The solution revolves around a modular design, with clear separation of configuration, data extraction, analysis, and file handling. Here's how I approached solving the problem:

1. **Environment and Configuration Management:**

**- .env file:** This file stores various paths (input, stopword, positive/negative words, output) as environment variables. These variables allow flexibility in configuring the file paths without hardcoding them.

**- config.py:** I loaded the `. env` file using `dotenv` and used `os.environ` to fetch the values, making them accessible across different scripts.

**2. Data Extraction (data\_extraction.py):**

- I wrote a function **`extract\_data`** to retrieve data from URLs using `requests` and `**BeautifulSoup`** for web scraping. The function extracts article titles and content from each URL.

- I read the list of URLs from the input Excel file, looped through them, and saved the extracted content to a specified directory.

**3. Data Analysis (data\_analysis.py):**

- I created several functions to analyse the text, such as calculating positive/negative scores, polarity, subjectivity, and various readability metrics (like sentence length and fog index). These functions tokenize the text and calculate scores based on word lists (positive and negative words) and complexity of words.

- The **`analyze\_text`** function combines these metrics into a dictionary, which is returned for further processing.

**4. Main Application Logic (app.py):**

- I implemented the main logic to process the extracted text files. For each article data analysis. I am calling the “analysis\_text” function which will return the all the metrics as a dictionary.

- The results were stored in dictionaries, then combined into dataframes and saved as Excel

- Also I made one more output file with the name **Output\_noStopwords.xlxs** . In this file, I analysis the data after removing the all the stop words given in the stopwords folder. For each article, I first analysed the text with stopwords included, then removed stopwords using the `remove\_stopwords` function and performed the analysis again.

- files, including both versions: with and without stopwords.

By structuring the solution this way, I ensured separation of concerns and made the code modular, reusable, and easy to maintain.