding spend projection > 80% => notify product & SRE.
    - rec API P95 > 350ms => page SRE.
    - real-time P99 > 500ms => page SRE.
    - provider error rate > 2% => alert.

Security & RLS (expanded)

- Supabase Auth canonical; store roles in users.role; RLS policies to use auth.uid().
- Example RLS:
    - users_select_own: USING (auth.uid() = id)
    - recommendations_select_user: USING (auth.uid() = user_id OR current_setting('is_internal') = 'true')
- Residency enforcement:
    - Pre-write validators in service middleware check users.data_residency and set request-local current_setting('target_region') to block non-matching writes. Where feasible, add DB-level triggers that check a request header or current_setting to ensure writes target correct region.
- Token storage:
    - refresh_token_encrypted in DB is KMS envelope encrypted. Decrypt only in backend workers with least privilege.
- No PII in embedding payloads; redact PII before sending to providers.

Performance and scaling considerations

- Socket.IO cluster with Redis adapter. Monitor per-node capacity; introduce Kafka/Redpanda when sustained events/sec > 5k.
- pgvector scaling: monitor table size, memory & rebuild times; consider vector DB or partitioning strategies beyond triggers (>500k vectors).
- BullMQ worker autoscaling via queue depth.

Acceptance criteria & gating (updated)

- Onboarding: ≥90% of test participants complete onboarding and see feed ≤60s.
- Accessibility: WCAG 2.1 AA automated + manual screen-reader tests.
- Embedding spend enforcement: 80% and 100% automated actions trigger and UI banners/notifications visible.
- EU compliance: users.data_residency set per precedence; embedding_meta.region == users.data_residency for EU users; tests verify no cross-region writes without override.
- Push pre-permission: shown after session_count >=2 OR first meaningful engagement; in-app push toggle default = OFF; OS prompt invoked only after in-app Enable and only once unless user re-triggers.
- Spotify Sync: background_sync default = OFF; refresh_token_encrypted persisted only with explicit consent; unlink deletes token and triggers immediate exclusion + purge job.

Risks and mitigations (top items)

1. Embedding cost overrun

- Mitigation: programmatic caps at 80/100, low-cost fallback, shadow budget limits, per-job cost attribution, immediate alerts, sandbox test APIs for QA.

2. EU non-compliance

- Mitigation: dedicated EU Supabase project, canonical users.data_residency with strict precedence, pre-write validators, two-approver overrides, EU sandbox QA, audit logs, automated tests verifying no cross-region writes.

3. Real-time scale / fanout thrash

- Mitigation: server aggregation (250–500ms), client coalescing (500–1000ms), counts-only fallback for large rooms, broker introduction triggers.

4. Provider outages / rate limits

- Mitigation: caching, circuit-breakers, retries with backoff, degrade to metadata heuristics, user-facing messaging, per-user opt-in for non-EU fallback and org override.

5. Privacy distrust for linking

- Mitigation: clear consent copy, export/delete UI, typed confirmation, immediate exclusion + 30-day purge guarantee, evidence via privacy_jobs and KMS key-deletion proofs where applicable.

Implementation roadmap & milestones (90-day view) Phase 0 (Week 0–2): infra & scaffolding

- Provision Supabase prod & sandbox (EU project), KMS setup, LaunchDarkly, Railway CI pipeline, Upstash Redis, Mock Provider + Mock EmbeddingsService dev images, admin/test API scaffolding.

Phase 1 (Week 2–6): core APIs & onboarding

- Implement users, users.data_residency logic and residency_changes audit, genre onboarding, feed seed logic (mv_genre_top_rooms).
- Client onboarding screens, residency UI (pre-fill via IP, require confirmation), first feed skeleton.
- Basic rec API GET /feed and pre-permission gating: show pre-permission after session_count >=2 OR first meaningful engagement.

Phase 2 (Week 6–10): provider linking & embeddings pipeline

- Spotify OAuth flow + persist refresh_token_encrypted only after consent; background_sync toggle (default OFF).
- Apple session import endpoint.
- EmbeddingsService adapter + OpenAI integration + Embeddings audit table; EU routing enforcement (users.data_residency).
- Background worker skeleton (BullMQ) and initial embedding compute (user taste vectors), and embedding spend metering.

Phase 3 (Week 10–14): real-time & presence

- Socket.IO server + Redis adapter, presence strip, toast logic, client coalescing.
- Optimistic join flow.

Phase 4 (Week 14–18): privacy flows & admin controls

- Export/Delete endpoints + re-auth flow, privacy_jobs, job updates via Socket.IO.
- Embedding spend cap enforcement & admin override APIs + CLI (two-approver).

- EU routing tests using sandbox-eu runner.

Phase 5 (Week 18–24): hardening, QA & rollout

- Instrumentation, SLO dashboards, LaunchDarkly flags and canary rollout plan.
- Accessibility & performance testing in sandbox (Medium + Large).
- Finalize A/B experiments and go-live gating.

Implementation patterns & coding best practices

- Code organization: modular NestJS services (api, embeddings, provider-connectors, realtime, jobs, admin).
- Contracts: OpenAPI for REST endpoints; validation using class-validator; idempotency middleware for mutating endpoints.
- Error handling: classify errors (retryable vs fatal); consistent error codes & x-degraded header usage.
- Secrets: never in repo; use environment injection; rotate keys; audit access.
- Tests: unit tests (80% for core libs), integration tests for DB + pgvector with sandbox, end-to-end for critical flows.
- Observability: trace every user request path across services using OpenTelemetry; attach job_id and embedding_meta in traces.

Clarifying operational notes & test hooks (engineering deliverables)

- Residency enforcement test hooks:
    - Provide sandbox-eu and sandbox-us with seeded profiles and test credentials.
    - CI smoke: create user with data_residency='eu', trigger embedding, assert embedding_meta.region='eu' and vectors stored only in EU project.
    - Negative test: attempt non-EU write for EU user; must be rejected.
- Spend enforcement test hooks:
    - Provide POST /admin/test/embeddings/simulate-spend for sandbox to force 80%/100% enforcement.
    - Provide GET /admin/embeddings/actions and metrics for QA to validate actions.
- Override workflow:
    - Admin APIs plus CLI to create, approve, execute, and observe override lifecycle. Tests to assert two distinct approvers required.
- Privacy delete/backups:

    - Provide sandbox tools to test privacy_job purge and KMS key destruction proof flows.
- Push pre-permission:
    - Server flags for session_count and meaningful_engagement_flag, and test users to exercise pre-permission flow.

Open questions (resolved in Q&A)

- Canonical residency signal and precedence resolved: users.data_residency with precedence billing/contract > user_declare > kyc > subscription > ip_suggested.
- EU fallback policy: require two-approver override for system-wide fallback; per-user opt-in supported; all fallback uses audited and time-boxed.
- Embedding fallback providers/models pre-approved sequence and constraints established; lower-dim fallback allowed only with compatibility plan.
- Push pre-permission logic defined: shown after 2 sessions OR first meaningful engagement; in-app toggle default OFF; Spotify background-sync default OFF.

Acceptance gates & product metrics (restated)

- Engagement: ≥20% of sessions include rec-driven activity.
- Quality: ≥70% of recs listened >30s; skip rate ≤15%; dismiss rate ≤10%.
- Growth: ≥10% of new artist follows from recs; +5% retention uplift.
- Delight: ≥4/5 satisfaction; 30% diversity.
- Operational: Embedding spend enforcement triggers at 80/100% with documented actions; residency routing verified by CI tests; privacy deletion and backup handling verifiable.

Risks and mitigations (detailed)

- Embedding cost overrun: enforce programmatic spend caps; simulate in sandbox; provide admin/test endpoints.
- EU non-compliance: enforce canonical residency, protect write paths, provide audit trails and CI checks.
- Real-time fanout: aggregation + coalescing + counts-only fallback.
- Provider outages: circuit-breakers, standby providers, degrade gracefully.
- User privacy trust: explicit consent, in-app notifications, export/delete flows, audit proofs.

Appendix: Implementation checklist (practical)

1. DB schema additions:

- users.data_residency, users.data_residency_source, users.data_residency_changed_by, users.data_residency_changed_at.
- residency_changes audit table.
- linked_accounts.refresh_token_encrypted, background_sync_enabled, sync_status.
- embedding_meta enhancements (region, override_id).

2. Service middleware:
   - pre-write residency validator; set current_setting('target_region').
   - embedding routing layer consults users.data_residency.
3. EmbeddingsService:
   - provider adapters (OpenAI, Cohere, LocalRunner).
   - batcher, circuit-breaker, cost meter; fallback sequence implementation.
4. Admin APIs & CLI:
   - override request/approve/execute; spend-status; test simulate-spend (sandbox).
5. Privacy flows:
   - re-auth flow; privacy_jobs worker; backup/purge runbooks; crypto-delete support.
6. Provider integrations:
   - Spotify OAuth flow with explicit consent for background sync (default OFF).
   - Unlink immediate exclusion + enqueue purge.
7. Client UX:
   - onboarding residency pre-fill via IP (mandatory confirmation).
   - pre-permission push modal: show after 2 sessions OR first meaningful engagement.
   - in-app push toggle default OFF.
8. Observability:
   - instrument embedding_use events with model/provider/region/override_id; spend metrics; override lifecycle events.
9. Testing & sandbox:
   - provide sandbox-eu + sandbox-us, Mock Provider & EmbeddingsService, Event Replayer; support failure injection knobs and spend simulation.

10. CI/CD:

    - automated tests for residency enforcement, override flows, spend cap actions; canary rollout and operational gates.

If you want, next deliverables I will produce:

- Concrete OpenAPI definitions for the admin/test and privacy endpoints, including request/response schemas and error codes.
- Starter NestJS module layout with sample middleware for residency pre-write validation and sample RLS SQL snippets to enforce region constraints.
- A minimal CLI spec for embedctl with exact commands and example outputs for SRE.

This engineering specification is actionable and prescriptive: implement the canonical residency field and enforcement layers, embed spend caps with admin/test tooling, require two-approver overrides for emergency cross region processing, and provide sandbox/test hooks so QA and SRE can validate controls end to end.
