⍰ I created an AI chatbot that can answer questions about my resume.

⍰ The chatbot runs completely locally and does not use any cloud API.

⍰ I built it using a lightweight language model called TinyLLaMA.

⍰ First, I take my resume in PDF format.

⍰ I extract all the text from the PDF using a PDF reader library.

⍰ After that, I clean the text and prepare it for processing.

⍰ Then I split the resume text into small parts called chunks.

⍰ Each chunk is around 300 characters and is split by sentences.

⍰ Chunking helps the model understand the resume content more clearly.

⍰ After chunking, I convert each chunk into embeddings.

⍰ For embeddings, I use the Sentence Transformer model **all-MiniLM-L6-v2**.

⍰ Embeddings convert text into numbers so similar meanings can be matched.

⍰ These embeddings are stored in a vector database called FAISS.

⍰ FAISS helps in fast similarity search between the user question and resume content.

⍰ When a user asks a question, the system understands the meaning of the question.

⍰ The resume content is used as context so the model does not hallucinate.

⍰ For answering questions, I use the TinyLLaMA 1.1B chat model.

⍰ The model is loaded using llama.cpp and runs on CPU.

⍰ It is a quantized model, so it is fast and memory efficient.

⍰ I use a strict prompt that tells the model to answer only from the resume.

⍰ If the information is not present, the model is instructed not to guess.

⍰ I also keep the temperature low to get accurate answers.

⍰ The chatbot interface is built using Streamlit.

⍰ It looks like a chat application and stores chat history.

⍰ Users can ask questions about skills, projects, education, and experience.

⍰ The system also handles errors properly.

⍰ If no resume is uploaded or the model file is missing, it shows an error message.

⍰ Overall, this project shows how LLMs, embeddings, vector databases, and prompt engineering work together in a real application.

Question:

Q1. What is RAG in your project?

Answer:

RAG means the model first retrieves relevant information from my resume and then generates the answer using only that information, instead of answering from its own memory.

Q2. Why did you use RAG instead of a normal LLM?

Answer:

A normal LLM can hallucinate. RAG ensures the model answers only from resume data, so the responses are accurate and controlled.

Q3. Is your project a pure RAG system?

Answer (BEST ANSWER 🔥):

The retrieval pipeline is implemented using FAISS and embeddings. Since the resume is small, I use full-document grounding for better accuracy, and retrieval-based context is ready for larger documents.

Q4. What embedding model did you use and why?

Answer:

I used all-MiniLM-L6-v2 from Sentence Transformers because it is fast, lightweight, and works well for semantic similarity search.

Q5. What are embeddings?

Answer:

Embeddings convert text into numbers so that the system can compare meaning instead of exact words.

Q6. Which vector database did you use?

Answer:

I used FAISS for storing embeddings and performing similarity search.

Q7. What similarity metric are you using?

Answer:

I am using L2 distance for similarity comparison.

Q8. Why FAISS?

Answer:

FAISS is fast, efficient, open-source, and works well for local and offline applications.

Q9. How do you prevent hallucination?

Answer:

I use RAG, strict prompt instructions, low temperature, and restrict the model to answer only from resume content.

Q10. Why did you choose TinyLLaMA?

Answer:

TinyLLaMA is lightweight, fast, and suitable for local inference with low computational resources.

Q11. Why not GPT or cloud models?

Answer:

I wanted a fully offline and private solution without dependency on APIs or cost.

Q12. What is quantization in your model?

Answer:

Quantization reduces model size and memory usage, which helps in faster inference with minimal accuracy loss.

Q13. How do you handle long documents?

Answer:

For long documents, I retrieve only the top-k most relevant chunks and pass them as context to the LLM.

Q14. How do you choose chunk size?

Answer:

Smaller chunks give better semantic matching. I used around 300 characters to balance accuracy and context.

Q15. What happens if the answer is not in the resume?

Answer:

The model is instructed to clearly say that the information is not present instead of guessing.

Q16. What is temperature in LLMs?

Answer:

Temperature controls randomness. I use a low temperature to get more accurate and deterministic answers.

Q17. How do you evaluate your chatbot?

Answer:

I manually tested it using resume-based questions and checked for accuracy and hallucination.

Q18. Can this system scale?

Answer:

Yes, by using better FAISS indexes and chunk-level retrieval, it can scale to multiple documents.

Q19. What did you learn from this project?

Answer:

I learned how RAG, embeddings, vector databases, and prompt engineering work together in real applications.

Q20. What improvements will you add next?

Answer:

I will enable full chunk-level RAG, add multi-document support, and improve retrieval quality.

LLM aur Rag ke Basice Question:

Q1. What is an LLM?

Answer:

An LLM is a large language model that understands and generates human-like text.

Q2. What does an LLM do?

Answer:

It reads text input and generates a meaningful text output.

Q3. Give an example of an LLM.

Answer:

TinyLLaMA, GPT, LLaMA.

Q4. What is TinyLLaMA?

Answer:

TinyLLaMA is a lightweight open-source language model designed for fast and local use.

Q5. Why use TinyLLaMA?

Answer:

It is fast, runs locally, and does not require high compute.

Q6. What is hallucination in LLMs?

Answer:

Hallucination means the model gives an answer that is not true or not present in the data.

Q7. How do you reduce hallucination?

Answer:

By giving proper context, using RAG, and keeping low temperature.

Q8. What is temperature?

Answer:

Temperature controls how random or accurate the model's answers are.

Q9. What does low temperature mean?

Answer:

Low temperature gives more accurate and predictable answers.

Q10. What is RAG?

Answer:

RAG means the model first retrieves information and then generates the answer.

Q11. Why is RAG used?

Answer:

To give accurate answers and avoid hallucination.

Q12. What does retrieval mean in RAG?

Answer:

Finding relevant information from stored documents.

Q13. What does generation mean in RAG?

Answer:

Creating the final answer using the retrieved information.

Q14. What data do you retrieve in your project?

Answer:

Resume content.

Q15. What is a vector database?

Answer:

A database that stores text as numerical vectors for similarity search.

Q16. Which vector database did you use?

Answer:

FAISS.

Q17. What are embeddings?

Answer:

Embeddings are numerical representations of text meaning.

Q18. Why are embeddings needed?

Answer:

To compare text meaning instead of exact words.

Q19. How does RAG help in resumes?

Answer:

It ensures answers come only from resume data.

Q20. RAG vs normal chatbot?

Answer:

Normal chatbot guesses; RAG chatbot answers from actual data.