Lecture 30: RAG Implementation

Introduction:

- Implement **Retrieval Augmented Generation (RAG)** in Spring AI.
- Ensure that when a user asks a question, the response is not generic but based on **our product details document** stored in the vector store.

Steps in Implementation:

- **Create API Endpoint**
  - Define a POST endpoint. /api/ask
  - Accept the query as a **request parameter**.
  - Return the **string response** directly.

```java
@PostMapping("/api/ask")
public String getAnswerUsingRag(@RequestParam String query) {
    return chatClient
            .prompt(query)
            .advisors(new QuestionAnswerAdvisor(vectorStore))
            .call()
            .content();
}
```

- **Use QuestionAnswerAdvisor**
  - Instead of sending a query directly to LLM, add an **advisor**.
  - QuestionAnswerAdvisor is linked with the **vector store**.
  - This ensures that responses are generated using **document embeddings.**

How Does it Work?

- User sends a query (e.g., “Need details about Art kit for kids”).
- Query is combined with **relevant chunks from the vector store**.
- Chat model generates an answer based on **both the query and the retrieved context**.
- Result → **Accurate, document-based response**.

Key Points:

- **Vector Store**: Holds embeddings of product documents.
- **QuestionAnswerAdvisor**: Connects query to vector store.
- **Accurate Retrieval**: Prevents hallucinations and keeps responses relevant.
- **Supports Multiple Formats:**Not limited to text; can also work with **PDFs or external APIs**