Yiqu Xiao

424-428-4523 | yiquxiao@gmail.com | github.com/felicixawe | felicixawe.github.io

Education

University of California, Los Angeles — Los Angeles, CA
Bachelor of Science in Mathematics of Computation — *Sept. 2021 – Dec. 2023*

Technical Skills

Languages: JavaScript, Python, Java, Yaml, Json, SQL, HTML, CSS
Tools and Frameworks: MERN(MongoDB, Express, React, Node), Django, SpringBoot, MySQL, DynamoDB, Maven, Npm, Docker, AWS, Postman, JUnit, JEST, PyTest, Jenkins, Github Actions

Internship Experience

Software Engineer Intern — Apr. 2021 – Jun. 2021
Bell AI Inc. — *Shenzhen, China*

- Designed and created the frontend advanced websites(layouts, navigation, animation button and icons) with **Bootstrap**, font-awesome using **HTML5**, **CSS3** and **JavaScript**. Connected the frontend with RESTful APIs through **React Hooks** (state hooks and effect hooks). Created frontend tests using **Jasmine** framework to validate behaviors.
- Participated in the backend development using **Node.js**, and **MongoDB**, implemented RESTful and GraphQL endpoints.
- Conducted manual API testing through **Postman** and automated basic integration tests using **JEST** testing framework.
- Built the application using **Yarn** and created CI/CD testing and deployment workflows in **Jenkins**.

Projects

MERN Based Shopping Platform — May 2022 – Jun. 2022

- Designed and developed a shopping platform based on **MERN**(MongoDB, Express, React, Node.js) and **AWS**.
- Implemented **MongoDB** based CRUD operation using **Node.js** and connected the frontend with backend APIs through **React** (UseEffect and UseEvent).
- Enabled session based user authentication and authorization using **JWT** and **Cookie**.
- Created docker-compose files to build the micro-services and deployed the services through **Docker**.
- Conducted manual API testing using **Postman** and wrote unit tests for isolated services and integration and end-to-end tests based on **JEST** testing framework.
- Deployed the service on **AWS** using **Docker** and automated the build and deployment pipeline through **Github Actions**.
- Setup monitoring system for the application based on Google open source **cloud-prober** and created a dashboard to visualize the status in **Grafana**.

Reviewing and Voting System — Sept. 2021 – Dec. 2021

- Designed and developed a reviewing and voting system based on **Python Django**, **MySQL** and **AWS**.
- Persisted data in **MySQL** and built basic CRUD operations using **django-mysql**.
- Designed and implemented **RESTful** APIs to support review and vote features using Django.
- Enabled authentication and authorization based on permission and token using Django Security **Middleware**.
- Created cache system for better performance based on Django Cache **Middleware**.
- Deployed the system on an **AWS EC2** instance through AWS CLI, conducted API tests using **Postman** to validate the API correctness and created automated test cases using Python **PyTest**.

Distributed TinyURL Service — May. 2021 – Jul. 2021

- Designed and developed a TinyUrl service based on Python **Django** and **PostgreSQL** which generated short urls and persisted the data upon the retention policy.
- Deployed the service on 4 **AWS EC2** instances using AWS **AutoScaling** Group and used a Zookeeper service to register and manage IDs for the distributed suffix generator. Persisted the url paris in pgsql and created a cache layer using **Redis**.

Student Registration Application — Jan. 2022 - Mar. 2022

- Designed and developed a Student Registration application using Java **SpringBoot**, **React** and **DynamoDB**.
- Used SpringBoot **H2** in-memory database to cache the frequently used student info based on **LRU**.
- Provided both session-based authentication and authorization via **JWT** and **Spring Security**.
- Built the frontend based on Ant design library and **JSX** and connected the frontend with APIs through **React Hooks**.
- Built the application using **Maven** and deployed on an **AWS EC2** instance, created a **DynamoDB** table through AWS console and wired up them using an **IAM** account.