RAGAVARSHINI R

FULL STACK ENGINEER · CHECKOUT, APIS & AGENTIC AI TOOLING

Chennai, India | +91 8526620740 | ragavs95@gmail.com

varshini39.github.io | linkedin.com/in/ragavarshini1812 | github.com/varshini39

SUMMARY

Full stack engineer with 9+ years delivering end-to-end product features across fintech and enterprise SaaS. Owns the whole path from React and Next.js interfaces through REST and GraphQL APIs to the event pipelines and databases underneath. Works in TypeScript, Java/Spring Boot, Node.js, and Python, and builds AI-assisted developer tools using agentic RAG, MCP servers, and multi-agent frameworks.

EXPERIENCE

Senior Software Engineer | PayPal | Chennai 05/2025 – Present

- Delivered features across the checkout stack, from React, Next.js, and TypeScript interfaces through to Java/Spring and Node.js services over REST and GraphQL APIs, for platform releases adopted in 52+ EU countries.
- Re-engineered Amex delegation flows to reduce authentication steps in the user journey, raising transaction success rates.
- Optimized checkout performance end to end, cutting latency and improving conversion.
- Led conversion initiatives for enterprise merchants, diagnosing friction points in live checkout flows and shipping the fixes.
- Built internal diagnostic and reporting tools on Datadog and BigQuery that give engineers self-serve visibility into funnel leakage.
- Architected an agentic AI tool that queries production telemetry and surfaces root causes of user drop-offs in natural language.
- Served as Scrum Master, running Agile ceremonies and coordinating delivery across the team.

Member of Technical Staff | Zoho Corporation | Chennai 05/2017 – 05/2025

- Built and shipped features across the Marketplace framework, from API layer to data access, improving query execution time by 25%.
- Architected low-latency solutions for encounter functionality, improving responsiveness and stability by 30%.
- Integrated Apache Kafka for asynchronous, event-driven communication and real-time webhook processing.
- Optimized high-traffic query patterns, reducing database load and sustaining 99.9% uptime SLAs.
- Executed zero-downtime database migrations for a platform serving millions of users.
- Streamlined logging across distributed services, improving traceability and reducing MTTR by 20%.
- Translated customer requirements into shipped product features, working directly with stakeholders on scope and design.

TECHNICAL SKILLS

Frontend: React, Next.js, TypeScript, JavaScript, HTML5, CSS3

Backend & APIs: Java, Spring Boot, Spring Framework, Struts, Python, Node.js, Express.js, REST and GraphQL API design, microservices, event-driven architecture

Databases & Data: MySQL, MongoDB, BigQuery, query optimization, zero-downtime migrations

Messaging & Streaming: Apache Kafka, Zookeeper, webhook processing

DevOps & Infrastructure: Docker, Kubernetes, Jenkins, Maven, Gradle, Bitbucket, Git

Observability: Datadog, Kibana, Jupyter Notebooks

AI Engineering: Agentic RAG and retrieval routing, CrewAI, AutoGen, LangChain, MCP servers, multi-agent orchestration, prompt engineering, Pinecone, LangSmith

PROJECTS

Multi-Agent Conversion Diagnostics PayPal · Internal

- Architected a multi-agent pipeline that diagnoses checkout conversion dips end to end, scoped by country, merchant, or experiment.
- Structured it as an escalation ladder: a BigQuery conversion check gates the run, a diagnostic agent localizes the dip to a checkout stage from the drop-off funnels, and an analyzer agent runs a fixed query set per known problem class.
- Reserved open-ended search for a hypothesis agent invoked only when the deterministic tiers are inconclusive, querying event tables, Datadog, and call logs within a bounded surface and retrieving from a corpus of prior incident analyses, so recurring issues are recognized rather than re-investigated.
- Closed the loop with a report agent producing stakeholder-ready output: conversion movement, drop-off stage, root cause, and supporting statistics.

- Exposed BigQuery and Datadog to the agents as typed, read-only MCP servers with argument validation and scoped credentials, keeping production access out of the agent context.
- Stack: Python, multi-agent orchestration, MCP SDK, BigQuery, Datadog API.

AutoGen Analysis Service GitHub

- Built a FastAPI microservice exposing an AutoGen pipeline of four AssistantAgents, and wired it into an n8n workflow so the analysis runs as an automated job rather than a manual script.
- Added a mock mode flag so the service runs end to end without hitting live LLM APIs, making the pipeline testable and demonstrable offline.
- Wrote every run to a JSON Lines log for traceability, capturing inputs and agent outputs per invocation. Stack: Python, AutoGen, FastAPI, n8n.

Payments Ledger with Read-Only MCP Server GitHub

- Built a double-entry ledger whose five correctness invariants are enforced by PostgreSQL rather than the service: zero-sum via a deferred constraint trigger, append-only via a BEFORE UPDATE OR DELETE trigger, idempotency by unique constraint, and balances derived from entries rather than stored.
- Paired it with a read-only MCP server that holds no database credentials and reaches the ledger only through read HTTP endpoints, the same boundary any external consumer faces; every response cites the exact request it made.
- Deliberately omitted a raw-SQL tool, which would be the most flexible tool on the server and would move the trust boundary from the API to the model's judgment.
- Tested with Testcontainers against real PostgreSQL, never H2; an invariant contract test attacks each rule through raw SQL, bypassing the service, to prove the database is what refuses the write.
- Stack: Java 21, Spring Boot 3, PostgreSQL 16, Liquibase, Python MCP SDK, JUnit 5, Testcontainers.

Agentic RAG Router (CrewAI) GitHub

- Built an agentic RAG pipeline that routes each natural-language question to the retrieval path that fits it: semantic search over a PDF corpus, live web search, or a direct LLM answer with no retrieval at all.
- Split the system into two sequential crews, a router crew that classifies the question and a retriever plus answer crew constructed afterwards with exactly one tool bound to it, so an agent physically cannot call a tool outside its route. Conditional branching sits in plain Python between the crews, since CrewAI sequential process has no native branching.
- Separated retrieval from answer generation into two chained tasks so each agent output is inspectable, and constrained the answer agent to the retriever evidence. Out-of-scope questions return an explicit refusal rather than a hallucinated answer.
- Wrapped a LangChain Tavily search tool in a CrewAI BaseTool subclass to satisfy CrewAI tool validation, and reworked the pipeline onto kickoff_async so it runs inside the Jupyter event loop.
- Logged question, route, router reasoning, tool used, retrieved evidence, and final answer for every run to a CSV reasoning trace. Stack: Python, CrewAI, OpenAI GPT-4o-mini, Tavily, pandas, Jupyter.

Flight Booking Application GitHub

- Microservices-based booking platform covering authentication, flight scheduling, and ticketing, containerized and deployed to Kubernetes.
- Stack: Spring Boot, Docker, Kubernetes, Maven, REST microservices.

Patient Health Information on Blockchain GitHub

- Solidity smart contracts on Ethereum for tamper-evident storage and permissioned sharing of patient health records.
- Stack: Solidity, Ethereum, Web3.

EDUCATION

M.Tech, Software Engineering, BITS Pilani

B.Tech, Information Technology, Karpagam College of Engineering, Coimbatore

CERTIFICATIONS

- Professional Certificate Program in Blockchain, IIT Kanpur
- Applied Agentic AI: Systems, Design & Impact, Microsoft & Simplilearn