Kusum Bhattarai Sharma

xcm15@txstate.edu LinkedIn: kusum-bhattarai-sharma GitHub: kusum-bhattarai

Education

B.S. in Computer Science (Physics + Math minor), 3.9/4.0 **August 2023 – May 2027**
Texas State University (TXST) *San Marcos, Texas*

- **Coursework:** Data Structures & Algorithms, OOP, Computer Architecture, Algorithms Analysis, Operating Systems
- **Achievements:** AI Collective Hackathon 3rd prize (**$2000**), TXST Datathon 2025 **Winner**, TXST Achievement Scholarship (**$104,000**)

Technical Skills

- **Languages:** C++, Java, Python, C, JavaScript, TypeScript, SQL, Bash, HTML/CSS
- **Frameworks & Libraries:** React, Node.js, FastAPI, Spring Boot, LangChain, Kafka, Redis, PySAL
- **Tools:** Docker, GCP, AWS, Terraform, GitHub Actions, FAISS, Google Benchmark, Linux, CI/CD, Nginx

Experience

Undergraduate Research Assistant **December 2025 – Present**
Department of Engineering, TXST *San Marcos, Texas*

- Built low-latency voice XR agent for **Magic Leap 2** with async **FastAPI** + **Socket.IO**, achieving **<150ms** audio latency over full-duplex streams.
- Deployed **LangChain** + **ChromaDB** knowledge engine indexing 3 safety manuals to **40 students**, achieving **>90%** user satisfaction.

Student Software Developer **September 2025 – Present**
Center of Analytics and Data Science, TXST *San Marcos, Texas*

- Built **Hybrid RAG pipeline** with **FAISS** vector search and cosine similarity ranking for AI-driven grant proposal optimization.
- Reduced faculty evaluation time by **80%** via polymorphic web scraping with **k-means clustering** and weighted 0–10 Fit Score. GitHub

AI & Data Researcher **January 2026 – Present**
Translational Health and Research Center, TXST *San Marcos, Texas*

- Merged 5 years of **CDC BRFSS** survey data with MSA-level **Distress Community Index** spatial statistics into a master population health dataset.
- Implemented **Global/Local Moran's I** (PySAL, k=6 KNN weights) and **LISA clustering** to map health disparity patterns across U.S. metro areas.

Projects

Incremental Dataflow Engine | *C++17, Python, pybind11, Spring Boot, Google Benchmark* **March 2026 – Present**

- Propagates delta updates through a typed operator DAG with retraction semantics; **40× faster** than batch recompute at 0.1% change rates on **1M-row datasets** (0.2ms vs 8.2ms).
- Optimized **GroupByCount** to **zero heap allocations** per cycle via pre-allocated buffers, XOR-based `RecordHash`, and materialized-view state; benchmarked **7 optimizations** with Google Benchmark.
- Three-layer design: **C++ core**, **Python DSL** via pybind11, and **Spring Boot** REST/SSE — validated with **70 Google Test cases** and live Wikipedia SSE + Alpaca stock feed demos. **GitHub**

Virtual Memory Simulator | *C++17, CMake, Google Test* **March – April 2026**

- Implemented per-process page tables, **FIFO / LRU / OPT** (Belady's) replacement policies, and **O(1) LRU** via doubly-linked list + hash map.
- Built **Copy-on-Write fork** sharing physical frames via read-only mappings with reference-counted deferred copies; validated with **14 CoW Google Test cases**. **GitHub**

Clash Royale-Inspired Game Engine | *C++17, CMake, Google Test* **December 2024 – June 2025**

- Built OOP engine across **43 source files** using **Factory pattern** + polymorphic inheritance; runs **15+ concurrent entities** on a 40×35 grid at 100ms cycles.
- Autonomous AI with **A* pathfinding**, armor/damage modifiers, and **zero memory leaks** via **RAII** and smart pointers. **GitHub**

GitHub-to-Portfolio | *Java 21, Spring Boot, Kafka, Redis, PostgreSQL, React* **March – April 2026**

- Parses **7 GitHub API endpoints** concurrently via `CompletableFuture.allOf()` into a typed `RepoSnapshot` before any LLM call, cutting extraction from 3–5s to **<2s**. **Live**
- Deployed on **GCP Cloud Run** with **Terraform**, **Neon** serverless Postgres, **Upstash** Redis, and **Confluent** Kafka; **Redis + PostgreSQL dual-write** for O(1) polling with durable history.
- Chose `@Async` over Kafka pipeline in prod to eliminate 30–60s consumer group rebalancing on **scale-to-zero** cold starts; Kafka infrastructure retained for horizontal scaling. **GitHub**