Nikita Lobanov

Canadian Citizen | nikita@nikitalobanov.com | nikitalobanov.com | github.com/nikitalobanov12 | linkedin.com/in/nikitalobanov

Education

British Columbia Institute of Technology — Expected April 2026
Full Stack Web Development Specialization — *Vancouver, BC*

Relevant Coursework: Distributed Systems, Operating Systems, Database Management, Linear Algebra
Extracurricular: BCIT Computing Club member, Peer mentorship

Skills

Languages: TypeScript, Python, Go, Java, SQL, Bash, Lua
Full Stack: React, Next.js, Node.js (NestJS, Express), Tailwind CSS, FastAPI, gRPC
Infrastructure: AWS (EC2, Lambda, RDS), Docker, Terraform, CI/CD, Linux, Grafana
AI & ML: RAG Pipelines, Prompt Engineering, Vector Embeddings (pgvector, Chroma), LlamaIndex

Experience

Seaspan Corp — May 2024 – Aug 2024
Software Engineer Intern — *Vancouver, BC*

- Architected a real-time telemetry dashboard using React and Recharts to process and visualize 500+ MB of daily Oracle logs, enabling maritime operators to detect engine anomalies 10x faster than manual parsing.
- Engineered a high-throughput Spring Boot microservice to ingest and validate 1,400+ complex accounting files monthly, integrating directly with ERP systems to eliminate 160+ hours of manual data entry.
- Optimized legacy SQL reporting queries by implementing composite indexes and materialized views, reducing execution time from minutes to sub-seconds while enforcing strict RBAC protocols for financial data.

Affistash — March 2023 – April 2024
Software Developer — *Remote*

- Eliminated I/O blocking bottlenecks in the Node.js backend by refactoring sequential batch processing into concurrent Promise pools, reducing p95 latency by 22% and cutting serverless compute costs by 30%.
- Engineered a distributed sliding window rate limiter using Redis and Lua scripts to prevent API abuse, ensuring 99.9% system availability during high-traffic bursts while supporting tiered usage limits.
- Scaled the analytics engine to support 1,000+ brand records by implementing efficient server-side faceted search with dynamic query building, delivering sub-50ms response times for complex multi-filter requests.

Projects

Stochi | *Next.js 16, Go, Postgres, pgvector, HuggingFace* — Dec 2025 – Present

- Engineered a health optimization platform to detect nutrient interactions and timing conflicts using Go and Next.js.
- Achieved 0ms perceived search latency for supplement logging by implementing a browser-side MiniLM-L6-v2 transformer model in a Web Worker for real-time fuzzy matching, eliminating unnecessary server round-trips.
- Optimized interaction checks to ¡10ms by architecting a Go engine that traverses graph-based stoichiometric rules; implemented Michaelis-Menten kinetics to model saturable absorption and elemental weight conversions across 17 database models, enforcing hard safety limits from NIH and FDA data sources.
- Reduced AI hallucinations by building a RAG pipeline with Llama 3.1 8B to ground advice in 500+ research studies.

Panday | *Next.js 15, Go, PostgreSQL, pgvector, Redis* — Sep – Dec 2025

- Led a cross-functional team of 8 (5 developers, 3 designers) to architect an AI-driven career planning platform that transforms unstructured government regulations into interactive roadmaps to address the skilled trades shortage.
- Implemented a semantic search pipeline using OpenAI embeddings and pgvector, integrating a Redis cache to store frequent queries which reduced external API costs by ∼80% and ensured sub-second response times.
- Solved dev/prod parity issues by engineering a custom Redis Proxy in Go that intercepts TCP requests and translates them to HTTP, bridging the gap between local Docker containers and the serverless Vercel production environment.
- Optimized the interactive graph visualization engine by reducing collision detection complexity from $O(n^2)$ to $O(n)$ via grid-based spatial partitioning, enabling smooth 60fps rendering for 100+ nodes.

Dayflow | *React 19, Tauri 2.0, Supabase Edge* — Jun – Aug 2025

- Built a personal AI task planner that grew to 300+ active users, utilizing Supabase Edge Functions and Generative AI to autonomously schedule tasks based on historical velocity and user constraints.
- Achieved 0ms perceived latency for task operations by implementing optimistic UI state management strategies combined with a custom client-side LRU cache to manage updates and rollback on failure.
- Engineered a unified cross-platform architecture using Tauri 2.0, enabling a single TypeScript codebase to compile into native binaries for Windows, macOS, and Linux with platform-specific window controls.
- Secured Google Calendar synchronization by implementing a server-side OAuth 2.0 flow with token encryption and automatic refresh, ensuring zero client-side exposure of sensitive credentials.