Company Comparison — AI Equity Research

Author: Sankar Kumar Palaniappan **Date:** 2026-07-26 **Status:** Shipped — retrospective PRD, documented post-launch (live at sankar.work/company-comparison) **Stage:** Hybrid of Stage 4 (Solution Review) and Stage 6 (Impact Review) — written after the fact to document real decisions, not to pitch a hypothetical.

---

1. Hypothesis

We believe a small, fully-functional AI product — not a slide, not a static write-up — that a technical reviewer can open, use, and inspect the architecture of in under two minutes, will land harder in a hiring conversation for AI PM / AI engineering roles than a resume bullet or a case-study PDF, because it lets the reviewer verify the claim ("I built X") themselves instead of taking it on faith.

Secondary, product-level hypothesis (the thing that has to be true for the first one to hold): a tool that turns "compare Company A vs Company B" into a structured, cited, 10-section analyst report with an interactive relationship graph — generated live from a user-supplied API key, with zero backend — is genuinely useful enough that it isn't obviously a toy. If the underlying tool doesn't hold up on its own merits, it fails as a portfolio artifact too.

---

2. Problem

Who has this problem: Hiring managers and technical recruiters screening candidates for AI-adjacent PM/engineering roles, who need a fast, high-confidence signal of "can this person actually ship with LLMs" — not just "can they talk about it."

How they solve it today: - Resume bullets and interview storytelling (unverifiable without deep follow-up questions). - Linked GitHub repos with no live demo (requires the reviewer to clone and run code — most won't). - Static case-study write-ups / PDFs (readable, but not inspectable — no way to confirm it actually works, let alone poke at edge cases). - Toy chatbot wrappers ("ask GPT a question in a box") — common enough to read as derivative rather than differentiating.

None of these combine (a) a genuinely non-trivial analytical task, (b) real UI complexity (custom charts, an interactive force-directed graph), (c) a live, clickable URL, and (d) documented architecture/security reasoning a reviewer can read.

Evidence: `[NEED: data]` — no formal survey run. This is a reasoned bet based on how AI-adjacent hiring has visibly shifted toward "show me something you shipped," not a data-backed problem statement. Flagging this honestly rather than inventing numbers.

Secondary (product-level) problem this solves for an actual end user: manually researching a two-company comparison — financials, valuation, competitive moats, SWOT, risks, relationships — by hand across multiple browser tabs and data sources typically takes 30–90 minutes, and most people don't have access to a paid equity-research terminal (Bloomberg, CapIQ, etc.).

---

3. Strategic Fit

Why this shape (bring-your-own-key, zero backend): the single biggest blocker to publicly shipping a working, non-trivial LLM demo is API cost/abuse liability. A backend holding a shared key needs auth, rate limiting, and a budget — none of which is the point of the artifact, and all of which adds real risk (a public demo with a hidden shared key is a magnet for abuse). BYOK removes the problem entirely, and it's disclosed to the user rather than hidden.

Why `gpt-4o-mini-search-preview` specifically: financials and stock prices go stale within a quarter; a plain chat-completions model with no browsing would either hallucinate numbers or refuse to answer confidently. This model's built-in web search plus citation annotations were a deliberate, load-bearing architectural choice — not the default model picked out of convenience.

Alternatives considered and rejected: | Alternative | Why rejected | |—|—| | Backend proxy holding a shared server-side key | Reintroduces cost/abuse liability and would misrepresent "your key never leaves your browser" as a real property of the system | | General-purpose "ask me anything about companies" chat box | Weaker signal — no structured output, no forcing function to build real UI | | `react-markdown` + a charting library (e.g. Recharts) for rendering | Would ship faster, but the hand-rolled markdown parser, SVG chart renderer, and D3 force-graph component are exactly the parts that demonstrate building primitives rather than wiring up dependencies — a deliberate trade-off of more code for more signal |

Competitive context: AI-assisted stock/company comparison tools exist commercially (fintech copilots backed by paid data providers, account systems). This project's differentiation as a portfolio piece isn't "better data" — it's architectural transparency (visible prompt, visible parsing pipeline, no data broker, no account) plus depth (10 structured sections + an interactive relationship graph) in a single static page with no infrastructure behind it.

---

4. Solution

4.1 User flow

1. User lands on the page: hero copy, a 3-field form (Company 1, Company 2, OpenAI API key — masked with a Show/Hide toggle), and a light/dark theme toggle fixed top-right.

2. **Compare** stays disabled until all three fields have trimmed, non-empty content, and again while a request is in flight.
3. On submit, the button shows a spinner and “Researching & analyzing…” (set expectations — this call takes up to ~60s given `search_context_size: "high"`).
4. The app builds a system + user prompt (the 10-section “format contract,” detailed below) and `POST`s directly from the browser to `api.openai.com/v1/chat/completions` using the user’s key in the `Authorization` header. The exact request body and raw response are logged to the console for transparency and debugging.
5. **Success:** the response text is parsed by a hand-written block parser (headings, paragraphs, lists, tables, fenced `chart` / `graph` JSON) and rendered — tables as real `<table>` elements, `chart` blocks as inline SVG bar/line charts, the `graph` block as an interactive D3 force-directed relationship graph, and any citation annotations as a linked source list (with an explicit disclaimer if none came back).
6. **Failure:** a plain-language error banner (OpenAI’s own error message, passed through) replaces the report area. Nothing else on the page breaks.
7. Theme preference toggles instantly and persists via `localStorage`; it’s read and applied *before* React mounts (inline script in `index.html`) so there’s no flash of the wrong theme on reload.
8. The app is stateless across sessions by design — no history, no accounts, no saved reports. Refreshing discards the current report.

4.2 The 10-section report

1. Company Snapshot — business model, products/segments, revenue mix, geography, headcount, leadership
2. Financial Comparison — side-by-side table + multi-year trend chart
3. Stock & Valuation — price, market cap, multiples, 1yr/5yr performance, analyst targets
4. Competitive Edge / Moats — scored 1–5 across brand, network effects, switching costs, cost advantage, IP/regulatory, distribution
5. SWOT + Head-to-Head
6. Market Position — share, TAM/SAM, growth rate, key customers/partners
7. Risks — company-specific, competitive, regulatory, macro
8. Future Trends & Outlook — 3–5 year growth thesis
9. Interactive Relationship Web Graph — force-directed graph of both companies’ executives, products, suppliers, partners, competitors, investors, and markets, with anything touching *both* companies visually flagged (amber ring/edge)
10. Bottom Line — investment view + competitive view, explicitly framed as analysis, not financial advice

The model is instructed via a strict “format contract” — numbered `##` headings, GitHub-flavored markdown tables, and fenced ```` ```chart ```` / ```` ```graph ```` JSON blocks in an exact schema — so the frontend can parse the response reliably instead of guessing at free-form

prose.

4.3 AI behavior spec (input → expected output)

#	Input	Expected behavior
1	“Apple”, “Microsoft”	Full 10-section report; rich graph with real executives/suppliers/investors; shared nodes (e.g. TSMC, index funds) visually flagged; citations present
2	“Coca-Cola”, “PepsiCo”	Full report; moat matrix scored 1–5; each company appears as a `competitor` node on the other’s graph, marked shared
3	“netflix”, “disney” (lowercase input)	Model resolves proper casing in the report; app does no casing normalization itself
4	“Apple”, “Apple” (same company twice)	**Known gap** — not guarded against client-side; relies entirely on the model noticing and narrating that there’s no real head-to-head
5	“Acme Rockets Inc” (fictional), “Boeing”	Model should flag unverifiable/nonexistent data per its “flag anything you couldn’t verify” instruction rather than fabricate financials
6	“Epic Games”, “Valve” (both private)	Model should note privately-held status and limited disclosed financials rather than inventing precise numbers; sparse citations expected
7	“Ford”, “Spotify” (unrelated industries)	Graph should surface the nearest connecting path (e.g. a shared institutional investor) or explicitly state there’s no direct relationship, per the prompt’s instruction — not fabricate a false link
8	“　　” (Tencent), “Sony” (non-Latin script)	Model handles multilingual input; app does no language filtering

#	Input	Expected behavior
9	Empty/whitespace-only field	**Compare** stays disabled client-side (trimmed-length check); no API call made
10	Valid inputs, revoked API key	OpenAI 401 → `data.error.message` passed straight through to the error banner
11	Valid inputs, quota exceeded	OpenAI 429 → same passthrough pattern
12	Network drops mid-request	`fetch` throws → caught in `handleSubmit`, generic "Something went wrong…" shown
13	Well-formed prose, truncated/malformed `chart` JSON	`JSON.parse` fails inside a `try/catch`; that block is silently skipped (`console.warn` only) — rest of the report still renders
14	Malformed `graph` JSON	Same graceful-degradation path as #13
15	Graph includes a node with no edge to either company hub	Not validated against the prompt's own rule; renders as a visually orphaned node — **known gap**
16	Model omits the `graph` block entirely	Parser simply produces no `graph` block; page doesn't break, section just has no visual under its heading
17	A full paragraph pasted into a "company name" field by mistake	No client-side length cap today; sent as-is, likely fails silently as an unrecognizable company — **known gap**
18	"Google", "Alphabet" (parent/subsidiary pair)	Prompt has no explicit parent/subsidiary edge type — model may model them as two independent hubs rather than one owning the other — **known gap**
19	Search-preview model returns a rate-limit error specific to that model tier	Surfaces as whatever error OpenAI returns; not special-cased

#	Input	Expected behavior
20	User toggles dark mode while a request is loading	No interruption — theme is pure CSS-variable state, independent of app/network state
21	User double-clicks **Compare** rapidly	Second click is a no-op — `canSubmit` includes `!isLoading`, blocking duplicate in-flight requests
22	Model returns duplicate citation URLs	`extractSources()` de-dupes by URL via a `Set` before rendering
23	A very long report pushes close to the 5,500-token ceiling	The `graph` block is requested near the *end* of the output, right before Bottom Line — real truncation risk on verbose reports (see Open Questions)
24	Graph search box: term matches no node	`handleSearchSubmit` no-ops and returns early; no "no results" affordance shown — **known gap**
25	User hides every node type in the graph legend	All nodes fade to 0 opacity; graph appears blank but doesn't crash; no "nothing to show" state — **known gap** (low severity, requires a deliberate all-off click)

Rejection criteria: the app never blocks a request based on the *content* of the company names (no allow-list, no validation) — rejection is delegated entirely to the model, which is instructed to flag rather than fabricate when it can't verify something.

Eval criteria (informal, not automated today): does the response parse into all 10 sections without breaking the renderer; does at least one `chart` block render; does the `graph` block render with both company hubs present and at least one shared node/edge (or an explicit "no relationship" statement); do citations appear for claims with hard numbers. No automated eval harness exists — this is currently manual spot-checking, called out honestly rather than implied to be tested infrastructure.

---

5. Non-Goals

- **Not** building user accounts, auth, or saved report history — the BYOK model deliberately needs no backend and no persistent user data at all.
- **Not** proxying or storing the OpenAI key server-side — this is an architectural non-goal, not a missing feature (see Strategic Fit for why it was explicitly rejected).

- **Not** supporting model choice (Claude, Gemini, other OpenAI models) in this version — locked to `gpt-4o-mini-search-preview` specifically for its web-search + citation capability.
- **Not** validating that input strings are real companies before calling the API — delegated entirely to the model's own verification instruction; no autocomplete or company lookup.
- **Not** adding usage analytics or telemetry in this version — ties directly to the "no third-party trackers" claim in the README. This is why every metric in Section 6 below is currently unmeasured.
- **Not** rate-limiting or cost-capping the user's own OpenAI usage — their account, their responsibility, stated explicitly in the UI copy.
- **Not** building a mobile app or native wrapper — a single responsive web page only.
- **Not** supporting PDF export or shareable report links — a report exists only for the current session; refreshing discards it.

Considered and cut: - Server-side proxy with a shared demo key — rejected for cost/abuse liability. - General open-ended chat interface — rejected for weaker portfolio signal. - Off-the-shelf markdown/charting libraries — rejected in favor of hand-rolled rendering, explicitly trading more code for more demonstrated capability.

---

6. Success Metrics

Every metric below requires adding lightweight, privacy-respecting instrumentation — which is currently out of scope per Non-Goals. Proposed here as targets to hit *if and when* that trade-off is made, not as things being tracked today.

Metric	Type	Baseline	Target	Status
Report completion rate (Compare click → rendered report, no error)	Primary	`[NEED: instrumentation]`	`[NEED: baseline before setting a number]`	Not instrumented
Time-to-first-report (click → rendered report)	Secondary	`[NEED: instrumentation]`	< 45s median	Not instrumented
Graph interaction rate (% of completed reports where the user drags/clicks/searches the graph at least once)	Secondary	`[NEED: instrumentation]`	`[NEED: baseline]`	Not instrumented — this is the metric that actually tells us if the flagship interactive feature is used or ignored

Metric	Type	Baseline	Target	Status
Error rate (bad key, quota, network, parse failure)	Guardrail	`[NEED: instrumentation]`	< 15% of submitted comparisons `[NEED: validate this target against real data once available]`	Not instrumented
Portfolio conversion (resume/portfolio link → live app open; qualitative interview feedback referencing the project)	Primary (given this app's stated real purpose)	`[NEED: process to track — not an app-instrumentation problem]`	`[NEED: target]`	Not tracked in any form

Open call to make, not a decision made here: whether it's worth breaking the "no analytics" non-goal to make the top three rows measurable. Flagged in Open Questions.

---

7. Rollout Plan (retrospective — what actually happened)

- **No staged rollout** — single-developer project at portfolio scale, not consumer scale, so a full release with no A/B test or percentage ramp was the right call.
- **Deployment mechanism:** a GitHub Actions workflow (`.github/workflows/deploy.yml`) builds the Vite app and deploys `dist/` to GitHub Pages on every push to `main`.
- **A real incident, caught and fixed:** the initial deployment served the *raw, unbuilt* repo (GitHub Pages was set to "legacy" branch-root serving) instead of a production build — the browser was requesting `/src/main.jsx` directly, which 404'd, producing a blank page. Root cause: no `base` path configured in `vite.config.js` for the `/company-comparison/` project subpath, and Pages wasn't building anything at all. Fixed by setting `base: '/company-comparison/'`, adding the Actions workflow above, and explicitly switching the repo's Pages build source from `"legacy"` to `"workflow"` via the GitHub API. `node_modules` and a stale `dist/` were also found committed to the repo with no `.gitignore` — cleaned up in the same fix.
- **Rollback plan:** `git revert` + push; Actions redeploys automatically (~30s build). Because Pages serves a static artifact swap rather than a rolling server, rollback is effectively instant once the workflow completes.
- No feature flagging exists or is warranted at this traffic scale.

8. Open Questions

Question	Owner	Status
Should the form block identical company names (A == B) client-side, or is relying on the model's own narrative handling sufficient?	Sankar	`[NEED: decision]`
Should the graph schema get an explicit `parent_of` / `subsidiary_of` edge type for cases like Google/Alphabet?	Sankar	`[NEED: decision]`
The `graph` block is requested near the *end* of a ~5,500-token budget, right before Bottom Line — real truncation risk on long reports. Reorder the format contract, or raise the token ceiling further?	Sankar	`[NEED: decision + real truncation-rate data, which requires instrumentation]`
Is it worth adding minimal, privacy-respecting analytics to make Section 6's metrics measurable, given it's in direct tension with the current "no third-party trackers" security claim?	Sankar	`[NEED: decision]`
Should company-name fields get a sanity length cap (e.g. 100 chars) to avoid degenerate prompts?	Sankar	`[NEED: decision, low priority]`
Should a CSP meta tag be added to the deployed page for defense-in-depth, even though the primary XSS mitigation (never rendering LLM output as HTML) already holds without one?	Sankar	`[NEED: prioritization]` — already flagged as a known gap in the README's security section

---

This PRD documents a shipped, single-developer project. Data points marked `[NEED: ...]` are genuine gaps, not omissions — no analytics exist today, so no usage claims are made that can't be backed up.