Selangor, Malaysia
Linkedin
GitHub

Teh Hung Wei

(+60) 18-323 0302
hwteh1997@gmail.com

I am a full-stack software and AI engineer especially interested in building web application experiences, scaling systems up, and producing reliable AI applications. I seek a full-time role to apply my skills, embrace challenges, collaborate with diverse teams, and contribute meaningfully to an organization.

Work Experience

AI Software Engineer **Anhsin Technology Sdn. Bhd.** **April 2024 – Jan 2025**
AI & Backend team Kuala Lumpur, Malaysia

- Developed and deployed an AI-powered chatbot using RAG architecture, integrating OpenAI, LangChain, and FastAPI, improving response accuracy by 40%.
- Designed and optimized the system backend in Python, reducing API response time by 30% through efficient database indexing and caching.
- Implemented scalable infrastructure on AWS (EC2, S3, RDS), improving system uptime to 99.9% and reducing cloud costs by 15%.
- Managed and optimized NoSQL databases (MongoDB), ensuring seamless data storage and retrieval for chatbot sessions.
- Integrated streaming APIs using Server-Sent Events (SSE) for real-time chatbot responses, enhancing user engagement.
- Developed frontend components using TypeScript and Next.js, building the homepage and login page with a responsive design.
- Implemented authentication middleware for secure access token and refresh token mechanisms, ensuring seamless user authentication and session management.

Technologies and Languages

- Languages: Python, JavaScript, TypeScript, Go
- Frameworks: React, NextJS, NestJS, FastAPI
- Databases: MySQL, Postgres, MongoDB, ChromaDB
- Cloud: AWS, Google Cloud Platform
- Other: Redis, Docker, CI/CD

Education and Certifications

- **M.Sc. Computer Science,** University of Teknologi, Malaysia. **2022 - 2023**
- **B.Sc. Industrial Physics,** University of Teknologi, Malaysia. **2017 – 2021**

Projects

1. Scalable Backend System with Go & gRPC Microservices
    a. Overview:
    Developed a robust full-backend system in Go, architected as microservices communicating via gRPC, this design enables high scalability and maintainability while ensuring efficient inter-service communication.
    b. Technologies:
        i. Languages: Go
        ii. Communication & Documentation: gRPC, Swagger
        iii. Infrastructure: Docker, Consul

iv. Databases: PostgreSQL, MongoDB, Redis
v. Messaging & Monitoring: RabbitMQ, Jaeger

c. Impact & Outcome:
i. Streamlined inter-service communication, supporting rapid scaling and reliable service integration.
ii. Enhanced system observability and accelerated issue resolution with Jaeger tracing.
iii. Established a robust, self-documented API layer that improved team collaboration and external integration efforts.

d. Challenges Addressed:
i. Implementing gRPC connections for seamless data exchange between microservices.
ii. Configuring service discovery and load balancing with Consul.
iii. Integrating diverse technologies to maintain a consistent API documentation process while ensuring data integrity across multiple databases.

e. Link:
GitHub Repo Project Link - Fullbackend
GitHub Repo Project Link - Microservices

2. Chatbot Backend with Self-Hosted LLM Integration
a. Overview:
Engineered a dynamic chatbot backend in Go that integrates with a self-hosted Large Language Model (LLM) using Ollama. The LLM service is deployed on a separate local machine, and an HTTP connection is established between the services through strategic DNS and port forwarding configurations.

b. Technologies:
i. Languages: Go
ii. AI integration: Self-hosted LLM via Ollama
iii. Infrastructure: Docker, MongoDB
iv. Networking: DNS Configuration, Port Forwarding, HTTP communication

c. Impact & Outcome:
i. Enabled seamless integration between the chatbot backend and the LLM service, ensuring efficient and reliable communication over HTTP.
ii. Achieved a modular architecture that decouples the AI model from the backend, allowing independent scaling and maintenance.
iii. Improved accessibility and security by configuring DNS and port forwarding to maintain a stable connection between distributed services.

d. Challenges Addressed:
i. Establishing secure and stable HTTP connections between services hosted on different local machines.
ii. Configuring DNS and port forwarding to overcome between services hosted on different local machines.
iii. Integrating a self-hosted LLM model into the production environment, balancing performance with system reliability.

e. Link:
Github Repo Project Link - LocalLLM