Sanjana Mali

Davis, CA sanjana.mali1583@gmail.com linkedin.com/in/sanjanamali github.com/sanjanamali8 sanjanamali.com

EDUCATION

University of California, Davis *MS in Computer Science (GPA: 4.0/ 4.0)* *Expected Graduation: Jun 2027*

- **Coursework:** Advanced Computer Architecture, Design & Analysis of Algorithms, Machine Learning & Discovery, Software Engineering, Distributed Database Systems, Computer Security

Pune Institute of Computer Technology, Pune, IN *BE in CS (GPA: 9.35/ 10)* *Jul 2020*

- **Coursework:** Data Structures, Object Oriented Programming, Operating Systems, Computer Networks

TECHNOLOGIES

Languages: C++, C, CUDA, Python, Java, TypeScript, JavaScript, SQL, Bash
Systems / HPC: Gunrock (GPU graph framework), Nsight Compute, Linux, Git
Architecture: Parallel algorithm design, memory hierarchy optimizations, microarchitectural analysis, performance profiling
Compilers / DSLs: Tensor-algebra (Einsum) IR and interpreter design, specification verification, EDGE
Web / Data: React, Angular, Node.js, AWS (Lambda, EC2, S3), CI/CD (Jenkins, GitHub Actions), Jest, SQL (MySQL, PostgreSQL), MongoDB, Redis

RESEARCH & TEACHING EXPERIENCE

Graph Algorithms with EDGE - *UC Davis (Prof. John Owens)* *Jun 2025 – Present*

- Represented graph algorithms (direction-optimizing BFS, Bellman-Ford, Max-flow) using a novel tensor algebra-based programming model (**EDGE**), separating computation from execution to analyze parallel execution strategies, memory access patterns, and mapping to efficient GPU kernels across the HW/SW stack.
- Co-authored write-ups of the DOBFS and Bellman-Ford cascades (**arXiv** in preparation) and gave feedback on the notation to the EDGE language authors.
- Built **edge-workbench**, a pipeline that cross-checks a tutorial, its cascade and an independent verifier, runs the cascade on an **IR**, and generates an interactive iteration-space walkthrough, catching **4 bugs** in a newly written set-cover cascade.
- Translated **Bellman-Ford** from EDGE tensor algebra specification to **Gunrock** GPU graph framework, implementing negative cycle detection and mapping EDGE execution semantics to GPU-parallel operator abstractions.
- Profiled GPU kernel performance using **Nsight Compute**, analyzing occupancy, memory throughput, and roofline model to identify optimization opportunities for graph workloads on **NVIDIA V100**.

Teaching Assistant – ECS 193: Capstone Projects - *UC Davis* *Jan 2025 – Dec 2025*

- Mentored 8 capstone teams (32 students) through full SDLC, guiding system design, architecture, and implementation, and supporting technical decisions and scalable feature development across diverse projects.

EXPERIENCE

Software Engineer - *ACI Worldwide, Pune* *Aug 2020 – Aug 2024*

- Architected and shipped a **multi-factor authentication** system across payment channels handling **high-throughput workloads**, achieving **70%** client adoption within 30 days.
- Developed backend services with **REST APIs** and database queries, optimizing request handling and improving system reliability for large-scale **distributed** payment workflows.
- Designed and implemented multi-payment workflows for **Apple Pay, Google Pay, PayPal, and Venmo**, optimizing transaction flow and reducing payment completion time by **60%**.
- Led migration of four large-scale applications from **Angular 7** to **Angular 18**, refactoring legacy code for maintainability and verifying with **Black Duck and Checkmarx** scans to ensure **zero security** vulnerabilities.
- Developed unit tests with **Jest, TestBed, and Karma**, achieving **95%** code coverage and improving software reliability.
- Rapidly integrated **third-party payment APIs**, enabling call center agents to process secure remote transactions and **retaining 2** enterprise clients during COVID-19 lockdowns.
- Provided **on-call production** support for critical payment infrastructure, debugging system issues and optimizing performance through log analysis and efficient hotfix deployment.

PROJECTS

CUDA GEMM Optimization – *UC Davis Research* *Feb 2026*

- Implemented GEMM from scratch with 6 progressive optimizations, achieving **10,200 GFLOPs/s (64% of cuBLAS)**. Optimizations included global memory coalescing, shared memory tiling, 1D/2D register blocking, and vectorized loads; profiled each variant using **Nsight Compute** to validate improvements in occupancy and memory throughput.
  Tools: CUDA, C++, Nsight Compute, Linux

Computer Architecture Performance Analysis – *UC Davis (ECS 201A)* *Winter 2025*

- Conducted microarchitectural performance experiments using the **gem5 simulator**, analyzing the impact of pipeline width, ROB size, and register count on ILP; evaluated cache architectures and blocking strategies on matrix multiply workloads; diagnosed **false sharing** bottlenecks in parallel code using cycle-level cache simulation.
  Tools: gem5, C++, Python, Linux