# **RAGChatbot – Documentation**

## **Project Overview**

The **RAGChatbot** is a Retrieval-Augmented Generation (RAG) based assistant designed to help users navigate the [Inforens](https://www.inforens.com) website. It scrapes webpage content, processes it into searchable chunks using vector embeddings and FAISS, and uses an LLM to answer user queries **only using the retrieved website data**.

Since I didn’t have direct access to the content, I initially scraped it from the website. However, scraping can now be removed, and instead, a pre-prepared notepad (text file) can be provided to the model. The bot will then refer to this notepad for answering questions.

## **Technology Stack**

| **Component** | **Purpose** |
| --- | --- |
| LangChain | Framework for building modular LLM workflows |
| Ollama | Local LLM and embedding inference (need to replace with Open Ai or Perplexity as it provides better responses. Ollama was used for testing purposes). |
| FAISS | Vector database for similarity search |
| BeautifulSoup + Selenium | Web scraping (Not required) |
| RecursiveCharacterTextSplitter | Efficient text chunking |
| PromptTemplate | Custom LLM prompt creation (to provide developers’ instruction to the LLM. Example : Answer the query of user and redirect them to corresponding link) |
| RetrievalQA | Combines document retrieval and LLM answering |

## **Workflow Overview**

1. Scraping Website Content (Not required)

* Uses **Selenium** to load and interact with JavaScript-rendered pages.
* **Closes popups** (e.g., on the Inforens homepage) using simulated user interaction.
* Extracts all visible text using **BeautifulSoup**.
* Saves to a .txt file for caching and reusability. (replace this text file with the content that we have access to). Content in new webpages (blogs) have to be added in the text file automatically.

2. Text Chunking & Embedding

* The raw scraped text is chunked using RecursiveCharacterTextSplitter.
* Each chunk is embedded using OllamaEmbeddings.
* Embeddings are stored in a **FAISS index** for similarity search.

3. Vector Store Creation

* Each chunk is wrapped in a Document object.
* Documents are stored using InMemoryDocstore.
* A retriever is created from the FAISS vector index.

4. Query Answering

* User question is passed through a **custom prompt** that:
  + Restricts answers strictly to the website content.
  + Redirects the user to relevant webpages when applicable.
* A RetrievalQA chain retrieves the most relevant chunks.
* The OllamaLLM model generates a context-aware response.

## **Scope of the Project**

Current Features

* RAG-based retrieval for **accurate, grounded answers**.
* Interactive CLI-based chatbot interface.

Potential Extensions

| Feature | Description |
| --- | --- |
| Scheduler | Regularly update the existing content with content from new websites |
| API Mode | Serve chatbot as a REST API |
| Long-term Memory | Add persistent storage for user chat history |
| Feedback Loop | Log answers and user feedback for improving relevance |
| **Access through admin dashboard** | **Access users’ search history through admin dashboard (Required)** |

## **Workflow Diagram**

