## **Identity & Sync Architecture – GDPR-Compliant Profile Backup & Model Delivery**

### **Context & Goals**

The WaW desktop MVP (PyQt UI, on-device ML) currently manages user profiles locally. By Q3 2025, we need:

* **Profile Backup:** Secure cloud storage & cross-device sync.
* **Model Delivery:** Remote updates of model.bin.
* **GDPR Compliance:** Opt‑in, data minimization, right to erasure.

**Non‑Goals:** No UI/design changes beyond sync; no large-scale multi‑tenant infra; no explicit auth.

### 

architecture diag.

### **Architecture Overview**

**Client:**

* **UI:** PyQt interacts via gRPC to local services.
* **Identity Service:** gRPC CRUD on encrypted profile.db (SQLite+SQLCipher/AES‑256).
* **Sync Service:** Background process polling or watching for changes; performs REST calls.

**Cloud:**

* **Cloud Sync API:** REST (POST /v1/profile, GET /v1/model/latest) over HTTPS.
* **Profile Backup Store:** Encrypted store (prototype: in‑memory or file; prod: blob DB).
* **Model Repository:** Static file or object store for model.bin.

### **Data Flows**

**Profile Backup:**

1. UI → Identity Service (UpdateProfile via gRPC).
2. Identity Service → encrypted DB, updates timestamp.
3. Sync Service detects change → POST /v1/profile JSON.
4. Cloud API stores encrypted profile → acknowledges.

**Model Update:**

1. Periodic GET /v1/model/latest with current checksum.
2. If new, download model.bin → verify SHA‑256 → replace/retain old copy.
3. ML component reloads or schedules update.

### **Key Decisions**

* **Protocol:** gRPC locally (efficient, typed); REST for cloud (simple, firewall‑friendly).
* **Encryption:** SQLCipher for DB; TLS for transport; SHA‑256 for model integrity.
* **Versioning:** API versioning (/v1/); model and schema migrations.
* **Resilience:** Idempotent REST, retry/backoff, independent Sync process, rollback.

### **Privacy & Compliance**

* **Opt‑in:** Backup off by default; UI toggle.
* **Erasure:** DeleteProfile clears local and cloud data.
* **Minimization:** Only essential profile fields; no history unless requested.
* **Audit:** Log sync events; enforce data residency.

### **Phased Rollout**

1. **Local Prep:** Encrypt DB; expose gRPC methods.
2. **Mock Sync:** Implement Sync Service with mock API.
3. **Cloud API:** Spin up REST service; basic endpoints.
4. **Integration Tests:** End‑to‑end offline/online and opt‑in/out scenarios.
5. **Beta Rollout:** Sync disabled by default; enable for testers.
6. **General Release:** UI opt‑in control; monitor and iterate.

### **Future Enhancements**

**Option A: S3 Intelligent‑Tiering + CDN**

* Auto‑tier storage, global distribution, low ops.

**Option B: MongoDB + RAG Sync**

* Change Streams, semantic deltas via Atlas Vector, cost‑effective sharding.

**Roadmap:**

* Phase 1–2: Rust modules for sync loops; feature‑flag RL agent.
* Phase 3: Event‑driven sync (WebHooks/gRPC streams).
* Phase 4: Test S3 and MongoDB paths; A/B measure.
* Phase 5: Select/hybrid approach; retire legacy store.