Interview Questions on AI Therapist Project

1. Vector Database Optimization

I chose ChromaDB for its ease of integration with the LangChain ecosystem and its support for persistent local storage, which is crucial for rapid prototyping and cost-effective deployment in research or small-scale production. Compared to FAISS, which is highly performant but less user-friendly for persistent storage, ChromaDB offers a more developer-friendly API and built-in persistence with minimal setup. I considered Pinecone for its scalability and managed cloud offering, but for this project—where privacy and local control were priorities—ChromaDB was the better fit. The trade-off is that ChromaDB may not scale as well as a managed cloud solution for millions of records, but it is sufficient for the current document set and user base.

2. Embedding Model Selection

I selected `all-MiniLM-L6-v2` primarily for its balance between speed and semantic accuracy. Its 384-dimensional output allows for efficient vector storage and fast similarity search, which is important for latency-sensitive applications like chatbots. Larger models like `all-mpnet-base-v2` offer marginally better accuracy but at a significant cost in inference time and resource consumption. In practice, I found that `all-MiniLM-L6-v2` achieved over 90% retrieval accuracy on test queries, with sub-100ms embedding times, making it ideal for interactive use.

3. Prompt Engineering Challenges

The initial prompt was too generic, leading to responses that sometimes lacked empathy or context relevance. For example, when asked, "I'm feeling hopeless," the early version responded with factual definitions rather than supportive language. I iterated by emphasizing compassion and context in the prompt, and I tested responses using a set of sensitive mental health queries. I also added explicit instructions to avoid medical advice and instead encourage users to seek professional help when appropriate. This resulted in more thoughtful, human-like responses.

4. Chunking Strategy

I chose 500-character chunks with 100-character overlap after experimentation. Smaller chunks led to loss of context, especially for complex mental health topics, while larger chunks sometimes included irrelevant information. The 100-character overlap ensures that important information at chunk boundaries is retained, reducing the risk of context fragmentation. I validated this by running retrieval tests and measuring answer completeness and relevance, finding that this configuration minimized both information loss and retrieval latency.

5. Rate Limit Handling
To handle Groq API rate limits, I would implement a local queueing mechanism with exponential backoff and user notifications for delays. For critical queries, I would cache recent responses and consider a fallback to a smaller, locally hosted model (such as DistilGPT-2) to ensure basic functionality remains available. This hybrid approach ensures continuity of service even if the primary model is temporarily unavailable.

6. Multi-Document Versioning
Currently, new PDFs require manual re-indexing. For production, I would implement file hashing and timestamp checks to detect changes, triggering incremental re-embedding and updating only the affected vectors in ChromaDB. This avoids unnecessary recomputation and ensures the knowledge base is always up to date.

7. Concurrency in Gradio
To support 100+ concurrent users, I would deploy the Gradio app behind a load balancer and use a managed vector database (or ChromaDB with a REST API) to handle concurrent access. I would also use asynchronous request handling and model inference batching where possible. Ensuring thread safety in ChromaDB is critical, so I would use process-level locks or move to a cloud-native vector store for higher scalability.

8. Emergency Response Protocol
For queries indicating self-harm or crisis, I would implement keyword detection and immediately provide crisis hotline information and encourage the user to seek help. I would also log such queries (with user consent) for review and, in a production setting, consider escalation to a human moderator or mental health professional.

9. HIPAA Compliance
To achieve HIPAA compliance, I would encrypt all stored data at rest and in transit, implement strict access controls, and ensure that all user data is anonymized where possible. I would also perform regular audits and ensure that any third-party services (such as Groq) are HIPAA-compliant or do not process PHI.

10. Model Specialization Trade-offs
I used LLaMA 3 for its general conversational ability and strong performance on open-domain queries. While domain-specific models like MentalBERT offer better accuracy on clinical terminology, they often lack the conversational nuance needed for empathetic dialogue. In testing, LLaMA 3 provided more natural, supportive responses, while MentalBERT was more rigid and technical. For future iterations, I would consider fine-tuning LLaMA 3 on mental health dialogue datasets to combine both strengths.

11. Hybrid Retrieval Approach
I considered hybrid retrieval (semantic + keyword/BM25) to improve recall, especially for clinical terms that might not be well-represented in the embedding space. BM25 could help surface relevant passages that semantic search might miss, particularly for rare or technical terms. Integrating both methods would likely increase answer coverage and accuracy.

12. Hallucination Mitigation
To reduce hallucinations, I constrained the model to only answer based on retrieved context, using the RetrievalQA chain. I also monitored responses for unsupported claims and measured hallucination rates by comparing answers to source documents. In the future, I would add answer verification steps, such as fact-checking against the retrieved context.

13. Failure Case Analysis
In one case, the system responded to "What are the side effects of medication X?" with a generic answer not present in the PDFs. To address this, I tightened the prompt to require explicit citation of context and added a fallback message ("I'm sorry, I don't have information on that topic") when the retrieval confidence is low.

14. Architectural Evolution
If rebuilding:

- I would use LlamaIndex for more flexible document parsing and knowledge graph construction.
- Add query routing to direct clinical queries to specialized models.
- Integrate real-time monitoring and analytics for continuous improvement.

15. Cross-Disciplinary Impact
Integrating with Azure Health Bot would provide robust compliance and scalability. Azure Cognitive Services, such as Text Analytics for sentiment detection and Translator for multilingual support, could enhance user experience. I would also leverage Azure's secure data storage and monitoring tools for enterprise-grade deployment.