## 2 MINUTE READS - ARTICLE SUMMARIZER

**OBJECTIVE:**

To create a tool that condenses lengthy blog posts and articles into summaries.

**FEATURES:**

- Web-crawlers to gather content from articles
- NLP for summarization.
- User interface for input and displaying results.

**HOW IT WOULD WORK:**

1. DEVELOPING THE WEB CRAWLER

- Set Up Crawler: Use BeautifulSoup to create a crawler that can search the web based on the input and collect URLs of relevant articles.
- Parse Content: Extract the text content from the collected URLs

2. IMPLEMENT THE SUMMARIZATION ALGORITHM

- Implement a basic text summarization model using a pre-trained T5 model from the Hugging Face Transformers library.
- Fine-Tuning: Fine-tune the model on a dataset of articles and their summaries if necessary.

3. BACKEND/FRONTEND:

- Backend: Set up a server using Flask to handle the input, trigger the web-crawler, and process the summarization.
- Frontend: Create a simple UI using HTML, CSS, and JavaScript or a framework like React to allow users to enter a search string and view the summarized results.

4. TESTING AND OPTIMIZATION

Repository: https://github.com/shravanisaraf/two-minute-reads-summarizer.git

---

Project Plan: Developing an Article Summarization Tool

1. Develop the Web Crawler:

Set Up Crawler:
- Objective: Create a web crawler to take input as url and provide content as output.
- Tools: Use BeautifulSoup, a beginner-friendly library for web scraping.
- Steps:
  - Install BeautifulSoup (`pip install beautifulsoup4`) and other necessary libraries.

Parse Content:
- Objective: Extract text content from the collected URLs.
- Implementation: Utilize BeautifulSoup to parse HTML content and extract relevant text.
- Considerations: Handle different HTML structures and text formatting to ensure accurate content extraction.

2. Implement the Summarization Algorithm:

Choose a Pre-trained Model:
- Objective: Select a pre-trained T5 model from the Hugging Face Transformers library for text summarization.
- Model Selection: Choose a model size (e.g., 't5-small', 't5-base') based on computational resources and performance requirements.
- Benefits: T5 offers a beginner-friendly interface and strong performance for text summarization tasks.

Fine-Tuning:
- Objective: Fine-tune the selected model on a dataset of articles and their summaries if necessary.
- Implementation: Utilize transfer learning techniques to adapt the pre-trained model to the specific summarization task.
- Considerations: Ensure proper data preprocessing, model hyperparameter tuning, and evaluation metrics selection.

3. Backend/Frontend Development:

Backend (Using Flask):
- Objective: Set up a Flask server to handle user input, trigger the web crawler, and process summarization.
- Implementation: Write Python scripts to define routes, handle requests, and interact with the web crawler and summarization algorithm.
- Considerations: Keep the backend logic simple and modular for easier maintenance and scalability.

Frontend (Simple UI):
- Objective: Create a simple user interface using HTML, CSS, and JavaScript (or a frontend framework like React) for user interaction.
- Implementation: Design input forms for users to enter search strings and display summarized results.
- Considerations: Prioritize user experience and ease of use in the UI design.

4. Testing and Optimization:

Testing:

- Objective: Thoroughly test the application to ensure functionality, accuracy, and performance.
- Testing Approaches: Conduct unit tests, integration tests, and end-to-end tests to validate each component's behavior.
- Considerations: Include edge cases, error handling, and performance benchmarks in the testing process.

Optimization:
- Objective: Optimize the application for performance, scalability, and user experience based on feedback and testing results.
- Optimization Techniques: Identify and address bottlenecks, improve code efficiency, and optimize resource usage.
- Considerations: Continuously monitor and iterate on the application to enhance its performance and usability.