HENRY O'NEAL

Frontend Engineer (UX/UI Systems, Dev-Rel/DevEx)

REACT/TYPESCRIPT/NEXT.JS + BLAZOR WASM/.NET/C#

Costa Mesa, CA | henryoneal24@gmail.com | 708-560-4165

GITHUB
@decentranaut

LINKEDIN
@decentranaut

PORTFOLIO
henryoneal.vercel.app

SUMMARY

Frontend Engineer focused on workflow-heavy product UI, specializing in state modeling, deep-linkable navigation, guarded flows, and reliable persistence. Strong collaborator between product/design and engineering teams; turning requirements into explicit UI states, edge cases, acceptance criteria, and clean integration seams. React/Next.js/TypeScript is my primary stack, but I also have experience with Blazor WASM UI (.NET/C#) in a robust monorepo.

SKILLS

Frontend: React, TypeScript, Next.js, JavaScript, HTML, CSS
UI + styling: Tailwind, Bootstrap, responsive/mobile-first UI
Workflow UI: multi-step flows with URL/query-driven state, guarded CTAs, state restoration, and safe redirects/empty states when inputs or plan data are missing
Routing: deep links, URL/query-driven state, refresh/back/forward restoration
API integration (frontend UI): REST + JSON, typed request/response models, error/validation mapping
State + reliability: loading/empty/error states, edge-case handling, browser persistence (light schema versioning when needed)
Forms + data: React Hook Form, basic Zod, React Query (mutations/invalidation)
Enterprise Frontend: Blazor WebAssembly (Razor), scoped CSS, UI-layer C# contributions
QA Delivery: DevRel-facing QA, acceptance criteria, edge cases, regression checks, demo readiness, limited automated testing
Delivery: Git + PR workflow, conventional commits, CI/CD exposure, documentation/spec writing

EXPERIENCE

Frontend Engineer (UI Systems + Technical Delivery)

Lunarspace.io | JAN 2022-present

Stack: **Next.js, React, TypeScript, Tailwind, React Query, Zustand, React Hook Form, Zod (basic), Git/GitHub**

- Built and shipped workflow-heavy product UI in React/TypeScript/Next.js, focusing on predictable states, guardrails, and clean UX transitions.
- Implemented role-gated surfaces (wrapper-based access control) to keep privileged tools scoped to the right account types.
- Shipped operator workflows (e.g., create user) with typed payloads and conditional visibility, keeping high-privilege options constrained.
- Built multi-step onboarding flows with browser persistence, prerequisite checks, and safe redirects when required inputs are missing.
- Implemented consistent async behavior using typed mutations (success/error feedback + invalidation patterns) to keep UI reliable after writes.
- Integrated UI-side wallet/balance checks and typed client calls for account setup flows (UI-side integration; no contract implementation claim).
- Designed workflow UI and smart contract interface behavior for on-chain modules (instantiate/execute/query expectations, admin controls, permission boundaries, failure cases) so smart contracts and gating could be implemented predictably by engineering partners.
- Partnered cross-functionally on requirements, edge cases, and acceptance criteria to deliver QA-ready increments.

HENRY O'NEAL

Frontend Engineer (UX/UI Systems, Dev-Rel/DevEx)

REACT/TYPESCRIPT/NEXT.JS + BLAZOR WASM/.NET/C#

Costa Mesa, CA | henryoneal24@gmail.com | 708-560-4165

GITHUB
@decentranaut
LINKEDIN
@decentranaut
PORTFOLIO
henryoneal.vercel.app

EXPERIENCE

Frontend Engineer / Delivery Lead

NDA Contract - "Training Pilot Platform" | AUG 2024-JAN 2026

Stack: **Blazor WebAssembly (.NET/C#), Razor, scoped CSS, Bootstrap**

- Owned the frontend UX/UI layers of a multi-role portal with operational dashboards and guided workflows, built mobile-first with scalable desktop layouts per user role.
- Partnered with the backend / full-stack engineering on edge cases and acceptance criteria, delivering reviewable milestones that stayed demoable.
- Shipped multi-step intake/onboarding flows with conditional UI states and validated inputs, designed to keep transitions predictable and QA-friendly.
- Defined explicit UI behavior for complex screens: loading/empty/error handling, validation messaging, and recovery behavior for missing/partial data.
- Built modular page composition so multiple sections could live on a stable route while remaining deep-linkable and refresh-safe.
- Implemented navigation/state restoration (refresh/back/forward returns users to the correct section/context) to improve continuity and reduce confusion.
- Integrated UI with API-backed workflows; collaborated on payload shapes, error cases, and completion criteria to keep integration predictable.
- Ran client-facing QA cycles: acceptance criteria, edge cases, regression checks, and demo readiness (limited automated testing).

Private Portfolio App — Next.js / TypeScript / Tailwind (Vercel)

Case Study: **Client Dashboard — a practical example of how I build state-reliable, data-dense UI**

- Built a Next.js portfolio app that embeds interactive demos inside a reusable shell (mobile + desktop rails) with consistent framing and navigation.
- Implemented deep-linkable demo flows where URL state drives what the user sees, so screens are shareable and predictable.
- Added versioned, namespaced persistence for workflow runs (schema v1 → v2) so progress can restore safely after refresh or navigation.
- Enforced style isolation so each demo can have its own UI without leaking CSS into the surrounding site.

Read: Client Dashboard case study

Demo: **Next.js, React, TypeScript, Tailwind, GitHub Actions, Vercel**