

STATEMENT OF QUALIFICATIONS

This position describes a challenge I've already faced on my own: **turning flat AI-generated HTML into structured, production-ready CMS pages**. I needed a repeatable system to sit alongside an HTML pipeline that generates 3,000–8,000-line single-page applications (SPAs). I built a four-phase SPA→CMS pipeline that takes a prototype through audit, component extraction, editing-interface generation, and idempotent database seeding. Every build in my portfolio is now eligible for rapid deployment through that pipeline. After a year spent acclimating to LLMs and AI tools, I'm energized to take on new challenges in this future-facing role, **defining new AI-to-CMS workflows**.

I have worked across WordPress Shopify/Liquid Drupal Headless stacks, and my marketing-operations experience spans MailChimp, Zapier, Community Brands (Spiceworks), Eventbrite, GA4, and GTM. I don't have direct HubSpot experience yet, but every role I've held has introduced a new technology — picking up HubSpot and a .NET CMS alongside the Celigo stack is a natural fit as I prefer MVC architecture. I aim to be fluent within the onboarding window.

Operationalizing AI output into a CMS — connecting forms, validating lead routing, fixing snippets, maintaining brand standards across a high-velocity pipeline — is an area where I'll find comfort. I can **design the system that makes that work faster, more reliable, and progressively more automated** — which is where the real value sits for a team defining this workflow from scratch. I aim to provide tremendous value as we keep building with AI.

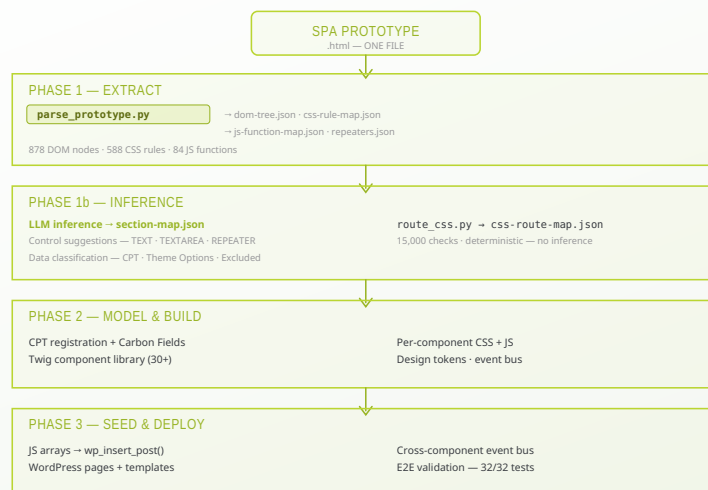


FIG 01 — SPA→CMS PIPELINE · DETERMINISTIC CODE + AI INFERENCE

Portfolio with live builds and technical walkthroughs at erad17.github.io. Supporting documents attached — the SPA→CMS pipeline architecture and its direct application to this role.

Marho Eradiri

Every prototype I build becomes the production site — structured, editable, and CMS-native.

A four-phase extraction pipeline turns flat AI-generated HTML into WordPress themes with structured editing surfaces. The same pipeline that shipped 3 complete themes from 3 prototypes is the pattern I'd bring to your AI-to-CMS workflow.

PIPELINE ARCHITECTURE

PHASE	AI HANDLES (INFERENCE)	CODE HANDLES (DETERMINISTIC)
PHASE 1 Extract	Section boundary detection — is the hero one component or two? Component naming — "Method Ledger," "Testimonial Rotator"	parse_prototype.py reads HTML/CSS/JS → structured JSON DOM tree · CSS rule map · JS function map · repeater patterns 878 DOM nodes, 588 CSS rules, 84 JS functions <1 second
PHASE 1B Inference	LLM pass converts DOM tree → section map CSS routing — which rules belong to which component? Control suggestions — TEXT vs TEXTAREA vs REPEATER Data classification — CPT vs Theme Options vs Excluded	route_css.py computes CSS-rule-to-component mapping Mathematical — 15,000 checks in milliseconds, no inference needed CSS routing config generation is purely deterministic
PHASE 2 Model & Build	Twig template generation — HTML → {% for %} loops Component library structuring	CPT registration — register_post_type() with typed Carbon Fields Twig component library — 30+ components Per-component CSS + JS · Global design tokens · Event bus
PHASE 3 Seed & Deploy	— (all deterministic from here)	Prototype JS arrays → wp_insert_post() calls WordPress pages + templates created Cross-component event bus wired 32/32 E2E tests — deterministic assertions

3 THEMES FROM 3 PROTOTYPES	4 CPT TYPES PER PROJECT (AVG)	30+ TWIG COMPONENTS GENERATED	588 CSS RULES ROUTED (DOLPHIN)	32/32 E2E TESTS PASSING
-------------------------------	----------------------------------	----------------------------------	-----------------------------------	----------------------------

01 Dolphin Academic Prep Tutoring platform. 10 pages, 4 CPTs, booking engine with database-level concurrency. FERPA-minded, AES-256. 37 components routed from 588 CSS rules.	02 Telehealth Psychiatry Practice HIPAA-compliant circadian design system with five time-of-day ambient shifts. Serial essay architecture — 14 installments. 485 CSS rules → 42 components.
03 Personal Portfolio Self-drawing SVG pipeline diagram. WCAG 2.2 AA. 252 CSS rules → 23 components. 5 CPTs. Same extraction pipeline, different output.	04 Additional Builds Elementor + CPT/ACF integrations, Timber v2 + Twig (zero paid-plugin lock-in), headless WordPress explorations. Pipeline-adaptable to any stack.

WHY THIS MATTERS FOR YOUR AI WEB DEVELOPER ROLE

Your team is building the bridge between **AI-generated content and production CMS pages**. My pipeline is that bridge — already built, already proven, already documented. The split between AI inference (semantic reasoning) and deterministic code (parsing, routing, seeding) is **exactly the architecture that makes AI-to-CMS reliable rather than fragile**. I'd bring not just the technical pattern, but the SOPs, admin guides, and architecture decision records that make it repeatable by a team.

BEFORE THE PIPELINE → AFTER THE PIPELINE

× MANUAL PROCESS (3-4 WEEKS)	✓ PIPELINE PROCESS (1 CONVERSATION)
Read prototype, manually identify sections and components	parse_prototype.py extracts DOM tree — 878 nodes in <1 second
Manually register CPTs and configure fields one by one	AI classifies data arrays → CPT + Carbon Fields registration
Manually build Twig templates for each of 30+ components	AI generates 30+ components from section map + DOM tree
Manually extract and scope CSS per component — error-prone	route_css.py routes 588 rules to 37 components deterministically
Manually seed sample data — copy/paste from prototype	Python seed script extracts JS arrays → wp_insert_post() calls
QA: spot-check manually, hope nothing breaks	32/32 E2E tests — deterministic assertions, every deploy
Documentation: write after the fact, if at all	Admin guides, component specs, ADRs — generated alongside the build

DIRECT APPLICATION TO GOENGINEER'S REQUIREMENTS

Turn AI-generated HTML into CMS pages

→ **My pipeline's entire purpose**

3 complete themes shipped from AI-assisted extraction

Review pages for brand & technical standards

→ **Phase 2: structured editing surfaces enforce standards**

Carbon Fields typed controls prevent brand breakage

Develop repeatable AI-to-web workflows

→ **The pipeline IS a repeatable workflow**

4-phase architecture, documented, deterministic at both layers

Integrate forms, validate submissions, lead routing

→ **Booking engine with atomic INSERT IGNORE concurrency**

32/32 E2E tests; MailChimp + Zapier + Eventbrite in production

Troubleshoot front-end, fix bugs, custom code

→ **10 years across WordPress, Shopify, headless stacks**

CAS: 11 WordPress sites, Sucuri WAF, SQL remediation [1]

Create & maintain SOP documentation

→ **Admin guides, component specs, ADRs on every build**

Component spec doc adopted for vendor handoff — still in production

HubSpot forms & workflows

→ **MailChimp + Zapier form automation — same category**

Platform-agnostic integration approach; fluent within onboarding

Shopify / Liquid

→ **Shopify/Liquid in my active skills matrix**

Storefront customization, theme development experience

QA across devices and browsers

→ **32/32 Playwright E2E tests; WCAG 2.2 AA on every build**

Semantic HTML, ARIA, keyboard flows — HCI-trained, not patched-on

THE BOTTOM LINE

You're hiring someone to **operationalize AI output into production CMS pages**. I've already built the system that does exactly that — from parsing to routing to seeding to validating. The pipeline is deterministic where it should be, AI-assisted where it adds value, and fully documented at every step. I can run it, improve it, and teach it. That's the role.