PROBLEM STATEMENT- ASSIGNMENT 2

1. **Where would you rate yourself on (LLM, Deep Learning, AI, ML). A, B, C [A = can code independently; B = can code under supervision; C = have little or no understanding]**

Ans: I will rate myself on

LLM – B
Deep Learning – C (have little bit understanding theoretically)
AI – B
Machine Learning – B

I have rated myself honestly based on my current hands-on experience. I am very interested in learning new things that I don't know, and I am always open to learning. I consider myself a self-learner.

I believe that with structured guidance, I can perform much better. I am always trying to strengthen my knowledge in AI and ML and remain open to learning and unlearning as part of my growth.

2. **What are the key architectural components to create a chatbot based on LLM? Please explain the approach on a high-level**

Ans:

The Five Core Layers of AI Chatbot Architecture

- **User Interface Layer**: Handles multi-channel input/output (web widgets, WhatsApp, Slack, voice). Must be channel-independent to reuse core logic seamlessly. Voice requires speech-to-text and text-to-speech technology. The important idea is that the same chatbot logic should work across all platforms.
- **Natural Language Understanding (NLU):** Responsible for parsing user inputs via tokenization, intent classification, and entity extraction. Extracts useful details like dates, product names, or locations (entities).

  Example: **For example:**
  "I want to change my delivery to tomorrow afternoon"

  * Recognizing CHANGE as intent

  * DELIVERY as entity,

  * and TOMORROW AFTERNOON as entity value.

- Dialogue Management: The conversational brain that maintains context and decides bot responses. It will keep track of previous messages and also decides what to do next. It will exactly know when to ask the follow up questions and always maintain context relevance throughout the conversation
- **Response Generation**: Response generation means the chatbot's reply.

  There are usually two approaches:

  Template-based responses for important and predictable information, its like a script which it follows accordingly.

  AI-generated responses for natural and flexible conversations

  Good chatbots combine both to stay accurate while still sounding human.

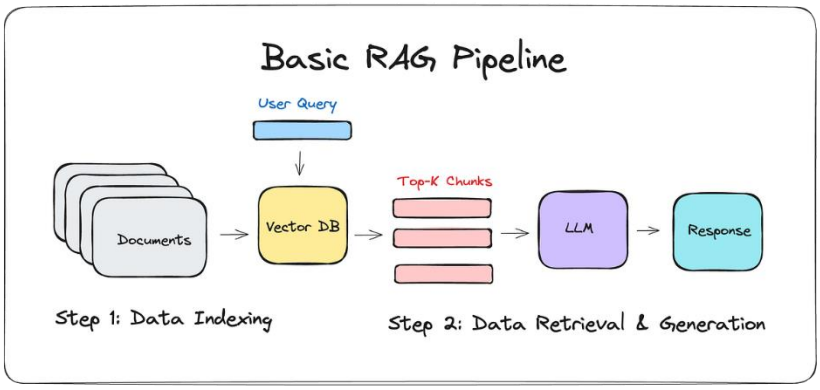
- **Backend and Integration:** Without backend integration, a chatbot can only answer FAQs and cannot perform real actions.Only with backend integration it will connects to External Systems to fetch necessary data.Like Databases, CRM systems,Internal tools etc.The backend manages sessions and context it will stores conversation state, tracks user sessions,maintains context across messages

LIMITATION

One major limitation of AI models is that they only know what they were trained on. They don't automatically know recent updates.Therefore model gives hallucinations.

SOLUTION

Retrieval Augmented Generation (RAG) : Before the AI generates an answer RAG adds an extra step.It will searches relevant documents or databases and then finds the most related information and then adds that information to the AI's input and then generates a response using trusted data.



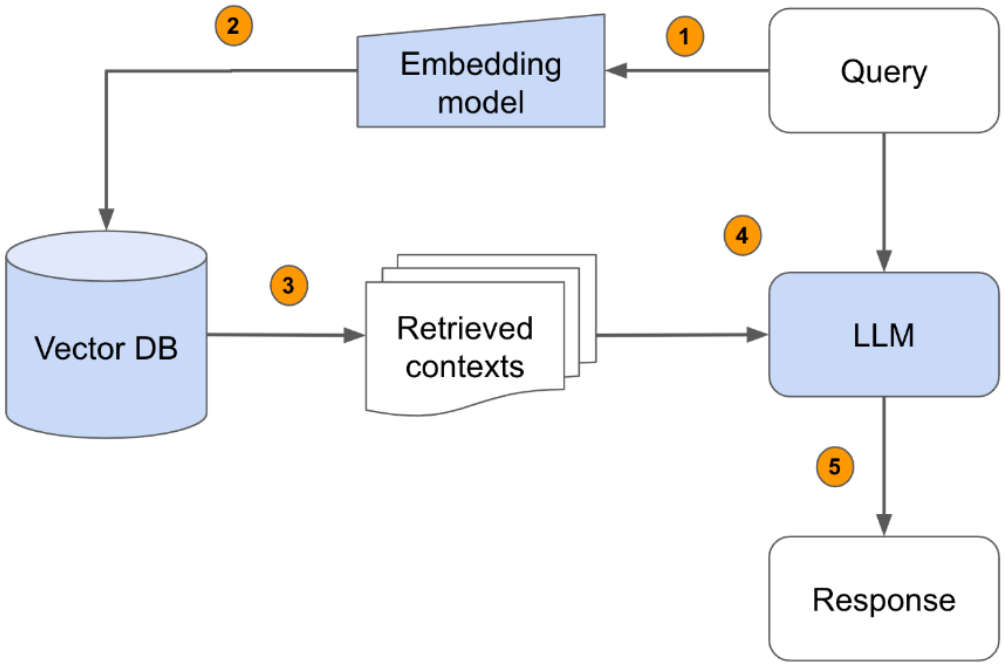
High-Level Approach:
When we talk about high level approach, when a user sends a message in the chatbot, the UI forwards it to the backend. The NLU layer understands the intent and extracts entities. The dialogue manager uses this information along with past context to decide the next action. If required the backend fetches data from databases or external systems. In case any specific area knowledge is needed, RAG retrieves relevant documents and passes them to the LLM(free from hallucinations). Finally, the response generation layer produces a clear reply.

3. Please explain vector databases. If you were to select a vector database for a hypothetical problem (you may define the problem) which one will you choose, and why?

Ans:

A vector database is a type of database designed to index and store vector embeddings for fast retrieval and similarity search. These specialised databases handle and process vectorised data which are arrays of numerical values representing points in a high dimensional space.They mainly store **Vector embedding** (numbers), Metadata and raw text.

Vector databases are used for similarity search, semantic search, multi-modal search, recommendations engines.Like finding similar images, documents, or audio files based on content, themes, sentiment, or style.



There are many vector databases available, including Qdrant, Pinecone, Milvus, Chroma, Weaviate, and others. Each has its own advantages, limitations, and best use cases. Below is a comparison of popular vector databases to help you decide which one fits your needs.

Vector database	When to use	Key features
Pinecone	When you want a fully managed service that's easy to set up for text or semantic search	Managed service, very user-friendly, but can get costly
Milvus	For large-scale projects with images, videos, or massive data, especially if you want to self-host	Open source vector database, and scalable, but requires more setup and maintenance
Chroma	Ideal for small projects or quick local testing	Lightweight and simple to use, but not designed for huge datasets
Weaviate	When you need hybrid search combining vectors and metadata filtering, with cloud or open-source solutions	Supports complex queries but requires schema planning
Faiss	Best for research or custom use cases needing fast similarity search libraries	It's a library, not a full database; you must handle storage and management
Elasticsearch	When you want combined traditional keyword and vector search in one system	Powerful but complex and resource-intensive
Qdrant	For fast, real-time search with filtering needs and easy self-hosting	High performance, strong filtering features, open-source

Hypothetical Problem

For my college project I am building an automated interview platform where the system reads a candidate's resume, understands past projects, and dynamically generates technical interview questions using **RAG**.
The system must retrieve relevant resume sections quickly and accurately to ask personalized follow-up questions during live interviews

Vector Database I woud choose will be FAISS

1. Best Fit for Resume-Based RAG so in my plaform which I am building resumes are text-heavy,relatively small in number,accessed frequently during a single interview.FAISS performs very fast similarity search on dense embeddings
2. Perfect for prototyping : This is especially useful during development and testing of an AI interview system where fast iteration matters. It Runs completely locally, does not require a server.
3. 3.Smooth Integration :FAISS integrates directly with LangChain and Hugging Face embeddings, which makes document chunking easier,embedding storage and retrieval logic better

When I wouldn't use FAISS

If my platform were scaled to:

- Millions of resumes
- Heavy metadata filtering