Zackary Santana

Miami, FL | zsant014@fiu.edu | linkedin.com/in/zackary-santana | zackaryjamessantana.com | attach.fly.dev

Education

Florida International University *Graduation Date: August 2023*

- *Bachelor of Science in Computer Science* *GPA: 3.6*
- *Advance UNIX Programming, Algorithm Techniques, Artificial Intelligence, Data Mining, Data Structures, Database Administration, Database Management, Computational Geometry, Human-Computer Interaction, Mobile App Dev, Programming Languages, Operating Systems, Software Engineering*

Work Experience

Junior Software Engineer *January 2023 – Present*

Addigy

- Developed among a fleet of **microservices** that all interconnect and are responsible for a different aspect of the product. This meant a business outcome often affected many microservices, 3-4 on average.
- Developed on 5+ **microservices** that expose CRUD endpoints and store data via a **MongoDB** and **SQLite** databases, deployed in a **Kubernetes** cluster using **Lens** to monitor the cluster.
- Created a new **webhooks** callout consumer service and webhooks manager service in **Golang** to sync 200 different accounts with around 3000+ users using **3rd party integrations**.

Software Engineer Intern *June 2022 – August 2022*

MongoDB

- Direct experience working on distributed systems internal facing **CI/CD** testing tooling, handling 1-2 engineering requests every week from other engineers at MongoDB, dog-feeding and pushing those updates live to increase other team velocities.
  - This ranged from changing the **CLI** to accept new flags, represent new database models, or fix commands under certain conditions like overriding defaults, to creating/adjusting **REST** endpoints to better align with the docs.
- Concurrently worked on uncoupling archiving and restarting logic for tests to allow for specific grouping of tests to restart only failed- teams often used these groupings with **800+ tests** that would run on average for **2 hours**. Restarting them before would restart all tests, with my changes it would cut down the restarted tests from 800+ → 1-5, this reduced restart time/machine usage by up to **90%**.
- Used tools like **Splunk** to query and test about the data of restarted tests, creating new pipelines in **MongoDB** to support the database model changes, **Golang** as the primary language for the tooling, **TypeScript/React.js** for UI changes.

Software Engineer Intern *June 2021 – August 2021*

MetLife

- Improved the UX (user experience) of the department-wide smoke test website used by **200+ teams**. The site previously had blocked executions of similar requests in multiple sections of the site and would not cache responses even on the same client session. Implemented coinciding execution of requests bringing the site from a **10 seconds-30 seconds** on a cold start **down to 1 second**.
- Migrated queries to **MongoDB** to the backend (**Restifiy**) and pinging endpoints (**JVM's**) to the frontend (**Angular.js**) which reduced server load, improved overall server response time, and resulted in a fast TTFB (Time-to-first-byte).

Skills

Languages: TypeScript, JavaScript, Golang, SQL, Python, Java, Dart, AWK, Bash, HTML, CSS

Web Technology: Solid.js, Solid Start, React.js, Next.js, Svelte 3, Svelte Kit, Vite, Rollup, Webpack, Express.js, Drizzle-ORM, Prisma, Node.js/NPM, Tailwind, SCSS, Flask, Streamlit, Progressive Web Applications (PWA)

General Technology: Git, Docker/Docker Compose, UNIX/Linux/WSL, Kubernetes, Firestore, PlanetScale MongoDB, PostgreSQL, SQLite, Prometheus, BigQuery, Looker Studio, Grafana

Additional Technology: Fresh, Deno, Preact.js, Twind, Remix.js, Cypress, Jest

Productivity: JIRA, Figma, Notion, Microsoft Teams, Trello, Slite, Slack

Projects

CodeConstants (https://codeconstants.com) *October 2022 - January 2023*

Web Application

- Large-scale application hosted via various custom docker images, it includes the main application which is a **React.js** and **Express.js** server that fetches from 3 different microservices.
  - **Golang** microservice: Provides an API that receives data from the frontend, authenticating requests with the TypeScript microservice, and "promoting" certain results to the Python microservice.
  - **Python** microservice: Responsible for a stable API that shows posts/info that are personally verified by an admin of the site, or automatically scrapped using a cron job and stored.
  - **TypeScript** microservice: Handles general utility cases for all microservices like authentication and emailing verification.