

Ship & operate · Engineering handbook

Local setup, repository map & API

🕒 3 min read · Updated 25 September 2026 · Verified against [f25ab97](#) on 2026-09-25

Running the platform on a developer machine: installing the pinned database, the repository map, the API quick reference and the iOS pair.

<https://docs.gurbanisoul.com/engineering/local-setup/>

Running, rebuilding and navigating the project on a developer machine. New to the project? The step-by-step version with troubleshooting is [Run it locally](#).

What this is

An **offline**, “**sovereign**,” **zero-dependency** study app over the full Granth (Angs 1–1430). The corpus and SQLite/FTS5 database are built reproducibly from the source edition in [sggs-data](#)¹; here a Python stdlib server serves them and a prebuilt Astro multi-page UI (search, reader, themes, lineage, study trail, concept constellation, insights).

- **Current build:** APP_VERSION / APP_BUILT in `webapp/serve.py` (served by `/api/meta` and `/api/health`). The traditional-saroop display toggle is **default-ON** as of v2.10.1 (reader can switch to verbatim).
- **Corpus:** 60,658 line records · 1,430 Angs · FTS5 full-text.
- **Data:** pinned by `dataset.lock.json` to a published [sggs-data](#)¹ commit (the source PDF and the rebuild live there).

Run it

```
# easiest: double-click "Start SGGS App.command"
cd webapp && python3 serve.py          # stdlib only; no pip install needed
# → http://localhost:7777    (override port with SGGS_PORT)
```

Run one service of the platform split from the same code with `SGGS_MODULES=search` (or any comma list of `reader,search,verify,insights,knowledge`; default `all`) and point it at its own database slice with `SGGS_DB=/path/slice.sqlite`. In split mode the server refuses to start unless the database holds every table the enabled contexts declare. `/healthz` (liveness) and `/readyz` (200 only when every declared table is present) are the platform health checks.

The server needs `db/sggs.sqlite` to exist (~104 MiB). The database is owned by [Algorythmos-AI/sggs-data](#)¹; this repository consumes the exact object pinned in `dataset.lock.json`. Run `make dataset` after cloning — it downloads the pinned object,

verifies its sha256 and size, and installs it atomically; `make dataset-check` proves `sggs-data` publishes that object at the pinned commit. (Start `SGGS App` command does this on first run.)

A new dataset

The corpus and database are rebuilt from the source PDF in [`sggs-data`](#)¹ (`make rebuild` there: reconcile char-exact, golden checks, deterministic double build, editorial ledger, fingerprints). To take a new dataset here, update `dataset.lock.json` to the published `sggs-data` commit (its `db/sggs.sqlite` LFS pointer gives the sha256 and size), run `make dataset`, regenerate the contract (`make contract`), and open a PR — `make dataset-check` and CI prove the pin.

Toolchain: `serve.py` = Python 3 stdlib only. The UI source is in `frontend/` (Astro + Tailwind v4, Node); its build output is synced into `webapp/static/.node_modules/`, `dist/`, and `webapp/static.bak/` are git-ignored.

Repository map

```
dataset.lock.json      # the sggs-data commit + database sha256/size this platform
serves
db/sggs.sqlite        # installed by `make dataset` (git-ignored); opened READ-ONLY by
the app
webapp/serve.py       # composition root: build identity, ROUTES table, HTTP layer,
startup
webapp/sggs/          # bounded contexts: core (DB handle + shared state), search,
reader,
                        # verification, insights, knowledge — each becomes a service in
the split
webapp/verify.py      # quotation-verification engine (Layer 3)
webapp/romannorm.py   # the query-time Roman fold (byte-identical to sggs-data's index-
time fold)
webapp/static/        # prebuilt Astro MPA (served); static.bak/ = local backup, ignore
frontend/             # Astro UI source (build → static/)
contract/             # golden vectors + OpenAPI (the API's behaviour; vendored by the
iOS app)
tools/                # gen_golden_vectors, gen_openapi, contract_http,
api_superset_check, harnesses
qa/chaos/             # adversarial search inputs (tools/chaos_harness.py)
scripts/              # ci / release / ops / data (fetch_dataset.py) tooling
```

API quick reference (`webapp/serve.py`)

`/api/search?q=&mode=&limit=&offset=` (modes: `auto,gurmukhi,roman,english,first,theme`) · `/api/ang/{1..1430}` · `/api/shabad/{comp_id}` · `/api/random` (complete Hukam unit) · `/api/verify?q=&ang=` · `/api/word?w=` · `/api/meta` · `/api/health` · `/api/themes/network` · `/api/analytics/{author,raag,progression,resonance,vaars,vaar,constellation}` · `/api/related` · `/api/line_concepts` · `/api/neighbors` · `/api/timing/*` · `/api/forms` · `/api/banis` · `/api/bani/{key}`. The generated, complete table is [API routes](#).

`/api/health` is the fast integrity check (asserts 60,658 lines, 1,430 distinct Angs, FTS works, verbatim Mool Mantar, ≥ 560

ੴ

, live verify). Use it after any DB change.

iOS app

The iOS app, its bundled database pair and its local-development checks (`make ios-db-check`, `make ios-db-repair`) live in **Algorythmos-AI/gurbani-soul-ios**².

Links

1. <https://github.com/Algorythmos-AI/siggs-data>
2. <https://github.com/Algorythmos-AI/gurbani-soul-ios>

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 25 September 2026. Verified against f25ab97 on 2026-09-25.
This copy may be out of date; the page is at <https://docs.gurbanisoul.com/engineering/local-setup/>