Local Research Assistant using Langchain + Ollama

CIS 483/583 – Deep Learning

Project 3

1. Introduction:

This project aimed to create a fully local research assistant that can load text documents, break them into manageable parts, retrieve relevant information with vector search, and generate context-aware answers using a Large Language Model (LLaMA 3) running locally through Ollama.

Modern LLMs are powerful, but they have fixed context windows. Retrieval-Augmented Generation (RAG) offers a practical solution by embedding external documents and retrieving only the most relevant sections during queries. This allows the model to operate as if it has a much larger context window.

In this project, I set up a complete RAG pipeline using LangChain components like RecursiveCharacterTextSplitter, HuggingFaceEmbeddings, FAISS for vector storage, and ChatOllama for inference. After loading and indexing a collection of documents about AI, neural networks, and healthcare applications, the system was tested on factual, contextual, and summarization tasks. The result is a self-contained research assistant that runs entirely on a local machine without any cloud dependencies.

2. System Design and Implementation

2.1 Windows Environment Setup

Create and activate a virtual environment

python -m venv venv

Activate the virtual environment

venv/Scripts/activate

2.2 Ollama Installation

Install Ollama if not already installed

Go to https://ollama.com/download

Download the installer

Test to make sure it is running

Go into the VS Code Terminal

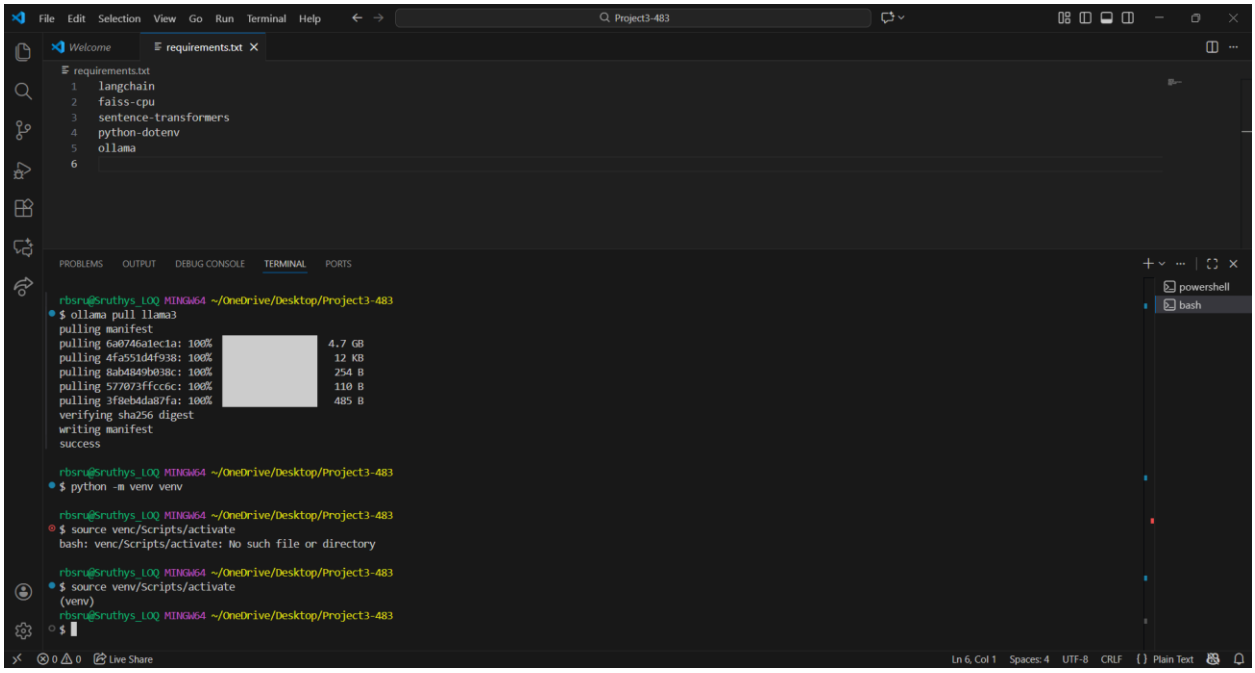
ollama list

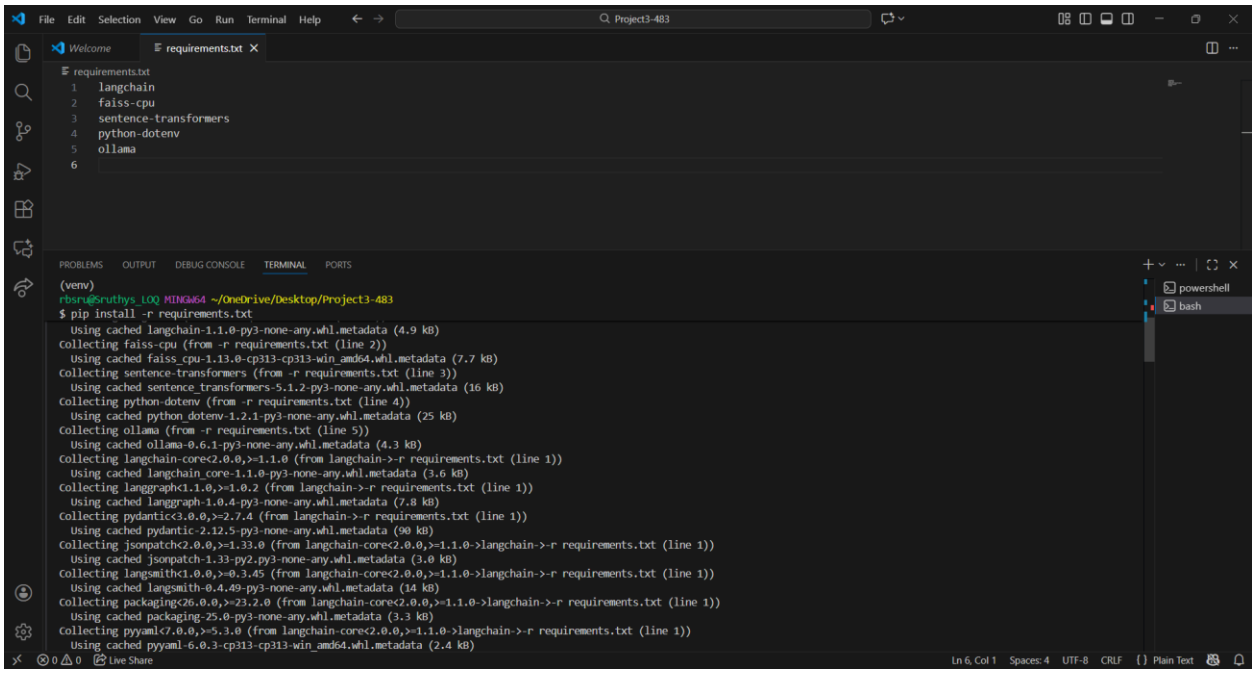
ollama pull llama3

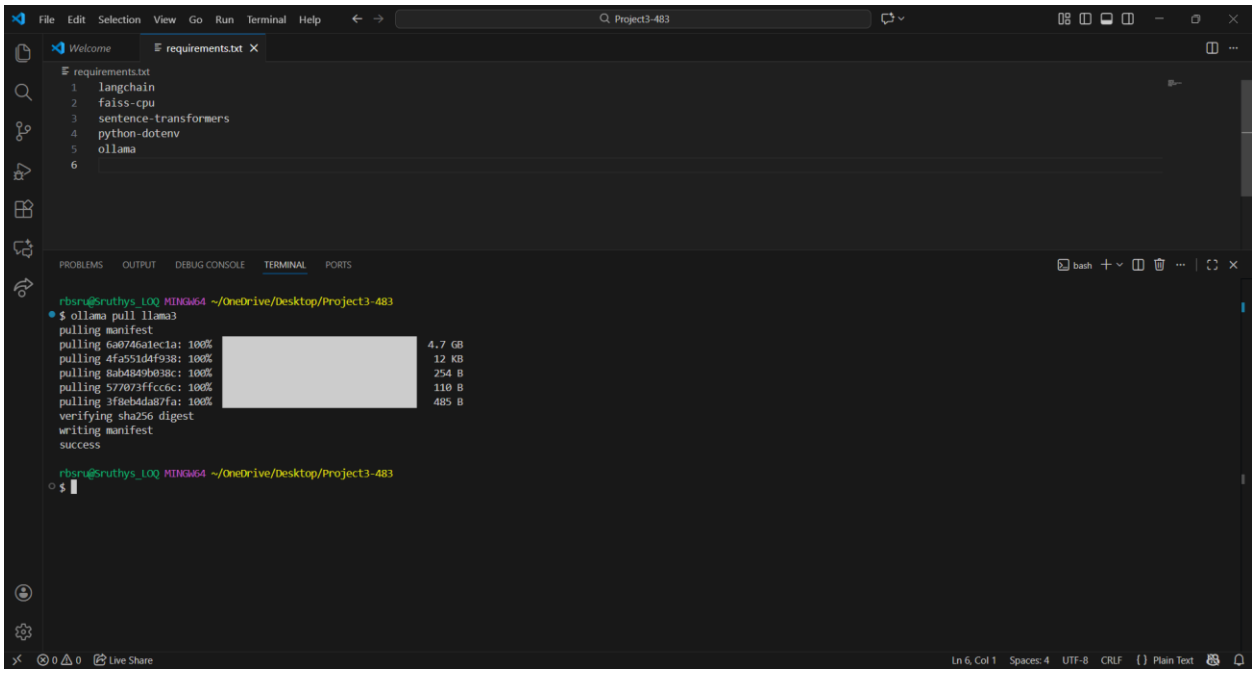
Pull the embedding model: ollama pull nomic-embed-text

2.3 **Dependencies**

Install required packages
pip install langchain langchain-community langchain-core langchain-ollama faiss-cpu pip install langchain langchain-community langchain-ollama langchain-text-splitters faiss-cpu







2.4 Document Loading and Preprocessing

The system loads a folder of .txt files that cover topics like neural networks, transformers, and AI in healthcare. Each file is read and split into overlapping chunks using LangChain's RecursiveCharacterTextSplitter.

Configuration used:
Chunk size: 800 characters
Chunk overlap: 100

This keeps semantic continuity while still producing manageable embeddings.

2.5 Embeddings and FAISS Index

Each chunk is embedded using sentence-transformers/all-MiniLM-L6-v2, a fast and lightweight embedding model. These embeddings are stored in a FAISS index, which allows for quick similarity searches.

If the FAISS index already exists, the program loads it; if not, it rebuilds the index. This makes the pipeline efficient for repeated use.

2.6 Retrieval and RAG Chain

A retriever is created from the FAISS index using as_retriever(search_kwargs={"k": 5}). When a query is made:

1. The user's question goes into the RAG chain.
2. The retriever selects the top five most relevant chunks.
3. A custom prompt is built dynamically:

   "You are a highly intelligent research assistant.
   Use ONLY the retrieved context to answer the question."

4. The prompt is sent to LLaMA 3 running locally via ChatOllama.
5. The answer and sources are returned.

This pipeline ensures that responses are based solely on the loaded documents.

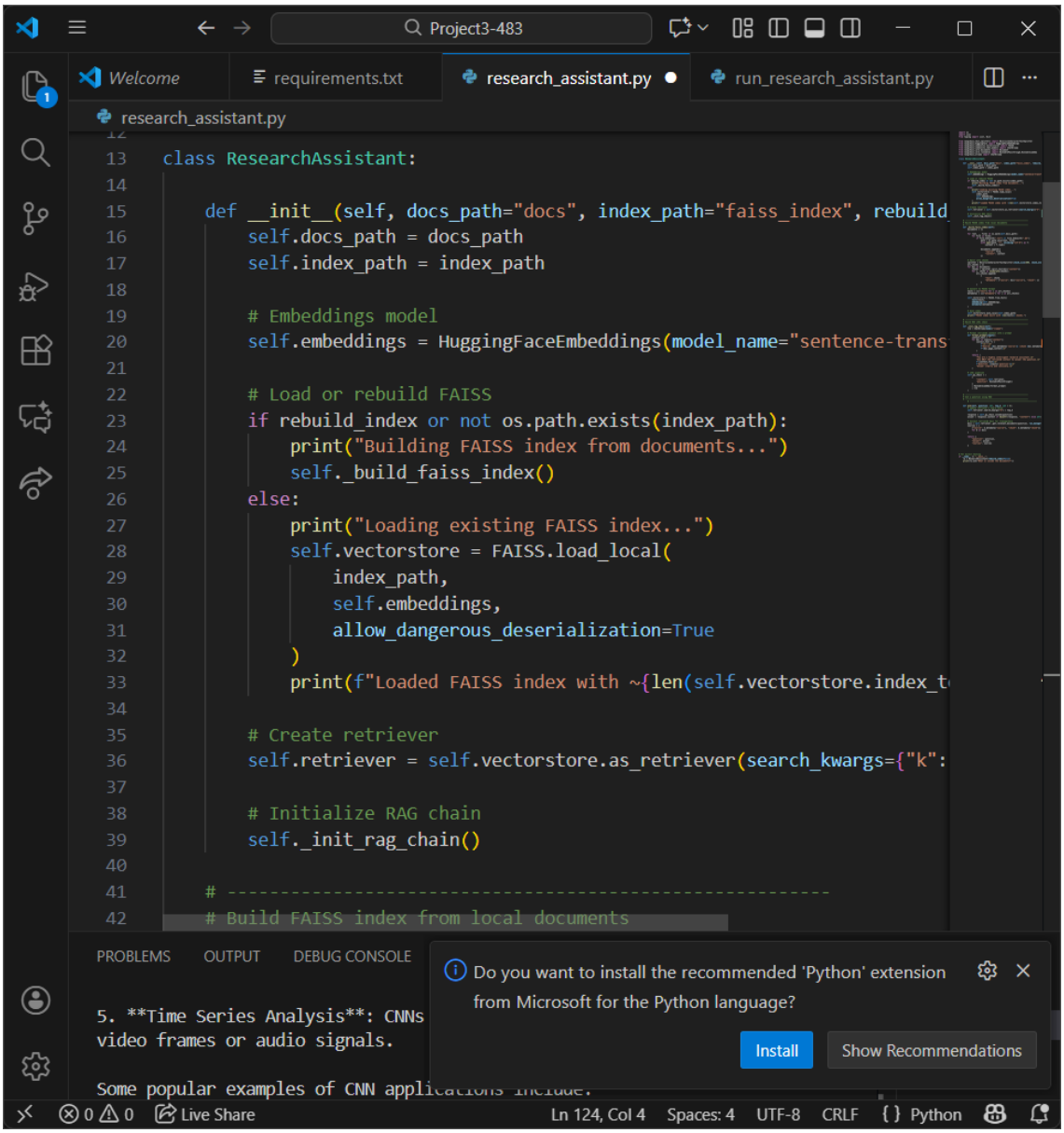
2.7 Demonstration Script (run_research_assistant.py)

The test script shows three required interactions:
1. A general factual question
2. A document-based follow-up question
3. A summarization request

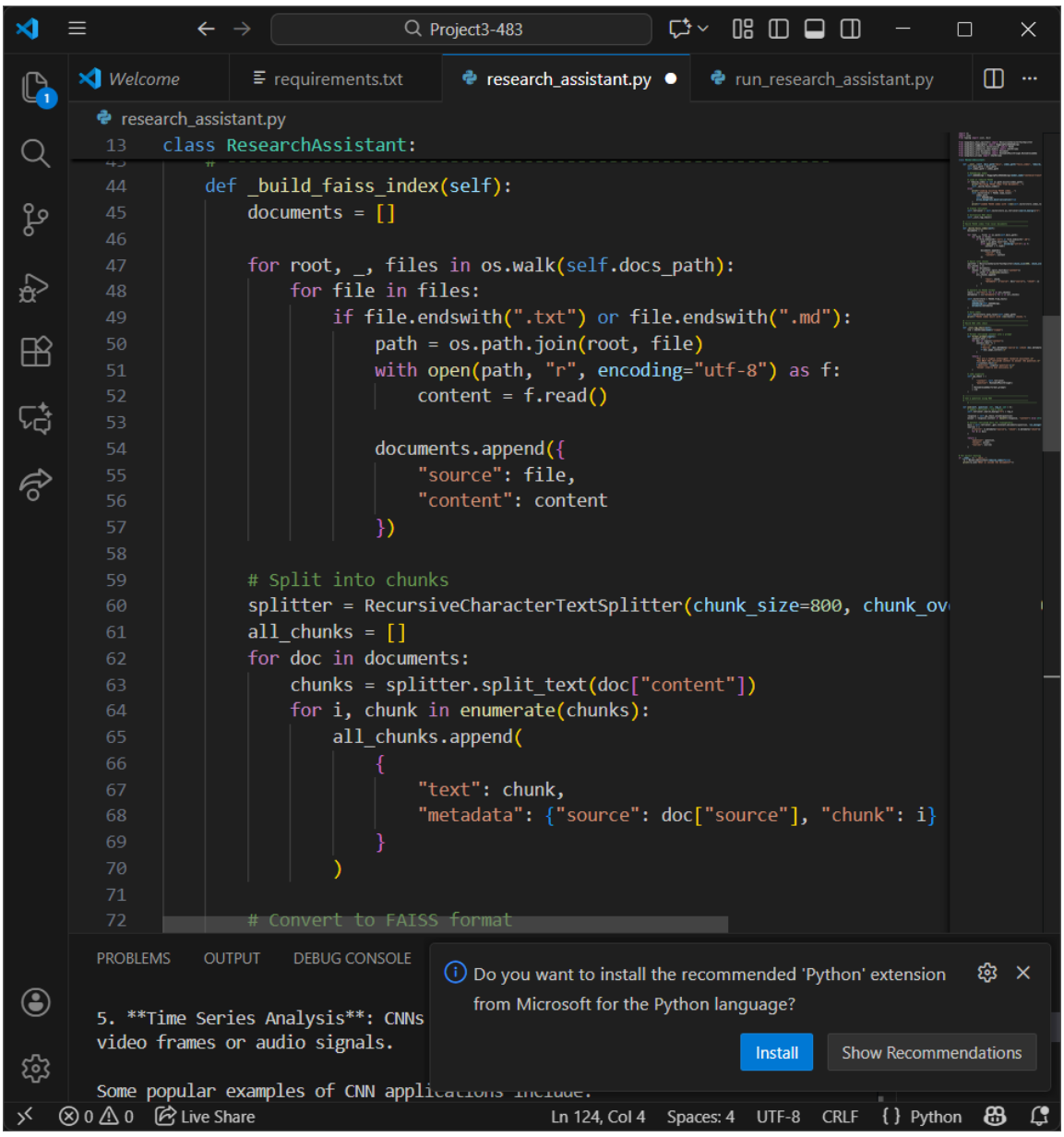
The script prints the question, answer, and retrieved source chunks, and saves the outputs to a JSON file.

3. **Screenshots**

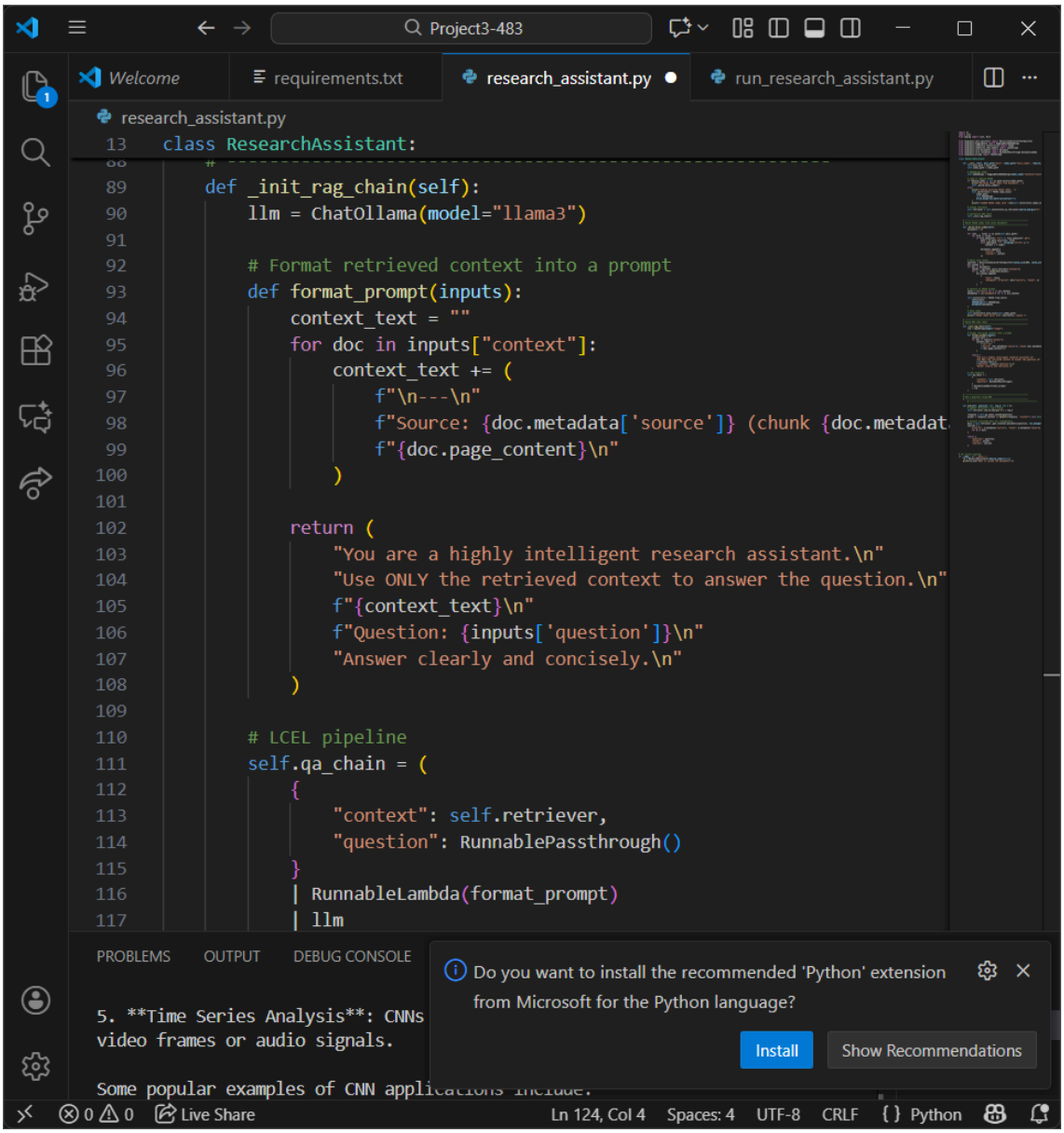
A. Code Snippets



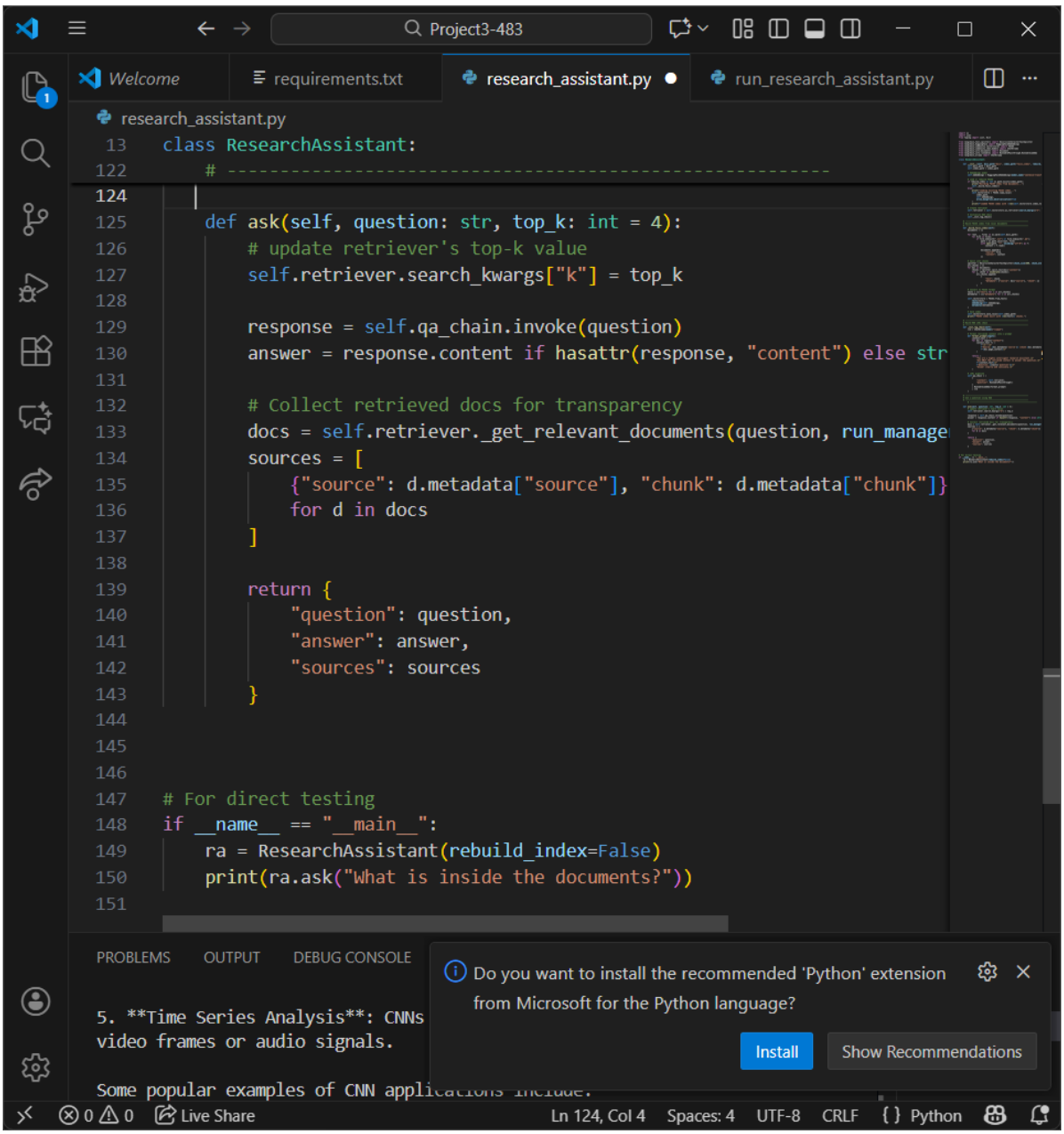
Initialization code (__init__)



Chunking / FAISS building code

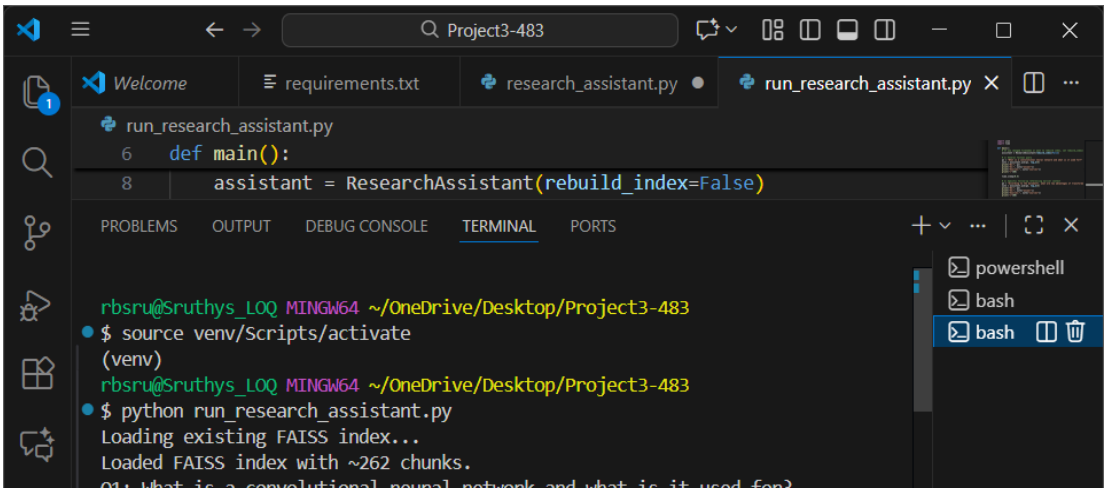


RAG chain setup (_init_rag_chain)



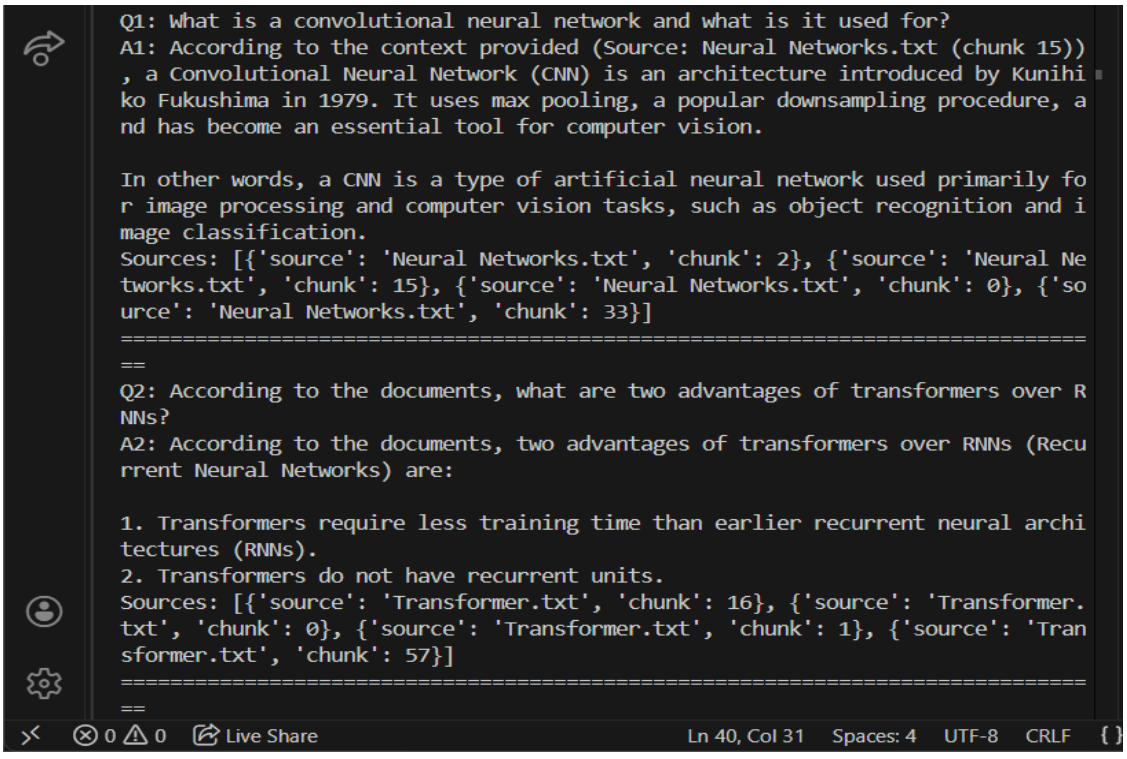
The ask() method

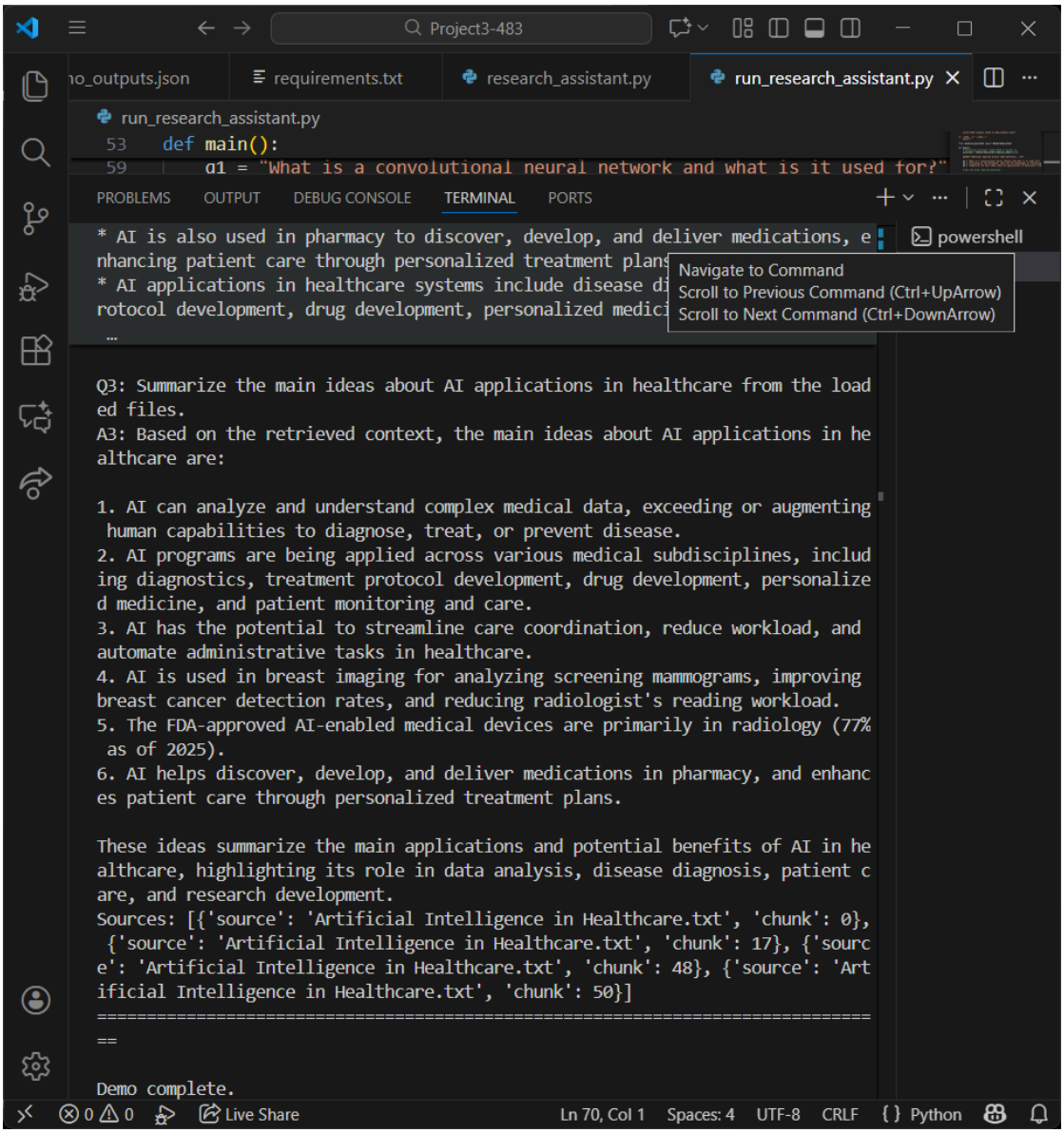
B. Terminal Output

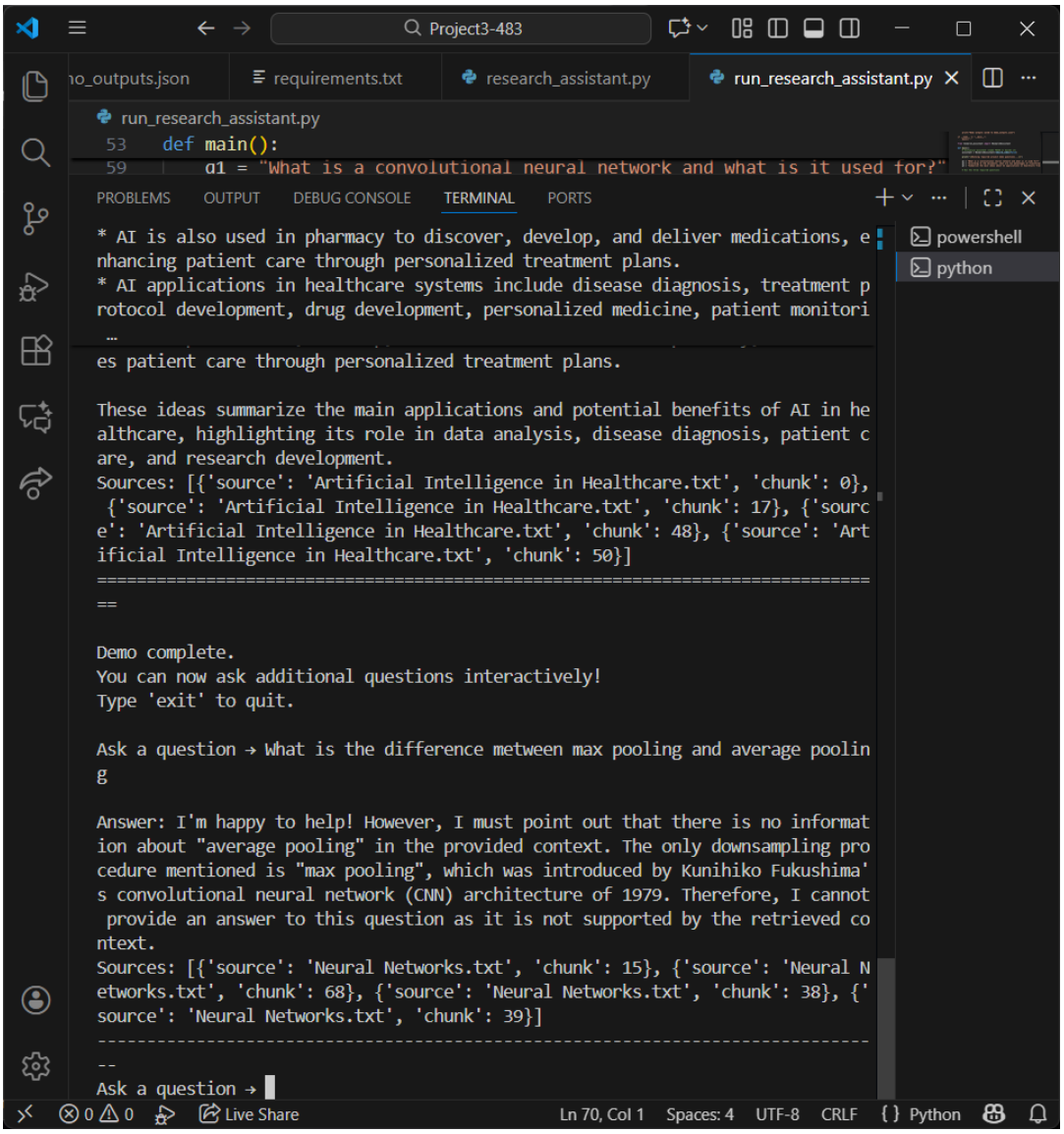


FAISS loading message:

Loading existing FAISS index...
Loaded FAISS index with ~262 chunks.




Project3-483
no_outputs.json
requirements.txt
research_assistant.py
run_research_assistant.py
run_research_assistant.py
53 def main():
59 q1 = "What is a convolutional neural network and what is it used for?"
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
* AI is also used in pharmacy to discover, develop, and deliver medications, e
nhancing patient care through personalized treatment plans
* AI applications in healthcare systems include disease di
rotocol development, drug development, personalized medici
Navigate to Command
Scroll to Previous Command (Ctrl+UpArrow)
Scroll to Next Command (Ctrl+DownArrow)
Q3: Summarize the main ideas about AI applications in healthcare from the load
ed files.
A3: Based on the retrieved context, the main ideas about AI applications in he
althcare are:
1. AI can analyze and understand complex medical data, exceeding or augmenting
human capabilities to diagnose, treat, or prevent disease.
2. AI programs are being applied across various medical subdisciplines, includ
ing diagnostics, treatment protocol development, drug development, personalize
d medicine, and patient monitoring and care.
3. AI has the potential to streamline care coordination, reduce workload, and
automate administrative tasks in healthcare.
4. AI is used in breast imaging for analyzing screening mammograms, improving
breast cancer detection rates, and reducing radiologist's reading workload.
5. The FDA-approved AI-enabled medical devices are primarily in radiology (77%
as of 2025).
6. AI helps discover, develop, and deliver medications in pharmacy, and enhanc
es patient care through personalized treatment plans.
These ideas summarize the main applications and potential benefits of AI in he
althcare, highlighting its role in data analysis, disease diagnosis, patient c
are, and research development.
Sources: [{'source': 'Artificial Intelligence in Healthcare.txt', 'chunk': 0},
{'source': 'Artificial Intelligence in Healthcare.txt', 'chunk': 17}, {'sourc
e': 'Artificial Intelligence in Healthcare.txt', 'chunk': 48}, {'source': 'Art
ificial Intelligence in Healthcare.txt', 'chunk': 50}]
==
Demo complete.
Live Share
Ln 70, Col 1
Spaces: 4
UTF-8
CRLF
Python



Output for Q1, Q2, and Q3

4. Demonstration Results (Actual Outputs)


```
{
 "interaction_1": {
  "question": "What is a convolutional neural network and what is it used for?",
  "answer": "According to the context provided (Source: Neural Networks.txt (chunk 15)), a Convolutional Neural Network (CNN) is an architecture introduced by Kunihiko Fukushima in 1979. It uses max pooling, a popular downsampling procedure, and has become an essential tool for computer vision.\n\nIn other words, a CNN is a type of artificial neural network used primarily for image processing and computer vision tasks, such as object recognition and image classification.",
  "sources": [
   {
    "source": "Neural Networks.txt",
    "chunk": 2
   },
   {
    "source": "Neural Networks.txt",
    "chunk": 15
   },
   {
    "source": "Neural Networks.txt",
    "chunk": 0
   },
   {
    "source": "Neural Networks.txt",
    "chunk": 33
   }
  ]
 },
 "interaction_2": {
  "question": "According to the documents, what are two advantages of transformers over RNNs?",
  "answer": "According to the documents, two advantages of transformers over RNNs (Recurrent Neural Networks) are:\n\n1. Transformers require less training time than earlier recurrent neural architectures (RNNs).\n2. Transformers do not have recurrent units.",
  "sources": [
   {
    "source": "Transformer.txt",
```

```
      "chunk": 16
    },
    {
      "source": "Transformer.txt",
      "chunk": 0
    },
    {
      "source": "Transformer.txt",
      "chunk": 1
    },
    {
      "source": "Transformer.txt",
      "chunk": 57
    }
  ]
 },
 "interaction_3": {
   "question": "Summarize the main ideas about AI applications in healthcare from the loaded files.",
   "answer": "Based on the retrieved context, the main ideas about AI applications in healthcare are:\n\n* AI can analyze complex medical data to diagnose, treat, or prevent diseases faster and more accurately than humans.\n* AI can automate administrative tasks, prioritize patient needs, and facilitate seamless communication among healthcare teams.\n* AI can be used in breast imaging for analyzing screening mammograms and improving breast cancer detection rates.\n* AI can help discover, develop, and deliver medications, as well as enhance patient care through personalized treatment plans.\n* AI has the potential to streamline care coordination, reduce workload, and improve patient monitoring and care.\n* AI can assist clinicians with disease diagnosis by processing large amounts of electronic health records (EHRs) and analyzing symptoms.\n* AI can also aid in early prediction of diseases such as Alzheimer's and dementia.\n\nThese applications have the potential to revolutionize healthcare systems, especially in developing nations where resources may be scarce.",
   "sources": [
    {
      "source": "Artificial Intelligence in Healthcare.txt",
      "chunk": 0
    },
    {
      "source": "Artificial Intelligence in Healthcare.txt",
      "chunk": 17
```

```
    },
    {
     "source": "Artificial Intelligence in Healthcare.txt",
     "chunk": 48
    },
    {
     "source": "Artificial Intelligence in Healthcare.txt",
     "chunk": 50
    },
    {
     "source": "Artificial Intelligence in Healthcare.txt",
     "chunk": 3
    },
    {
     "source": "Artificial Intelligence in Healthcare.txt",
     "chunk": 55
    }
   ]
  }
}
```


5. Reflection

Transformers and RNNs are two different approaches to sequence modeling, and this project shows why modern systems rely heavily on Transformer architectures along with retrieval methods.

RNNs, such as LSTMs and GRUs, process information sequentially, maintaining a hidden state from one step to the next. This creates a bottleneck: early information must be compressed into a single state vector. As sequences get longer, the model finds it hard to keep information because gradients fade over time. Even structures designed to address this, like LSTMs, still face long-range dependency issues.

Transformers replace recurrence with self-attention, allowing every input token to connect directly to every other token. Instead of remembering details step-by-step, the model calculates relationships in parallel, making it more efficient and significantly better at capturing long-range structures. However, Transformers have a fixed context window and cannot natively handle documents longer than a certain number of tokens.

This is where Retrieval-Augmented Generation (RAG) becomes important. By embedding documents and storing them in FAISS, the system retrieves only the most relevant text when a question is asked. The model then receives the retrieved chunks directly in the prompt, effectively giving it access to information far beyond its context limits.

This hybrid system combines:

- the flexible, non-sequential reasoning of Transformers,
- with the long-term memory provided by vector retrieval.

In practice, RAG greatly extends an LLM's usable memory while keeping inference efficient. This setup shows why modern AI systems rely less on internal sequence memory (like RNNs) and more on structured external memory systems that allow for more accurate and context-rich responses.

6. Conclusion

This project successfully created a local RAG-based research assistant using LangChain and Ollama. By integrating text splitting, embeddings, FAISS retrieval, and a custom prompt pipeline, the system showed how retrieval can enhance a Transformer model's reasoning abilities.

Through three test interactions—factual queries, context-aware follow-ups, and summarization—the system demonstrated correct retrieval behavior and grounded answers based on the loaded documents. The project provided practical experience with LLM pipelines, vector databases, and modern retrieval methods commonly used in real-world AI applications.