Mini Clinical Note & Coding Assistant (Node.js full-stack)

Project: Mini Clinical Note & Coding Assistant (Node.js full-stack)

Problem statement

Build a small web app that turns a short primary-care consultation transcript into:

1. a structured clinical note (SOAP),
2. suggested problem list & ICD-10 codes,
3. a draft billing hint (e.g., likely E/M level or CPT short-list), and
4. an AI-generated compliance banner explaining limitations/safety (“not a medical device,” “for clinician review only”).

No telephony or live diarization is required—assume you’ re given a short transcript (or upload a text file). Keep it lightweight, but production-minded.

Requirements (must-have)

A. Stack & architecture

- **Single repo, Node.js throughout.**
  - Backend: Node.js + Express (or Fastify).
  - Frontend: React (CRA, Vite, or Next.js) with TypeScript preferred.
  - Storage: in-memory or a lightweight DB (SQLite/LowDB) for session history.
- **AI integration:** Call *one* LLM (of your choice) via REST from the backend only. No keys in the browser.
- **Deterministic interfaces:** Define clean API routes, return JSON with explicit schemas (Zod/TypeBox ok).

B. Core features

1. **Transcript input**

- Paste text or upload a `.txt` file (<= 5 KB). -- Can use some consultation examples on the internet or youtube, etc.
- Show a read-only view with basic formatting (speaker labels optional).

2. **Note generation (Documentation)**
   - Produce a **SOAP** note with: Subjective, Objective, Assessment, Plan.
   - Include a **problem list** extracted from the transcript with brief rationales (1–2 lines each).
3. **Coding suggestion (Billing & Coding)**
   - Suggest **up to 3** ICD-10 codes ranked by relevance (code + description).
   - Provide a **billing hint**: either a likely **E/M level** or **up to 3 CPT codes** with one-line justification each.
   - Add **model self-confidence**: low/med/high per suggestion.
4. **Compliance & guardrails**
   - Prepend a **compliance banner**: "Draft only; clinician review required; not a medical device; may be inaccurate."
   - Implement **two simple guardrails**:
     - If transcript contains emergencies ("chest pain," "suicidal," etc.), show a **high-risk flag** and require a user checkbox ("I understand this is not a triage tool") before revealing suggestions.
     - **No definitive medical claims**: convert absolutes ("will cure") to cautious language ("may help") before displaying.
5. **Traceability**
   - Show the **exact prompts** sent to the model (redact secrets).
   - Keep a **decision log** in the UI with the key steps (e.g., "extracted symptoms → mapped to problem list → mapped to ICD-10").
6. **Research notes**
   - Add a small **"Research.md"** in the repo listing the public sources you used to understand ICD-10/CPT/E/M basics (links + 1-line takeaways). Aim for reputable sources.

C. UX expectations

- Clean, minimal, keyboard-friendly UI.
- A single page is fine; use cards/tabs for: Transcript | Note | Codes | Billing | Trace.
- Provide a **copy** button for each section and **export JSON** of the whole result.

Evaluation rubric (100 pts)

- **Problem understanding & scope fit (10)** – Delivers exactly what's asked; reasonable assumptions documented.
- **Frontend craft (20)** – Component design, state management, accessibility, and polish (loading/empty/error states).
- **Backend/API quality (15)** – Clear endpoints, validation, error handling, and separation of concerns.
- **AI engineering (20)** – Prompt design (few-shot or structured), output schemas, guardrails, failure modes, and sensible retries/timeouts.
- **Clinical structure & coding logic (15)** – SOAP clarity, problem extraction quality, reasonable ICD-10/CPT/E/M hints with justifications.
- **Security & compliance thinking (10)** – Secrets handled server-side, basic PII hygiene, disclaimers, and "no medical device" guardrails.
- **Research & internet use (5)** – Thoughtful, cited sources in Research.md; evidence of reading up (not guesswork).
- **DX & delivery (5)** – Clear README, run scripts, and a short demo GIF or Loom link.

Stretch goals (pick 1–2 if time allows; not required)

- **Simple RAG**: Allow the user to toggle "ground coding suggestions on uploaded ICD-10 CSV" and show which entries influenced the answer.
- **Model comparer**: Compare two models (e.g., API A vs. API B) and show diff of SOAP and codes.
- **Basic PHI scrub**: Before sending to the model, replace names/phones with placeholders and show a mapping table in the Trace tab.
- **Unit tests**: 3–5 focused tests, especially around parsing/model output shaping.

Deliverables

- **Repo with:**
  - `README.md` (setup, run, design choices, assumptions, time spent).
  - `Research.md` (links + 1-line insights).
  - `docs/` (optional diagrams: sequence/flow).
  - **Demo**: short GIF or Loom (⩽2 min) walking through transcript → outputs → trace.

- **Working app**: `npm run dev` (or `pnpm/yarn`) to start both server and UI.
- **One sample transcript**: add a short synthetic 1–2 minute primary-care interaction (no real PHI).

Suggested approach (time guide, not mandatory)

1. **Scaffold & data path (1–2h)**: Set up Node/Express, React, env vars, and a simple POST `/analyze`.
2. **Prompting & shaping (1–2h)**: Design a single "analysis" prompt that returns a typed JSON (SOAP, problems, codes, risks). Use a schema validator.
3. **Guardrails (0.5–1h)**: Keyword flagging for emergencies + claim softener.
4. **UI polish (1–2h)**: Tabs/cards, copy/export, loading/errors.
5. **Docs & demo (0.5–1h)**: README, Research.md, 90-second walkthrough.

Acceptance criteria (functional)

- Upload or paste transcript → click **Analyze** → see SOAP, problems, ICD-10, billing hint, banner, and trace.
- Refresh the page and the last result still appears (simple persistence).
- Prompts & model responses visible in Trace, with PII placeholders if you implemented PHI scrub.
- No API keys in client code; errors are human-readable.

Implementation hints

- **Prompting:** Ask the model to output **strict JSON**; validate it. Provide few-shot examples inline to stabilize structure.
- **Coding hints:** You may use simple **string matching + heuristics** (e.g., map "Type 2 diabetes" → E11.9); document your mapping sources in Research.md.
- **Risk words list:** start tiny (e.g., "chest pain", "shortness of breath", "suicidal", "anaphylaxis").
- **Compliance banner:** hard-code plus model-generated expansion (e.g., "This draft may miss context…").

- **Testing:** Add at least one unhappy path (malformed JSON from model → show friendly error and raw text in Trace).

What we’ re assessing (why this works)

- **Front-end strength** via a focused but polished UI.
- **Full-stack fundamentals** (auth not required, but server boundaries, validation, and error handling are).
- **Applied AI** skill—prompting, schema enforcement, guardrails, basic safety/compliance thinking relevant to healthcare.
- **Research & judgement**—how you interpret ICD-10/CPT/E/M at a lightweight level, and how you cite sources succinctly.
- **Delivery**—clear docs, runnable project, thoughtful trade-offs.

Submission

Send a Git repo link or demo the project face to face if proceed to the next round.