Tejasvi

Toronto, ON, Canada | (263) 880-6268 | t3jasvii@gmail.com | linkedin.com/in/t3jasvi | github.com/tejasvi541

Education

Concordia University — Montreal, QC, Canada
Masters in Computer Science (Applied) — *Jan 2024 – May 2025*

Kurukshetra University — Kurukshetra, India
B.Tech. Computer Science & Engineering — *Aug 2019 – Sep 2023*

Skills

- **Languages:** Java, Python, TypeScript, JavaScript, C++, SQL, Shell, Bash, HTML/CSS
- **Backend Frameworks:** Spring Boot, Hibernate, JPA, Node.js, Next.js (Backend), Express.js, Hono.js, FastAPI
- **Frontend Frameworks/Libraries:** Angular.js, React.js, Next.js (Frontend), React Native
- **UI/Styling:** Tailwind CSS, Shadcn UI, Redux
- **Databases & Messaging:** PostgreSQL, MySQL, MongoDB, Redis, Kafka, Zookeeper, Elasticsearch, Drizzle ORM, Prisma
- **Cloud & DevOps:** AWS (ECS, CloudWatch, S3, EC2), Docker, Kubernetes, CI/CD, Jenkins, Github Actions, Microservices
- **Tools & Other:** Git, REST, gRPC, WebSockets, JWT, Postman, Linux, TensorFlow, PyTorch, RAG, Langchain, GenAI

Experience

Limelight Software — Toronto, ON, Canada
Full Stack Engineer — *Oct 2025 – Present*

- Engineered a unified **OLAP state synchronization layer** bridging in-memory cubes with bulk database reload APIs, eliminating frontend-driven stale-state heuristics and enforcing a strict **backend single-source-of-truth** model that prevented cross-session data divergence during concurrent reloads.
- Stabilized multi-dimensional query execution by introducing **pre-validation guards** on OLAP **INTERSECT/search endpoints**, normalizing cross-hierarchy operands at the API boundary and eliminating recurring **Bad Request cascades** without adding latency.
- Refactored backend orchestration into a deterministic **Persist → Cache Mutate → Publish** pipeline, decoupling transactional DB writes from async eventing while consolidating bulk reload APIs to enforce **ordered state transitions**.
- Eliminated race conditions, duplicate cache invalidations, and non-deterministic state during high-volume reloads, restoring **thread-safe execution** and consistent cross-service data synchronization.
- Extended **ANTLR-based parsing and AST evaluation** with custom error-node propagation and centralized lazy-evaluation control, also adding **dependency-aware stale marking** to deterministically re-trigger formula recomputation in topologically consistent order, preventing computation graph failures (#LAZY!, null states) while preserving partial execution across multi-threaded evaluation pipelines.

Hilo Design — Remote, India
Full-Stack Developer Intern — *Nov 2022 – May 2023*

- Developed Node.js backend (JWT auth, MySQL) optimizing API response to sub-200ms p-95 & used Spring Boot (Hibernate) microservices for specific data processing, enhancing modularity.
- Designed responsive React.js frontend (Redux), improving load speed by 40%, and deployed containerized microservices (Docker) on AWS ECS, configuring CI/CD and CloudWatch monitoring.
- Implemented MySQL indexing & Redis caching, reducing DB query latency by 30% & improving API throughput and system resilience by 50% under load.

Projects

Distributed Leader Election Service | GitHub | Java, Spring Boot, Zookeeper, Maven, Kafka Jun 2024

* Engineered a fault-tolerant leader election module using Java and Apache Zookeeper, leveraging ephemeral sequential znodes and Watcher events (NodeChildrenChanged) within a Spring Boot framework for robust distributed consensus.
* Designed as a reusable coordination component for microservices, ideal for managing singleton jobs, service discovery, or orchestrating stateful operations like Kafka consumer group leader assignments.

Learning Management System | GitHub | Next.JS, TypeScript, PostgreSQL, Stripe, Mux Sep 2024

* Engineered a full-stack LMS using Next.js 14 (App Router) & TypeScript, featuring SSR/ISR, Stripe payments, Mux video streaming, and PostgreSQL DB via Drizzle ORM.
* Developed intuitive course creation/management UI with drag-and-drop (Tailwind CSS, Shadcn), secured by Clerk RBAC for role-specific access control.

Scalable Chat Application | GitHub | TypeScript, React, Sockets, Kafka, PostgreS, Redis, Docker May 2024

* Developed a distributed, real-time chat backend using TypeScript, Socket.io (WebSockets), and Kafka for high-throughput, fault-tolerant message queuing, storing messages in PostgreSQL.
* Architected a Dockerized microservices system (incl. Kafka, Redis) enabling horizontal scaling (10K+ msg/sec at <100ms P99 latency) with a responsive React.js frontend.

VLM Number Plate Recognition | *PyTorch, Transformers (ViT/RoBERTa), Flask, Attention* | *Paper* April 2025

* Fine-tuned a **330M TrOCR** (ViT encoder/RoBERTa decoder) on **CENPARMI LPR dataset** (1600 images, 12 epochs), achieving **86.8% accuracy** (2.3% CER) & **≤1-s inference** on consumer GPU.
* Engineered custom **SafeForwardModel wrapper** & data-collator to stabilize training on heterogeneous data, enabling robust **convergence** & achieving **2.3% CER** with high generalization.
* Implemented aggressive **data augmentation** (rotations, jitter, affine) & normalization (384x384), harnessing **self-attention** for superior context modeling & robust performance.

Multimodal Vision Transformer (PaliGemma/Siglip) | *PyTorch, ViT, Gemma, Docker* | *Github* Oct 2024

* Implemented **Siglip ViT** & **PaliGemma** multimodal model (PyTorch), focusing on patch/positional embeddings, multi-head self-attention, and MLP blocks.
* Developed efficient inference pipeline leveraging **KVCache**, top-p/temperature sampling, & GPU acceleration via **Docker** (NVIDIA CUDA) using **Safetensors**.
* Engineered robust image preprocessing (resizing, normalization) & text integration via custom **PaliGemmaProcessor** for effective **conditional text generation**.

LLaMA 2 Implementation | *PyTorch, LLaMA Stack, Docker* | *Github* Sept 2024

* Architected **LLaMA 2 model** from scratch (PyTorch), implementing key components like **RoPE**, **GQA**, and **RMS normalization** for enhanced efficiency.
* Developed robust inference logic using **top-p sampling** & **temperature scaling** with pre-trained weights; used **GPU-enabled Docker** for scalable deployment.
* Implemented efficient **KV caching** during autoregressive generation, reducing computational load & **memory footprint** for faster, optimized inference.