Review the Audit Trail

Admin Read the history of consequential changes, annotate and correct it, but never rewrite it.

Screens Audit trail `audit-trail.html` · Audit detail `audit-detail.html`

Writes note edits in place `audit_log.note` · corrections as new rows `corrects_entry_id`

Key fields who `actor_id + actor_label` · what `action` · on what `target_type · target_id` · before / after `before_state · after_state` · explanation `note` · supersedes `corrects_entry_id` · which run `run_id`

1 · Given / When / Then

Human wording first; the exact table/field in `mono`.

GIVEN	The signed-in person is an **admin**. Consequential changes across the install already write to `audit_log`.
WHEN	On the **Audit-trail Screen** they filter and read the history, open an entry on the **Audit-detail Screen** to see the exact `before_state` and `after_state`, add an explanatory `note`, or, when an entry turns out to be wrong or misleading, post a correction.
THEN	Adding a note is the one edit the trail allows: it updates only that entry's `note` column and creates no new row. A correction, by contrast, never touches the original — it appends a new `audit_log` row whose `corrects_entry_id` points at the entry it supersedes, and the original is never rewritten or deleted. Either way the record shows both what was originally logged and any later explanation or correction. **reads** `audit_log` **writes** a `note` edit, or a new correction row

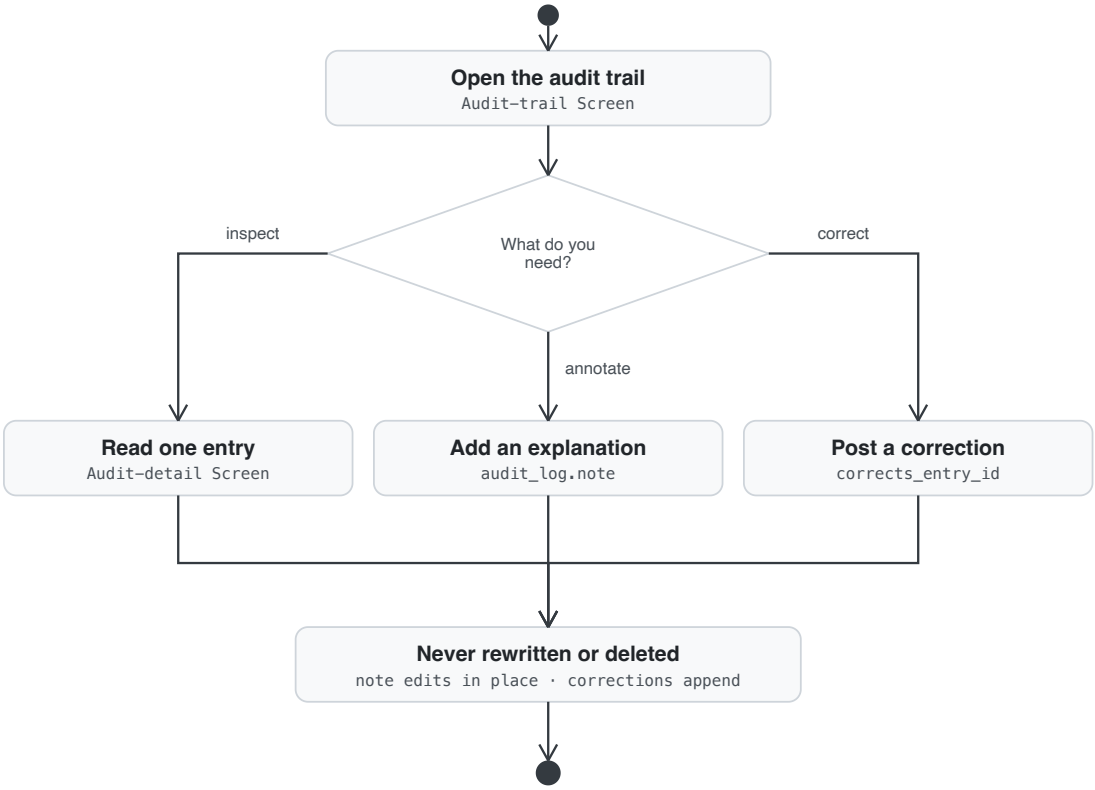
Variations and exceptions

IF…	THEN…
The actor's profile was later disabled, or its handle changed	Entries stay readable: `actor_label` stores the name as it was at the time.
A correction is itself wrong	It is corrected by another entry pointing at it; the chain may be more than one link long.
The run an entry belongs to is cleaned up or purged	Its run-scoped entries (`audit_log.run_id` set) go with it. Install-level entries — roles, catalog, documents — have no `run_id` and are unaffected.

Acceptance test — *Given* an existing audit entry, *When* an admin posts a correction, *Then* a new `audit_log` row exists with `corrects_entry_id` set, the original row is byte-for-byte unchanged, and the detail screen shows both.

2 · The Journey, Screen by Screen

Three things you can do; none of them destroys anything.



3 · In Plain English

The audit trail answers one question: who changed what, and what did it look like before and after. The consequential things are recorded here: roles granted, models retired, documents deleted, runs purged.

It is append-only but for one field, and that is the whole design. The single exception is the `note`: you can annotate an entry in place to explain it, which updates only that note and nothing else. To fix a wrong entry you never rewrite it — you add a new entry that points back at the old one and says so. Anyone reading later sees the original claim, any note, and the correction, which is exactly what makes the record worth trusting. A trail you can quietly rewrite proves nothing.

Entries record the actor's name as it was at the time, so the history stays legible even after people leave or change their handle. And entries tied to a particular run travel with it, so purging a study's data does not leave orphaned references behind.

Fits the schema cleanly. The trail is write-once but for one field: row-level security allows updating only the `note` column (the in-place annotation) and otherwise never updates or deletes `audit_log`; `corrects_entry_id` is the supersede link and `run_id` the run-scope tie.