Cursor AI Prompt — Build: **Autonomous Sourcing & Backhaul Demo (Investor-grade, multi-page web app)**

Paste the entire block below into Cursor (or a single task in your Cursor project). This is the *complete* spec — UI, pages, mock data, behaviors, interactions, repo & deploy instructions, and acceptance tests. The app must feel like a revolutionary integration of **Generative AI + Agentic AI** while remaining deterministic and demo-friendly (no real ML needed — fake/ scripted intelligence that *acts* smart).

> **Context / Goal (single paragraph)**
> Build a multi-page interactive demo that visualizes a Fully Autonomous Sourcing / Backhaul Agentic system: scout → outreach → sample → QA → cost analysis → negotiation → shortlist → handoff. The demo must show both GenAI (natural language, draft emails, route/milk-run suggestions) and Agentic AI (feasibility scoring, multi-party negotiation, execution). Use the screens and flows in the Figma prototype: https://www.figma.com/design/dZmruum92tEC8Nsb43VmNI/Procurement-Prototype?node-id=0-1&p=f&t=kMrJjxJ7OQsNFFYJ-0 as the visual reference and follow the investor-focused prototype guidance in the attached brief. The problem and solution context come from project docs (empty-mile problem and federated AI freight exchange). Logistics Solution AI-powered Backhaul Freight Man... Logistics prototype instructions

---

High level requirements (must obey)

1. **Multi-page app** — each page is a real route (URL) and all buttons/links are clickable and navigate between pages.
2. **Visual fidelity** — pages should visually match the provided Figma screens (layout, hierarchy, colour/spacing), but exact fonts can be approximated.
3. **Interactive/dynamic behavior** — change inputs and the app responds (mock-AI outputs update, charts animate, negotiation messages change, feasibility scores recalc). All AI behavior is deterministic and scripted for investor demos.
4. **GenAI + Agentic AI feel** — include visible "agent" widgets, chat logs, provenance traces (why agent made a suggestion), and a simulated voice-call demo (playback of a short canned audio snippet) where appropriate.
5. **Clickable flows** — from Landing → Dashboard → Matchmaking → Negotiation → Execution → Analytics → SRM handoff. Ensure 3-5 click path length to reach final PO.

6. **Deployable** — produce a git repo with instructions to run locally and a one-command deploy to Netlify or Vercel.

Tech stack (recommended — make this explicit for Cursor)

- **Next.js + TypeScript** (pages or app router)
- **TailwindCSS** + shadcn/ui components (or simple Tailwind components)
- **Recharts** or Chart.js for graphs
- **React Context / Zustand** for app state (mock AI engine runs against this)
- **Framer Motion** for micro-animations
- **Lucide icons**
- Optional small Node API (pages/api) to serve deterministic endpoints (no external ML)
- Deploy: **Netlify** (or Vercel) with a single build command.

(If you prefer React + Vite, replace Next.js with Vite in the card below.)

Pages & components (explicit list — implement all)

1. **Landing (/)**
    - Hero summarizing the problem → "Revolutionizing Strategic Procurement & Backhaul with GenAI + Agentic AI"
    - CTA buttons: `See Demo` (→ /dashboard), `View Prototype` (Figma), `Investor One-pager` (downloadable PDF)
    - Quick stat cards (Empty-mile $ impact, Typical cycle time reduction) — mock numbers.
2. **Login / Role select (/login)**
    - Role toggle: Load Owner / Fleet Manager / Quality Head / CPO — picking a role changes the workspace layout & permitted actions.

3. **Dashboard (/dashboard)**
    - KPI cards: Empty-mile ratio, Utilization %, CO_2 saved, Avg lead time. (Animated numbers.)
    - Map with pins for suppliers/trucks (mock geo tiles).
    - "Start sourcing" button → /matchmaking
4. **Matchmaking / Scouting (/matchmaking)**
    - Form: Product spec, Qty, Quality, Deadline, Target budget.
    - `Run Scouting` button triggers a simulated "Scouting Agent" progress sequence (3 staged steps: Discover → Validate → Outreach candidates).
    - Results table: suppliers with **match score**, estimated landed cost, contact method. Each supplier row has `Request Sample` and `Contact` actions.
    - A side rail: Agent explanation card (Why agent shortlisted these suppliers — show features used like location, certifications, reliability).
5. **Outreach & Communication (/outreach)**
    - Inbox-like view listing outreach attempts (auto-drafted emails by GenAI, auto-calls by Agent). Show message drafts and a `Play Call` button that plays a canned audio snippet.
    - Provide `Resend`, `Schedule Call`, and `Escalate` actions — all scripted to change supplier status.
6. **Sample Tracking & QA Handoff (/qa)**
    - Sample status timeline per supplier: Requested → Shipped → Received → Testing → Approved/Rejected.
    - When QA approved, enable `Cost Analysis` step.
7. **Cost Analysis / Landed Cost (/analysis)**
    - Dynamic calculator: Inputs (unit cost, distance, fuel factor, duties, packing). When the user edits any input, compute `Total Landed Cost` and a `feasibility score`.
    - Show AI-generated market benchmark (GenAI paragraph) that adapts to values. (E.g., if landed cost > benchmark, the GenAI text suggests negotiation points.)
8. **Negotiation Agent (/negotiation)**

- Chat-style negotiation log between our Agent and supplier Agent. Agent messages should be generated deterministically from the current numbers (e.g., if landed cost > target, negotiation opens with structured proposals).
- Provide `Auto-negotiate` toggle (on/off). If on, simulate multi-round negotiation up to N rounds then escalate to `Human Review`. If successful, show `Execute Contract` CTA. Include visible guardrails (min margin, max detour) quoted in the UI to show governance. Logistics Solution

9. **Execution / Contract & Payment (/execution)**
   - Display a mock e-contract PDF with prefilled terms. `Execute Now` triggers a payment flow (fake escrow), then shows driver trip creation (route, checkpoints). Show driver view link.
10. **Analytics & ROI (/analytics)**
    - Graphs: Savings over time, Empty miles trend, Acceptance rate.
    - A "What changed?" GenAI summary box that writes a 3-line executive summary based on the demo values.
11. **SRM Handoff (/srm)**
    - Export package: supplier profile, sample reports, negotiation transcript, contract history. Button: `Export to SRM` (simulate sending, show success toast).
12. **Admin / Agents Console (/agents)**
    - Show list of active Agents (Scouting, Outreach, Negotiation, Execution) with state machine status. Allow toggling each Agent's aggressiveness (slider) that changes behavior deterministically.

Mock data & deterministic "AI engine"

- Include a `data/mock/*.json` set: 8 suppliers, 6 trucks, 3 warehouses, a ledger of prior contracts. Use realistic names and ₹ values as in prototype guidance. Logistics prototype instructions
- Build a small deterministic rules engine in the frontend or API:
  - **Feasibility score** = weighted sum of (match attributes, distance penalty, equipment match, reliability). Keep weights fixed but editable (admin).

    - **Negotiation outcomes**: define 3 negotiation profiles (cooperative, neutral, stubborn). If `Auto-negotiate` ON, choose deterministic strategy: cooperative → settle in 1 round at 95% ask; neutral → 2 rounds at 90%; stubborn → escalate to human. Show exact calculation in the negotiation log. (This gives agentic behavior without ML.)
- **GenAI-style text**: provide prewritten templates and a small templating function that fills templates based on values. Example templates for outreach email, negotiation rationale, executive summary. These must change text based on input numbers to feel generative (not random).

Visual & UX instructions (copy/paste into Cursor)

- Use **Tailwind** layout tokens from Figma. Keep spacing consistent with prototype.
- Use card components, rounded 2xl corners, soft shadows, and a neutral investor palette with purple/teal accent.
- Add subtle Framer Motion entrance animations for KPI cards and map pins.
- Provide a small audible cue for agent actions (soft chime).
- Make every button & link use `<Link href="...">` (or react-router) so navigation is real.

Repo structure & files to produce

```
bashCopy code/freight-exchange-prototype
  /public/assets (icons, sample audio)
  /src
    /pages (or app) - routes per page above
    /components - KPI, Map, AgentCard, NegotiationChat, SampleTimeline
    /lib
      mockEngine.ts (deterministic AI rules)
      templates.ts (GenAI text)
    /data
      suppliers.json
      trucks.json
      shipments.json
  package.json
  README.md (run + deploy instructions)
  netlify.toml (or vercel.json)
```

Dev tasks for Cursor AI (explicit step list for the agent to complete)

1. Initialize Next.js + TypeScript project, install Tailwind, shadcn/ui, recharts, lucide-react, framer-motion.
2. Create pages and wire routes exactly as listed.
3. Implement deterministic engine in `/lib/mockEngine.ts` with exported functions:
   - `runScouting(req): Supplier[]`
   - `computeLandedCost(supplier, params)`
   - `negotiate(supplier, params, profile)`
   - `generateText(templateId, context)` (templating for GenAI text)
4. Create seed JSON mock data and import into pages.
5. Add agent UI components: AgentCard showing reasoning trace; NegotiationChat that displays each negotiation step and logic; AgentConsole (admin).
6. Add Map view using leaflet or simple static svg; ensure pins are clickable to open supplier detail modal.
7. Add audio snippet for automated call playback and ensure `Play` is deterministic.
8. Add deploy config and README with exact commands:
   - `npm install`
   - `npm run dev`
   - `npm run build && netlify deploy --prod` (or Vercel link)
9. Create a sample "investor walkthrough" script in `/README.md` with step-by-step clicks that produce the 2-minute demo (5 clicks to final PO).
10. Commit to git and output repository URL.

(Implement unit tests for `mockEngine` functions if possible — simple Jest tests.)

Acceptance criteria / demo checklist (Cursor must output pass/fail on each)

- ☐ All pages exist and navigation works (no dead links).
- ☐ Scouting flow: fill form → run → suppliers appear.
- ☐ Outreach: draft emails visible, `Play Call` plays audio.
- ☐ Sample: a supplier moves through the sample status and triggers QA notification.
- ☐ Cost analysis updates immediately when inputs change.
- ☐ Negotiation: auto/negotiate runs deterministic rounds and either results in contract or escalate.
- ☐ Execution: clicking `Execute` shows mock e-contract, and `Create Trip` appears for driver.
- ☐ Analytics page updates numbers consistent with actions.
- ☐ "Agent trace" is visible for at least one agent (scouting or negotiation) explaining reasoning.
- ☐ One-click deploy works and README contains the deploy URL.

UI text & copy style (investor tone)

- Keep copy concise, metric-driven, and executive-friendly. Use short GenAI snippets for emails, negotiation rationales, and executive summaries. Example:
    - Outreach email subject: `Opportunity: Supply 10k units — Quality-certified supplier request`
    - GenAI executive summary: `Agentic synthesis: Scouting returned 6 suppliers; 3 passed QA; estimated landed cost ₹X — negotiation reduced price by Y% saving ₹Z.`

Use the attached docs for tone and language examples. AI brief slide content Logistics value chain analysis

---

Small UX polish items (wow moments)

- Animated "agent thinking" dots with log entries appearing as the agent "reasoned".

- A “playback” of the negotiation timeline (time-lapse).
- A single “wow moment” on landing: show instantaneous “Empty-mile saved” micro animation when negotiation succeeds.

Deliverables (what I expect back from Cursor AI)

1. A GitHub repo (link) with full source and commit history.
2. Live deployed demo URL (Netlify/Vercel).
3. README with run & deploy instructions, and a 2-minute investor demo script (click sequence).
4. A short video (optional) — 60–90s screen capture of the end-to-end flow (if Cursor supports export).

References (use to shape behavior & messaging)

- Problem & federated backhaul solution summary. Logistics Solution
- Screens & stepwise user story for backhaul / agentic negotiation. AI-powered Backhaul Freight Man...
- Prototype build guidance & timelines. Logistics prototype instructions
- Tone and AI evolution text for GenAI vs Agentic phrasing. AI types and comparison

---

Developer notes for Cursor (explicit implementation constraints)

- Everything should be deterministic: avoid calls to external LLMs or live APIs. All “AI” outputs must come from local templates & deterministic functions.
- Make it trivial to replace mock engine with real APIs later: centralize engine behind `/lib/mockEngine.ts`.
- Keep assets and audio small (<1MB total).

- Accessibility: all interactive controls must be keyboard accessible and have ARIA labels.
