Pilot Project: ChatGPT × Linear × Obsidian-Style Knowledge Tool

Compensation: $40/hour

Time Budget: 10-25 hours (you choose how many, paid hourly)

Goal

Build a MVP that feels like a mix of **ChatGPT, Linear, and Obsidian** — collaborative problem solving/tracking for knowledge work.

The goal is not to replicate these tools fully but to show how fast you can ship and polish similar functionality. Simply get as far as you can given the allotted time.

Core Features

1. **Omnibox Input**
    - Single text field where user can type, submit entries, attach files/images, and begin chatting with AI (similar to ChatGPT/Claude/Gemini/Grok omnibox).
2. **Problems and Other Objects**
    - Main object is **Problems** (like Linear's "issues"), each with status, priority, and owner.
    - Each new chat from the omnibox either creates a new problem or attaches to an existing one.
    - Problems can have multiple chats.
    - Problems are named automatically as clearly and concisely as possible (user can edit).
    - Other objects:
        - chats
        - artifacts (files, images, etc.)
        - conjectures (possible solutions to problems, text-based)
        - criticisms (data, evidence, tests, or other refutations of conjectures, text-based).
    - All other objects must link to at least one problem. Criticisms must also link to a conjecture.
3. **Dashboard + Knowledge Graph**
    - **Dashboard:** Manage problems with a Linear-like dashboard and views.
    - **Knowledge Graph:** Visualize problems and linked objects in an Obsidian-style graph.

Stack (suggested, flexible)

- Frontend: Next.js (React + TypeScript)
- Backend: NestJS (TypeScript)
- Database: Postgres
- Deployment: Vercel + Railway/Render

Above is only a suggestion. Use whatever stack lets you work fastest.

Deliverables

1. Working deployed app (link).
2. GitHub repo with code.
3. README with:
    - Setup instructions
    - Design decisions
    - AI tools used and most impactful prompts

Evaluation Criteria

- **Velocity:** Speed of shipping usable features.
- **AI-first workflow:** Effective AI use for speed and polish.
- **Code quality:** Structure, readability, maintainability.
- **Polish:** Deployment, clean UX, minimal bugs.