**Data Management — Operational Spec (2-page version)**

**Overall purpose. Persist user identity/links, contact-request workflow, device push tokens, and per-call scam analysis/alerts. The store is optimized for mobile workloads (short writes, indexed reads) and simple cross-collection joins via string keys (no DBRefs).**

**Common conventions (all collections).**

* **Identifiers. Users identified by phoneNumber (E.164, unique) and optional firebaseUid (unique, sparse). Other collections reference users via userId (Firebase UID) and/or phoneNumber.**
* **Timestamps. createdAt, updatedAt set by Spring Data; soft delete via deletedAt (readers filter deletedAt == null).**
* **Indexes. Uniques and compound indexes enforce integrity and speed lookups (detailed per collection).**

**1) users — Canonical user profile**

**Purpose. Single source of truth for user identity, current links (trusted contact / protegee), and historical links.**

**Construction & Updates.**

* **Create at registration or first profile setup.**
* **Update on profile edits or link changes (approval/denial/cancel).**
* **Soft delete by setting deletedAt (retain for audit/restore).**

**Key fields (selected).**

* **phoneNumber (unique, E.164), firebaseUid (unique sparse), name, userType (Protegee|TrustedContact|Both).**
* **linkedContacts[] (current) — items contain phoneNumber, name, relationship, contactType (TRUSTED\_CONTACT|PROTEGEE), connectedAt, removedAt:null.**
* **oldLinkedContacts[] (archive) — same shape; entries moved here with removedAt populated.**

**Indexes & integrity.**

* **{ phoneNumber: 1 } unique; { firebaseUid: 1 } unique, sparse.**
* **Application logic ensures a contact cannot be simultaneously in current and archived sets.**

**Typical reads/writes.**

* **Get profile by auth: users.findOne({ firebaseUid, deletedAt:null }).**
* **Link add/remove: modify linkedContacts atomically; move removed entries to oldLinkedContacts.**

**2) contact\_requests — Linking workflow inbox**

**Purpose. Tracks requests to form trusted contact/protegee relationships, including unregistered targets (by phone).**

**Construction & Updates.**

* **Create when a user sends a request (requesterUid, targetPhoneNumber, contactType, optional names/relationship).**
* **Transition status on approve/deny/cancel/expire; set respondedAt on decision.**
* **Soft delete optional; otherwise keep for audit.**

**Key fields (selected).**

* **requesterUid (sender, indexed), requesterName.**
* **targetPhoneNumber (indexed) and targetUid (filled when target registers/linked).**
* **contactType (TRUSTED\_CONTACT|PROTEGEE, indexed), relationship (free text).**
* **status: PENDING|APPROVED|DENIED|EXPIRED|CANCELED, createdAt, respondedAt, deletedAt.**

**Indexes & integrity.**

* **Compound (non-unique):**
  + **{ requesterUid:1, targetPhoneNumber:1, contactType:1 } (validation/dup detection).**
  + **{ targetPhoneNumber:1, status:1 } (recipient inbox).**
  + **{ requesterUid:1, status:1 } (sender outbox).**
* **Application prohibits conflicting active requests (e.g., multiple PENDING for same triple).**

**Typical reads/writes.**

* **Inbox for target: by targetPhoneNumber + PENDING.**
* **Approve/Deny: update status, move link into users.linkedContacts on approval.**

**3) device\_tokens — Per-device FCM registration**

**Purpose. Maintain one record per (user, device) plus a global uniqueness on FCM tokens to ensure clean push routing and rollout targeting.**

**Construction & Updates.**

* **Upsert on app register/refresh using (userId, deviceId) as the key.**
* **Replace fcmToken when the client refreshes; update platform, appVersion, lastUpdated.**

**Key fields (selected).**

* **userId (Firebase UID, indexed), deviceId, fcmToken (unique).**
* **platform (android|ios|web), appVersion, lastUpdated (epoch millis).**
* **createdAt, updatedAt, deletedAt.**

**Indexes & integrity.**

* **Unique compound { userId:1, deviceId:1 } (one row per device).**
* **Unique { fcmToken:1 } (no duplicates across ecosystem).**
* **{ userId:1 } for fan-out and cleanup.**

**Typical reads/writes.**

* **Send notifications: fetch tokens by userId.**
* **Cleanup: soft delete or overwrite on unregister.**

**4) call\_records — Per-call analysis and alerts**

**Purpose. Persist results of scam analysis per call/event: risk, transcript, scoring, timing, and outbound message linkage.**

**Construction & Updates.**

* **Create on incoming report; fill userId, eventId (if provided), metadata.**
* **Update when model analysis completes or when notification messageId is known.**

**Key fields (selected).**

* **userId (indexed), eventId (with userId unique), callerNumber.**
* **riskLevel, modelScore, modelAnalysis, transcript, durationInSeconds, occurredAt.**
* **messageId (FCM reference), createdAt, updatedAt, deletedAt.**

**Indexes & integrity.**

* **Unique compound { userId:1, eventId:1 } (prevents duplicate ingestion of same event for a user).**
* **{ userId:1 } for history queries, { eventId:1 } supports de-dup checks.**

**Typical reads/writes.**

* **Recent calls: by userId sorted by occurredAt desc.**
* **De-dup: upsert on (userId,eventId) when client sends an external call ID.**

**Cross-collection identity & relationships**

**Keys.**

* **User identity: phoneNumber (primary, unique) and optional firebaseUid (unique, sparse).**
* **References: other collections store userId (Firebase UID) and/or targetPhoneNumber. No DBRefs—plain strings to keep queries simple and portable.**

**Core flows.**

1. **Invite/linking: create contact\_requests → on approval, push linked pair into users.linkedContacts for both sides (as appropriate) and archive prior links to oldLinkedContacts.**
2. **Notifications: resolve device\_tokens by userId; broadcast with fcmTokens.**
3. **Alert ingestion: write call\_records; later enrich with modelAnalysis and messageId.**

**Data hygiene, privacy, and retention**

* **Auditability: createdAt, updatedAt across all collections.**
* **Soft delete: set deletedAt; all app reads exclude deletedAt != null.**
* **Integrity: unique/compound indexes listed above; service layer rejects conflicts (e.g., duplicate active requests, token collisions).**
* **PII handling: phoneNumber, transcripts, and analysis are PII/PHI-adjacent—restrict access by role; redact transcripts as required; enable TTL/retention policies if mandated (e.g., archive old call\_records after N days).**
* **Backups & recovery: snapshot MongoDB; verify index recreation in restore procedures.**