Technical Due-Diligence Report

HumanityAI'd Museum AR Smart Glass Guide

Backend (musem_API) + Glasses App (musem_ANDROID) — v1.0

July 2026

CONFIDENTIAL — PREPARED FOR PROSPECTIVE PARTNERS & ACQUIRERS

1. Purpose & scope

This report is an honest, buyer-grade technical assessment of the two codebases that make up the Museum AR Smart Glass Guide:

- **musem_API** — the on-premises AI backend (~6,300 lines Python): FastAPI, PostgreSQL 15, Redis 7, CLIP ViT-L/14 + FAISS recognition, Chatterbox multilingual TTS, WebRTC ingest, nginx, Prometheus/Grafana.
- **musem_ANDROID** — the RayNeo X3 Pro glasses app (~1,600 lines Kotlin/Java): Camera2 → WebRTC H.264 streaming, DataChannel result overlay, ExoPlayer narration.

It is written to survive scrutiny by a buyer's own engineers. It states strengths plainly and flaws plainly, and it distinguishes what is **product IP** from what is **POC-grade** and must be hardened before a production sale. A conservative hardening estimate is included.

2. Verdict in one paragraph

The product contains genuine, differentiated engineering — a dwell-based "look and listen" recognition UX, a multilingual content pipeline with an Arabic human-approval workflow, and a low-latency latest-frame-wins streaming design — and it is **demonstrably working end-to-end today** (10 exhibits, 8 languages, 80 approved narrations, recognition verified at 0.995 similarity). It is a **hardened proof-of-concept, not yet a hardened product**: the largest gaps are authentication, transport security, automated tests, and single-node capacity. None of these are architectural dead-ends; the realistic hardening effort is **8–14 engineer-weeks across both repos**, and five significant backend defects were already professionally resolved in the most recent hardening pass — a positive signal about code trajectory.

3. System architecture

```
RayNeo X3 Pro glasses
  │  (Camera2 → HW H.264 → WebRTC, DTLS-SRTP media; DataChannel results)
  │  (or snapshot JPEG POST /recognize)
  ▼
nginx (TLS, rate-limit, static cache)
  ▼
FastAPI  "museum_api"  (single uvicorn worker)
  ├─ webrtc_stream_service (aiortc, latest-frame-wins buffer=1)
  ├─ recognition_service (dwell state machine: scan→detect→confirm→narrate→cooldown)
  │     ├─ CLIP ViT-L/14 (open-clip)  → 768-d embedding
```

```
  │     └─ FAISS IndexFlatIP + position_mapping.npy
  ├─ tts_service → Chatterbox Multilingual → opus (ffmpeg loudnorm)
  ├─ PostgreSQL 15  (museums→galleries→exhibits→{images,translations,embeddings,narrations}; sessions, events, tours,
api_keys, admin_users)
  └─ Redis 7 (tag-based cache)
Admin panel (React) · Prometheus :9091 · Grafana :3002
```

Data model is well normalized with UUID keys, sensible composite indexes, timezone-aware timestamps, and a narration status workflow (`pending → generating → pending_review → approved/failed`) with reviewer attribution and an Arabic-requires-approval gate. Alembic manages migrations.

4. Strengths (sellable engineering)

Backend

- Clean layered structure (routers → services → models), Pydantic v2 schemas, async SQLAlchemy 2.0 with `selectinload` to avoid lazy-load traps.
- Real-time design done right: size-1 latest-frame buffer with drop counting, FPS-throttled recognition, a versioned DataChannel result schema (`museum.exhibit_results.v1`), in-band language/gallery switching.
- The **dwell-state machine** ("look at an exhibit, hear narration") is genuine product IP with tunable trigger/cooldown knobs.
- **Human-in-the-loop content workflow**: statuses, review/approve/reject, AI-generated flag, Arabic approval gating — a strong story for cultural institutions.
- **i18n depth**: 8 languages including Arabic/Chinese/Hindi/Russian, `name_ar` RTL-aware columns, per-language scripts and narrations, engine support for 23 languages.
- **Museum-tuned audio**: fade in/out, peak normalization, ffmpeg loudnorm, 80 kbps Opus for quiet galleries.
- **Ops posture**: multi-stage CUDA Docker image, non-root user, healthchecks, startup-order gating, nginx TLS + rate limits, Prometheus/Grafana, `/health` vs `/health/ready`, privacy purge endpoint, no-frame-persistence option.

Glasses app

- Small, readable, sensibly packaged (streaming / recognition / overlay / audio / session).
- Correct architecture: WebRTC with hardware H.264 + DataChannel; media is DTLS-SRTP encrypted.
- `ExhibitSmoother` temporal hysteresis (3-frame confirm / 5-frame lost) is a real UX quality touch.
- Modern toolchain (Gradle 9.4.1, AGP 9.2.1, compileSdk 36), scoped network-security config, minimal permissions, 8-language string resources incl. RTL.

5. Findings (prioritized, both repos)

Severity: **C** Critical · **H** High · **M** Medium · **L** Low.

#	Sev	Area	Finding	Fix
B1	C	Backend auth	`Authorization: Bearer <anything>` passes the API-key middleware; `verify_api_key` accepts any signed JWT of any role; admin role not enforced	Validate JWT in middleware; per-route RBAC; remove blanket Bearer pass-through
B2	C	Backend secrets	Live `.env`, TLS private key, and DB creds in the tree; `ADMIN_SESSION_SECRET` defaults to empty → forgeable admin JWTs	Secrets manager / install-time generation; fail-fast on empty secret; scrub tree
A1	C	App secrets	A hardcoded `API_KEY` ships in an unobfuscated APK (R8 off). *(The value is not reproduced in this report, which is distributed; it is cited by file and line in our internal record and must be treated as compromised.)*	Per-device provisioned token in Keystore; rotate key; enable R8
A2	C	App signing	Release keystore passwords committed in `build.gradle.kts`; keystore ships in tree	New keystore; passwords via gitignored `keystore.properties`
A3	C	App transport	All signaling/REST is cleartext HTTP incl. API key; 8 dev IPs whitelisted; user CA trust enabled	TLS + pinning; drop user anchors; strip dev IPs from release

#	Sev	Area	Finding	Fix
B3	H	Backend streaming	WebRTC video frames likely never consumed (no `recv()` pump) — headline stream path needs on-hardware verification	Add track-consumption task; load-test
B4	H	Backend multi-tenancy	`museum_id` ignored in FAISS search — cross-museum matches possible	Filter FAISS by museum_id / per-museum indexes
B5	H	Backend privacy	Unapproved audio + all images publicly reachable under `/static`	Signed URLs or auth on static
B6	H	Backend perf	CLIP inference & FAISS writes block the single event loop (esp. CPU mode)	`asyncio.to_thread` / inference worker
B7	H	Backend perf	TTS generation runs synchronously in the request → 504 past nginx 30 s	Background job + status polling
A4	H	App robustness	No reconnect/recovery on ICE failure — a Wi-Fi roam kills the session until app restart	Backoff restart state machine; wire NetworkMonitor
A5	H	App analytics	Museum/gallery selection unimplemented → sessions/analytics never run on a fresh install	Fetch museums in onboarding / infer server-side
A6	H	App resources	Camera/encoder teardown races init; VideoSource/Track never disposed	Per-resource cleanup on worker thread; CameraEventsHandler
A7	H	App deps	Unofficial WebRTC M119 (Oct 2023) repackage, ~2.5 yrs stale	Move to maintained build; pin upgrade cadence
A8	H	App build	Build not reproducible — AARs, launcher art, keystore untracked; a clean clone fails	Commit/host binaries with licenses; gate signing on file existence
B8/A9	H	Both	**No automated tests, no CI in either repo**	pytest+httpx (backend), JUnit (pure-logic classes), CI pipelines
—	M/L	Both	FAISS↔DB drift durability, N+1 hot-path queries, dead code, thermal-hostile 1440p/20 Mbps capture on glasses, login lockout, dev-compose mount mismatch, Prometheus metrics declared-but-unset	See roadmap §8

6. Already resolved during hardening (credit as fixed)

These were found and professionally fixed in the most recent pass and are verified present in the code:

- **Exhibit-create 500** — relations now eager-loaded (`db.refresh(..., attribute_names=["images","audio_narrations"])`) so Pydantic serialization no longer triggers a lazy load outside the async context.
- **TTS never initialized** — lazy, lock-serialized provider init on first use.
- **Multilingual TTS + watermarker crash** — switched to `ChatterboxMultilingualTTS` ; `perth.DummyWatermarker` fallback for the broken `PerthImplicitWatermarker` in the base image.
- **Event-loop freeze during model load/synthesis** — `asyncio.to_thread` wraps both `from_pretrained` and `generate` , so the healthcheck no longer kills the container mid-load.
- **Persistent-storage data loss** — volume mounts corrected from `/backend/...` to `/backend/app/...` (matching the app's path derivation), so images, FAISS index, and audio survive restarts.

7. Dependencies & licensing

- **Clean for commercial use**: Chatterbox TTS (MIT), open-clip + CLIP ViT-L/14 weights (MIT), FAISS (MIT), aiortc (BSD), OkHttp/Media3 (Apache-2.0), loguru (MIT). **No insightface** in this product — none of its non-commercial model-license baggage applies here (worth stating to buyers).

- **Attention items**: `faiss-gpu-cu12` bundles NVIDIA CUDA runtime (EULA redistribution terms when shipping the image); distro **ffmpeg** may include GPL components (use an LGPL-only build to be safe — only libopus/BSD paths are actually used); `pynvvideocodec` is NVIDIA-proprietary and currently **unused** (remove from requirements to simplify the story); the TTS watermark is currently disabled via the Dummy fallback (a conscious decision to document).
- **Security hygiene before sale**: `python-jose 3.3.0`, `passlib 1.7.4`, `python-multipart 0.0.9`, `starlette 0.38.6` all predate known CVEs/fixes — run `pip-audit` and upgrade. RayNeo AARs (MercurySDK, RayNeoIPCSDK) are vendor binaries with no license files in the tree — clear redistribution rights in any acquisition.

8. Hardening roadmap & effort

Phase	Work	Effort
0 — Demo blockers	Rotate all credentials; new app keystore + secrets out of source; enable R8; commit/host AARs + launcher art with licenses	~1 week
1 — Security	Backend JWT validation + RBAC + fail-fast secrets; TLS + pinning on app signaling/REST; drop dev IPs & user CA trust; per-device tokens	2–3 weeks
2 — Correctness/ robustness	Verify & repair WebRTC frame path; `museum_id` FAISS filter; gate `/static` audio; thread-pool CLIP + background TTS jobs; app reconnect state machine, camera events, teardown fix	3–4 weeks
3 — Maturity	pytest + JUnit suites, CI, `pip-audit`, dependency upgrades; FAISS rebuild-from-DB durability; DB-backed hashed API keys/RBAC; museum selection in app; thermal-adaptive capture ladder	3–5 weeks
4 — Ops polish	Wire declared Prometheus metrics / real GPU exporter; remove dead modules; gate `/docs` in prod; README + protocol docs	ongoing

Total to production-grade: ~8–14 engineer-weeks across both repos.

9. What a buyer is actually acquiring

- A **working, differentiated product** with proven multilingual hands-free museum UX and an on-prem, data-sovereign architecture that fits GCC procurement.
- A **clean, readable codebase** a competent team can own within days, on a modern toolchain.
- **Defined, bounded technical debt** with a costed remediation plan — not open-ended risk.
- Genuine IP in the dwell-recognition UX and the Arabic-aware content/approval pipeline.