AI20Labs – Software Engineer Technical Assignment
Assignment Duration: 1 Week
Role: Full Stack Software Engineer
Location: Remote

🧠 About AI20 Labs

AI20 Labs is a next-generation AI development and strategy studio based in **Austin, Texas**. Founded by four serial entrepreneurs with a history of building multiple startups from inception to over $50 million in revenue, we possess the expertise to develop **Minimum Viable Products** and scale them effectively.

In today's rapidly evolving landscape, artificial intelligence is transforming every facet of business operations. Organizations must adopt AI to enhance revenue streams and optimize cost efficiencies. At AI20 Labs, we specialize in crafting tailored AI strategies, seamlessly integrating AI into existing systems, and rapidly developing AI-powered prototypes to validate innovative ideas.

With a team of highly skilled AI engineers and data scientists, we leverage deep industry knowledge to deliver transformative AI solutions. Our collaborative approach ensures that we partner closely with clients to achieve shared goals.

Embrace the AI revolution with AI20 Labs.

📌 Assignment Overview

As part of the interview process at **AI20Labs**, we invite you to complete a short technical assignment that evaluates your ability to build and integrate a React-based frontend with a Python backend to create an intelligent document Q&A system.

🎯 Goal

Develop a web-based application that allows users to:

1. **Upload documents** (PDF and `.txt` files supported)
2. **Interact with an intelligent agent** that can answer questions based on the uploaded document content
3. **Maintain chat history across sessions**, even after a browser refresh

🧰 Technical Requirements

Frontend

- Framework: **React.js**
- Requirements:
  - Implement an interactive agent interface with the following features:
    - **New Chat**
    - **Chat History**
    - **Clear Chat**
  - Persist chat history using **sessionStorage** or **localStorage**
  - UI/UX should be **intuitive and clean**
  - Use **Type Safety** (TypeScript preferred)
- Deployment:
  - Deployment is **flexible** – AWS S3, Vercel, Netlify, or any platform of your choice

Backend

- Language: **Python**
- Framework: **FastAPI** or **Flask**
- Responsibilities:
  - Handle document uploads and parsing
  - Use **LlamaIndex** for:
    - Ingesting and chunking documents
    - Embedding generation (via OpenAI's `text-embedding-ada-002` or a free alternative)
    - Indexing and querying
  - Store embeddings using:
    - **Pinecone**, or
    - **ChromaDB**
  - Respond to user questions based on document context
  - If a question is **unrelated to the document**, the agent should respond that it **doesn't know the answer**
- Deployment:
  - Use **Docker** to containerize the backend
  - Deploy to **AWS EC2**

⚙️ Bonus (Optional) - Raises the Bar

Important: Completing these tasks will significantly boost your chances of being selected. This list is prioritized.

- Use llamaIndex dappier tool to have the agent access to the web too. (Feel free to modify the system prompt accordingly if you considering to add this feature)
- Store ChatHistory in a database (e.g., MongoDB) to separate each session. Display all sessions in a sidebar navigation.
- Basic user session management
- CI/CD integration for both Frontend and Backend. (e.g., GitHub Actions)

✅ Submission Checklist

Please share:

- A public **GitHub repository** containing:
  - Clean, well-structured code
  - `README.md` with:
    - Setup instructions
    - Demo link or short screen recording of the functionality

---

📐 Final Acceptance Criteria

- Successfully upload and process both PDF and text documents
- Use **LlamaIndex** for document orchestration and Q&A handling
- Generate and store embeddings using **OpenAI** or a free/open-source alternative
- Persist embeddings in **Pinecone** or **ChromaDB**
- Return a clear fallback response when the question is unrelated to the document
- Support agent features including:
  - **New Chat**
  - **Chat History**
  - **Clear Chat**
- Persist chat data across page reloads using sessionStorage or localStorage
- Be deployed with working endpoints and accessible frontend
- Contain TypeScript code on the frontend and Python code on the backend
- Include a clear `README.md` with setup steps and a short demo

Reach out to us at **dev@ai20labs.com** if you have any questions or need clarification.