

Search & verification

Harnesses and golden vectors

🕒 4 min read · Updated 27 September 2026 · Verified against [aced1a0](#) on 2026-09-27

What keeps search and verification honest: three regression harnesses that measure recall on real seeker behaviour, and the golden contract that pins every engine's output across deploys and the app.

<https://docs.gurbanisoul.com/search/harnesses-and-golden-vectors/>

Two kinds of proof guard the engines. **Harnesses** measure how well search serves real seeker behaviour, so a change can be judged corpus-wide rather than on a few examples. **Golden vectors** record exact outputs, so no change to the fold, the waterfall or the verifier can ship unnoticed — here, over HTTP against every deploy, or in the app's Swift port.

The harnesses

Harness	What it types	What it measures	Run
<code>tools/roundtrip_harness.py</code>	every corpus line's <i>own</i> transliteration (or a sample, an Ang range, a word cap)	does the line come back at rank 1 / in the top 3 — the “does every line resolve when a user types it” metric; failures bucketed by form and Ang range	<code>python3 tools/roundtrip_harness.py [--sample N]</code>
<code>tools/casual_quote_harness.py</code>	the first ~6 words of a real line, casualised : modern postpositions (<code>kai</code> → <code>ka</code> , <code>kee</code> → <code>ki</code>), dropped aspirates (<code>khelio</code> → <code>kelio</code>), shortened vowels (<code>kool</code> → <code>kul</code>)	top-1 / top-3 recall for the way seekers half-remember a verse (the class that produced the <i>jamuna ka kul</i> <i>khel kelio</i> bug)	<code>python3 tools/casual_quote_harness.py [--sample N]</code>

Harness	What it types	What it measures	Run
tools/chaos_harness.py	attack files: {agent, behavior, query, canonical_translit, canonical_ang}	PASS iff a top-3 result is the canonical line (transliteration match, Ang±1), against a running server	python3 tools/chaos_harness.py "qa/chaos/attacks_*.jsonl" qa/results/chaos.jsonl

The first two run `do_search` in-process and write per-line JSONL under `qa/results/`; `make harnesses` runs both. Re-run them before and after any change to a tier, a weight or the fold, and compare the summaries: the waterfall is tuned corpus-wide, and a local improvement often costs recall elsewhere.

Where the numbers stand

Measured on the pinned dataset at integration `aced1a01` (2026-09-27). Use these as the baseline a change is compared with:

Harness	Scope	pass@1	pass@3
round-trip	all 55,510 verse lines, their own transliteration (15 s)	55,114 (99.3%)	55,490 (100.0%)
casual-quote	600 lines, seed 7 (1 s)	585 (97.5%)	596 (99.3%)
chaos	200 attacks against a running server	—	156 (78%)

- **Round-trip:** most of the lines not at rank 1 are refrains printed several times; the weakest Ang range is 1401–1430 (97.9% at rank 1).
- **Chaos, by behaviour:** 20/20 for bilingual, emoji, honorific, fragment, substitution and SMS queries; grammar 18/20; sound-alike spelling 14/20; voice-to-text 3/20; keyboard-smash 1/10 with spaces and 0/10 without. The committed `qa/chaos/chaos_FINAL.jsonl` records 159/200 (the first run, `chaos_baseline.jsonl`, 73/200); today's run is three below it, and the cause has not been traced.

None of the three gates CI on a threshold. `casual_quote_harness.py` runs in the python job and fails it only if it crashes; the round-trip harness runs with `|| true`; the chaos harness is not run by any workflow and expects the server on port 7777. The golden contract below is what blocks a change.

The golden contract

`tools/gen_golden_vectors.py` drives the **real** functions — `roman_norm`, `difflib`, `do_search`, `verify`, the reader, timing, banis, analytics — and records their exact outputs under `contract/`. The files and their record counts, from `contract/_meta.json`:

contract/_meta.json · lines 1-20

```
{
  "db_sha256": "4e6777c54140d7c4ff76fd06b5453991daf4cabcb2c98479d216908d4c31455b",
  "files": {
    "golden_analytics.ndjson": 56,
    "golden_banis.ndjson": 28,
    "golden_constellation.ndjson": 31,
    "golden_difflib.ndjson": 49,
    "golden_pahar.ndjson": 2183,
    "golden_reader.ndjson": 27,
    "golden_roman_norm.ndjson": 24719,
    "golden_search.ndjson": 68,
    "golden_timing.ndjson": 83,
    "golden_verify.ndjson": 18
  },
  "generator": "tools/gen_golden_vectors.py (+ frontend/scripts/gen-pahar-vectors.mjs)",
  "tool_versions": {
    "python": "3.14.7",
    "sqlite": "3.53.4"
  }
}
```

Read from the source at build time — [view these lines on GitHub at c3d9360](#) .

Source: [contract/_meta.json](#)² — the counts and the database the vectors were built against.

File	Pins
golden_roman_norm.ndjson	the fold: 24,719 input → output pairs (The Roman fold)
golden_difflib.ndjson	<code>SequenceMatcher.ratio()</code> values, repr-exact, so the port's scorer equals Python's
golden_search.ndjson	real queries through the waterfall and their results
golden_verify.ndjson	claims and their full verdicts (Verification engine)
golden_reader.ndjson	Ang, shabad and Hukam projections
golden_timing.ndjson, golden_banis.ndjson, golden_analytics.ndjson	the knowledge, Nitnem and insight layers
golden_pahar.ndjson	the pahar clock (generated by <code>frontend/scripts/gen-pahar-vectors.mjs</code>)
openapi.json	the APIs shape, inferred from real responses (Contract and OpenAPI)

`_meta.json` also records the `db_sha256` the vectors were generated from; the scripture-integrity workflow requires it to equal the [dataset pin](#).

Where the contract is enforced

Where	How
this repository, every PR	<code>make contract</code> : regenerate and <code>git diff --exit-code contract/*.ndjson</code> — any drift must be a deliberate, reviewed regeneration

Where	How
every deploy (staging and production), the API image	<code>tools/contract_http.py --base <origin></code> replays the same projections over HTTP and compares by parsed-JSON equality
through the staging gateway	the whole contract passes through <code>sggs-staging.vercel.app</code> , proving the split services answer exactly like the monolith
production, every 6 hours	the data canary replays the contract against production
gurbani-soul-ios	<code>vendor.lock.json</code> pins every contract file by sha256 at a platform release; byte-parity tests hold the Swift port to them

A change to the engines therefore has one honest path: change the code, re-run the harnesses, regenerate the contract, review the diff, release, and let the app re-vendor at the new tag.

Links

1. https://github.com/Algorythmos-AI/sggs-platform/blob/c3d936031f0c283283c5d1102705affe1f8cf404/contract/_meta.json#L1-L20
2. https://github.com/Algorythmos-AI/sggs-platform/blob/integration/contract/_meta.json

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 27 September 2026. Verified against aced1a0 on 2026-09-27. This copy may be out of date; the page is at <https://docs.gurbanisoul.com/search/harnesses-and-golden-vectors/>