Kayan Shih

Providence, RI | kayan_shih@brown.edu | 401-248-8545 | linkedin.com/in/kayans | github.com/kayans

Education

Brown University Providence, RI

Master of Science in Computer Science – GPA: 4.0/4.0 *September 2023 – May 2025*

- Relevant Coursework: Database Management, Computer Networks, Computational Linguistics, Computer Systems

New York University New York, NY

Bachelor of Arts in Computer Science – GPA: 3.7/4.0 *September 2019 – May 2023*

- Relevant Coursework: Operating System, Computer System Organization, Machine Learning, Deep Learning, Responsible Data Science, Algorithm, Data Structure
- Honor: Dean's list 2019 & 2021

Technical Skills

Languages: Python, Java, Golang, C/C++, JavaScript, TypeScript, HTML/CSS, Solidity
Frameworks: React, Node.js, Express.js, Next.js, JQuery, Jest, Supertest, JMeter
Developer Tools: Git, Docker, Kubernetes, AWS, VS Code
Databases: MySQL, PostgreSQL, RocksDB, MongoDB, DynamoDB, Redis

Experience

Software Engineer Intern June – Aug 2023

SlowMist *Xiamen, China*

- Improved the security of 10+ Ethereum-based projects by identifying and resolving vulnerabilities in ERC20 and ERC1155 token contracts written in **Solidity**, optimizing token-vault interactions and restructuring permissions.
- Collaborated with 3 cross-functional teams using **Agile** methodologies, delivering reports that led to security improvements, mitigating risks and ensuring safer deployments for high-value DeFi projects.

Projects

Event Ticketing Platform June – Aug 2024

- Developed a scalable ticket booking web application to handle ticket browsing, ordering, reselling, and payment processing, using **Node** and **Express**; containerized microservices with **Docker** and deployed with **Kubernetes**.
- Improved event transfer reliability by implementing **NATS Streaming** Server as an event bus, enabling seamless communication between microservices and ensuring safe concurrent transactions through **optimistic locking**.
- Enhanced payment and subscription features by integrating Stripe API and webhooks to manage payment processing and ensure consistent handling of concurrent user transactions.
- Developed dynamic and responsive frontend pages using **React**, **Next.js**, and **TypeScript**, leveraging **JQuery** for event handling and **MongoDB** for managing order data, providing users with a smooth experience.
- Increased deployment efficiency and test automation by building a CI/CD pipeline using GitHub Actions, creating unit tests with **Jest** and **Supertest**, conducting end-to-end usability tests on concurrency control with **JMeter**.

Distributed Key-Value Storage System June – Aug 2024

- Designed and developed a distributed key-value storage system using **Multi-Raft** architecture, achieving 99.9% availability and supporting 1 million QPS by implementing linearizable K-V read/write operations in **Golang**.
- Improved data distribution across shards by leveraging **consistent hashing**, and implemented efficient data migration strategies for dynamic load balancing between **Raft Groups**.
- Reduced read latency by **42%** by implementing **async Apply** for non-blocking log entry processing, utilizing **ReadIndex** for non-blocking reads, and integrating **FollowerRead** to allow followers to handle requests directly.
- Optimized system performance by integrating multiple storage engines (**RocksDB**, B+ trees, hash tables), customizing each for different storage requirements, and implementing multi-version concurrency control (**MVCC**) to ensure efficient concurrency handling and data integrity.

AI-Powered Recipe Generation App June – Aug 2024

- Developed a serverless web application that generates recipes based on user-provided ingredients, leveraging **AWS Amplify** and **Amazon Bedrock** with the Claude 3 Sonnet foundation model, and hosted frontend on CDN.
- Designed and implemented a **GraphQL API** with custom types to handle dynamic input like ingredient arrays, optimizing queries with batching and resolvers for efficient processing.
- Created an efficient data model in **DynamoDB** for user and recipe data with sparse indexing, enabling Auto Scaling for the database to scale.
- Built the backend using **AWS Lambda** to process user-submitted ingredients, invoking appropriate functions after user sign-up confirmation for real-time request handling without server management.
- Enhanced the web application's 2Auth by implementing an Email one-time-password authentication with **JWT** using **Amazon Cognito**.