**WebScrapingAPI Technical Task**  
*Mihai Trandafir*

**Technology Choices**

**Fastify – Why ?**  
I opted for fastify because of my familiarity with it. It has a reputation as one of the fastest ( if not the fastest) web framework for Node.Js

**The scraping – Puppeteer**

I chose Puppeteer as my scraping because of it’s support for XPath. Although I explored Cheerio, it was limited to classes and HTML elements.

The scraping function targets each post directly. Selecting the container that has all the posts in it, I am able to iterate over every child element. This way the function is independent of the number of the posts.

To get the words of each article was a challenge. The text was wrapped in different elements and it was hard to identify. I had to extract every paragraph, list and subtitle independently, use regex to eliminate punctuation and special characters such as “\n” and “\r” .

If the process takes too long I’ve implemented a timeout feature of 10 seconds.

**The sentiment analysis**

This was a though one, not because of how hard it is but because of the constraints. Not being allowed to use A.I to analyze the text was a challenge to make it accurate.

So what I’ve done is:

* Got 2 lists, one of positive words and one of negative words. [Positive Words](https://gist.github.com/mkulakowski2/4289437) and [Negative Words](https://gist.github.com/mkulakowski2/4289441).
* Tokenize the text (reused the function for counting)
* Implement **scoring** mechanism that analyzes every word in the post.
* If the word is in the positive words list the score will increment by 1. If it is in the negative words list, it will decrement by 1.
* If the final score of the post is positive, the sentiment would be positive, if it’s 0 would be neutral and if it’s unde 0 it will be negative.

I wanted to add weight to each word in the list but I found it very hard to determine a balance so I had to trade that accuracy for time. Also maybe implementing a threshold on the neutral state would improve it. For example a post should be considered positive only if it’s above 10 score and negative if it’s only under -3.

**The additionaly implemented features and the proposed ones**

Implemented ones:

* Extracts all the data regarding the images and stores it inside the image object.
* Gives the author and their profession of each article
* Returns the post content
* Returns the date of the article

Proposed:

* Use A.I for better sentiment analysis.
* Endpoints that return specific website components (e.g., images, links, or only post titles) or filters for endpoint content inclusion.

**Web Application Development with Qwik**

React, despite its immense popularity, never quite resonated with me. Qwik was not just about trying a new technology; it was a conscious challenge I set for myself to showcase adaptability. Among the big 4 frameworks (Vue, React, Angular, and Svelte), Qwik is way faster than any of them.

**The bonus task**

Implemented the **metrics** endpoint which will give the following data about the posts:

* **Average Post Length**
* **Top Terms**
* **Sentiment Distribution**
* **Author Contribution**

Metrics are very important nowadays especially in the A.I era, especially for training data. That’s why I’ve implemented this feature, I thought that’s something the end use would like.   
  
  
GITHUB Link: