

Trung Nguyen

trungn21@student.ubc.ca | +1 (672)-855-0321 | [linkedin.com/in/tna21](https://www.linkedin.com/in/tna21) | github.com/trungnguyen21 | [technically.engineer](#)

EDUCATION

University of British Columbia

Bachelor of Science, Computer Science (GPA 3.7/4.0 - Dean's List)

Participated in ICPC Campus Qualifier (2025); Open source contribution: Databricks Omnigent (2026)

Sept 2023 – May 2028

Vancouver, BC, Canada

EXPERIENCE

Software Engineer Intern - Performance Engineer

Bank of Montreal

May 2026 – Present

Toronto, ON

- Redesigned a transaction table schema and index cluster to use **table partitioning** and partition switching with staging tables in SQL Server, cutting job processing time by 65% and doubling insert throughput.
- Decreased database query latency by 15%** across 20 stored procedures by decoupling an asynchronous logging solution using SQL Server Service Broker.
- Prototyped an invoice-processing agent to assist users making bill payments using AWS AgentCore, Textract, Bedrock Gateway, and Lambda functions.

Software Engineering Intern

WorkSafeBC

Jan 2025 – Aug 2025

Vancouver, BC

- Developed an LLM-powered browser testing automation tool** using Microsoft Autogen and Playwright to streamline UI testing, saving 100+ hours annually for the QA team.
- Owned the Planner Agent end-to-end in a two-phase plan/execute design: it explores the application once and records key page states and element locators, which an Executor Agent turns into Playwright scripts, saving 20% token usage.
- Identified an **Out of Memory** issue with the FastAPI server due to many concurrent Playwright browser instances during peak traffic, introducing Celery task queue with worker memory cap to prevent the issue.
- Cut agentic correction loops by **50%** and **reduced 75% token costs** per run by adding custom LLM prompts and instructions for interacting with non-standard UI elements.

PROJECTS

FinancialsWatcher (Github) | Python, Kafka, Confluent Cloud

June 2026 – Current

- Architected an event-driven, low-latency data pipeline in Python to detect a company's 8-K earnings filing, analyze its financial implications, and alert users; the bot gained 7000 users year-to-date.
- Built a fault-tolerant message queue system to decouple ingestion from AI analysis, scaling the pipeline to support a dynamic ticker watchlist.
- Routed deterministic trading signals to users via a custom Discord bot and PostgreSQL subscriber database.

Inference Engine From Scratch (Github) | C++, Qwen

September 2026 – Current

- Built a C++ CPU inference engine for Qwen2.5-0.5B-Instruct, implementing 24-layer transformer execution, grouped-query attention, RoPE, and greedy text generation.
- Developed a memory-mapped GGUF loader and custom matrix-vector kernels supporting four quantization formats (Q4_K, Q5_0, Q6_K, Q8_0), computing directly on packed weights.
- Parallelized matrix-vector multiplication with OpenMP and added a per-layer KV cache, raising decode throughput from 0.06 to 5.7 tokens/sec (95x) on an M2 MacBook Air.

Fraud Detection Engine (Github) | Python, Terraform, AWS

July – September 2026

- Built a cloud-native real-time credit card transaction fraud detection system, handled 1k TPS with 400ms p50.
- Developed a multi-threaded Python transaction generator capable of simulating high-throughput data stream, with configurable fraud ratio.
- Designed a modular, heuristic-based Scoring Engine capable of independently evaluating transactions against risk patterns (e.g., rapid velocity, high amounts, odd hours) to assign fraud probability scores.
- Deployed entirely via Infrastructure as Code (IaC) using Terraform.

TECHNICAL SKILLS

Languages: Python, JavaScript/TypeScript, C/C++, Java, SQL, T-SQL

Frameworks: LangChain/LangGraph, Autogen, Google Agent Development Kit, FastAPI, Express, React

Technologies: Git, Prometheus, Grafana, Docker, PostgreSQL, SQL Server, NGINX, Azure, Google Cloud Platform