Smart Chatbot (Summarizer → RAG)

Part 4 — Multi-File Support + Chunking

🎯 Goal

Load one or more files or a URL.
Features:

1. **Load multiple local files** (`.pdf`, `.txt`, `.md`) from a folder.
2. **Chunk the text** into 500–1000 character segments.
3. **Store chunks in FAISS** with metadata (file name, file type, source path, etc.).
4. **Prepare them for RAG (Retrieval-Augmented Generation)**.

🧠 Concepts

- **Multi-file ingestion** → Instead of just one file, the script can load and process multiple file types (`.txt`, `.md`, `.pdf`) from your `data/` folder.
- **Chunk metadata** → Store filename, file type, and chunk ID so you know where each answer came from.
- **Reusable pipeline** → Drop any file into `data/`, and it will automatically be split, embedded, and made queryable.
  📚 Docs you might want later:
- LangChain loaders
- LangChain text splitters

📊 System Flow (Summarizer → RAG)

```
flowchart TD
    A[Start] --> B{Input Source?}
    B -->|URL| C[Fetch Article Text with BeautifulSoup]
    B -->|Files| D[Load Files from /data folder
(.txt, .md, .pdf)]

    C --> E[Summarize with GPT (concise/detailed/etc.)]
    D --> E[Summarize with GPT (concise/detailed/etc.)]

    E --> F[Split into Chunks
(500–1000 chars)]
    F --> G[Attach Metadata
(filename, type, chunk ID)]

    G --> H{FAISS Index Exists?}
    H -->|Yes| I[Load FAISS Index and Add New Chunks]
    H -->|No| J[Create New FAISS Index from Chunks]
```

```
    I --> K[Save Updated Index]
    J --> K[Save Index]

    K --> L[User Q&A Loop]
    L --> M[Embed Query with OpenAI Embeddings]
    M --> N[Retrieve Relevant Chunks from FAISS]
    N --> O[Ask GPT with Retrieved Context]
    O --> P[Return Answer to User]

    P --> L[Repeat Until Exit]
```