Adeel Akhani

Personal Portfolio | adeelakhani | adeelakhani | (647) 513-3926 | aakhani@uwaterloo.ca

Education

University of Waterloo
Bachelor of Software Engineering *Present*

Technical Skills

Languages: TypeScript, JavaScript, HTML, CSS, Python, Java, C++, C, C#
Concepts: RESTful API Design, CI/CD, Object-Oriented Programming, Data Structures & Algorithms
Technologies: AWS, Google Cloud Platform, Docker, PostgreSQL, MongoDB, Supabase, Vercel, Render, Unity
Libraries & Frameworks: Next.js, React, Tailwind CSS, Node.js, Express.js, FastAPI, Flask, Django, LangChain, CrewAI, OpenAI Agents SDK, Playwright, Mongoose, EJS, Scikit-learn, Pandas, NumPy, Matplotlib, Java Swing/AWT
Tools: Git, Github, GitLab, Postman, Unix

Experience

Software Developer Intern May 2025 – Present
Script Runner
- Developing software solutions for pharmacies using **React/Redux**, **Express.js**, **PostgreSQL**, **Redis**, and **GCP**
- Helped implement the **MVP** for **Uber Direct integration** using their **API**, **pricing algorithms**, and **validation systems** to enable third-party delivery services.
- **Developed workflows** for scheduling, payment tracking, and dashboard improvements to streamline operations
- **Designed** and **performed testing** across staging environments, resulting in **30+** bug fixes for production release

Software Engineering Intern Jul 2022 – Aug 2022
SoftSages Technology
- Implementing ML models using **scikit-learn**, **pandas**, and **NumPy** to provide software solutions
- Developed a **logistic regression model** with a **93% accuracy** to predict an employee's future based on HR data
- Made a email spam detection model using a **TfidfVectorizer** for NLP pre-processing, achieving a **90% accuracy**

Projects

LoopyAI | *FastAPI, PostHog API, OpenAI Agents SDK(via Gemini), Next.js Supabase(PostgreSQL)* Jun 2025
- LoopyAI analyzes session replays to detect key issues and deliver agentic insights with contextual playback
- Used **FastAPI** and **OpenAI Agents SDK** to analyze session data from DOM snapshots via the **PostHog API**
- Used a **Supabase**-hosted **PostgreSQL** database to store structured data on important user sessions
- Designed an aesthetic frontend with animations using **Next.js**

LetsCook | *TS, Next.js, Express.js, Supabase(PostgreSQL, Auth), Docker, GCP, Vercel* Feb 2025 – Jun 2025
- Full-stack social cooking platform which allows users to share cooking challenges, and earn points
- Utilized **Express.js** in developing **RESTful APIs** for profile management, creations/submissions, and workflows
- Leveraged Supabase for image uploads and operating a **RLS** enforced **PostgreSQL** database with SQL functions
- Deployed **Dockerized** backend to **Google Cloud Run** with **GitHub Actions CI/CD**
- Deploying an appealing frontend on **Vercel** using **React** with **Next.js**, **Tailwind CSS**, and **Shadcn/ui**
- Integrated authentication via **Google OAuth 2.0** through Supabase

MakeSomething | *React, Node.js, Express, Vercel, Render, Claude AI API, Axios* Jan 2025
- Full-stack web application for generating recipes based on available ingredients reaching **30+** users
- Created a **RESTful API** with **Node.js** and **Express** to process data and generate recipes using **Claude API**
- Engineered a dynamic frontend with **React** and **Vite**, featuring real-time ingredient recommendations
- Deployed on **Vercel** with backend hosted on **Render**

RememberGranny | *Flask, PyTorch, LSTM, HTML, CSS, Groq API* Oct 2024
- Web application for elderly users that provides password protection tools and story-based memory assistance
- Implemented password strength classification via a **LSTM network** in **PyTorch** trained with **760,000** passwords
- Developed a RESTful API using **Flask** to handle backend logic and ML model interaction
- Provided **story generation** using **Groq API** based on the users password to help them remember it
- **Password generation** using Groq API based on a personal questionnaire, that are guaranteed strong passwords
- Achieved a effective and simple frontend using **HTML** and **CSS** and **Flask** backend for API calls