Take-Home Exercise: Text Analysis-as-a-Service (Full-Stack)

Exercise

Overview

You will build a small application that allows a user to analyse a text file and get metrics on the contents of the file. The key goals are to produce a useful app that gives the user and end to end experience, from upload to metrics, as well as clean, maintainable code.

Important Note:

- You may use any modern language or framework that you might use yourself in a professional environment. You may use multiple languages for FE and BE, or you may choose to use the same one.
- You must provide a `Dockerfile` and/or `docker-compose.yml` so we can easily build and run your service locally.
- You do not need to worry about authentication/authorization for your API, nor about production deployment scenarios.
- Code as you usually would. You're free to use Google, open-source libraries, ChatGPT, etc. Just to tell us (see submitting section) how much you used it, where did it work well and where did ChatGPT struggle and how you worked around it.

We don't expect a fully production-grade application. Roughly 3 hours of work is suggested, but feel free to spend more time if you'd like. During a follow-up interview, we'll discuss your solution and possibly add new requirements.

Core Requirements

1. **Upload UI**
    - Web form (or drag-and-drop) for a single .txt file.
    - Show upload progress + success/failure states.
2. **Text-Metrics Engine**
    - On the server, read the text and compute **three** of:
        - Total word count
        - Unique word count
        - Average sentence length
        - Flesch–Kincaid readability score
    - Top 10 most frequent words (excluding stopwords)
    - Package results into JSON: metric name, value, and (where relevant) line/paragraph references.
3. **Results Presentation**
    - FE fetches the JSON and renders a clear list/table of metrics.
    - For “top 10 words,” show a simple bar chart or list.
4. **History**
    - The user should be able to see a history of all the files that have previously been analysed and their metrics

Data Persistence

- You may use a lightweight local database (SQLite, Postgres, etc.) or even an in-memory map if time is limited.
- If you choose a DB, include instructions or migrations so we can run it easily in Docker.
- You may select whatever you feel is appropriate for file storage.

Running Your Service

- Provide a Dockerfile and/or docker-compose.yml so we can run docker-compose up (or equivalent) to start your service.
- Include a README.md with clear instructions:
  - How to build and run the application in Docker.
  - Any required environment variables.
  - (If required) example curl commands or requests to test the endpoints.
- Also include a section in the README.md with ideas you have for future additions if you had a week to implement this.

Submitting

Zip all your code, and send it to us over email (or provide us with a link to download it, e.g. Google Drive), by submitting to lewis@cogna.co .

Please indicate the amount of time you spent on it.

Please also indicate (see submitting section) how much you used ChatGPT (or equivalent), where did it work well and where did ChatGPT struggle and how you worked around it.

If you have any questions at all, email lewis@cogna.co