Tools + Languages

Frontend: React + Javascript/Typescript
Backend: Python / Node. Flask/FastAPI
LLM / Chatbot: OpenAI or Claude or Gemini or Anthropic
Geolocation: Google Maps API
Grocery Store Data: Apparently it's realistic to only choose 1-2 stores, so: Walmart or Target API
Dev Tools: Github, env vars, HTTP client
Other: Parallel API calling for speed, Vercel for deployment

Tasks:

1) Frontend Chat (Create the main page for a Chat input UI)
2) LLM (prompt design + JSON output)
3) Store (walmart/target) API integration (product results + mock data if necessary)
4) Business Logic (ingredient and product matching)

Github Link: https://github.com/TheAsianFish/BuyList

Gpt Generated Overview:

Frontend Chat UI
Build the chat page + results view. Work against a mocked `/plan` response (static JSON). Done when: user can submit a message, see loading/error, and render a shopping plan grouped by store.

LLM → Ingredient JSON
Produce a single function/endpoint that takes user text and returns **strict JSON**: `{ dish, servings?, ingredients:[{name, quantity, unit, notes?}] }`. Done when: schema is consistent + robust on common inputs (cake, pasta, tacos) and failures return a safe fallback.

Store Connector
Implement one "search products" interface with two backends: (a) real API (Walmart/Target/whatever you choose), (b) mock catalog JSON. Output normalized to: `{store, items:[{title, price, url?, availability?, matched_terms}]}`. Done when: same interface works with mock and real.

Matching + Fallback Logic
Pure deterministic module that takes `ingredients[]` + `storeResults[]` and returns a final

plan: `{stores:[{store, purchases:[{ingredient, product, price}]}], missing:[...], totals?...}`. Done when: it minimizes store count and deterministically assigns items using mock data.