SRIRAM VIVEK

Charlotte, NC | +1 (631) 428-3788 | sriramv1202@gmail.com | linkedin.com/in/sriram-vivek | github.com/SriramV1212

Experience

Galatea Associates LLC — **Boston, Massachusetts**
Financial Software Engineer Intern — June 2025 – August 2025

- Contributed to an AI proof-of-concept for a major financial services client designed to help financial advisors interpret portfolio optimizer trade suggestions, by building a RAG service that supplied the LLM with retrievable domain context for narrative generation
- Built a document ingestion pipeline extracting text and tables from a 600-page optimizer specification PDF using PyMuPDF, chunking content with LangChain's Recursive Character Text Splitter, generating embeddings via text-embedding-3-large, and storing vectors in a PGVector store for similarity retrieval
- Exposed the RAG service through a query endpoint so team members could test the ingested specification and validate that it surfaced usable context to inform the main narrative prompt
- Wrote SQL queries against MS SQL Server to extract and reshape portfolio positions, constraints, and trade outputs into structured tables for LLM prompt consumption, and authored technical documentation for the RAG service and the broader POC in Confluence while tracking deliverables in JIRA under Agile methodology

Projects

Distributed Microservices Orchestration using gRPC | **GitHub**

- Designed a 3-service distributed backend (gRPC, Protocol Buffers) with a central orchestrator aggregating calls to independent User and Search services, using server-side streaming for real-time result delivery
- Added a circuit breaker on the Search service call path and exponential backoff retries on both downstream calls to contain repeated failures, and secured the User and Search services with mutual TLS requiring client certificate verification on every request
- Instrumented full-stack observability across all 3 services with OpenTelemetry tracing, Prometheus metrics, and Grafana dashboards, running the observability stack via Docker Compose to monitor latency, throughput, and error rates under simulated failure conditions
- Validated the system end-to-end with dedicated client scripts covering the standard request flow, live streaming updates, and an injectable failure-simulation mode to confirm retry and circuit-breaker behavior under real conditions

Real-Time Event-Driven Payment Processing Backend | **GitHub**

- Architected an event-driven payment backend (FastAPI, Apache Kafka) decoupling synchronous API ingestion from asynchronous downstream processing, backed by a PostgreSQL state machine managing transitions across pending, processed, and failed states
- Built idempotent event processing using a tracked-event-ID table with conflict-safe inserts, combined with manual Kafka offset commits, to prevent duplicate transaction execution and data loss on consumer crashes
- Isolated failed events on a dedicated dead-letter queue topic with structured error context for recovery and debugging, and load-tested the pipeline end-to-end with 1,000 simulated payment requests against a 4-partition Kafka topic
- Designed around Kafka's durability and replayability guarantees, allowing failed or missed events to be safely reprocessed during debugging and recovery, and enabling multiple consumers to process messages in parallel as traffic scales

Agentic RAG System with Custom MCP Server | **GitHub**

- Built a retrieval-augmented generation (RAG) backend from scratch, designing a markdown-header-aware chunking pipeline with a token-bounded fallback and a self-hosted Qdrant vector store that indexed 3,100+ document chunks in under 60 seconds end-to-end
- Engineered a custom MCP (Model Context Protocol) server in Python exposing 4 retrieval tools as the sole data-access layer, enforcing zero direct database access from any client, and verified it with a standalone test client exercising all 4 tools directly against the server
- Containerized and deployed the full stack (Docker, systemd, Nginx/TLS reverse proxy) to a self-managed Linux VPS, building a GitHub Actions CI/CD pipeline with a self-hosted runner that rebuilds and redeploys the entire stack on every push to main
- Built a Next.js/TypeScript frontend with a retrieval-inspector UI surfacing citation sources and similarity scores, integrating in real time with the FastAPI backend for full answer traceability

Education

Stony Brook University — **New York, United States**
Master of Science in Computer Science and Applied Mathematics — Aug 2024 – May 2026

Courses: Data Structures and Algorithms, Computer Networks, Programming Abstractions, Theory of Computation, Data Management, Big Data Systems, Big Data Analysis, Probability, Data Analysis, Statistical Learning, Statistical Computing

SSN College of Engineering — **Chennai, India**
Bachelor of Engineering in Electrical and Electronics Engineering — Nov 2020 – May 2024

Courses: Python Programming, Object-Oriented Programming, Linear Algebra and Calculus, Partial Differential Equations

Skills

Programming Languages: Python, SQL, Linux

Backend & Data: FastAPI, gRPC, PostgreSQL, MySQL, MS SQL Server, MongoDB, Apache Kafka, Redis

Infrastructure & DevOps: Docker, Kubernetes, Nginx, systemd, AWS (S3, EC2, Lambda), GitHub Actions, Git

Observability: Prometheus, Grafana, Jaeger

AI & ML (Applied): RAG, LangChain, Model Context Protocol (MCP), Qdrant

AI-Assisted Development: Claude Code CLI (skills, workflows, agent teams, agent sessions, goals), Codex, Cursor

Frontend: Next.js, TypeScript, Vercel