Tridhatri Sontena

tridhatri.sontena@gmail.com
tridhatri.sontena@intel.com
+91-7337337537
github.com/Tridhatri
linkedin.com/in/tridhatri-sontena
tridhatri.vercel.app

Skills Summary

- **Programming Languages**: Python, C/C++, Lua, SQL, Vim Script, Java, JavaScript, Bash, PowerShell, typst
- **Systems & Debugging**: Windows Performance Analyzer (WPA), GPUView, WinDbg, WPT, GPU Debugging, Graphics IP, Display Debugging, ETL Analysis, Performance Analysis, Log Analysis
- **Tools**: Git, GitHub, NeoVim, Vim, tmux, kitty, LaTeX
- **Platforms**: Linux (Arch, Ubuntu, Debian), Unix, Windows, macOS

Experience

- **Intel Corporation** — On-site
  GPU SDE Debugger Intern — *July 2025 – Present*
  - **Agentic AI Debug App**: Architected and built an end-to-end Agentic AI command-line application to automate GPU driver defect analysis, reducing manual triage time from 10–20 minutes to an average of 2.7 minutes (87% improvement). Interfaced with internal defect-tracking REST APIs to pull live issue metadata, attachments, and engineer comments as structured LLM context.

    Designed a 6-phase multi-agent pipeline: (1) automated data ingestion and log classification (ETL, GOP, crash-dump, burnin, performance traces), (2) interactive attachment selection with a user-driven per-item analysis loop, (3) LLM-orchestrated per-category analysis using specialized agent tools and skills, (4) RAG-powered triage checklist compliance and best-known methodology retrieval from internal knowledge bases, (5) structured JSON report generation via a dedicated agent tool, and (6) rendered HTML report artifact with a plaintext terminal summary. Built 19 tool integrations and agent skills spanning 20 defect categories (Display, Media, BSOD, TDR, Gaming, AI/ML, etc.). Orchestrated two external AI analysis pipelines via subprocess with dynamic prompt construction derived from defect context. Applied multithreading, content-hash caching, and context-window compression to minimize LLM token overhead. Built a custom QA evaluation framework for continuous accuracy measurement.
    Stack: Python, Enterprise LLMs, Agentic AI, RAG, LLM Tool-Use, Multi-Agent Systems, ETL, WPA, WinDbg, GPUView, WPT, Roff, HTML/CSS, JSON, REST APIs
  - **Graphics Driver Debug Core Work**: Debugging low-level Graphics IP issues with a focus on display technologies and OS-Graphics Driver-External Device interactions. Involved in identifying, analyzing, and resolving Graphics Driver and GOP driver faults to improve stability and correctness.
- **C.R. Rao Advanced Institute of Mathematics, Statistics and Computer Science** — On-site
  Research Intern (Part-time) — *May 2024 – August 2024*
  - **Limited Dataset Analysis**: Conducted research on cybersecurity datasets under limited-data constraints, focusing on anomaly detection and improving data quality and balance using synthetic data generation and limited dataset analysis techniques to enable effective machine learning.
- **Kellton Tech** — On-site
  Student Developer (Part-time) — *May 2023 – July 2023*
  - **Domain-Specific Chatbot**: Built a custom domain-specific chatbot using OpenAI APIs with Python and Jupyter Notebooks; performed internal payroll data analysis and generated automated summaries and insights.

Education

- **University of Hyderabad** — Hyderabad, India
  Integrated Master of Technology (I.M.Tech) – Computer Science — *Nov 2021 – May 2026 (Expected)*
  - **Relevant Coursework**: Operating Systems, Unix Network Programming, Machine Learning, Neural Networks, Algorithms, Virtualization, DBMS, Computer Organisation and Architecture, Internet Technologies, Essentials of AI

Academic Projects

- **Minimal Operating System – 1000-Line Kernel**: Built a bootable RISC-V kernel in C and Assembly featuring physical and virtual memory management, process scheduling, system calls, and paging. Implemented trap handling and context switching using OpenSBI, and validated kernel functionality using QEMU-based emulation.
  Link: github.com/Tridhatri/RISC_V-OS
- **Computer Vision Referee Assistant (HoopsEye)**: Developed an automated basketball foul and ball detection system using YOLOv8 and OpenCV, integrated with a Flask backend for real-time inference and event visualization. Designed the pipeline for frame preprocessing, object detection, and inference optimization to assist referees during live gameplay.
  Link: github.com/Tridhatri/Automating-Basketball-Referee
- **Corotu – Coroutine Runtime with Priority & EDF Scheduling**: Built a Go-inspired coroutine runtime in C addressing Go's lack of task prioritization. Implemented M:N cooperative scheduling with 8-level priority queues and EDF (Earliest Deadline First) scheduling – provably optimal for meeting task deadlines. Features custom ARM64 assembly context switching (replacing broken `ucontext` on macOS), per-coroutine `mmap`'d stacks with guard-page overflow protection, and work-stealing across OS threads. Benchmarked against Go goroutines, libco, and libmill on context switch throughput and deadline miss rate.
  Stack: C, ARM64 Assembly, pthreads, mmap
  Link: github.com/Tridhatri/corotu
- **TCP Concurrent Student Database**: Implemented a multi-client TCP concurrent server in C using processes and sockets to manage student and course records. Designed streamlined request/response protocols, ensured synchronization across clients, and handled concurrent read/write operations as part of Unix Network Programming coursework.
  Link: github.com/Tridhatri/TCP-Student-Database-in-C