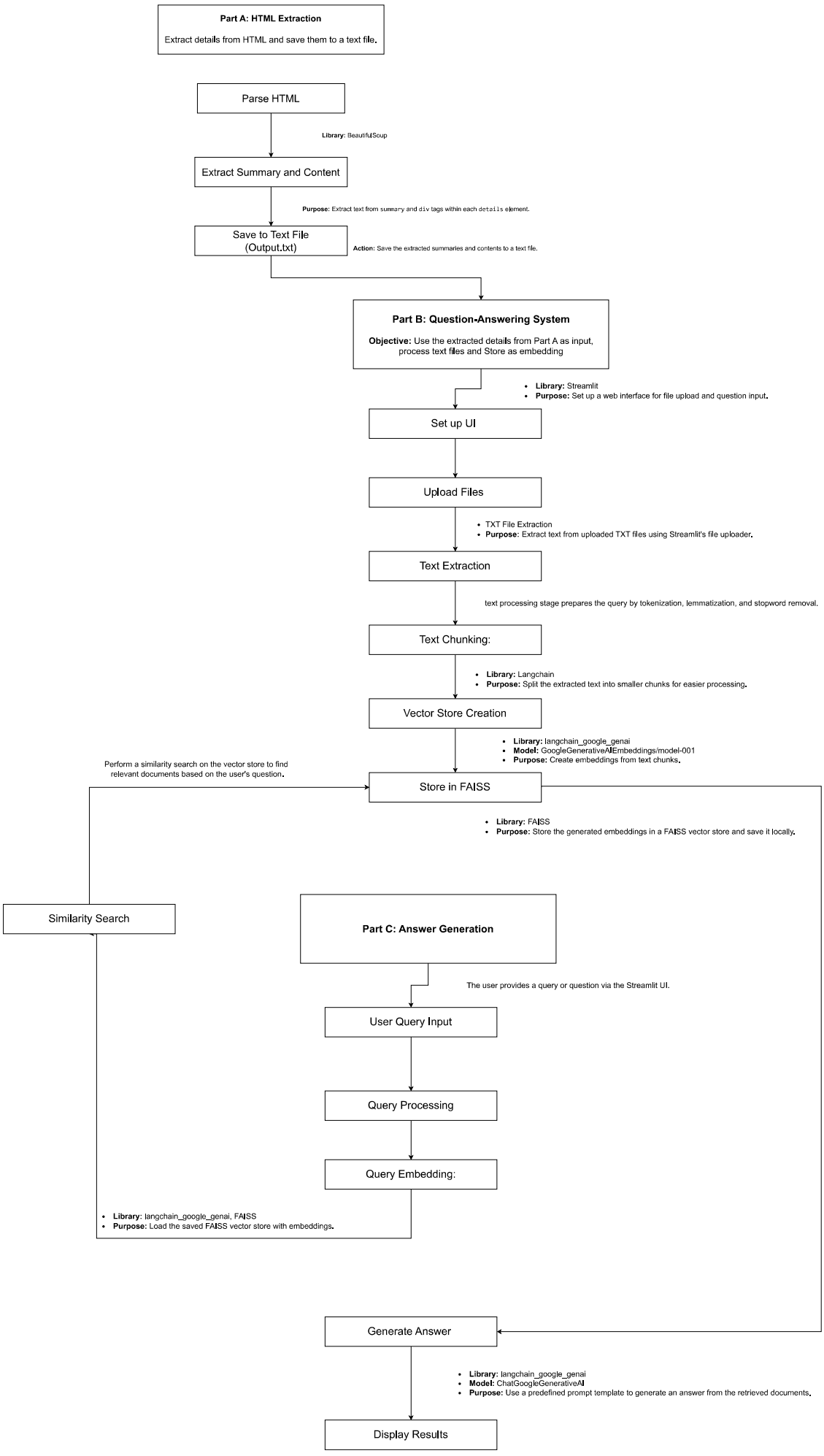
Part A: HTML Extraction
Extract details from HTML and save them to a text file.
Parse HTML
Library: BeautifulSoup
Extract Summary and Content
Purpose: Extract text from summary and div tags within each details element.
Save to Text File (Output.txt)
Action: Save the extracted summaries and contents to a text file.
Part B: Question-Answering System
Objective: Use the extracted details from Part A as input, process text files and Store as embedding
Library: Streamlit
Purpose: Set up a web interface for file upload and question input.
Set up UI
Upload Files
TXT File Extraction
Purpose: Extract text from uploaded TXT files using Streamlit's file uploader.
Text Extraction
text processing stage prepares the query by tokenization, lemmatization, and stopword removal.
Text Chunking:
Library: Langchain
Purpose: Split the extracted text into smaller chunks for easier processing.
Vector Store Creation
Library: langchain_google_genai
Model: GoogleGenerativeAIEmbeddings/model-001
Purpose: Create embeddings from text chunks.
Perform a similarity search on the vector store to find relevant documents based on the user's question.
Store in FAISS
Library: FAISS
Purpose: Store the generated embeddings in a FAISS vector store and save it locally.
Similarity Search
Part C: Answer Generation
The user provides a query or question via the Streamlit UI.
User Query Input
Query Processing
Query Embedding:
Library: langchain_google_genai, FAISS
Purpose: Load the saved FAISS vector store with embeddings.
Generate Answer
Library: langchain_google_genai
Model: ChatGoogleGenerativeAI
Purpose: Use a predefined prompt template to generate an answer from the retrieved documents.
Display Results