Chenyu Wei

c63wei@uwaterloo.ca | 249-994-0088 | LinkedIn | GitHub | Portfolio

Education

University of Waterloo **Waterloo, ON**
Bachelor of Mathematics in Computer Science, Human-Computer Interaction Specialization *Sep 2024 - May 2029*

- Coursework: Object-Oriented Programming, User Interface, Operating Systems, Assembly Language, Algorithms

Experience

Dameng Database Company **Wuhan, China**
R&D Research Intern, Technology Research Department *May 2026 - Aug 2026*

- Reimplemented and evaluated an **LLM** prototype generating **C/C++** functions for **SQLite, PostgreSQL, and DuckDB**; automated a generate-compile-test-validate loop that fed compiler and unit-test errors back to the model for iterative repair, reducing manual execution and inspection.
- Collaborated on a **Dify/MCP** workflow that classified and summarized **378** database papers into structured JSON for research review; designed prompts, added schema checks, and refined outputs from reviewer feedback.
- Documented setup, architecture, results, and limitations; demonstrated the prototype internally and handed off code and reports to R&D. Published walkthroughs: DBCooker/OpenCook, Semantic Operator Estimation.

Black Sesame Technologies **Wuhan, China**
AI Application Intern *May 2025 - Aug 2025*

- Designed and implemented the training-data pipeline for an AEB multimodal decision project and supported **Qwen-VL** fine-tuning; matched **50,000+** road images with radar data, vehicle signals, and labels by frame ID.
- Automated prompt construction and internal model-service calls in **Python**, adding JSON field/type validation, bounded retries, error logging, and resumable processing to reduce manual corrections.
- Ran model-training workloads on the company's **Kubernetes**-based platform using **Docker** images, configuring training commands and monitoring execution logs.
- Used **OpenCV** to select key frames from 20-frame sequences, reducing image input by **about 80%** while retaining the final frame; grouped samples by driving scenario for comparative experiments.

Projects

FeedbackFlow | *React, FastAPI, PostgreSQL, Fastify, Docker, Kubernetes, Azure* *Aug 2026 - Oct 2026*

- Independently designed and built a full-stack platform that turns user feedback into traceable requirements and engineering tasks, owning architecture through **Azure** deployment; used Codex as a coding assistant.
- Built a **React/TypeScript** dashboard with **TanStack Query** and a typed API client, including a triage inbox with server-side search, filters, pagination, bulk status updates, and inline tags and assignees.
- Designed **FastAPI** REST APIs on **PostgreSQL** with user login, separate data for each team workspace, and requirements that stay linked to their original feedback as they move from draft to delivered.
- Built a dependency-free **JavaScript** widget that any product site adds with one script tag, letting end users send feedback straight into the team's inbox through a rate-limited, duplicate-safe public API.
- Built background analysis that drafts requirements with an OpenAI-compatible **LLM**, schema-validating output and versioning runs for retries; exposed the same services as **MCP** tools.
- Wrote a **Node.js**/TypeScript (**Fastify**) notifier; the API publishes events after responding so delivery never blocks users, and the notifier fans out Slack and HMAC-signed webhooks with exponential backoff.
- Shipped multi-stage **Docker** images to **Azure Container Apps** via **ACR** and **GitHub Actions**, provisioning Azure with **Bicep** and OIDC identities; wrote **Kustomize** manifests for **Kubernetes**, run on **kind**.

Macro Coach (Amazon AppDev Hackathon) | *MCP, AWS Lambda, DynamoDB, CDK, Bedrock* *Oct 2026*

- Built a voice-first nutrition coach for **Alexa+**: users say what they ate, and a **TypeScript MCP** server logs and corrects meals, returning macros, remaining budgets, meal ideas, and weekly reports as structured data for screens plus a short spoken summary for voice.
- Used **Amazon Bedrock** to parse free-text meals into food, quantity, and unit, then matched each item to **USDA FoodData Central** so every number comes from real nutrition data rather than model guesses.
- Deployed a stateless server on **AWS Lambda** and **API Gateway** with **DynamoDB** storage, provisioned in one **AWS CDK** command along with a nightly **EventBridge** job and **S3/CloudFront** hosting.
- Built a **React** Alexa+ simulator with push-to-talk voice and spoken replies that acts as a real MCP client against the deployed server, showing each tool call live next to macro-progress rings.

Text-to-SQL Agent | *FastAPI, React, SQLGlot, SQLite, LLM APIs* *Jun 2026 - Aug 2026*

- Extended a command-line **LLM** SQL agent into a typed **React/FastAPI** app for natural-language queries with SQL validation, result exploration, and query history kept apart from analytical data.
- Rejected invalid or unsafe SQL with **SQLGlot** AST checks and **SQLite EXPLAIN** under read-only limits, feeding errors to a bounded self-correction loop; reached **92% (23/25)** accuracy on a 7-table benchmark.

Technical Skills

Languages: Python, TypeScript/JavaScript, SQL, C/C++, Java, Bash | **Tools:** GitHub Actions, Linux, Git, Pytest