Hemanth Kumar Mangalapurapu

New York, NY (Open to Relocation) | (669) 260-4439
hemanth.mangalapurapu@gmail.com | linkedin.com/in/mangalapurapu | github.com/hemu1808

Profile

Software Engineer with a Master's in Computer Science and 2 years of production experience building distributed, cloud-native systems. Strong in data structures and algorithms, concurrency, API design, and failure handling. Proficient in Go, Python, and React, with a focus on correctness, performance, and system design.

Technical Skills

Languages: Go (Golang), Python, Java, TypeScript/JavaScript, SQL, GraphQL
Frontend: React.js, Next.js, Redux, Tailwind CSS, WebSockets, Jest
Backend: Node.js, Express, PostgreSQL, DynamoDB, MongoDB, Redis, gRPC, FastAPI, REST API
Cloud & DevOps: AWS (Lambda, Fargate, ECS, S3, DynamoDB, AppSync), Docker, CI/CD, Terraform, Grafana, Prometheus
Machine Learning: LangChain, Vector Databases (ChromaDB), Ollama, PyTorch, RAG Pipelines

Education

Auburn University at Montgomery, Master of Science in Computer Science *Aug 2023 - May 2025*
MLRIT, Bachelor of Technology in Computer Science & Engineering *Jul 2018 - Aug 2022*

Experience

Speeler Technologies, *Software Engineer, India* Aug 2021 - Aug 2023

- Led redesign of backend **GraphQL** services for a high-traffic e-commerce platform serving 50k+ daily requests
- Reduced P95 latency by 30% by implementing resolver batching and AppSync caching.
- Designed a multi-tenant **DynamoDB** Single-Table Architecture, utilizing sparse indexes and GSI sharding to eliminate hot partitions and support high-cardinality access patterns.
- Engineered a fault-tolerant, event-driven pipeline using **Lambda** and **SQS Dead Letter Queues (DLQ)** to process images reducing operational infrastructure costs by 70%, ensuring idempotency and 99.99% data durability.
- Built collaborative UI layer in React/WebSockets supporting 10k+ concurrent users, reducing latency by 50ms.
- Orchestrated the migration to **AWS ECS Fargate**, implementing Blue/Green deployments for zero-downtime releases across a 50K+ user production system and defining self-healing health checks.

Key Projects

HGPT - RAG Knowledge Base | *Python LangChain ChromaDB Ollama* May 2025 - Present

- Built a fully local, privacy-first search engine featuring real-time Server-Sent Events (SSE) streaming, and **Llama 3** and **DSPy** designed to handle 10K+ Documents, built with Python and React.
- Implemented a hybrid search strategy using **ChromaDB, BM25** for semantic vector retrieval and Reciprocal Rank Fusion, Cross-Encoder re-ranking to maximize context relevance, improving accuracy by 50% over standard keyword search.
- Enabled high-performance ingestion pipeline using **FastAPI, Celery workers** to asynchronously process and chunk large datasets into embedded vectors with Chain-of-Thought(CoT) reasoning without blocking the UI.

Distributed Container Orchestration Engine | *Go PostgreSQL Docker API Grafana* Jan 2025 – May 2025

- Reduced node failure detection time to <30ms by engineering a custom distributed control plane in **Go** with a high-frequency **gRPC** heartbeat mechanism.
- Built a CLI and Dashboard utilizing **Grafana** to visualize real-time node metrics, exposing cluster state via RESTful API.
- Ensured cluster consistency and crash recovery by implementing a Write-Ahead Log (WAL) in **PostgreSQL**, creating a robust failover system for controller outages. Integrated Prometheus for distributed tracing and real-time observability
- Improved hardware utilization by 25% by designing a custom scheduler using Bin Packing algorithm to optimize Memory allocation across worker nodes.

High-Concurrency Transit Reservation System | *Node.js React Redis MongoDB* Aug 2024 – Dec 2024

- Reduced double-booking conflicts by 95% during burst traffic events by architecting a booking engine utilizing Redis Distributed Locks (Redlock) and Optimistic Concurrency Control (OCC).
- Secured PCI-compliant payments across React web and React Native mobile interfaces by integrating Stripe API webhooks within serverless functions for asynchronous verification.
- Scaled real-time inventory updates to 1,000+ concurrent clients with MongoDB versioning to manage seat inventory state, <50ms latency by integrating WebSockets for bidirectional state synchronization.