**Technical Architecture Overview**

This AI-powered help documentation system follows a **Retrieval-Augmented Generation (RAG)** architecture. It combines **web scraping, vector-based search (FAISS), and GPT-4-powered answers** to provide efficient, document-based question answering.

**System Components**

| **Component** | **Purpose** |
| --- | --- |
| **Web Scraper** | Extracts help documentation from a given website (help.something.com). |
| **Text Embeddings (Sentence Transformers)** | Converts text into numerical vectors for similarity search. |
| **FAISS (Vector Search Engine)** | Stores embeddings and retrieves the most relevant document based on query similarity. |
| **GPT-4 (OpenAI API)** | Generates answers based on retrieved documents. |
| **JSON Storage** | Stores scraped documentation for future queries. |
| **User Interface (CLI)** | Allows users to scrape new websites, view stored JSON files, and query documentation. |

**User Input (CLI Interface)**

* + The user selects either:
    - Scraping a new website (help.something.com).
    - Querying an already scraped website.
  + If a new website is entered, the system validates the URL format.

1. **Web Scraping**
   * Uses requests and BeautifulSoup to extract text.
   * Cleans unnecessary elements like headers, footers, and scripts.
   * Saves the extracted data in JSON format (scraped\_data/{domain}.json).
2. **Document Indexing**
   * Loads the scraped documents.
   * Converts text into embeddings using all-MiniLM-L6-v2.
   * Stores embeddings in a FAISS vector index for fast retrieval.
3. **Query Processing**
   * Converts user queries into embeddings.
   * FAISS searches for the most similar document chunks.
   * If no relevant results are found, the system returns:  
     "This information is not found in the documentation."
4. **Answer Generation (GPT-4)**
   * The retrieved text is formatted into a structured prompt.
   * GPT-4 generates an answer strictly based on retrieved documentation.
   * The system returns the answer along with source URLs.

**Technologies Used**

| **Technology** | **Usage** |
| --- | --- |
|  |  |
| **Python** | Core language for implementation. |
| **Beautiful Soup** | Extracts text from help documentation. |
| **Sentence Transformers (all-MiniLM-L6-v2)** | Converts text into numerical vectors (embeddings). |
| **FAISS** | Efficient vector search engine for retrieving relevant documents. |
| **OpenAI GPT-4** | Answers user queries based on retrieved documentation. |
| **JSON Storage** | Stores scraped documentation for querying. |

**Future Scalability Considerations**

1. **Use a More Advanced Embedding Model**
   * Replace all-MiniLM-L6-v2 with multi-qa-mpnet-base-dot-v1 for improved search accuracy.
2. **Implement Distributed FAISS for Large-Scale Search**
   * Use FAISS with GPU acceleration or distributed indexing for handling millions of documents.
3. **Expose the System as an API**
   * Convert the CLI-based system into a **Fast API or Flask web service** for seamless integration.

This architecture ensures that the system efficiently **scrapes, stores, retrieves, and answers questions based on help documentation** while maintaining high-speed search and retrieval.