Shunqi(Chris) Liu

shunqil@usc.edu | 213-245-7331 | Los Angeles, CA | https://www.linkedin.com/in/shunqil/

EDUCATION

University of Southern California (GPA: 4.0) — **Los Angeles, CA**
Master of Computer Science — **Dec. 2022**

Shandong University (GPA: 3.8) — **China**
Bachelor of Software Engineering — **Jun. 2020**

SKILLS

- **Programming:** Java, Python, Go, C, C++, C#, MATLAB, HTML, SQL, PromQL
- **Framework:** SpringMVC, Spring Boot, Junit, RxJava, Log4j, Gin, Swaggo, Django, Swagger, Scrapy
- **Tools:** Cucumber, Lombok, Maven, Azure, MongoDB, Redis, Docker, Linux, MySQL, Anaconda, Prometheus, Grafana

WORK EXPERIENCE

Software Engineer Intern | VMware — **Feb. 2021–Jun. 2021**

- Worked on **VMware Horizon** about a component of lifecycle management for VMs in different cloud environments.
- Used **Java and RxJava** to implement Inventory updates in the situation of restarting a powered off VM on ZeroCloud.
- Applied **Mockito** with Java **Spring** based on **Azure** real and mock server and fixed 5 bugs accordingly.
- Developed 70 unit tests and changed 5000+ lines of code to improve code coverage to 70% based on actions to VM by utilizing **Cucumber** and **Junit**.
- Collected and cleaned 200,000 CPU and memory usage records from cluster through **Wavefront** and **Swagger** and helped to plan when/where to run chaos experiments.
- Supported scenarios that credentials should be updated in credential service and interacted with the Azure token process.

Software Engineer Intern | OPPO — **Sept. 2020–Dec. 2020**

- Worked for middleware team in **OPPO Cloud** about a store system to replace Redis and its auxiliary control platform.
- Applied **Gin, Monkey, Require, Sqlx** to finish development and testing tasks of clusters control platform in **Go**.
- Used **Swagger** to automatically generate API documents and avoided manual maintenance on API documents.
- Designed new **Grafana** monitoring pages and metrics through **PromQL** and developed automatic registration function based on **Prometheus** and **Consul** and simplified 90% of manual operations.
- Conducted a performance comparison test of new KV storage system in Intel non-volatile memory and Intel SSD by operating **Benchmark** and found a 30% efficiency improvement in read and write rates.

Software Engineer Intern | China Sports Lottery Center — **Oct. 2019–Dec. 2019**

- Automatically acquired historical data on football matches by using **Xpath** and Python **Request**.
- Utilized Python **Selenium** to enable automatic login and user status change and abnormal situations handling.
- Analyzed football lottery impact factors and stored 330,000 records in **MySQL** database.

PROFESSIONAL EXPERIENCE

Online Monitoring Website for Soybean Price Trend — **Jul. 2020–Sept. 2020**
(Full Stack Website, Django, MongoDB)

- Designed an online monitoring website for soybean price trend based on **MVC** framework.
- Accomplished all front-end design of website through **Django**, **Semantic UI**, including navigation pop-up box and other functions.
- Deployed a database with **MongoDB** to store 200,000 soybean price records gathered from exchange websites.
- Applied **Highcharts** to visualize soybean price data and implemented an interactive line chart based on soybean price fluctuations over time.
- Completed intranet mapping and server by using **Sunny-ngrok**.

Distributed Crawler System for Football Lottery Information — **Feb. 2020–Jun. 2020**
(Crawler System, Redis, Python, Grafana)

- Constructed a **Python** distributed crawler system to collect Football Lottery Information with multiple nodes.
- Applied **Scrapy** Asynchronous Network Framework to extract data and improved crawler efficiency by 300%.
- Established IP proxy pools and a download middleware to automatically change IP and user-agent information.
- Utilized **Redis** for distributed crawling to store request information and centralize management of crawler.
- Implemented a collector to monitor running status of crawlers and figures were displayed in **Grafana**.
- This system ran on 10 nodes with 33,000 records updated daily and 6,000,000 records stored in **MongoDB** database.