# Goal: Answer a Question Using PDF Knowledge

You're using:

- **Pre-indexed PDF chunks** with **FAISS**
- **LLaMA3** (via Ollama) as the answering engine
- **Hugging Face embedding model** for semantic search

---

# Query Time Process: In-Depth Steps

---

## **Step 1: User Asks a Question**

**Input Example:**

```
"What are the objectives of the NDPS Act?"
```

This question will be processed through the following pipeline:

---

## **Step 2: Convert Question into Vector (Embedding)**

sentence-transformers/all-MiniLM-L6-v2

This converts:

```
"What are the objectives of the NDPS Act?"
```

Into a **dense vector** like:
`[0.21, -0.44, ..., 0.12]` (768-dimensional)

This vector **captures the meaning** of the sentence — not just the words.

## **Step 3: Retrieve Top-k Similar Chunks from Vector DB**

Now you query FAISS:

```
retriever = vectorstore.as_retriever(search_kwargs={"k": 5})
```

🛍️ **FAISS** searches for the **5 most similar chunks** based on **cosine similarity** between:

- the question vector
- the stored PDF chunk vectors

 Example Match (Chunk from PDF):

"The NDPS Act aims to control and regulate operations relating to narcotic drugs and psychotropic substances, to prevent their misuse, and to implement international treaties."

This chunk has high **semantic similarity** to the user's question, even if the exact words don't match.

## **Step 4: Send Retrieved Chunks + Question to LLM (LLaMA3)**

Now you form a **prompt** like:

Context:

"The NDPS Act aims to control and regulate operations relating to narcotic drugs and psychotropic substances..."

Question:

"What are the objectives of the NDPS Act?"

Answer:

🔁 This prompt is passed to **Ollama (LLaMA3)** through the RetrievalQA chain:

```
qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)

result = qa_chain.run(question)
```

## **Step 5: LLM Generates a Final Answer**

Using the context, **LLaMA3** generates:

"The NDPS Act primarily aims to regulate and control narcotic drugs and psychotropic substances, prevent their misuse, and fulfill international treaty obligations."

✅ This answer is not **just copied** — it's **abstractive generation**:
LLM understands and rephrases the context into a human-friendly answer.

## **Step 6: Show Answer to User (via Streamlit)**

In your Streamlit app:

```
st.write("Answer:", result)
```

Optional:
Also show the **source chunk** to build user trust:

return_source_documents=True

🔄 Summary of Flow (Visual Representation)

[User Question]

↓

[Hugging Face Embedding (MiniLM)]

↓

[FAISS Vector DB → Top-k Relevant Chunks]

↓

[Prompt + Chunks → LLaMA3 via Ollama]

↓

[LLM-Generated Answer]

↓

[UI Display in Streamlit]

# ❓ Why This Is Powerful

- Combines **factual accuracy** (from documents) + **natural language** generation (LLM)
- Handles **semantic questions**, not just keyword matches
- Can work **offline** (great for sensitive or secure applications)