# JACKIE LIU

wenyanli@andrew.cmu.edu • (412) 999-2183 • https://www.linkedin.com/in/wenyanl/

## EDUCATION

**Carnegie Mellon University, Pittsburgh, PA** *Sep 2018 – May 2020*
*Master of Science, Electrical and Computer Engineering* **GPA: 3.80 / 4.00**

- *Main Courses:* Foundations of Computer Systems, Data Structures for Application Programmers, Distributed Systems, Advanced Cloud Computing, Internet Services, NoSQL Database Management, Objected Oriented Analysis and Design

**Porto Business School, Porto, Portugal** *Sep 2017 – Aug 2018*
*Master of Business Administration* **GPA: 15.0 / 20.0**

**Wuhan University of Technology, Wuhan, China** *Sep 2011 – June 2015*
*Bachelor of Science, Electrical and Information Engineering* **GPA: 89.5 / 100.0**

## PROFESSIONAL EXPERIENCE

**Solve Education Holdings Pte Ltd, Palo Alto, USA** *July 2020 – Present*
*Backend and Web Development Intern*

- Worked on a QA based educational game that inspires students' interest by adding a sense of competition within.
- Implemented the server-side using **Java and Node.js**.
- Developed the client-side using **Node.js, PHP, and Express.js** and implemented a separate internationalization for students across the globe.
- Designed and developed the infrastructure that scales the platform using **AWS CloudWatch, AWS Lambda, Terraform**.
- Designed a load balancer using PHP-FPM Fast CGI with HAProxy and Nginx.
- Optimized server's performance by adding cache alongside **MariaDB, and MySQL**.
- Bridged client-side and server-side with **RESTful API**.
- Corporate with over 50 members and followed **Agile** Development approach to minimize communication overhead.

**Wuhan GDCC Intelligent Technology Co., Ltd, Wuhan, China** *Aug 2015 – Dec 2016*
*Software Engineer Intern*

- Designed actual toy teddybears that enable long-distance couples to communicate by waving hands, sending emojis, and delivering messages via a centralized server.
- Designed a **multi-thread** server using a thread pool based on producer and consumer model that enables a high request-per-second (RPS).
- Implemented a frontend alongside the actual toy bear that enables couples to visualize various data, and message log.
- Implemented the backend with **Node.js, Express.js, and MongoDB**.
- Implemented an **android** application with Android Studio that controls the behaviors of the toy alongside the toy bear itself with various teammates.
- Bridged frontend and backend using CRUD operation for **RESTful API** service.

## TECHNICAL STRENGTHS

**Programming Languages: Java - proficient**, **C - proficient**, **JavaScript - advanced**, **Python - advanced**, **PHP - advanced**, C++ - intermediate, html - intermediate, CSS - intermediate
**Frameworks: Kafka**, **Spark**, **Spring Boot**, **AWS Lambda**, Redis, **MongoDB**, **Docker**, Bootstrap, **Java RMI**, Android, **Restful API**, **Kubernetes**

## PROJECTS

**Microsoft Talent Program – High Efficiency Realtime News Recommendation System (Spring Boot, Java, Spark)**

- Develop a highly scalable real-time news recommendation system based on **Java, Spring Boot, Kafka, and Elasticsearch** which could support a **querie-per-second (QPS) over 1200.**
- Used Spring Cloud Schema Registry to support data model evolution over time, with the related schema information stored in **JSON**.
- Implemented a high concurrency real-time web crawler which get millions-level news data, preprocessing and feeding the data into **Kafka** with designed protocols and doing **ETL** processing and news data classification with **Spark ML**.
- Optimized Spark ML program structure to gain a higher degree of efficiency which saved 20% of the budget.
- Designed and implemented a news publisher application with Java to read data from Elasticsearch and publish data to clients through **Restful API.**
- Develop a high-efficiency news client application with Java to provide a User interface to see the news with the optimized **WebSocket** protocol.

**Autoscaling Distributed Cloud Application based on Terraform (Terraform, AWS Lambda)**

- Developed a **cloud infrastructure** based on **Django** that supports an image-classification engine.

- Enabled high-performance autoscaling in more than 4 request patterns, including abrupt start and stop pattern, consistent high workload pattern, continuous increasing pattern, and continuous decreasing pattern using **AWS EC2, Terraform, AWS CloudWatch, and AWS Lambda**.
- Designed scaling rules and tuned parameters within Terraform by dynamically monitoring the clients' load using **AWS CloudWatch**.
- Designed security rules with **Terraform and AWS CLI** to manage user privilege.

### Web Blog for Large-scale Concurrent User Access (JavaScript, HTML, CSS, MySQL, Spring Boot)

- Developed a full stack web blog that supports users' login, blog posting, commenting, and deleting.
- Implemented backend using **AWS EC2, Apache Tomcat, Servlet, MVC model, and Spring Boot.**
- Implemented frontend using **HTML5, CSS, JavaScript, jQuery, and Bootstrap**.
- Set up a relational database with **MySQL and Generic DAO** to store users' input and prevent **SQL Injection attacks**.
- Managed concurrency issues within the system that allows **over 100k simultaneous accesses**.

### File Caching Proxy for Distributed File System (Java RMI, Concurrency, Multithread)

- Created a baseline file server and a client-side file caching proxy with low concurrent file operations latency.
- Utilized **check-on-use approaches** by ensuring any requests checked the latest version before granting data access.
- Implemented an optimized version file caching proxy using **lease-based protocols** and a mechanism that **tuned lease time** based on **matching detected patterns**.
- Wrote appropriate tests that generated 27 patterns based on the results to realize an **intelligent pattern match** function.
- Increased read-ops of optimized file caching proxy system by **19.7%** and write-ops by **25.7%.**

### Iterative Machine Learning Training on AWS using Apache Spark (PySpark, Machine Learning)

- Conducted **Extract, Transform, Load (ETL)** on a raw dataset and trained models iteratively with PySpark.
- Optimized **logistic regression and gradient descent** machine learning algorithm to process a KDD2010 dataset in less than 31 minutes, KDD2012 dataset in less than 63 minutes, and Criteo dataset in less than 48 minutes.
- Achieved an efficiency that processes 1 WET file in less than one 1 minute and 100 WET files in less than 19 minutes.

## PUBLICATION

**Design of Obstacle Avoidance System for the Blind based on Fuzzy Control** (Yang Su, **Wenyan Liu**, Ri-hua Jiao, Xiao Liu)
NETINFO SECURITY 1671-1122(2014)06-0067-05