Aryan Mishra

aryanmishra4002@gmail.com | linkedin.com/in/aryan-mishra | github.com/aryan55254

Profile Summary

Information Technology undergraduate interested in building high-performance backend systems and scalable web applications. Experienced with low-latency C++ servers, real-time WebSocket communication, and distributed background processing. Comfortable working with concurrency, load testing, and system-level networking while building practical, production-style projects involving rate limiting, video processing, and AI-backed services.

Technical Skills

Languages: C++, TypeScript, JavaScript
Systems & Networking: POSIX Sockets, WebSockets, Multithreading, Lock-free Programming
Infra, Databases & Tooling: MongoDB, Redis, Docker, BullMQ, FFmpeg, Linux
Web & Auth: Node.js, Express.js, Next.js, React.js, OAuth, JWT, REST APIs

Education

Guru Tegh Bahadur Institute of Technology — New Delhi, India
Bachelor of Technology in Information Technology — *Aug. 2024 – June 2028*

Projects

Chronos | *C++, Multithreading, Lock-free Ops* — Code
- Engineered a low-latency job scheduler using a hybrid queuing architecture (Lock-Free Deque + MPSC Mailbox).
- Achieved less than **5µs P99 latency** and **2.1M+ jobs/sec** throughput using spin-wait spinning and randomized work stealing.
- Implemented custom memory management and cache-aligned structures to eliminate false sharing on the hot path.

Vortex | *TypeScript, Socket.Io, Redis, Docker* — Demo | Code
- Developed a multi-process monolith for video trimming, offloading heavy FFmpeg tasks to BullMQ workers.
- Implemented real-time progress streaming via WebSockets and secure Google OAuth authentication.
- Designed Redis-backed job queues capable of handling **4,000+ concurrent background jobs** reliably under load.

Nexus | *C++, POSIX Sockets, Multithreading* — Code
- Built a high-performance WebSocket group chat server from scratch using raw TCP sockets and thread-per-connection architecture.
- Implemented low-level WebSocket framing and handshake protocols without external network libraries.
- Load tested with **1,000 concurrent clients**, achieving a peak throughput of **17,000+ messages per second**.

Heritage | *Next.js 16, TypeScript, Groq API, Redis* — Live | Code
- Engineered an AI educational chatbot for Indian history using Llama-3.1-8b with context-aware follow-ups.
- Implemented custom stateless authentication and context retention using a Redis-backed sliding window algorithm.
- Enforced security limits (**10 chats / 60s**) to prevent LLM API abuse and ensure system stability.

Droplet | *C++, Node.js* — Code
- Designed a thread-safe Token Bucket rate limiter in C++ enforcing stable request rates with burst allowance.
- Implemented per-client isolation via BucketManager and automatic cleanup of inactive buckets.
- Used Node.js to build load-testing and simulation clients, validating behavior under **20,000 concurrent requests**.