Webpage Question Answering

The Webpage Question Answering is a use case scenario where the user enters a website link and a question he or she intends to know about from the website.

The process:

Step 1:
The contents of the web page is fetched using the requests library.
BeautifulSoup's HTML.parser parses the content retrieved from the website and a structured representation of the HTML document is obtained and stored into a variable (soup). Using soup.get_text(), we get the text from the HTML document, free from the tags. There are multiple line spaces between paragraphs in the text. The multiple spaces were eliminated using regular expressions.

Step 2:
The formatted text is then saved into a text file, which is then loaded as a document using TextLoader from langchain_community.
Then the document is split into chunks with a chunk size of 500 characters and with a chunk overlap of 60 characters.

Alternatively WebBaseLoader can be used from langchain to get the contents of the webpage. It uses BeautifulSoup as well. It would be faster and simpler than the traditional approach I followed here.

Step 3:
Vector embeddings are generated for the documents using OllamaEmbeddings and a vector store is created using FAISS (from LangChain).

Step 4:
The latest Llama 3 is the model of choice for this assignment. The Llama 3 model is accessed using the Groq API. Specifically, the 8 Billion parameters version of Llama 3 is being used for this assignment.

Step 5:
A prompt is created using ChatPromptTemplate, a document chain chain is created using create_stuff_documents_chain method which takes in the large language model and the prompt.

Step 6:
A vector store retriever is created using the vector store. A retrieval chain is created using the vector store retriever and the document chain. Using this retrieval chain, we can pass the input and get the output. In the output, a bit of processing is done where using replace method, the * in the text generated by the LLM is replaced with \n (a newline character).

The code for all the steps mentioned above is encapsulated into a function called qna_bot. The function takes in the url and the question as the arguments which are used in the code and returns the formatted output. If the question is not present in the webpage, it returns “I don’t know the answer”. This function is called and used within the flask app.

The API:

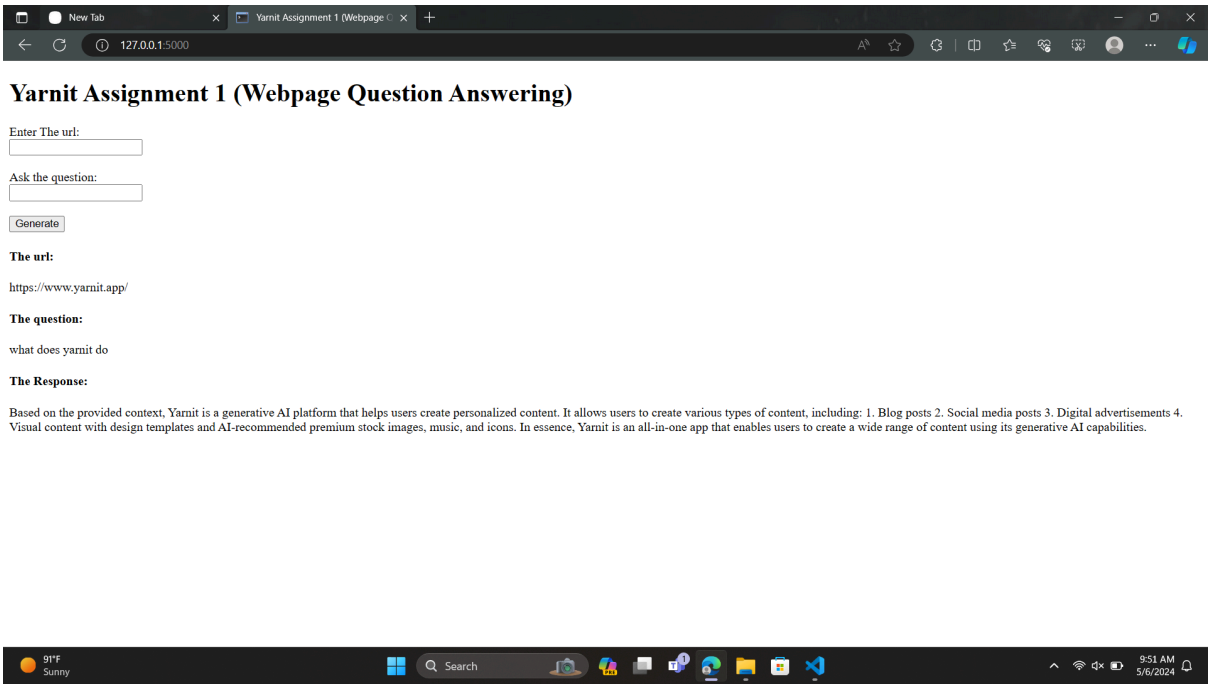
The API is created using the flask web framework. It is a simple app with just one route (the root) which handles both GET and POST HTTP methods. A function index() will be called when we access the root path. Using request.form.get("url") and request.form.get("question"), we get the url and question from the user using a HTML form. These are passed to the qna_bot function. The function returns a string (output from the LLM).

Then the index.html template is rendered using render_template and the values of url, question and output are passed as variables to the template.

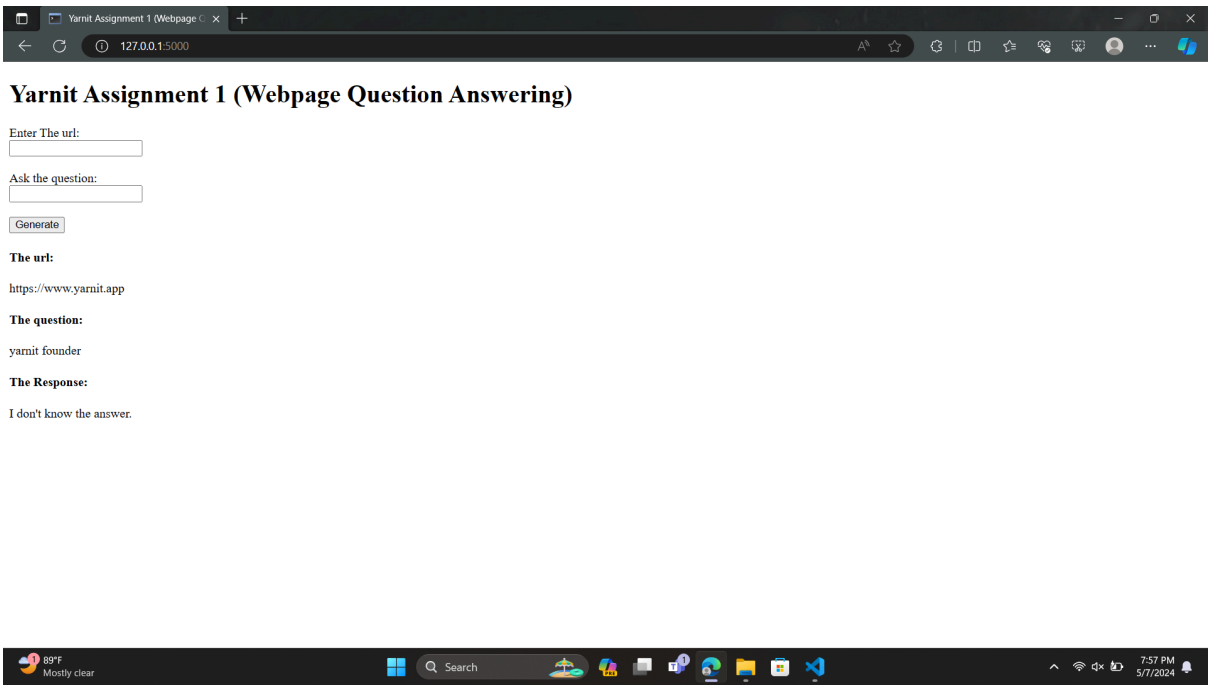
The HTML:

A basic UI is created to interact with the app. The index.html has 2 input text fields for url and question respectively and a submit button. The page displays the url and the question entered by the user and the response (the returned output) from the LLM.

Screenshot:



When the answer is there



When the answer is not there