Gautama Shastry Bulusu Venkata

+1 571-653-0056 | gautamashastry@gmail.com | https://linkedin.com/in/satya2603/ | https://github.com/GautamaShastry | Leetcode | https://gautamportfolio.com

Summary

MSCS candidate (GPA 3.85) with strong experience building frontend applications in React.js, Redux, HTML5, CSS3, and TypeScript that deliver scalable, performant, and user-friendly experiences. Skilled in optimizing UI rendering, enhancing usability, and collaborating with cross-functional teams to design and ship impactful features. Complementary backend expertise with Node.js, Spring Boot, and Flask, enabling full-stack problem solving. Passionate about turning complex business requirements into intuitive, high-quality user interfaces.

Education

George Mason University — Fairfax, VA
Master of Science in Computer Science - 3.85/4 — *Jan. 2024 – December 2025*

Andhra University — Visakhapatnam, AP, India
Bachelor of Technology in Computer Science - 8.15/10 — *Aug. 2019 – May 2023*

Technical Skills

Languages: Java, Python, SQL, JavaScript/TypeScript, HTML/CSS, C, C++
Backend & Systems: Spring Boot, Node.js, Express.js, Django, Flask, REST APIs, Microservices, Distributed Systems, Systems Programming (file I/O, multithreading, memory management, socket programming), Object-Oriented Design, Operating Systems, Linux
Frontend & Web: React.js, Redux, Vite, Tailwind CSS, HTML5, CSS3
Cloud & DevOps: AWS (EC2, S3, Lambda, RDS, Route 53, CloudFront, IAM, SQS, SNS, CloudWatch), Docker, Kubernetes, Jenkins, Rancher, Git, GitHub
Databases: PostgreSQL, MySQL, MongoDB, HikariCP, JPA/Hibernate
AI & ML: Python (spaCy, pandas, NumPy, TensorFlow, Keras), Natural Language Processing, Retrieval-Augmented Generation (RAG), LLM fine-tuning, Large Language Models, Machine Learning, Deep Learning
Tools & IDE: Visual Studio Code, IntelliJ IDEA, Eclipse, Postman, Maven, Confluence, Jupyter Notebook
Methodologies: CI/CD, Agile/Scrum, Version Control(Git)

Experience

Associate Software Engineer — Jan 2023 – December 2023
Backflipt — *Hyderabad, India*

- Faced a sluggish, hard-to-scale UI, so I **redesigned the front end into modular React components with Redux** by refactoring shared logic and state boundaries, **boosting user engagement by 20%** and improving maintainability.
- Hit rendering bottlenecks under load, so I **optimized the render path** by introducing React hooks, memoization, and granular component splits, **cutting re-render overhead by 40%** and **improving page load by 10%.**
- Saw latency from redundant network calls, so I **streamlined client–server data flow** by redesigning Redux state, normalizing entities, caching, and debouncing requests, **reducing response latency by 15%.**
- Onboarding was slow due to tribal knowledge, so I **documented UI/UX workflows and standards in Confluence** by mapping flows, patterns, and examples, **shortening new-dev ramp-up by 25%** and smoothing cross-team handoffs.

Projects

Student Survey Application | *SpringBoot, Docker, Kubernetes, Jenkins, AWS, Rancher* — March 2025 - May 2025

- We needed a production-ready survey backend with strong data integrity, so I built a Spring Boot REST API using Spring Data JPA and strict field validation (@NotNull, @NotBlank, @Email), which blocked bad inputs early and kept the dataset clean during the pilot.
- To keep reads/writes fast under concurrent load, I provisioned Amazon RDS MySQL 8 and tuned HikariCP (maxPool=20, minIdle=5) and connection settings, which stabilized throughput and kept latency low during traffic spikes.
- High availability was a must, so I containerized the service into a slim OpenJDK 23 image, published version-tagged images, and deployed a 3-replica Kubernetes workload on EC2 via Rancher (NodePort exposure), which **sustained 99.9% uptime.**
- Release cycles were manual and slow, so I wired a Git-triggered Jenkins pipeline (Maven build → Docker image → K8s rollout with health checks), which delivered **zero-downtime deployments** in **under 5 minutes end-to-end.**
- Before shipping, we needed confidence in behavior, so I wrote Postman suites and exercised live paths after rollout, which **processed 10+ production submissions** with **zero validation or runtime errors.**

Taskify-A task management Web app | *Vite + React.js, Express.js, Node.js, MongoDB, Docker* Jan 2025-Feb 2025

- We needed secure, role-based task workflows, so I built a full-stack React + Express + MongoDB app with guarded routes and CRUD flows, which gave teams a clean way to manage tasks end-to-end.
- Sign-ins were flaky and hard to trust, so I implemented **JWT auth** with **bcrypt hashing** and tightened route protection, which **reduced login issues by 25%** and solidified access control.
- Users couldn't find tasks quickly, so I added real-time filtering, search, and sorting (20+ combos) and tuned MongoDB queries with the right indexes, which **cut search time by 15%** and made task discovery feel instant.
- Dev environments kept drifting, so I containerized the stack with Docker and a simple compose file, which made local setup predictable and cross-platform deploys painless.
- Concurrent edits caused UI hiccups, so I designed clear REST endpoints and centralized state with Redux, which kept client–server interactions smooth and UI updates consistent under load.

Resume Analyzer | *React.js, SpringBoot, Python, Flask, PostgreSQL, Render, Vercel, Docker* Feb 2025-May 2025

- Recruiters needed faster screening, so I built a React app that lets them upload resumes/JDs and view AI fit scores behind JWT-protected routes, which **cut manual review time by 70%.**
- Secure data flow was a must, so I designed Spring Boot REST APIs with Spring Security (JWT) and Spring Data JPA on PostgreSQL, which **made uploads/retrieval reliable and audit-friendly.**
- Accuracy and speed mattered, so I shipped a Flask microservice using spaCy for NER and a scoring pipeline, which **hit 87% accuracy with less than 1.5s latency** on test runs.
- Cross-service reliability was a risk, so I engineered the integration between the Spring Boot (Java) API and the Flask (Python) microservice with typed contracts, timeouts, and retry logic, persisting scores and extracted skills in PostgreSQL, which **processed 15+ resumes** in two weeks with **zero API errors.**
- I needed to scale without blowing the budget, so I deployed backend + AI on Render and the frontend on Vercel with autoscaling and health checks, which kept **uptime at 99.95% for less than $50/month.**

Portfolio | *React.js, Tailwind CSS, AWS S3, AWS CloudFront, AWS Route 53, AWS ACM* Jan 2025 - July 2025

- I needed a fast, clean way to showcase my work, so I built a mobile-first single-page site in React with Tailwind and light, reusable components, which scored **90+ across all Lighthouse categories.**
- Global performance mattered, so I served static assets from **S3 behind CloudFront**, tuned cache/TTL and compression, and optimized images, which **delivered a 193ms average TTFB** and an **89% SpeedVitals grade (A).**
- Security was non-negotiable, so I **enforced HTTPS via ACM**, enabled **HSTS preload**, and locked S3 behind a CloudFront origin policy (no direct bucket access), which passed audits with **zero critical Security Hub findings.**
- Uptime and simplicity were goals, so I kept it **fully serverless** with Route 53 health checks and CloudFront monitoring, which sustained **99.95% uptime** without manual ops.
- Engagement needed a lift, so I added subtle motion (section reveals, card transitions, hover effects) with Framer Motion, which **increased user engagement by 15%.**

Certifications

- AWS Cloud Practitioner CLF-C02
- Deep Learning Specialization(Deep Learning.ai)