Technical Due-Diligence Report

HumanityAI'd Patrol Sight — on-premises smart-glasses face recognition

Swarm stack (34 services) + Face API v3.3.1 + RayNeo glasses app — v1.1.7

July 2026

CONFIDENTIAL — PREPARED FOR PROSPECTIVE CUSTOMERS, PARTNERS & ACQUIRERS

1. Purpose & scope

This report is an honest, buyer-grade technical assessment of the system deployed as the Docker Swarm stack `patrol-sight`:

- **The recognition engine** — `patrol-sight-ai` (Face API v3.3.1 "Scalable Edition"): FastAPI, SCRFD detection, ArcFace ResNet-50 embeddings on TensorRT FP16, GPU FAISS IVF-PQ search, PostgreSQL 15, Redis 7, NATS, MinIO.
- **The command & control plane** — C2 frontend and NodeHub device hub with AES-256-GCM payload encryption, plus auth-service (JWT), backend business API, and the Targets/labels/tracking-history domain.
- **The edge app** — the RayNeo X3 Pro Android streamer (Camera2 → hardware H.264 → RTP/UDP), with in-lens label overlay and client-side label smoothing.

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 **deployment-grade debt** that must be closed before a production security sale. A conservative hardening estimate is included.

2. Verdict in one paragraph

Patrol Sight is a **working, measured, end-to-end system** — not a prototype. Independently timed on real hardware, a face crosses from the officer's glasses to a labelled overlay in **~68 ms typical (49–103 ms range)** at 14–20 effective recognition fps, searching a **93,728-person GPU FAISS index** with a documented path to 13M vectors at 80–130 ms. The engineering contains genuine differentiated work: a two-layer tracker that skips the vector search on **97.6% of frames**, a newest-frame-wins ingest that refuses to build GPU backlog, an H.264-over-UDP transport chosen explicitly for battery- and thermal-limited head-worn devices, and a 34-service Swarm topology with per-tier update strategies and NATS Nkey zero-trust between services. It is, however, **a production deployment carrying deployment-grade security debt**: default and committed credentials in the tree (including a live master API key inside the Android source), datastore ports published on the host alongside nginx, a plaintext video path that is safe only because the glasses LAN is physically isolated, and a Docker-socket mount in the ops container. None of these are architectural dead-ends. The realistic hardening effort is **10–16 engineer-weeks**, and the most recent release notes show the team diagnosing and fixing real production incidents (pgbouncer pool exhaustion, TensorRT engine mispairing, false-positive cache layer) with candid write-ups — a positive signal about code trajectory.

3. System architecture

```
RayNeo X3 Pro glasses  (Camera2 -> MediaCodec HW H.264 -> RTP/UDP :5557)
  |                     (results returned as JSON datagram -> UDP :17391)
  v
nginx  (mode: global; intended sole public surface :80/:8080/:443/:9000)
  v
Tier 3 GPU:  app = patrol-sight-ai  (:5555 TCP, :5556 UDP JPEG, :5557 UDP H.264)
  |- h264_stream_service   NVDEC-preferred decode (~1-2 ms), newest-frame-wins reader,
  |                        FPS-capped async worker so the GPU never builds a backlog
  |- Pipeline V3           SCRFD det_10g -> 5-pt affine align 112x112
  |                        -> ArcFace w600k_r50 512-d on TensorRT FP16
  |                        -> FAISS IVF4096,PQ64x8 on GPU (nprobe 64)
  |- two-layer tracker     L1 IoU bbox -> tracking_id; L2 identity cache
  |                        (FAISS search skipped on 97.6% of frames)
  |- worker                FAISS shard rebuild + hot reload via app:5555
  \- ai-quality-engine     face-quality assessment (FQA v4)
  v
Tier 2:  recognition-server (NATS v1.vision.frame.* -> publishes v1.vision.alert)
         backend (FastAPI) -> pgbouncer -> PostgreSQL 15  |  auth-service (JWT)
         node-api (NodeHub device hub, AES-256-GCM payloads) -> C2 frontend
         ar-settings-api + ar_postgres (glasses configuration context)
  v
Tier 1:  PostgreSQL 15 (business humanity_aid_db + AI facedb) - MinIO S3 - Redis x4
Tier 4:  coturn (STUN/TURN) - nats - cloudflared
Tier 5:  museum-* services on an isolated overlay (separate product, same host)
```

Deployment model. One GPU host, one Swarm stack, 34 services across five tiers with per-tier update strategy (GPU tier: stop-first, parallelism 1). All 22 data volumes are declared **external** so `docker stack rm` cannot delete customer data. A single GPU is shared by CUDA time-slicing; the multi-GPU path (`generic_resources`, one GPU per replica) is written and commented out, ready to enable.

4. Strengths (sellable engineering)

Recognition engine

- **Measured, not claimed, latency.** A full stage-by-stage budget exists for the glasses path (capture 12 ms, transit 3 ms, decode 7 ms, detect 20 ms, embed 8 ms, search 5 ms, return 3 ms, render 7 ms), independently corroborated by a WebRTC path measured with clock synchronisation at **p50 26–32 ms / p95 63–85 ms**.
- **The two-layer tracker is genuine product IP.** IoU bbox matching yields a stable `tracking_id`; an embedding+identity cache then answers 97.6% of frames without touching FAISS — cutting average per-frame latency 65 → 30 ms and GPU utilisation 85% → 40%. This is what makes continuous patrol recognition affordable on one GPU.
- **Backpressure done right.** The H.264 ingest uses a newest-frame-wins reader thread plus an FPS-capped async worker, an explicit design choice "so the GPU never builds an unbounded backlog" — the failure mode that turns live video AI into a growing latency spiral.
- **Transport engineering with a documented rationale.** H.264 was chosen over JPEG for "~14–30× less bandwidth and far fewer packets per frame, which matters for battery/thermal-limited head-worn devices"; a written comparison of custom UDP/RTP vs WebRTC/RTSP/SRT/QUIC concludes with a reasoned decision rather than a default. NVDEC decode (1.3–2.7 ms vs 15–25 ms on CPU) and a 5 GHz WiFi 6 migration (ping ~100 ms → ~4 ms) are both quantified.
- **Search that scales on paper and in memory.** GPU FAISS IVF-PQ gives 27× compression (13M vectors: 962 MB vs 26 GB flat), with measured P95 of 8.5 ms over 1,931 real searches and a documented shard/nlist/nprobe plan to 13M. Batch search is 19.5× faster at 100 faces.
- **Model choice was benchmarked, not assumed.** A 0–90° pose study across 24 images showed ArcFace R50 beating MobileFaceNet in 19 of 21 comparisons (+0.054 mean similarity; **~60% fewer false negatives at extreme pose**), justifying the heavier model — exactly the evidence a security buyer needs, since off-angle faces are the operational norm.
- **Operational maturity in the engine**: adaptive nprobe tuning, a circuit breaker, index-integrity checks, cache preloading, GPU warm-up (removes ~170 ms cold start), a documented ADC-vs-cosine threshold calibration (`0.45 ADC ≈ 0.52 exact cosine`), and audit middleware with an operator-facing AuditLogs page.

Platform & ops

- 34-service Swarm topology with health-gated startup ordering, an Alembic migration gate (`orm`) that blocks the backend until schema is current, and nginx blocking on service DNS resolution before it execs.
- `deploy.sh` is a real operations tool (`up/down/restart/update/status/logs/health/pull/config/cleanup`) with hard-won specifics: it refuses to run outside Swarm, deploys with `--resolve-image=never` because "a registry hiccup otherwise wedges GPU services in 'preparing' forever", and polls teardown to completion.
- `gpu-startup-wrapper.sh` retries specifically on `Faiss assertion … CUDA error 999` after waiting for `nvidia-smi` — a boot race that bites GPU containers, handled deliberately.
- Boot autostart via a systemd oneshot unit plus Swarm state restoration; `.swarm-state/` keeps pre/post-deploy image snapshots, and each release keeps a byte-for-byte copy of the stack file that produced it.
- **NATS Nkey zero-trust auth per service** and **AES-256-GCM payload encryption** between NodeHub and the C2 frontend (raw `curl` against the API returns 400 by design) — meaningfully above the norm for an on-prem appliance.

Edge app

- Correct low-latency architecture: Camera2 straight into the MediaCodec hardware encoder surface, deliberately avoiding ImageReader/JPEG/NV21 conversion and per-frame pixel handling.
- Real UX quality work on label stability: EMA smoothing (alpha 0.45) with spatial proximity matching, 3-second stale eviction, IoU-based coasted-track suppression to kill double labels, and a fix where UUID tracking IDs had been parsed as integers.
- Colour-coded in-lens label pills via a compact `name@@label@@#colour@@` wire format — the officer sees threat category, not just a name.
- One-command provisioning (`deploy-glasses.sh`) that auto-detects the device, builds R8-optimised release, installs and launches.
- Per-device sessions keyed by `device_id` , and a **privacy-preserving mode that sends the 512-float faceprint (~2 KB) instead of the image (~100 KB)** — a strong answer in privacy-sensitive deployments.

5. Findings (prioritized)

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

#	Sev	Area	Finding	Fix
S1	C	Secrets in tree	A **live master API key is committed in Android source** (`StreamConfigUtils.java`) and ships inside the APK; the same key prefix appears in an ops runbook	Rotate the key now; per-device provisioned tokens in Android Keystore; scrub history
S2	C	Default credentials	Stack file hardcodes defaults for the admin password, master API key, database password and `ADMIN_SESSION_SECRET` ; TURN users and the Grafana admin password likewise. *(Values are not reproduced in this report; they are cited by file and line in our internal record and must all be treated as compromised.)*	Install-time secret generation; fail-fast on defaults; secrets manager or Docker secrets
S3	C	Network exposure	Postgres, Redis, NATS, backend, recognition, custom-recognition and AR services publish host ports alongside nginx — the file itself flags this and ships commented-out `LOCAL_IP_ADDRESS` bindings	Bind all non-nginx ports to the private LAN IP or drop them; make nginx genuinely the sole surface
S4	H	Transport security	Glasses video and returned identity results are **plaintext UDP** — safe only while the glasses LAN is physically isolated with no uplink	SRTP/DTLS or WireGuard on the glasses VLAN for any deployment that cannot guarantee isolation; keep the isolation requirement contractual until then
S5	H	Privilege surface	`db-manager` mounts the host Docker socket to run migrations via `docker exec` ; privileged `gpu-`	Replace socket mount with a migration job in the Swarm; drop capabilities; document the residual risk

#	Sev	Area	Finding	Fix
			`init` and `coturn` are flagged in-tree as "the highest-risk nodes in the topology"	
S6	H	Config correctness	The stack's default `TRT_ENGINE_PATH` points at a path that is a **directory** in the v3.2.0 image — load fails and **silently falls back to ONNX**, quietly losing the TensorRT speed-up unless `.env` overrides it	Fail loudly when the engine cannot deserialize; ship correct defaults; assert engine↔model pairing at startup
S7	H	Capacity	Single GPU shared by time-slicing across `app`, `worker` and FQA — one host serves one site, and concurrent-glasses capacity is not yet load-tested to a published number	Enable the pre-written multi-GPU path; publish a measured concurrent-device ceiling per GPU class
S8	H	Tests/CI	No automated test suite or CI is evident for the stack and edge app (the engine repo cites passing streaming/API/behavioural suites, but stack-level and Android coverage is absent)	pytest + httpx at engine level, JUnit for pure-logic Android classes, CI on both; smoke-test the deployed stack
S9	M	Watchlist integration	There is **no third-party watchlist connector** — the shipped equivalent is the C2 Targets/labels model plus per-person enrolment; external watchlist sync is listed as a future item	Build the connector, or scope it explicitly in every bid so it is priced rather than assumed
S10	M	Consistency	Recognition threshold appears as 0.35 / 0.45 / 0.50 across documents and detection as 0.5 / 0.60 / 0.65; glasses host IP differs across five documents	Single source of truth for tunables; per-site config profile; delete stale values
S11	M	Pooling	pgbouncer runs `pool_mode=session` with the pool shared by auth-service and backend; the v1.1.7 login-504 incident was fixed by raising the pool 10→40, documented in-tree as "headroom, not a cure"	Move to transaction mode once the asyncpg DSN sets `prepared_statement_cache_size=0`
S12	M/L	Docs & drift	`ARCHITECTURE.md` and the project README describe the older compose topology (bridge network, hotspot mode) and one README has corrupted opening text; a second `recapi` compose stack coexists on the host with older image tags	Regenerate docs from `docker-stack.yml`; retire or clearly separate the parallel stack

6. Already resolved during hardening (credit as fixed)

These were diagnosed and fixed on the live system, and the reasoning is recorded in-tree:

- **Login 504s under load** — traced to pgbouncer session-mode pool exhaustion shared between auth-service and backend; pool raised 10→40 with an explicit note that the durable fix is transaction mode.
- **False positives from the deepest cache layer** — the PostgreSQL cosine fallback (L4) was removed from the glasses path because it "caused false positives + 3–12 ms latency per miss".
- **False detections at high detector resolution** — raising `DET_CONF_THRESHOLD` from 0.35 to 0.60 removed 2–11 spurious detections per frame at det_size 1920.
- **Label instability in the lens** — EMA smoothing plus spatial matching, stale eviction, and IoU coasted-track suppression; UUID tracking IDs had been parsed as integers (returning array indices), now hashed as strings.
- **GPU cold-start and boot races** — a warm-up dummy search removes ~170 ms first-request latency, and the startup wrapper retries specifically on the FAISS CUDA-999 assertion after waiting for the driver.
- **Bandwidth and radio ceiling** — migrating the glasses link from a 2.4 GHz hotspot (72 Mbps, ~100 ms ping with power save) to 5 GHz WiFi 6 (480 Mbps, ~4 ms) enabled 2560×1440 at 20 Mbps.
- **Decode cost** — NVDEC brought H.264 decode from 15–25 ms on CPU to 1.3–2.7 ms; a DSCP-marking experiment was measured, found to double the p95 tail, and reverted rather than kept on faith.

7. Dependencies & licensing

- **Attention item — model licensing.** The recognition stack uses **InsightFace-lineage models** (SCRFD `det_10g`, ArcFace `w600k_r50`). The InsightFace pretrained models are published for **non-commercial research use**; shipping them in a commercial security product requires either a commercial licence from the rights holder, replacement with commercially-licensed weights, or training equivalent models on properly licensed data. **This is the single most important legal item to resolve before a paid deployment** and should be planned for explicitly, not discovered in diligence.
- **FAISS** (MIT), **PyTorch/ONNX Runtime** (BSD/MIT), **FastAPI/Starlette/Pydantic** (MIT/BSD), **NATS** (Apache-2.0), **MinIO** (AGPL-3.0 — note: AGPL obligations attach to modified network-served versions; the unmodified server used as a component is the common path, but the licence should be reviewed for a shipped appliance), **PostgreSQL** (PostgreSQL licence), **Redis 7** (RSALv2/SSPL for recent versions — pin an appropriate version or move to Valkey if redistribution matters), **coturn** (BSD).
- **NVIDIA components**: TensorRT, CUDA runtime and PyNvVideoCodec/NVDEC carry NVIDIA EULA redistribution terms when shipped inside an image; `faiss-gpu` bundles CUDA runtime similarly. **ffmpeg** from a distro may include GPL components — use an LGPL-only build for a shipped appliance.
- **RayNeo vendor SDKs/AARs** are binaries with no licence files in the tree — clear redistribution rights before shipping a fleet image.
- **Security hygiene before sale**: run `pip-audit` across the engine and C2 dependency sets and upgrade; the legacy Django C2 backend carries its own CRITICAL committed-secrets findings (`DEBUG=True`, weak keys) and should be retired or hardened rather than shipped.

8. Hardening roadmap & effort

Phase	Work	Effort
0 — Deployment blockers	Rotate the committed API key and all defaults; per-device tokens for glasses; secrets out of source and out of the APK; bind non-nginx ports to the LAN IP; fix the TensorRT engine default and fail loudly on fallback	~1.5 weeks
1 — Legal & compliance	Resolve model licensing (commercial licence or replacement weights); LGPL ffmpeg build; clear RayNeo AAR redistribution; produce the data-protection pack (retention, purge, audit) for PDPL/SIRA review	2–3 weeks (plus vendor lead time)
2 — Security	Encrypt the glasses transport (SRTP/DTLS or WireGuard); remove the Docker-socket mount; RBAC across C2 and engine APIs; hashed DB-backed API keys; drop or gate the legacy Django C2	3–4 weeks
3 — Maturity	pytest + JUnit + CI; deployed-stack smoke tests; pgbouncer transaction mode; single source of truth for tunables; published concurrent-device capacity per GPU class; multi-GPU path enabled and measured	3–5 weeks
4 — Product gaps	External watchlist connector; audio/haptic alert back to the wearer; behaviour/anomaly analytics; regenerate architecture docs from the Swarm file	2–4 weeks

Total to production-grade for a government security sale: ~10–16 engineer-weeks, of which the legal/licensing item is the longest-lead and least engineering-dependent.

9. What a buyer is actually acquiring

- A **working, measured system** — sub-100 ms glasses-to-label recognition against a ~94k-person index on a single mid-range GPU, with the stage-by-stage evidence to back every number.
- **Real algorithmic IP** in the two-layer tracker (97.6% search elision), the backpressure-correct ingest, and the transport/model choices that were benchmarked rather than guessed.
- A **complete operational envelope**: 34-service Swarm topology, one-command deploy/health tooling, boot autostart, external volumes, release snapshots, and per-service zero-trust messaging.
- **Defined, bounded debt** with a costed remediation plan — including the model-licensing item stated up front rather than buried.
- An **architecture that already fits GCC data-residency law**: recognition, watchlist and analytics never leave the customer's premises, which is where cloud face-recognition vendors cannot follow.