PRATYAKSH KRISHNNA

Delhi, India | +91 98211 68499 | pratyaksh.krish@gmail.com | github.com/pratyaksh-krishnna | linkedin.com/in/pratyaksh-krishnna

AI ENGINEER

AI engineer working on LLM application development in Python and TypeScript: retrieval-augmented generation, multi-agent orchestration with LangGraph, and guardrailed production LLM systems shipped on GCP and AWS. B.Tech CSE, Class of 2027, with internship experience delivering a production conversational analytics agent for an enterprise logistics client.

TECHNICAL SKILLS

- **Generative AI:** RAG, LangChain, LangGraph, agentic and multi-agent workflows, prompt engineering, context engineering, tool calling / function calling, structured output, LLM guardrails (prompt-injection, PII, toxicity), LLM evaluation (RAGAS), observability (LangSmith), Anthropic Claude and OpenAI APIs, Ollama, local LLM inference
- **Languages:** Python, JavaScript, TypeScript, SQL, C++, Rust, HTML, CSS
- **Backend & Cloud:** FastAPI, Node.js, Express.js, REST API design, Celery, BullMQ, Redis, Kafka, WebSockets/Socket.IO, async and concurrent processing, data pipelines / ETL, Docker, Nginx, CI/CD (GitHub Actions), GCP (BigQuery, Cloud Run, Eventarc, Cloud Storage, IAM), AWS (ECS, EC2, ECR, S3), Linux, Git
- **Data & Vector Stores:** PostgreSQL, pgvector, embeddings, semantic search, BigQuery, MongoDB (Mongoose), DuckDB, Redis, Drizzle ORM
- **Frontend:** React.js, Next.js, TanStack, Vite, Tailwind CSS
- **CS Fundamentals:** Data Structures & Algorithms, OOP, Operating Systems, DBMS, Computer Networks

EDUCATION

B.Tech, Computer Science & Engineering — Jaypee Institute of Information Technology, Noida | 2023 – 2027

- JEE Mains: 94th percentile (top 6% of 1M+ candidates). Active competitive programmer, applying graph, DP and amortized-analysis techniques to engineering work.

EXPERIENCE

Software Engineer Intern — Hoonartek (Remote) | Jun 2026 – Jul 2026

GCP (BigQuery, Cloud Run, Eventarc, Cloud Storage, IAM, Cloud Logging), SQL, Conversational Analytics, LLM guardrails

- Shipped a natural-language analytics agent on BigQuery Conversational Analytics for logistics client XpressBees, letting business users query profitability and lane performance without writing SQL.
- Built an event-driven ingestion pipeline (GCS → Eventarc → Cloud Run → BigQuery) that auto-loads 6 monthly logistics parquet feeds, replacing manual loads and adding per-file logging and error-folder routing.
- Authored a BigQuery stored procedure joining, deduplicating and applying business logic across sources into a curated silver layer, using an idempotent DELETE-then-INSERT keyed on revenue month so reruns stay safe.
- Restricted the agent to 4 purpose-built semantic views (margin, lane ranking via DENSE_RANK, cost breakdown, RCA) and wrote agent instructions and guardrails that block PII exposure, hallucination and out-of-scope answers.
- Enforced IAM role-based access and Cloud Logging across the pipeline and authored the end-to-end SOP now used by the client's engineering and business teams.

PROJECTS

Multi-Modal Enterprise RAG Platform | github.com/pratyaksh-krishnna/MULTI-MODAL-RAG | Demo video

FastAPI, PostgreSQL (pgvector), LangChain, LangGraph, Celery, Redis, AWS S3, Ollama, SSE, RAGAS

- Architected an asynchronous ingestion pipeline with Celery, Redis and S3 presigned URLs, moving document processing off the API layer so uploads process in the background instead of blocking requests.
- Implemented multi-modal parsing of 4 formats (PDF, DOCX, PPTX, URLs) with Unstructured, landing text, tables and images in PostgreSQL as 1536-dimension pgvector embeddings.
- Built hybrid retrieval combining vector and keyword search, multi-query expansion, reranking and Reciprocal Rank Fusion, improving retrieval accuracy by ~30% over single-strategy baselines.
- Developed a LangGraph multi-agent system with supervisor orchestration, three-layer input guardrails (toxicity, prompt injection, PII), citation tracking and SSE streaming of token and citation events.
- Enabled on-premises inference with configurable local LLMs (Ollama/LLaMA) for zero external data exposure, and validated answer quality with RAGAS (N = 30) against a naive RAG baseline.

RECLAIM — Bounded Multi-Agent Revenue Recovery | github.com/pratyaksh-krishnna/RECLAIM | Demo video

Node.js, Express, BullMQ, Drizzle ORM, PostgreSQL, Redis, Zod, React, TanStack, Anthropic Claude and OpenAI APIs, Docker

- Built a multi-agent system where LLM agents diagnose failed payments and overdue invoices with cited evidence while deterministic code computes every monetary amount, keeping agents out of the money path.
- Constrained agent output with forced structured output and Zod-validated shared contracts, so malformed or off-policy reasoning is rejected before it reaches an execution tool.
- Implemented a policy engine governing payment-link, SMS, WhatsApp and email interventions with contact and retry ceilings, webhook deduplication, idempotent execution and prompt-injection detection.
- Measured incremental recovery against a randomized holdout cohort with 95% confidence intervals, surfaced in a React and TanStack operator console with human-in-the-loop approval and editable policy rules.
- Covered the policy engine, state machine, idempotency and safety constraints with 121 tests, faking the LLM only at the integration seam so contracts, policy evaluation and tools run for real.