RYAN SADEGHI-JAVID

https://ryanjavid.com/ https://github.com/Ryan-Sadeghi-Javid javidryan7@gmail.com (416)-700-8414

EDUCATION

University of Waterloo *September 2023 - April 2028*
*Bachelor of Applied Science in **Electrical Engineering**, Honors, Co-op*
***Relevant Coursework**:Digital Circuits (FPGA design with VHDL), Algorithms and Data Structures (C++)*

TECHNICAL SKILLS

Languages:	C++, VHDL, C# Java, TypeScript, JavaScript, Python, Solidity, PowerShell, Tailwind CSS
Hardware & Systems:	Digital Logic, FPGAs, SCADA, Embedded Systems, Arduino, Serial Communication
Software & Tools:	Git, AWS, Spring Framework, Docker, Node.js, React, REST APIs, MongoDB, MySQL, Excel

WORK EXPERINCE

Unity Health Toronto — AWS, Python, Java, JavaScript, React, MongoDB *September 2024 - December 2024*
Software Engineer
- Developed the **electronic Asthma Management System (eAMS)**, a scalable application improving asthma care across Canada, serving **10,000+** patients. Contributed to **backend (Java - Spring Framework, Javascript)** API development and **frontend (React)**, creating a user-friendly doctor interface for patient management.
- Engineered **AWS** cloud infrastructure, optimizing healthcare application, security, and performance. Leveraged **AWS S3, EC2, and Lambda**, cutting infrastructure costs by **20%** and reducing API response times by **35%**.
- Built a physician application system for **Mainpro credits**, developing a **MongoDB database**, **Node.js** backend, and **React** frontend, deployed from development to production, automating 100% of the physician credit approval process.
- Automated data tagging and export processes using **Python**, integrating **AWS S3** to streamline healthcare data analysis and ensure compliance with data management best practices.

Crosslinx Transit Solutions (Eglinton LRT) — SCADA, Python, PowerShell *January 2024 - April 2024*
System Engineer
- Developed **automated PowerShell** scripts to streamline CCTV camera maintenance, enabling scheduled reboots and real-time status monitoring, reducing system downtime by **15%** and minimizing manual intervention by **60%**.
- Built Python scripts to analyze and process large Excel datasets (10,000+ entries), generating frequency reports of offline cameras, identifying failure trends, and improving preventive maintenance efficiency by **40%**.
- **Executed System Acceptance Testing (SAT)**, collaborating with **engineers and QA teams** to validate system functionality, identify defects, and ensure compliance with transit safety standards.
- Monitored and controlled transit infrastructure using **SCADA software**, analyzing system alerts and executing rapid issue resolution to maintain **optimal operational efficiency**.

PROJECTS

Healic – Winner of Best Web App at NextStep Hacks (https://healic.vercel.app/) — React, Node.js, Socket.io, OpenAI API
- Developed a peer-to-peer mental health platform with real-time chat, dynamic user matching, and a shared resource hub.
- Built **RESTful APIs** using **Node.js** and **Express.js**, and integrated **Socket.io** for real-time messaging between anonymous users
- Integrated **OpenAI** to deliver empathetic, AI-powered response suggestions and improve user engagement.
- Led backend architecture and deployment on **Render**; integrated the frontend with **React** and **Vercel** for seamless delivery.
- Designed a dynamic queuing algorithm to match users based on overlapping mental health concerns in real time.
- Worked in a cross-functional team to coordinate backend, **UI/UX**, and **API logic**, ensuring rapid iteration and feature completeness.
- Delivered a full-stack MVP under tight hackathon deadlines; awarded Best Web App at NextStep Hacks for innovation and impact.

ETL Graph Relationship Engine (github: ETL-pipeline) — C++, STL, Graphs, Trees, Parsing Algorithms
- Engineered a **C++** ETL pipeline to extract and analyze global trade data using a directed graph with efficient real-time queries.
- Designed **graph structures** where nodes stored country data and edges encoded unique relationships from statistical thresholds.
- Parsed 60^{+} years of time-series data using efficient **STL containers** (`unordered_map`, `set`, `BST`) and custom parsing logic.
- Built graph features including **pathfinding**, **edge updates**, and **relationship queries** to simulate real-world data operations.
- Optimized performance on large datasets by reducing memory usage and using **cache-friendly STL traversal patterns**.
- Developed preprocessing routines to validate and normalize raw input files, handling missing values, malformed records, and inconsistent time-series formats.