Jordan Michael Walsh

UI Engineer and Full-Stack Developer

jordan@jmwalsh.dev 405.479.3668

GitHub | Portfolio | LinkedIn

SUMMARY

Experienced UI Engineer and Full-Stack Developer adept at greenfield development and seamlessly integrating into existing codebases. Known for holistic problem-solving, I prioritize analytical thinking and deep understanding of both application requirements and user experience. Committed to maintaining well-documented, legible codebases, I foster collaboration through clear communication and shared standards. Passionate about delivering innovative solutions and continuously refining my craft.

SKILLS

Programming Languages: JavaScript, Typescript, Python, C#, Go, HTML, CSS,
Frameworks: React, Remix, Next.js, Astro, Svelte, Node.JS, Express, .NET, Flask
Database and Query Languages: Postgres, SQL, MSSQL, MySQL, Sqlite, Redis, GraphQL, MongoDB
Other skills and tools: Linux, Azure, Bash, Docker, Tailwind, Emotion, Styled Components, WebPack, Vite, PowerBI

EXPERIENCE

Lead UI Engineer/Full Stack Developer
Task Automation Partners | October 2021 - Present,
• Responsible for developing and optimizing enterprise applications for multiple clients such as Mitsubishi Electric and Insight Lean Solutions..
• Spearheaded the development of frontend architecture for Mitsubishi Electric's warranty processing infrastructure and internal tooling with React 18, TypeScript, and Vite--extending DTOs from the .NET 6 + SQL backend into Typescript with OpenAPI TypeGen.
• Collaborated directly with clients and stakeholders to gather and delineate project requirements, ensuring alignment between technical implementation and business objectives.
• Maintained and organized hundreds of asynchronous API calls and endpoints by providing facades separated by concern or entity, improving typesafety by strongly coupling the return types and accepted params to the types generated from the backend codebase.
• Utilized Azure DevOps and Git for coordination, documentation, and task assignment.
• Implemented navigation and data provisioning with React Router V6 and V7 to provide a strongly defined boundary between component logic and data loading, as well as synchronizing revalidations in complex and nested layouts.
• Relieved pressure on backend engineers and assisted with C#/.NET development, iterating on client requirements and updating controllers and methods as needed.

React Developer
TechPulse | September 2022 - September 2022,
• Application utilized by students and administrators for uploading and viewing Covid test results dispensed from kiosks being deployed in two U.S states.
• Coordinated with multiple teams and led the development of frontend functionality to quickly fulfill Phase 1 requirements in a multi-stage contract, delivering an enterprise application built with React 18.
• Rapidly iterated on UI designs provided by product manager and the design team via Miro by collaborating with and leading developers from multiple time-zones and locales.
• Translated multiple sources of documentation provided by other teams (Swagger, text documents, Miro, email correspondence) and ensured in-IDE access to data shape and endpoint documentation to limit context-switching and reduce confusion for members of the frontend team.
• Documented UI data requirements and worked with backend leadership to formalize data models and relationships to reduce costs associated with query complexity and frequency in an architecture utilizing AWS Lambdas and DynamoDB.
• Codified GIT version control procedures and organized assignments to mitigate conflicts, as well as implementing preview deployments on Vercel prior to deploying to an AWS staging environment prior to launch.

Full-Stack Engineer
DeFi Accounting LLC (Contract) | August 2022 - August 2022,
• Developed the API and database for an accountant working for a leading company in the NFT gaming industry:
• Delivered a custom-built solution for an accounting firm conducting a multi-year audit involving millions of transactions occurring across multiple blockchains and platforms, including Eth, Hive, Tron, BNB, and Wax.

• Built out a robust and typesafe system for initializing and maintaining transaction data, provided endpoints for seeding and updating new database instances, and leveraged Docker-Compose to allow future migration of the Postgres container and data to cloud or DbaaS providers.
• Analyzed the shape of data returned from multiple endpoints and developed schemas and interfaces to reduce complexity for the end user. Utilized Typescript and Node.JS to mutate the original data to conform to interfaces that corresponded with Prisma ORM schemas.
• Ensured type consistency between the Prisma ORM schema and the application layer, with considerations for expanding type safety to the presentation layer with Typescript and TRPC.
• Provided robust documentation with TSDoc syntax to provide in-IDE resources for future developers, and TypeDoc to provide top-notch external documentation regarding data shape and mutations for CPAs as well as engineers.

Web/React Developer
ByteLaunch | Feb 2021 - September 2021,
• Multiple migrations, performance audits, and optimizations of different React applications.
• Leveraged my understanding of various historical React patterns and the value judgements undergirding application structure and logic to quickly decipher design decisions and optimize performance.
• Audited handling of application state and data-fetching mechanisms to reduce excessive rerenders and synchronize component updates.
• Preserved design decisions that did not constitute an obstacle to improving performance or intelligibility, making decisions on whether to eliminate unnecessary complexity in the context of the application. This involved interfacing with codebases with little up-front context and navigating different design philosophies ranging from Flux-based architecture like Redux + RTK to applications that minimized boilerplate and reduced complexity by utilizing native React APIS, web-native APIs, and sessions for application state.
• Demonstrated an ability to interface with unfamiliar codebases written by complete strangers and to separate my own feelings about an architecture from an analysis of how it functions.
• Analyzed and refactored React applications from version 15.6, 16.00, 16.8 to 17+.
• Reduced code duplication and improved readability and performance by refactoring codebases into well-documented and reusable components and functions.

EDUCATION

Software Engineering Immersive
General Assembly • 2021

Philosophy & Critical Theory | Shimer College | Chicago, IL | 08/09 - 11/11
Shimer College • US, IL, Chicago • 2011