**Implementation Steps**

**Step 1 : Setting Up LLaMA with Ollama**

1 . **Install Ollama** on your respective OS.

Check out the platform <https://ollama.com/download> to download ollama

Once installed , Make sure its up and running

2. **Install Llama 3.2 Model**

Once you have Ollama installed, you need to download the Llama 3.2 model for your chatbot. Run the following command:

ollama pull llama3.2

Run the command using terminal

This command will download the Llama 3.2 model to your system, making it ready for local use.

### **Step 2: Installing Required Libraries**

Set up virtual environment ( Optional )

Run the below command to install the libraries

**pip install streamlit PyPDF2 langchain sentence-transformers faiss-cpu ollama**

If you have a GPU and want to utilize it, use faiss-gpu instead. ( optional)

**pip install -U langchain-community**

### **Step 3: Writing the Code**

Create a file called app.py and add the below code

import streamlit as st

import os

from PyPDF2 import PdfReader

from langchain.embeddings import HuggingFaceEmbeddings

from langchain.vectorstores import FAISS

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.chains import RetrievalQA

from langchain.llms import Ollama

from langchain.chains.question\_answering import load\_qa\_chain

*# Function to extract text from PDF*

*def* extract\_text\_from\_pdf(*pdf\_path*):

reader = PdfReader(pdf\_path)

text = ""

for page in reader.pages:

text += page.extract\_text()

return text

*# Function to create FAISS vector store*

*def* create\_faiss\_vector\_store(*text*, *path*="faiss\_index"):

splitter = RecursiveCharacterTextSplitter(*chunk\_size*=1000, *chunk\_overlap*=200)

chunks = splitter.split\_text(text)

embeddings = HuggingFaceEmbeddings(*model\_name*="sentence-transformers/all-MiniLM-L6-v2")

vector\_store = FAISS.from\_texts(chunks, *embedding*=embeddings)

vector\_store.save\_local(path)

*# Load FAISS vector store*

*def* load\_faiss\_vector\_store(*path*="faiss\_index"):

embeddings = HuggingFaceEmbeddings(*model\_name*="sentence-transformers/all-MiniLM-L6-v2")

vector\_store = FAISS.load\_local(path, embeddings,

*allow\_dangerous\_deserialization*=True)

return vector\_store

*# Build QA Chain*

*def* build\_qa\_chain(*vector\_store\_path*="faiss\_index"):

vector\_store = load\_faiss\_vector\_store(vector\_store\_path)

retriever = vector\_store.as\_retriever()

*# Load QA chain for combining documents*

llm = Ollama(*model*="llama3.2")

qa\_chain = load\_qa\_chain(llm, *chain\_type*="stuff")

qa\_chain = RetrievalQA(*retriever*=retriever,*combine\_documents\_chain*=qa\_chain)

return qa\_chain

*# Streamlit App*

st.title("RAG Chatbot with FAISS and LLaMA")

st.write("Upload a PDF and ask questions based on its content.")

uploaded\_file = st.file\_uploader("Upload your PDF file", *type*="pdf")

if uploaded\_file is not None:

pdf\_path = *f*"uploaded/{uploaded\_file.name}"

os.makedirs("uploaded", *exist\_ok*=True)

with open(pdf\_path, "wb") as f:

f.write(uploaded\_file.getbuffer())

text = extract\_text\_from\_pdf(pdf\_path)

st.info("Creating FAISS vector store...")

create\_faiss\_vector\_store(text)

st.info("Initializing chatbot...")

qa\_chain = build\_qa\_chain()

st.success("Chatbot is ready!")

if 'qa\_chain' in locals():

question = st.text\_input("Ask a question about the uploaded PDF:")

if question:

st.info("Querying the document...")

answer = qa\_chain.run(question)

st.success(*f*"Answer: {answer}")

Finally run the command

**streamlit run app.py**

This launches the chatbot on localhost:8501.