Website Content Search Tool – Technical Documentation

Overview:

The Website Content Search Tool is a full-stack web application designed to extract, chunk, embed, and semantically search content from any given website URL. It enables users to find meaningful content matches on webpages based on semantic similarity rather than keyword matching.

Objective

To create a seamless tool that:

- Takes a user-provided URL
- Extracts meaningful content from that page
- Splits the content into manageable tokenized chunks
- Embeds the content using an NLP model
- Accepts a user query
- Returns the most semantically relevant text chunks from the page

Use Cases

- Competitive analysis
- SEO content search
- Research and citation mining
- Smart document readers
- Legal & academic paper search

Tech Stack

Layer	Technology	Purpose
Frontend	**Next.js**, TailwindCSS	User interface, input, result display
Backend	**FastAPI**, Python	API logic, data processing
NLP	SentenceTransformers	Embedding content & queries
Parsing	BeautifulSoup, Requests	Web scraping and text extraction
State	ReactJS, React Hooks	Client-side state & form handling

Logic & Workflow

Step 1: User Input

- User submits a website URL and a search query via the frontend UI.

Step 2: Web Scraping

- Backend (FastAPI) fetches the webpage content using the `requests` library.
- HTML is parsed using `BeautifulSoup` to extract readable text from `<p>`, `<div>`, and other tag types.

Step 3: Tokenization and Chunking

- Text is split into **chunks of 500 tokens** using a tokenizer.

- This keeps processing efficient and contextually relevant for embeddings.

Step 4: Embedding

- Each text chunk is embedded using `SentenceTransformer('all-MiniLM-L6-v2')`.
- The user query is also embedded into a vector.

Step 5: Semantic Search

- Cosine similarity is calculated between the query vector and each text chunk.
- Top 10 most similar chunks are returned to the frontend.

Step 6: Result Display

- Frontend shows each result in a styled card format.
- Results can be expanded to preview full HTML or content.

File Structure

```
website-content-search/

|

├── backend/

│   ├── main.py              # API routes & logic

│   ├── chunker.py            # Tokenizer + chunk logic

│   ├── parser.py            # HTML parsing and cleaning

│   ├── vector_db.py          # In-memory vector search logic

│   └── requirements.txt      # Python dependencies

|

├── frontend/

│   ├── app/

│   │   ├── layout.js, page.js # UI components
│   │   └── globals.css       # Tailwind base styling
│   ├── public/               # Favicon and icons
│   ├── package.json          # Frontend dependencies
│   └── tailwind.config.js    # Tailwind config
│
├── README.md
└── .gitignore
```

🛠️ Setup Instructions

Backend Setup

```
cd backend

python -m venv venv

source venv/bin/activate  # Windows: venv\Scripts\activate

pip install -r requirements.txt

uvicorn main:app --reload
```

Frontend Setup

```
cd frontend

npm install

npm run dev
```

Access Locally

- Frontend: http://localhost:3000
- Backend API: http://localhost:8000

Features

- Semantic Search
- 500-token chunking
- Vector similarity ranking
- Expandable content preview
- Embedded with Transformers
- Scalable architecture (backend/frontend separation

Error Handling

- Invalid URL: Returns 400 with message
- Empty content: Returns a fallback response
- Backend/server errors are caught and displayed on frontend

Future Enhancements

- Integrate Weaviate or Pinecone for large-scale vector search
- Highlight matched sentences in preview
- Save search history & cache chunks
- Login system with session-based query
- Add support for PDFs, .docx, and RSS feeds