Ashutosh Siddh

APPLIED AI ENGINEER · LLM SYSTEMS · BACKEND & DATA

Gurgaon, India | Open to remote (global) | +91 70735 22808 | siddhashutosh@gmail.com | ashutosh-siddh.com | github.com/siddhashutosh

SUMMARY

Senior engineer with **11+ years** designing, leading and running enterprise backend systems, now building **production LLM systems end to end** — agent orchestration, retrieval, voice, governed tool access via the Model Context Protocol, evaluation harnesses and AI-specific observability. The through-line of the work is **making non-deterministic systems safe to operate**: deterministic policy engines the model cannot bypass, adversarial test corpora, CI quality gates that fail the build on regression, and failure taxonomies that say *why* a system broke rather than scoring that it did. Deep SQL Server / Oracle and regulated-delivery background across insurance, telecom, healthcare, automotive and HR — applied directly to AI-over-enterprise-data problems, where correctness, permissions and cost decide whether a demo ever ships. Ships on AWS (EC2, Lightsail, Lambda), Vercel, Railway and Render. M.Sc. Information Technology, DA-IICT.

TECHNICAL SKILLS

AI Engineering	Claude API (Opus / Sonnet / Haiku) · agent tool-calling & streaming · multi-agent orchestration · LangGraph · LangChain · Model Context Protocol (SDK, stdio + Streamable HTTP) · RAG & hybrid retrieval (BM25, RRF, HyDE, multi-query, LLM rerank) · structured outputs · prompt caching · context & token-budget engineering · LLM-as-judge evaluation · guardrails & adversarial red-teaming · AI observability (OpenTelemetry GenAI, OpenInference) · voice agents (Vapi, LiveKit, Deepgram) · open-weight models
Languages	C# · TypeScript · Python · SQL · JavaScript
Backend	.NET Core · ASP.NET Web API · MVC · microservices · Node.js 22 · FastAPI · Express · Next.js 16 (App Router / RSC) · REST · JSON-RPC
Data	SQL Server · Oracle · PostgreSQL / Neon · pgvector · SQLite · Drizzle · Prisma · star-schema modelling · row-level security · query cost & plan analysis
AWS	EC2 · Lightsail · Lambda · S3 · RDS · VPC · IAM · CloudWatch · Route 53 · ECS/ECR · CloudFormation · Secrets Manager
Platform & DevOps	Docker · Nginx · Terraform · GitHub Actions · TeamCity · Azure DevOps · Vercel · Railway · Render · Expo/EAS · CI/CD
Engineering	System design · SOLID & dependency inversion · threat modelling · SRS/HLD/LLD & ADR authorship · regression + adversarial testing · observability & on-call design · technical leadership · Agile/Scrum
AI Dev Workflow	Claude Code · Cursor · Codex · GitHub Copilot · Antigravity

SELECTED AI SYSTEMS — designed, built and shipped independently

Anvaya — AI failure observability & causal diagnosis

TypeScript monorepo · ~25k LOC · 230 tests

Node 20 · TypeScript · SQLite · React dashboard · OpenTelemetry GenAI / OpenInference ingest · Claude (opt-in judge)

- Answers the question tracing and eval tools do not: **why did the AI fail, and how**. AI's worst failures return **HTTP 200 with confident, well-formed, wrong output** — every classical monitor stays green. Anvaya ingests execution traces, detects failures, and **attributes them causally across the span tree**, so a bad answer points at the retrieval step that caused it rather than at itself.
- Built a **research-grounded failure taxonomy of 56 named modes across 8 families** (infra, context, retrieval, generation, agent control-flow, tools, security, economics) covering all 14 MAST multi-agent modes, all 7 Barnett RAG failure points and 6 OWASP LLM Top-10 (2025) entries — named codes with remediation, not a freeform confidence score.
- **Four-tier, cheapest-first detection**: L0 structural (span-tree cycles, status, timings), L1 heuristic (regex, schema, token overlap, set ops), L2 statistical (PSI/JS drift, z-score, MAD), L3 LLM-judge (off by default, billed). **L0–L2 cover 43 of the 56 modes at ~zero marginal cost** — driven by the finding that the most frequent failure modes are the cheapest to catch.
- Calibrated honesty by construction: findings render *possible / likely / confirmed* bands, and a lexical-overlap groundedness check is **capped so it can never present as confirmed** — false-positive fatigue kills observability tools faster than missed detections.
- **The SDK cannot break the app it observes** (ADR-0005): with the collector stopped, instrumented requests still complete at full speed. Content capture off by default; secret/PII redaction runs in-process *before* transport and again server-side; security findings record the secret *class*, never the value.
- Four packages (core · sdk · server · ui) with an **enforced acyclic dependency direction**; session-scoped detection for cross-turn modes a single trace can never see; full design set — design doc, SRS, HLD, LLD, taxonomy and **7 ADRs**.

MIT · `npm run dev` starts collector + dashboard with zero config, no Docker, no database, no API key

ConciergeAI — multi-tenant AI customer service with a non-bypassable policy layer

Next.js · TypeScript · ~12k LOC

Next.js · TypeScript · Claude · Postgres + in-memory adapters · BM25 retrieval (from scratch) · Vitest · n8n

- Single architectural commitment: **the language model never holds authority to commit the business**. The LLM understands and proposes; a **deterministic policy engine with no LLM inside** decides; the LLM only narrates the decision. If the model hallucinates *"I've refunded you £400"*, no refund happened — it never had the capability — and the failure degrades to a transcript contradiction the output rail catches and escalates.
- Nine enforced controls, each traced to a real public incident: server-side amount recomputation (the model's number never touches money), **verbatim-only policy retrieval** (against *Moffatt v. Air Canada* invented-policy liability), `toolCallId` idempotency against double refunds on retry, HMAC-bound approvals against forged client-side confirmations, PAN redaction before persistence/logging/model (keeping the platform out of PCI-DSS scope), and escalation on **leading** indicators rather than after the customer is already angry.
- **Allergen handling treated as safety-critical, not as an intent** — grounded in a peer-reviewed finding that only 39% of chatbot-supplied food-allergy sources were accurate (the rest expired, miscited or fabricated). The system never generates or infers an allergen answer and never asserts a dish is safe; the only permitted speech act is *"our records list X"*, always with the cross-contamination warning and escalation on uncertainty.
- Test strategy built for non-determinism: **assertions on structure and tool-call shape** with a mocked model instead of asserting on prose, plus a **frozen adversarial corpus** (prompt injection, data exfiltration, PII leakage) — regenerating attacks per run would make a regression indistinguishable from a new attack.
- Storage behind repository interfaces: no `DATABASE_URL` runs the seeded in-memory adapter, setting it hands over to Postgres which applies its schema on first request — a one-variable change, not a rewrite. Ships EU AI Act Article 50 disclosure (applies 2 Aug 2026) with an output rail that blocks any reply denying AI status.
- Documented **honest metrics and known limitations** rather than marketing numbers — containment vs. resolution gap, repeat-contact rate as the first-class metric, and a full list of accepted weaknesses with reasoning in the LLD.

Full research → SRS → HLD → LLD → C4/sequence/ER/deployment diagram set; retail, restaurant and human-agent console surfaces

AegisQuery — governed analytics Model Context Protocol server

TypeScript · Node 22 · MCP spec 2025-11-25

@modelcontextprotocol/sdk · node-sql-parser (AST) · SQLite/Postgres · Streamable HTTP · OAuth 2.1 / RFC 9728

- Gives LLM agents **safe, read-only warehouse access** through 4 workflow-oriented tools and 7 policy-annotated resources over a star schema — curated metrics preferred over model-authored SQL, with a `preview_query_cost` tool that validates and prices a query without executing it.
- **Nine-stage safety pipeline**: AST parse → single-statement read-only guard → reference extraction → table/column allowlist → PII masking → row-level policy injected by AST rewrite → planner-based cost guard → read-only execution → token-budget truncation → structured audit record per invocation.
- Reproduces and then **defeats the stacked-statement injection class** (`COMMIT; DROP SCHEMA public CASCADE;`) that bypassed a naive `READ ONLY` transaction in a widely used reference Postgres MCP server — structural AST validation as the primary control, a read-only connection as defence in depth. Verified by a **20-assertion adversarial harness with 9 injection rejections plus a post-attack data-integrity check**, and a 10-task golden eval scored against an independent oracle.
- **Stateless-first remote transport** — Streamable HTTP with no session id so it scales behind a round-robin load balancer; bearer tokens map to role + attributes, and unauthenticated requests get a 401 with RFC 9728 Protected Resource Metadata. stdio uses environment credentials per spec guidance.
- Live console demo runs the *same* governance engine on Vercel serverless: switch role (analyst / regional / finance / admin) and watch one query behave differently — PII masked for the analyst, raw for admin, cost column denied below finance, adversarial SQL rejected on the spot.

Live: web-beta-rosy-0ovpsr1q8m.vercel.app · Source: github.com/siddhashutosh/aegisquery-mcp

Dialoft — autonomous AI voice SDR (outbound sales & qualification)

Python · 39 offline tests

LangGraph · LangChain · Vapi (PSTN) · LiveKit (browser) · Deepgram Nova/Aura · Claude · FastAPI · Postgres + pgvector · Docker

- **One brain, many channels**: a single LangGraph state machine (8 nodes, 7 state-derived stages) exposed behind an **OpenAI-compatible `/v1/chat/completions` endpoint** that both Vapi and a LiveKit worker consume — telephony and browser voice share one conversation engine with zero duplicated logic.
- Routing is deterministic because the handling stage is *derived* from accumulated state (consent, BANT slots, classification), not chosen by the model: `open → discovery → qualify → objection → book | nurture → wrap`, with a 0–100 lead score (HOT/WARM/COLD) recomputed every turn.
- 8 LangChain tools — CRM, calendar, knowledge-base RAG, DNC, scoring — with objection handling grounded in the KB so it never invents pricing; **compliance as first-class graph steps**: consent disclosure, DNC enforcement, opt-out, and calling-window guards before any dial.
- **The entire flow verifies offline**: a deterministic fake LLM plus in-memory CRM/calendar/DNC back-ends run 39 pytest tests and a full scripted conversation with no telephony, speech or API keys — and the outbound endpoint returns a dry-run payload showing exactly what would be sent.

Source: github.com/siddhashutosh/Dialoft · SRS / HLD / LLD ship with the repo

Virtual Buddy — multi-tenant AI-employee SaaS for creators

Live in production · paid tiers · web + iOS/Android

Next.js 16 (RSC) · Claude API · Clerk · Stripe · Drizzle + Neon Postgres · Expo/React Native · Vercel Cron · Vitest · GitHub Actions

- Streaming Claude agent over tool-calling with **tier-based model routing across Haiku, Sonnet and Opus**; tier and persona are derived **server-side** (never client-supplied) and gated by per-creator/IP rate limits plus monthly plan quotas.
- Multi-tenant 14-table Drizzle schema on Neon with **Postgres row-level security enforced through a non-bypass `app_user` role** — tenant isolation verified end to end, not assumed from a `WHERE` clause. YouTube OAuth tokens **AES-256-GCM encrypted at rest**; OWASP-hardened with fail-closed auth/crypto in production, CSP, and redirect pinning.
- Production observability for unattended work: every cron run is recorded with status/counts/duration **even when the body throws**, `partial` is a distinct status from `ok`, failures fire a webhook alert, and a `CRON_SECRET`-authed **`/api/health/jobs` returns 503 when a job goes stale** — the half alerting cannot cover, because a cron that silently stops firing looks identical to a healthy one.
- Dead-grant handling as its own case: a revoked YouTube token flags the account `needs_reconnect`, tells the creator in-product and alerts ops — retrying never fixes it, only reconnecting does.
- **Adapter-first architecture** — the whole product runs on deterministic mocks with no keys, and each real integration (Claude, Neon, Clerk, Stripe, Google OAuth) activates automatically when its keys are present; CI gates lint, typecheck, 48 unit tests, key-gated AI evals, and a clean production build.
- Native **Expo/React Native app on the same API and database** — Clerk bearer tokens in the device secure enclave, streaming chat over `expo/fetch`, biometric app lock, push registration and haptics — so web and mobile cannot drift.

Live: virtualbuddy-hazel.vercel.app · design set incl. a 34-figure UML "system as built" document and a P0–P3 production-readiness gap list

evalharness — LLM evaluation framework with a CI quality gate

Python · pip-installable · 49 offline tests

Pydantic v2 · Typer · sentence-transformers · FastAPI · Next.js dashboard · GitHub Actions

- Reframes LLM quality as what it actually is — **test infrastructure with a probabilistic twist**. Versioned golden sets, **five composable scorers** (exact, contains, regex, local embedding via all-MiniLM-L6-v2, and a schema-validated LLM judge), then a gate that compares against a saved baseline.
- **Regression detection on three independent signals** — per-case pass→fail, aggregate pass-rate drop, and per-scorer mean drop, each with configurable tolerance — and `evalharness gate` **exits non-zero**, so a prompt or model change that degrades quality turns the PR red exactly like a failing unit test.
- **Content-addressed SHA-256 response cache** keyed on provider, model, prompt, system and params: re-runs are free and reproducible, and because the key changes when the prompt or model changes, **the cache can never mask a real regression**.
- Built to SOLID seams — `Runner` depends only on the `Provider` / `Cache` / `Scorer` / `Reporter` abstractions injected into it, so a new backend or scorer is one small class and no edit to the runner. Per-case fault isolation, a single `EvalHarnessError` base with typed subclasses, and library-safe structured logging that never configures handlers on import.
- Hosted dashboard doubles as a **prompt playground** across 10 AI-feature templates (support bot, sentiment, summariser, PII redactor, NL→SQL, grounded RAG Q&A…): weaken a prompt, run the gate, and watch it name exactly which cases and scorers regressed. Bring-your-own-key live mode keeps the key in the browser.

Live: evalharness-wine.vercel.app · Source: github.com/siddhashutosh/evalharness

RAG Playground — modular retrieval lab with side-by-side strategy scoring

Strict TypeScript · ~1.5k LOC · 2 runtime deps

Anthropic SDK · transformers.js (local embeddings) · from-scratch BM25 & Reciprocal Rank Fusion

- Every pipeline stage is a swappable module behind an interface (`EmbeddingProvider`, `Retriever`, …) composed from a `PipelineConfig`, so retrieval techniques can be switched on and off and **compared head-to-head on the same corpus**.
- Three retrievers (dense, BM25, hybrid) with **Okapi BM25 and RRF implemented from scratch** and reused across hybrid and multi-query fusion; query transforms (multi-query expansion, HyDE); listwise LLM reranking; three chunking strategies with overlap.
- Grounded generation that **resolves model-emitted `[n]` citations back to source chunks and drops unmappable markers**, with explicit "I don't know" abstention; reference-free LLM-as-judge scores context relevance, faithfulness and answer relevance, printing a ranked strategy table with token cost.

– Runs offline for ingestion via a dependency-free hash embedder; all Claude calls use **JSON-schema-constrained structured outputs**. CI runs typecheck, unit tests, an ingest smoke test and `npm audit`; threat model documented in SECURITY.md.

KESSLER & PENUMBRA — open-source orbital-risk stack (published packages, live deployments) Python · FastAPI · React/Three.js · Apache-2.0 · on PyPI

FastAPI · SGP4 · NumPy · SQLite cache · Vite + React + Three.js · AWS Lightsail (ap-south-1) · Claude analyst briefings · GitHub Actions CI

– **KESSLER** — an open conjunction-assessment toolkit filling a real gap: NASA's reference tooling is MATLAB-only and no lightweight open Python library covers CDM parsing plus collision-probability computation, just as the US TraCSS programme begins distributing standardised CDMs to every satellite operator. Ingests public Space-Track and CelesTrak feeds, parses CDMs (KVN + JSON), and computes **Foster 2-D collision probability cross-checked against Chan's series with a Max-Pc bound**, plus SGP4 screening and risk triage. **50 passing tests**; published to PyPI as `kessler-toolkit`, generated from the backend logic layer and **CI-verified to stay in sync**.

– Architected as pure-math logic under a service/orchestration layer under FastAPI, so the astrodynamics core has no framework dependency; **cache-first by design** with a client-side rate governor that respects Space-Track's published limits. Runs live on a $5 VPS with a 30-minute refresh, a six-stage visualised agent pipeline, a 3D dashboard, and Claude-generated analyst briefings that **fall back to deterministic templates when no key is set**. Ships aerospace-style SRS/HLD/LLD/QA documents and states its accuracy limits plainly ("triage-grade, not an operational collision-avoidance service").

– **PENUMBRA** — the phase-2 companion: **probabilistic forecasting of the space-weather drivers** (F10.7 solar flux, Kp) that govern satellite drag. Official operational products publish point values with no uncertainty; PENUMBRA attaches **empirical quantile bands by lead time derived from a leakage-free walk-forward backtest**, emits per-day storm-category probabilities instead of a misleading point value on a bounded index, translates the uncertainty into orbit-decay risk, and publishes a **calibration audit** — reliability/coverage, pinball loss, and skill score against NOAA's own 45-day forecast. 35 tests; on PyPI as `penumbra-toolkit`.

Live: 13.127.244.0 (KESSLER) · 13.127.244.0:8080 (PENUMBRA) · github.com/siddhashutosh/kessler

PROFESSIONAL EXPERIENCE

TrueBlue — Software Engineer Apr 2025 – Present

– Build and maintain a workforce-management product: backend services in .NET Core and ASP.NET Web API over SQL Server, deployed on AWS (EC2, S3, RDS, IAM, CloudWatch).

– Use Claude and GitHub Copilot as part of the day-to-day development and code-review workflow, including agentic coding tooling for refactors and test generation.

GlobalLogic — Associate Consultant (Tech Lead responsibilities) Jun 2021 – Mar 2025

– Led end-to-end delivery of an enterprise insurance platform (C#, ASP.NET Web API, MVC, .NET Core, SQL Server) from requirements through design, implementation and unit testing.

– Carried combined hands-on development and team-lead responsibilities in an Agile model — estimation, design review, code review and mentoring — owning source control and collaboration across TFS and Git.

Nagarro — Senior Associate Sep 2019 – Jun 2021

– Built a customer and dealership management platform (.NET, Oracle) with serverless components on AWS Lambda, and infrastructure/CD via Terraform and TeamCity.

– Delivered an HR staffing application as microservices with an Angular front end on Oracle, managing delivery through Azure DevOps.

LiquidHub (Capgemini) — Technical Lead Oct 2017 – Aug 2019

– Designed and built in-house Google and Microsoft authorization-API integrations end to end using .NET (C#, Web API) and SQL Server — OAuth flows, token lifecycle and downstream service integration.

GGK Tech — Software Engineer Jun 2017 – Oct 2017

– Built healthcare web products for one of the largest US health-benefits networks (C#, MVC, Web API, Angular, SQL Server).

Infosys — System Engineer Jul 2015 – Mar 2017

– Modernized a large telecom system for a major Australian telecommunications provider: migrated legacy VB to C# and Oracle to SQL Server, and built the data-migration and automated validation tooling that verified the migration.

ALSO BUILT

106-agent deep-research harness — a parallel multi-agent research pipeline (web search → primary-source fetch → **3-vote adversarial verification per claim, ≥2 refutations kills it** → synthesis) run over 722 tool calls and 3.5M tokens to assess a venture thesis; the output leads with its own coverage gaps and reports which claims were refuted. · **Sir Roasts-A-Lot** — persona and prompt-engineering study shipped as a Next.js 15 product: streaming Claude responses with prompt caching on a large persona prompt, four calibrated safety tiers with hard rules the model cannot be talked out of, and a curated offline corpus so the product degrades gracefully with no API key. · **SMSERP** — full-stack school ERP (Express, Prisma, PostgreSQL, JWT/bcrypt, Next.js) with role scoping enforced server-side rather than hidden in the UI.

EDUCATION

M.Sc. in Information Technology — DA-IICT (Dhirubhai Ambani Institute of Information & Communication Technology), Gandhinagar 2013 – 2015

Bachelor of Computer Applications — Ahmedabad University 2010 – 2013