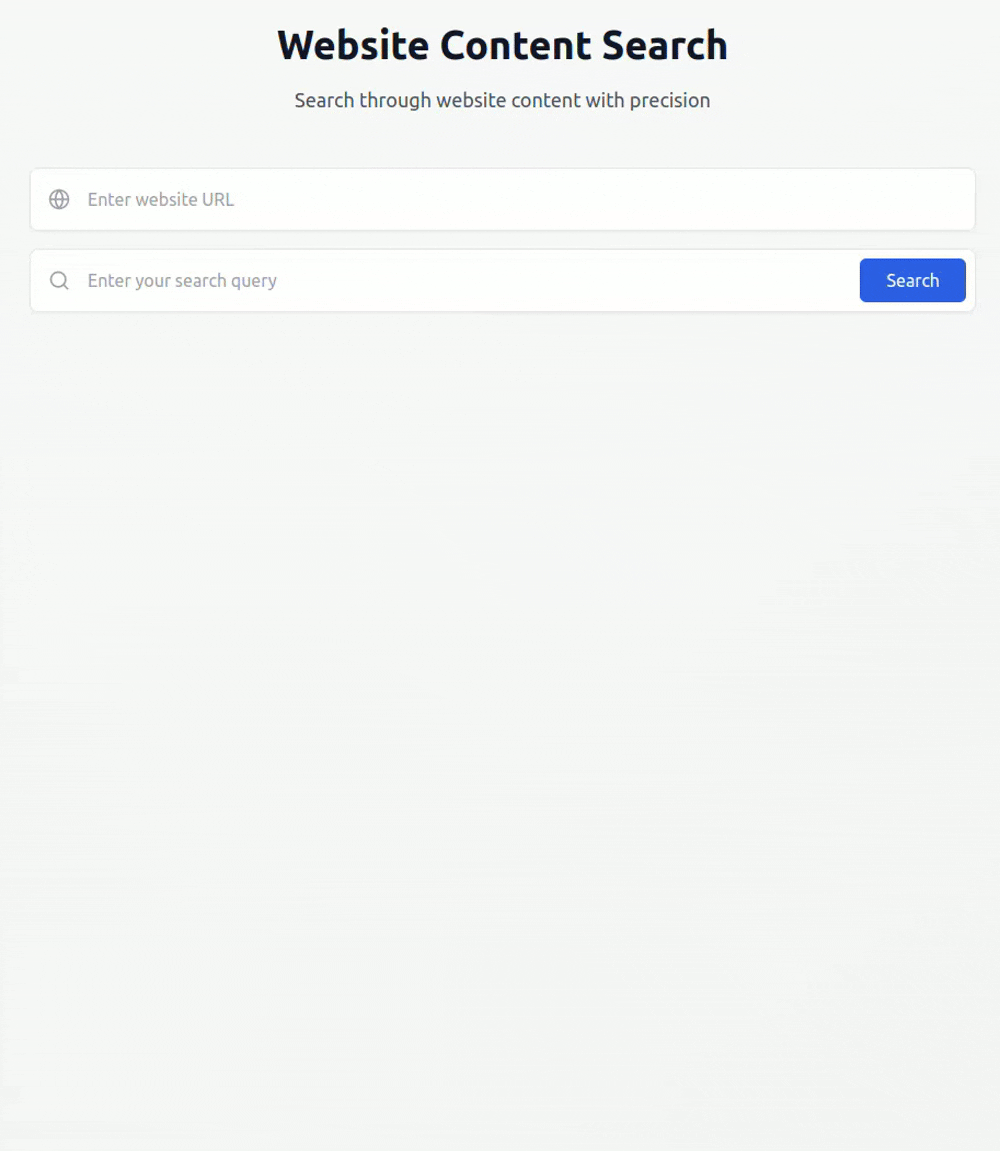
# Objective

Develop a single-page application (SPA) where users can input a website URL and a search query. The application will return the top 10 matches of HTML DOM content (chunks) based on the search query.

Figure: Sample Demo

## 

## Application Details

### Frontend Requirements:

* **Framework:** Use React or Next.js.
* **Features:**
  + A **form** with two input fields:
    1. **Website URL** (Text input for the target URL).
    2. **Search Query** (Text input for the query string).
  + A **submit button** to trigger the query.
  + Display the **top 10 matches** in a structured format (e.g., card or table layout), showing HTML content chunks with up to 500 tokens each.

### Backend Requirements:

* **Framework:** Use any Python-based framework (e.g., Flask, FastAPI, or Django).
* **Functionality:**
  + Fetch HTML content from the provided URL (assume no JavaScript is required to render the page).
  + Tokenize the HTML content and split it into chunks of a maximum of 500 tokens each.
  + Search for the query string across these chunks.
  + Return the top 10 matching chunks based on relevance.

### Vector Database Integration:

* Use an open-source vector database like **Milvus**, **Weaviate**, or **Pinecone** to:
  + Index the HTML chunks.
  + Perform semantic search to find the most relevant matches for the query.

## 

## Key Features to Implement:

1. **HTML Parsing:** Use a library like BeautifulSoup (Python) to extract and parse HTML DOM content. Clean the content to remove scripts, styles etc.
2. **Tokenization:** Use a tokenizer (e.g., from transformers or nltk) to split content into chunks of up to 500 tokens.
3. **Search:** Perform a semantic or keyword search using the vector database.
4. **Relevance Ranking:** Sort results based on relevance and return the top 10.

# 

# Submission Checklist

1. **Source Code:**
   * Include the complete source code for both frontend and backend.
   * Provide a README.md with the following details:
     + Instructions for setting up and running the project locally.
     + Prerequisites and dependencies.
     + Any additional configuration or setup steps for the vector database.
2. **Walkthrough Video:**
   * A short video (5–10 minutes) showcasing:
     + How the application works (UI/UX demo).
     + The workflow of submitting a query and retrieving results.
     + An explanation of the codebase structure.
3. **Slide Deck (5 Pages):**
   * Page 1: **Introduction** – Overview of the task and solution approach.
   * Page 2: **Frontend Design** – Explanation of UI/UX design and React/Next.js implementation.
   * Page 3: **Backend Logic** – Overview of the backend framework, HTML parsing, and tokenization.
   * Page 4: **Vector Database** – Details on the chosen vector database and integration for semantic search.
   * Page 5: **Conclusion** – Challenges faced, lessons learned, and potential improvements.