RAG + LangChain – Interview Q&A

Q1. How does LangChain help implement a RAG system?
A:

- LangChain provides **document loaders** for ingestion.
- **Embeddings** + **Vector Stores** for indexing & retrieval.
- **Chains** to connect retrieval → LLM.
- **Agents** to dynamically decide when retrieval is needed.
  So LangChain = “workflow engine” for RAG.

Q2. What role does a Vector Database play in RAG and how does LangChain integrate with it?
A:

- Vector DB stores embeddings → enables similarity search.
- RAG needs it for retrieving relevant chunks.
- LangChain integrates with vector DB’s FAISS, Pinecone, Chroma, → wraps them into retrievers.

Q3. How does LangChain Memory complement RAG retrieval?
A:

- **RAG retrieval** → pulls external knowledge from documents.
- **LangChain memory** → keeps **conversational context**.
  Together → answer is grounded in **docs + chat history**.

Q4. How do Agents in LangChain enhance a RAG system?
A:

- Agents decide dynamically:
  - Should I retrieve from DB?
  - Should I just answer with memory?
  - Should I call an external API?
    This makes RAG **adaptive** instead of fixed.

Q5. Compare RAG vs Fine-tuning, and explain how LangChain supports both.
A:

- **RAG** → retrieval at runtime (no retraining).
- **Fine-tuning** → update model weights with domain data.
- LangChain → supports RAG (via retrievers) + can call fine-tuned models (via LLM wrappers).

Q6. How can LangChain Output Parsers help in a RAG pipeline?
A:

- RAG answers may need **structured format** (JSON, dict).
- LangChain's **Output Parsers** enforce consistency → e.g., always return `"answer": "...", "source": "doc.pdf"`.

Q7. How would you design an enterprise chatbot using both RAG and LangChain?
A:

1. Ingest internal documents → embeddings → vector DB.
2. Use LangChain **RetrievalQA Chain**.
3. Add **ConversationBufferMemory** for multi-turn chat.
4. Use **Agents** to handle external API calls (e.g., HR system).
5. Return **answers with citations**.

Q8. What challenges arise when combining LangChain with RAG?
A:

- Retrieval may return irrelevant docs → LLM hallucination.
- Agents may select wrong tools.
- Latency increases (retrieval + LLM).
- Complex debugging (chains of multiple components).

Q9. How do LangChain, RAG, and LlamaIndex fit together?
A:

- **RAG** → the core technique (retrieval + generation).
- **LangChain** → orchestration layer (chains, agents, tools, memory).
- **LlamaIndex** → specialized for document ingestion + indexing for RAG.
  All three together → powerful retrieval-based LLM system.