Vibe Feature Request: Static Prerender (SSG) Build Mode for CMS-Backed Content Sites

From: Brett — Blackfeather Digital (Vendasta AI Certified Partner) **Re:** Enabling Vibe as the front-end for SEO-critical, CMS-driven websites **Status:** Feature request / roadmap question

---

The ask in one line

Add a **static prerender (SSG) build mode** to Vibe — the ability to fetch content from an external CMS API *at build time* and emit prerendered, static HTML per route. This is the single capability that would let us **build the design in Vibe, pull content live from a CMS, and ship faster custom websites** without sacrificing SEO.

If that build mode (or an equivalent server-render path) is on the roadmap, it changes our entire web-build strategy in Vibe's favor.

Why this matters (and why we're raising it)

We're an AI Certified Partner reselling the Vendasta stack to our agency clients, and we want to standardize web builds on Vibe. We love the proposition: faster, cleaner, compiled front-ends with built-in analytics and connectors, built without the WordPress plugin overhead.

The blocker is specific and narrow. For one tier of sites — **high-velocity blog / SEO content** — Vibe's current rendering model can't do the job, *not because of the data connection, but because of how the page is rendered.* That tier is a meaningful share of our book of business, and SEO + AI-crawler visibility is the entire reason those pages exist.

We've already validated that the data side works. The gap is rendering. This request is about closing that one gap.

What we confirmed Vibe can already do

In planning mode, Vibe confirmed it **can** connect to a headless content source:

- It can query a headless WordPress REST or WPGraphQL endpoint using native `fetch` or a library like `graphql-request`.
- The practical constraint is **CORS** — the WordPress server must allow requests from the Vibe app domain. (Solvable; standard config.)
- Because there's no backend, requests are read-only against public endpoints or use public/safe tokens. (Fine for published blog content, which is public anyway.)

So the data plumbing is not the problem. We can get content from a CMS into a Vibe app today.

The gap

In the same answer, Vibe described itself as a **"standard client-side React app"** / **"browser-based app"** with **"no Vibe backend server."** That means content is fetched and rendered **in the visitor's browser**, after JavaScript executes.

The consequence for SEO content:

- The **initial HTML response is effectively empty** — a root `<div>` plus a JS bundle. The blog content only appears after JS runs and the CMS fetch completes, client-side.
- Standard search crawlers *can* eventually render JavaScript, but it's slower, less reliable, and deprioritized versus static HTML.
- **AI crawlers** — increasingly central to discoverability — are far worse at executing JS, and many don't execute it at all. They would see a blank page where the blog content should be.

For **apps** (proposal builders, interactive tools) this is the *correct* architecture and a non-issue. For **brochure/landing sites** with thin, rarely-changing content it barely matters. But for a **content library whose purpose is to be found**, client-side rendering undercuts the one job the pages exist to do.

The capability we're requesting

A build mode where content from an external API is **pulled in at build time and baked into static HTML per route**, so each blog post is a complete static page rather than something assembled in the browser. Specifically:

1. **Static Site Generation (SSG):** At build, fetch from an external content API (e.g. headless WordPress REST/WPGraphQL) and write a complete, prerendered static HTML file for each route.

2. **Content in the initial HTML:** The body text, headings, and metadata are present in the raw HTML response *before* JavaScript executes — so any crawler or AI bot reads the full page instantly.
3. **Rebuild on publish (ideal):** A webhook from the CMS triggers a rebuild when content is published or edited, so the static pages stay current without manual redeploys.

A per-request **SSR** option would also solve the crawlability problem, though SSG is the better fit for blog content (content changes rarely relative to how often it's read, so rendering once at build beats rendering on every request).

What success looks like

- `view-source:` on a published blog page shows the **actual post content** in the HTML, not an empty root div.
- Google's URL Inspection / Rich Results tools see a fully populated page.
- A non-JS-executing crawler retrieves the complete article.
- Editors keep authoring in their CMS (e.g. WordPress); Vibe renders the design and bakes the content at build.

Scope note — we are not asking Vibe to stop being a SPA

The client-side SPA model is right for the things Vibe is already great at: interactive apps, tools, and thin brochure sites. We'd use Vibe + Supabase + SSO for app builds without hesitation. This request is for an **additional build mode** aimed at the content/SEO tier — not a replacement of current behavior.

If SSG/prerendering is feasible and on (or addable to) the roadmap, Vibe becomes our default for *every* tier of site, and the "Vibe the design, pull from the CMS, ship faster custom sites" story lands fully. That's the outcome we're after.

Prepared by Blackfeather Digital. Technical observations above are drawn directly from Vibe's own planning-mode responses; we've flagged what's confirmed vs. what we're requesting.