Aakarsh Kashyap

+91-9068014817 | souls.syntax@gmail.com | linkedin.com/in/souls-syntax | github.com/souls-syntax

Education

Ganeshi Lal Agrawal University (GLA) — Mathura, UP
B.Tech in Computer Science Engineering, CGPA: 8.51 — *Aug. 2024 – May 2028*

Indian Institute of Technology (IIT) — Madras, Tamil Nadu
B.S in Data Science and Applications, CGPA: 7.45 — *May 2024 – Aug 2028*

Experience

Technical Contributor, GigaVector (Open Source) — Apr 2026 – Present
GigaVector — *Virtual*

- Diagnosed and fixed a non-portable compile flag causing SIGILL crashes on WSL2 targets in a vector database with 500+ active users; traced the failure to ABI assumptions that did not hold across toolchain configurations and submitted a patch accepted upstream.
- Resolved a daemon thread lifecycle bug paired with an `is_running` liveness race that allowed silent data corruption on shutdown; the fix required reasoning about memory visibility across thread boundaries and correct teardown ordering.
- Debugged cross-platform DLL dependency failures on Windows involving libcurl linkage, MinGW runtime mismatches, and CFFI symbol resolution errors, restoring build reproducibility across Linux and Windows targets for all downstream contributors.

Projects

CUSTODIAN | *Go, Python, Redis, PostgreSQL, DistilBERT, Docker Compose* — Dec 2025 – Jan 2026

- Built a high-concurrency Go orchestrator implementing a scatter-gather pattern on every incoming query, fanning out simultaneously to PostgreSQL for audit logging and a Python BERT service for inference, with results joined before response.
- Designed a three-tier degradation chain: Redis cache under 1ms for known claims, local DistilBERT at 50ms for high-volume filtering, and async LLM escalation via a Redis job queue for ambiguous cases; each layer independently falls back to the one below it.
- Implemented write-through cache consistency: background worker LLM verdicts atomically overwrite both the Redis cache and the PostgreSQL record, ensuring the next requester always gets the highest-confidence result instantly.

Chirpy | *Go, PostgreSQL, JWT, Argon2id, REST* — Jan 2026 – Mar 2026

- Designed and implemented a production-style RESTful API in Go with zero external web frameworks, covering user registration, chirp CRUD, stateless JWT authentication, refresh token issuance and revocation, webhook ingestion, and admin observability metrics.
- Built the auth layer with short-lived JWT access tokens and revocable refresh tokens persisted in PostgreSQL; used Argon2id for password hashing and implemented shared-secret webhook verification for secure third-party event ingestion.

SlopGen – LLM Agentic Engine | *Go, OpenAI SDK, Ollama, JSON Schema* — Feb 2026 – Mar 2026

- Implemented a ReAct-style agentic loop in Go: the LLM reasons over tool call results iteratively (read, write, execute) until the task is resolved, with full conversation history passed on every inference request.
- Defined all tool interfaces via strict JSON schemas; built a human-in-the-loop confirmation gate for shell execution, balancing autonomy with safety in agentic workflows.
- Designed for model-agnostic inference: local Ollama (qwen2.5:7b) and remote cloud endpoints (GPT-OSS 20B hosted on Kaggle via Ngrok) switchable via a single CLI flag.

Technical Skills

Languages: Go, Python, C/C++, SQL (PostgreSQL, SQLite), JavaScript
Backend: REST API Design, Scatter-Gather, Write-Through Cache, JWT, Argon2id, RBAC, Webhook Verification
Databases: PostgreSQL, Redis, SQLite, SQLAlchemy ORM
Infrastructure: Docker, Docker Compose, Cloudflare Zero Trust, Git, Linux
AI / Agents: OpenAI-compatible SDKs, Ollama, Tool Calling, ReAct loops, Async LLM Pipelines