Matthew Ngo

714-362-1914

matthewngo98@gmail.com

https://matt-ngo.com/

Work Experience

Software Engineer Intern // ICU Medical // January 2021 – Present

- Full stack web application development with PHP, Laravel, and MySQL

Education

2019 - 2021 // University Of California Santa Cruz // (3.8 GPA)

- Degree: Bachelor of Science (BS)
- Major: Computer Science
- Relevant Courses:
  - Web Applications, Software Engineering, Computer Systems Design, Distributed Systems, Computer Architecture

2016 - 2019 // Orange Coast College // (3.9 GPA)

- Degree: Associate in Science (AS)
- Majors: Computer Programming, Mathematics

Skills

- Full-Stack Web Development
  - Front End: Proficient with React JS, HTML, CSS, JS, jQuery, Bootstrap, Material UI
  - Back End: Node JS, Express, OpenAPI, PHP, Laravel
  - Databases: MongoDB, PostgreSQL, MySQL
  - Testing Frameworks: Jest, Puppeteer, SuperTest
  -
- Other Programming Languages
  - C++, C, JavaScript, Python, Java
- Technologies
  - Unix, Git/GitHub/Gitlab, VirtualBox, Docker
- Methodologies
  - Agile, SCRUM

Projects

- Full-Stack Email Browser
  - Single page, Responsive web app with RESTful route validation and multi-user authentication supported using JSON Web Tokens
  - Developed with REACT, Material UI, Axios, OpenAPI, Node.js, Express, PostgreSQL
- Multithreaded HTTP Server & Load Balancer
  - HTTP Server responds to GET, PUT, and HEAD requests, with additional logging and health-checking features implemented. In testing, multithreading yielded avg 2.5x speedup compared to original single threaded server.
  - Load balancer designed to run in conjunction with multiple instances of HTTP Server to handle accepting concurrent client requests and distribute them over the set of servers. Running the load balancer with just 1 extra server resulted in an average 1.5x speedup when running the test suite.
- "CA" Sharded Key-Value Store:
  - Partition-tolerant, available, and causally consistent distributed key-value store built with Python, Flask, and Docker
  - Guaranteed fault tolerance with replication, eventual consistency via gossip, and causal consistency through vector clocks + causal context JSON objects