Aarjav Patni

+1 437 484 2405 | aarjav.patni@uwaterloo.ca | aarjavpatni | aarjavpatni | aarjav.xyz

Education

University of Waterloo — Waterloo, ON

Bachelor of Computer Science (Honours) — *Sep 2023 - May 2028 (expected)*

Courses: Object-Oriented Programming, Compilers, Computer Architecture, Data Structures & Algorithms, Cryptography

Experience

Voltra — Waterloo, ON

Software Engineering Intern — *Jan 2025 – Apr 2025*

- Led the development of a **Go REST** API for an ETL pipeline using **AWS S3** that unifies electrical grid data into a single schema across 18 formats and 8 North American regions; processed **200k+** records/day.
- Built internal **gRPC** coordination between the API components for task dispatch, retries, and progress tracking.
- Architected reusable **Terraform** modules to setup and standardize **Kubernetes** (k8s) clusters on **AWS** and **GCP**, enabling consistent deployments to scale backend services powering EV charging stations in Northern Virginia and California.
- Slashed setup time from **20 hours to 5 mins** by building a self-serve **GitOps** workflow in **ArgoCD** to spin up k8s clusters.
- Reduced latency of the **slowest 5%** requests from **1,150 ms to 450 ms (60%)** by deploying **OpenTelemetry** via **SigNoz** for distributed tracing and **Cilium Hubble** to isolate cross-service network issues in production.

Readwise — Toronto, ON

Software Engineering Intern — *May 2024 – Aug 2024*

- Developed a **Django API** powering the Readwise app's feedback ticket prioritization, handling **300+** daily requests.
- Built an internal **TypeScript** Raycast tool for the API, saving the customer experience (CX) team **50+** hours/month.
- Shipped document-search in **React** for the Readwise app, enabling search within documents with **1,000+** pages.
- Optimized desktop auto-update flow so that **React** UI accurately drives the **Rust** updater across **10,000+** app installs.
- Improved PDF to HTML conversion accuracy by **75%** by integrating Diffbot APIs into the document ingestion pipeline.

Recurse Center — New York, NY

Software Engineer — *Jul 2025 – Sep 2025*

- Engineered PyPoker – a **Python** multiplayer poker simulator with custom hand evaluation and real-time play via **WebSockets**; implemented server-side game state, turn/betting rules, and per-table broadcasts to synchronize players.
- Built Vortex – a **Go** project solving Fly.io's Gossip Glomers distributed systems challenges including implementing broadcast/gossip systems, grow-only counters, **Kafka**-style log, and totally-available transactions.
- Collaborated on mind-map-crdt, a **TypeScript** real-time multiplayer mind map using **Conflict-free Replicated Data Types (CRDTs)** for state replication and deterministic merging of concurrent edits.

Projects

Flux | *GitHub* | *Blog Post* — Jan 2026

- Built a low-latency **Rust**-based image processor comparing sequential, batched, and streaming pipelines; measured stage (download → process → save) latency + throughput, and tracked peak memory via PID polling every 100 ms.
- Achieved a throughput speedup from **1 img/s to 13 img/s** using stream vs sequential processing on **10,000** images that overlaps downloads, CPU-bound resizing/encoding, and disk writes with tuned, bounded concurrency.
- Used `tokio` to implement stage-bounded **concurrency** + backpressure, semaphores to cap concurrent downloads/processes/saves, and a blocking pool to run CPU-bound resize/encode to keep the async runtime responsive.

Orderbook Proxy API Server | *GitHub* — Nov 2024

- Developed a proxy API server in **Rust** to speed up trade analytics query execution (buy/sell counts, total volume) over time-series market data from raw CSV logs, cutting end-to-end runtime from **32 s to 11.5 s (64% decrease)**.
- Designed a custom hour-bucket caching algorithm backed by **LRU** eviction, reducing API calls from **1,000 to 164 (84%)**.
- Instrumented the system with structured logs (per-query timers, cache hit/miss counts, upstream call counts) and built a replay benchmark that runs **1,000** queries end-to-end to measure runtime, speedups, and caching tradeoffs.

Hostel Management System | *GitHub* | *Demo* — May 2024

- Developed a full-stack **React + Supabase Postgres** web application to digitize and streamline student tenant management.
- Implemented an admin interface to view, add, and update student and billing records using the **Supabase** API.
- Automated billing with dynamic rent/utility calculations, reducing admin workload by **10 hours weekly**.

Skills

Languages: Rust, Go, Python, C++, TypeScript, JavaScript, C, Java, Bash, PowerShell, Elixir
Technologies: Kubernetes, Terraform, Docker, AWS (EC2, EKS, S3, Lambda), GCP (GCE, GKE), PostgreSQL, Redis, gRPC, CUDA
Frameworks: OpenTelemetry, Django, React, Node.js, Next.js, PyTorch, TensorFlow, OpenCV, Pandas, NumPy, ROS 2