**Task 1: Chat with PDF Using RAG Pipeline**

**Requirements:**

pip install PyPDF2 langchain openai faiss-cpu

**Code Implementation**:

import os

from PyPDF2 import PdfReader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.embeddings.openai import OpenAIEmbeddings

from langchain.vectorstores import FAISS

from langchain.chains import RetrievalQA

from langchain.llms import OpenAI

# Step 1: Extract Text from PDF

def extract\_text\_from\_pdfs(pdf\_paths):

all\_text = ""

for pdf\_path in pdf\_paths:

pdf = PdfReader(pdf\_path)

for page in pdf.pages:

all\_text += page.extract\_text()

return all\_text

# Step 2: Chunking the Text

def chunk\_text(text, chunk\_size=500, chunk\_overlap=50):

text\_splitter = RecursiveCharacterTextSplitter(

chunk\_size=chunk\_size, chunk\_overlap=chunk\_overlap

)

return text\_splitter.split\_text(text)

# Step 3: Generate Embeddings and Store in Vector DB

def store\_embeddings(chunks):

embeddings = OpenAIEmbeddings(openai\_api\_key="YOUR\_OPENAI\_API\_KEY")

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

return vector\_store

# Step 4: Query Handling

def query\_vector\_store(vector\_store, query):

llm = OpenAI(model="gpt-3.5-turbo", openai\_api\_key="YOUR\_OPENAI\_API\_KEY")

retriever = vector\_store.as\_retriever()

qa\_chain = RetrievalQA.from\_chain\_type(

llm=llm, retriever=retriever, chain\_type="stuff"

)

return qa\_chain.run(query)

# Main Function

def main():

# Input PDFs

pdf\_paths = ["path/to/pdf1.pdf", "path/to/pdf2.pdf"]

raw\_text = extract\_text\_from\_pdfs(pdf\_paths)

# Chunking

chunks = chunk\_text(raw\_text)

print(f"Total Chunks Created: {len(chunks)}")

# Store Embeddings

vector\_store = store\_embeddings(chunks)

print("Embeddings Stored Successfully!")

# Query the system

query = input("Enter your query: ")

response = query\_vector\_store(vector\_store, query)

print("\nResponse:")

print(response)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Task 2: Chat with Website Using RAG Pipeline**

**Requirements:**

pip install requests bs4 langchain openai faiss-cpu

**Code Implementation**:

import requests

from bs4 import BeautifulSoup

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.embeddings.openai import OpenAIEmbeddings

from langchain.vectorstores import FAISS

from langchain.chains import RetrievalQA

from langchain.llms import OpenAI

# Step 1: Scrape Website Content

def scrape\_website(url):

response = requests.get(url)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, "html.parser")

text = soup.get\_text(separator="\n", strip=True)

return text

else:

print(f"Failed to fetch {url}")

return ""

# Step 2: Chunking the Content

def chunk\_content(content, chunk\_size=500, chunk\_overlap=50):

text\_splitter = RecursiveCharacterTextSplitter(

chunk\_size=chunk\_size, chunk\_overlap=chunk\_overlap

)

return text\_splitter.split\_text(content)

# Step 3: Generate Embeddings and Store in Vector DB

def store\_embeddings(chunks):

embeddings = OpenAIEmbeddings(openai\_api\_key="YOUR\_OPENAI\_API\_KEY")

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

return vector\_store

# Step 4: Query Handling

def query\_vector\_store(vector\_store, query):

llm = OpenAI(model="gpt-3.5-turbo", openai\_api\_key="YOUR\_OPENAI\_API\_KEY")

retriever = vector\_store.as\_retriever()

qa\_chain = RetrievalQA.from\_chain\_type(

llm=llm, retriever=retriever, chain\_type="stuff"

)

return qa\_chain.run(query)

# Main Function

def main():

# Input Website URLs

urls = [

"https://www.uchicago.edu/",

"https://www.stanford.edu/",

"https://www.washington.edu/"

]

all\_content = ""

for url in urls:

print(f"Scraping content from {url}...")

content = scrape\_website(url)

all\_content += content

# Chunking

chunks = chunk\_content(all\_content)

print(f"Total Chunks Created: {len(chunks)}")

# Store Embeddings

vector\_store = store\_embeddings(chunks)

print("Embeddings Stored Successfully!")

# Query the system

query = input("Enter your query: ")

response = query\_vector\_store(vector\_store, query)

print("\nResponse:")

print(response)

if \_\_name\_\_ == "\_\_main\_\_":

main()