Larris Xie

larris.xie@uwaterloo.ca · 437-974-6166 · larris.me · LinkedIn · GitHub

Education

University of Waterloo Sep 2024 - Apr 2028
Bachelor's of Computer Science **4.0/4.0 GPA**
Scholarships: *Nortel Institute, President's Scholarship, President's Research Award ($7500)*
Relevant Coursework: Object-Oriented Software Development (Enriched), Data Structures and Data Management (Enriched), Algorithm Design and Data Abstraction (Enriched)

Experience

Shopify May 2025 - Aug 2025
Software Engineering Intern

- Orchestrated a **100% uptime data migration** from legacy APIs to new service endpoints while ensuring full **backwards compatibility** through shadow writes, backfilling **5B+ rows**, and staged rollouts.
- Designed and implemented scalable **data models and GraphQL APIs** in **Ruby on Rails**, powering a fully extensible cash management experience for **2M+ active retailers**.
- Engineered full-stack features for Shopify Point of Sale (POS) in **React Native and TypeScript**, resolving critical race conditions in async workflows to reduce failure rate by **40%.**

Meta x MLH May 2025 - Aug 2025
Production Engineering Fellow

- Deployed and managed **Linux infrastructure** on a CentOS VPS, leveraging system calls and kernel-level debugging at the OS layer, and **Docker** and NGINX at the application layer to improve reliability.
- Configured system monitoring with **Prometheus, cAdvisor, and Grafana**, leveraging low-level resource metrics (CPU, cache misses, memory usage, I/O throughput) to cut bug detection time by **80%.**
- Automated **CI/CD workflows** with GitHub Actions, integrating testing frameworks, security scans, and performance monitoring to streamline deployments and improve system reliability.

University of Waterloo Jan 2025 - Present
Undergraduate Research Assistant — Security and Privacy in ML

- Built a privacy-preserving **vertical federated learning** system for fraud detection, leveraging a **GAN** and encrypted communication protocols to enable secure cross-party training on distributed time-series data.
- Achieved a **95% F1 score** under a 10000:1 class imbalance by training custom **transformer models in PyTorch** under federated learning and performing large-scale hyperparameter optimization.
- Engineered a **distributed secure inference** pipeline using **Facebook's CrypTen MPC** and **PyTorch's RPC**, providing membership inference resistant serving on GPUs and maintaining **100% of performance**.

Jobeyze Dec 2024 - Mar 2025
Software Engineering Intern

- Spearheaded the development of an **automated web scraper** using Selenium and Laravel, extracting **1200+ job postings daily** and eliminating manual collection entirely.
- Built a **RESTful API** with a structured **MySQL** schema, enabling real-time syncing of postings and application links across platforms used by thousands of active users daily.

Projects

On The Dot (2500+ users) | *Python, TypeScript, PostgreSQL, React, GCP*

- Launched a trivia game with persistent game state and anonymous user tracking—**10k plays in 3 months**.
- Architected a Python backend with custom **PostgreSQL** functions to achieve **20ms read/write latency** on **30K queries/day** for real-time statistics tracking and rate control.

LifeOS (Hack the 6ix Winner) | *Python, TypeScript, FastAPI, Redis, Docker, PostgreSQL, React, AWS*

- Engineered a **distributed video processing pipeline** using **Redis Queue** that orchestrates semantic analysis with TwelveLabs, embedding generation, and cloud storage via **asynchronous workers**.
- Deployed a **containerized API service** with **Docker**, FastAPI, Pydantic models, and JWT middleware; supported modular features like authentication, video ingestion, memory search, and chat.

QAI (Agentic QA in CI/CD) | *Python, TypeScript, FastAPI, PostgreSQL, GitHub Actions, React, AWS*

- Developed an autonomous, **multi-agent QA framework** orchestrating **computer-use agents** in virtual machines to execute web test suites in parallel with **async Python.**
- Built a **GitHub Action pipeline** that converts PR context into structured, executable tests and spins up the agents which have reasoning traces, video recordings via FFmpeg, and summaries of each test.