Engineering programming assignment

This programming assignment is designed to evaluate your technical implementation skills and your ability to reason about scalable system architecture. The problem mirrors real challenges we encounter at ANVA when processing large volumes of text data, and your solution will serve as the foundation for discussing how such systems evolve to meet production demands. We're interested not only in your code quality and testing approach, but also in understanding how you think about building robust, maintainable systems that can grow with our business needs.

Interview Process

Following your submission, we will conduct a technical interview that uses your implemented solution as the foundation for a broader technical discussion. During this conversation, we will review your code implementation choices and reasoning, then explore how your solution might evolve to address real-world challenges at scale.

The discussion will cover foundational technical concepts including performance optimization, data structures and algorithms, concurrent programming patterns, and testing strategies. We will also examine architectural principles for distributed systems, exploring topics such as scalability patterns, fault tolerance mechanisms, data consistency models, and service decomposition strategies.

This approach allows us to understand not only your coding abilities, but also your technical reasoning process and how you think about building robust systems that can grow and adapt over time. We encourage you to be prepared to discuss trade-offs in your implementation decisions and to think beyond the immediate requirements toward how your solution might need to evolve in a production environment serving millions of requests.

Case

ANVA processes a lot of different text documents. We are curious to find out how often a particular word is used in the text documents that we process.

Use the following definitions and assumptions for this case:

- **Word**: A word represents a sequence of 1 or more characters (a-z A-Z). For example: kjsHKDieh or Insurance;
- **Input Text**: The input text contains words which are separated by different types of separation characters;
- **Available memory**: There is sufficient memory available to store the entire entered text;

- **Capitalization**: For this case, there is no distinction made between uppercase and lowercase letters. For example: the sentence "The car is the color purple." will indicate that "the" occurs 2 times.

Technical specifications

- Latest version of Java
- Latest version of Spring Boot
- Feel free to choose any test framework, the following are common in our team:
    - JUnit5
    - AssertJ
    - Mockito
    - RestAssured

Implementation

Create a class that implements the following interface:

```
public interface WordFrequency {
  String getWord();
  int getFrequency();
}
```

Create a service that implements the following interface:

```
public interface WordFrequencyAnalyzer {
  int calculateHighestFrequency(String text);
  int calculateFrequencyForWord(String text, String word);
  List<WordFrequency> calculateMostFrequentNWords(String text, int
n);
}
```

Implement the three methods defined in the above interface:

- **calculateHighestFrequency** returns the highest frequency in the text (multiple words may have this frequency);
- **calculateFrequencyForWord** returns the frequency for the specified word in the specified text;
- **calculateMostFrequentNWords** returns a list of the “n” most frequent words in the specified text, with all words returned in lowercase. If multiple words have the same frequency, they are returned in alphabetical order.

  As an example: the text "The cat walks over the staircase" with n = 3 will return the following list: {("the", 2), ("cat", 1), ("walks", 1)}

All interfaces must be exposed via a REST API.

Acceptance criteria

- All interfaces mentioned under "Implementation" are implemented;
- All interfaces are exposed via a REST API;
- Delivered functionality is provided with Unit tests;
- Delivered REST APIs are provided with automated tests.

Delivery

Deliver this case via a public git repository (GitHub/GitLab/Other). Please include a brief README documenting your approach, any assumptions you made, and instructions for running your application and tests.

Questions

If you have any questions about this case, you can submit them via email to the person who shared the case with you.

Confidentiality

Please treat the contents of this assignment as confidential; do not share it or the solution with others.