Nam (Logan) Nguyen

Syracuse, NY | 253-391-7245 | nnguyen6@oswego.edu
namnguyen31.com | linkedin.com/in/logann131 | github.com/logann131

SKILLS

- Blockchain | Ethereum • Solidity • Hardhat • Waffle • Ethers.js • Web3.js • Truffle • OpenZeppelin • Remix
- Front-End | Next.js • React.js • TypeScript • HTML • CSS • Tailwind • Styled Component • WordPress
- Back-End | Golang • Open Liberty • Spring Boot • NodeJS • Express • MongoDB • MySQL • Firebase
- Deployment | AWS • Vercel • Netlify • Heroku • Docker • OnRender

EDUCATION

State University of New York (SUNY), College at Oswego Jan. 2021 – exp. May. 2023
Bachelor of Science in Computer Science **CGPA: 4.00/4.00**

- Courses: Software Engineering, Data Structures and Algorithms, Front-end Web programming, SQL Database

EXPERIENCE

Full-stack Software Engineer Internship, *Jobs4Interns – Domenix* Sep. 2022 – Dec. 2022

- Successfully practiced **Agile** methodologies to support collaborative team efforts, improve project transparency, and facilitate timely project completion
- Collaborated closely with teammates in a **Scrum** team of 6 to create efficient restful APIs utilizing the MERN stack, facilitating seamless data transfer between the front-end and back-end systems
- Experienced in using Atlassian Tool Suite (**JIRA**, **Confluence**, etc.) for project management and collaboration

Web Development Intern, *SUNY Center for Professional Development* Feb. 2022 - May. 2022

- Conducted proactive monitoring of website performance, swiftly addressing any technical issues, and troubleshooting as necessary to ensure optimal functioning of a highly complex network of 30 WordPress websites

PROJECTS

Spark Your Noble Story (SYNS Platform) *(Full stack + Blockchain)*
(Capstone project) – https://github.com/syns-platform - https://syns.vercel.app

- Utilized cutting-edge front-end technologies such as **Next.js**, **React.js**, and **TailwindCSS** to develop SYNS, a full-stack web 3.0 platform that combines a music-donation system, NFT marketplace, club membership, and social media features for musicians and fans to connect and share artworks through NFTs
- Deployed 5 **Solidity** smart contracts on **Polygon**, reducing gas fees by 90% and enabling cost-effective NFT transactions along with club membership and donations functionalities
- Utilized **OpenZeppelin**'s ERC-721 and ERC-1155 standards for secure creation and management of NFTs in the project
- Incorporated **Hardhat** framework to streamline the development of smart contracts, increasing the feedback loop up to 10 times through robust support for tasks such as contract editing, debugging, and deployment management
- Integrated **Ethers.js** for seamless NFT transactions and other smart contract functionalities in the client application
- Implemented 3 restful-based microservices for the backend using **Golang** and **MongoDB**, providing a robust and scalable infrastructure for off-chain data management and API services

Hashtology Decentralized App *(Front-end + blockchain)*
(Side project) - *https://github.com/logann131/hashtology-dapp*

- Designed and developed a **decentralized application** on the **Ethereum** network, facilitating seamless global cryptocurrency transfers and delivering a user-friendly experience for customers
- Implemented a fungible token smart contract using **Solidity**, enabling wallet-to-wallet transfer functionalities
- Employed a blend of **React.js** front-end framework and **TypeScript** programming language to enhance application performance and guarantee type safety

NFTir *(Golang Back-End)*
(Course/Individual Project) – *https://github.com/nftir*

- Developed a high-performing RESTful application server using **Gin-Gonic,** a Golang framework to manage individual NFTs with up to 40 times faster development process and web service performance
- Employed **AWS DynamoDB** to effectively store and manage metadata for individual NFTs obtained from the NFTGo API server, thereby enhancing data retrieval and processing efficiency within the NFTir project
- Leveraged **AWS EC2**, **ECR**, and **ECS** to deploy a containerized solution, significantly increasing scalability and efficiency for the application