**Task 1: To search the text in the text file and return the counts of the given words**

To achieve this, I have taken a Rest controller “WordController.java” in which I declared a post request “wordSearch()”. This request calls a service “WordsService.java” which in turn calls a repository “WordRepository.java” in which there is an initializer which is being used by 2 methods namely “wordSearchCount()” and “WordsAndTheirCounts()” and call the “wordSearchCount()” method.

Initially, in the initializer, I have separated the words in the text file using delimiter and converted them to lowercase. I have taken a hash map. Here I check for two conditions:

1. If the word is not present, in this case the word is added to the hash map with count as 1.
2. Else the count of the word in the hash map increments by 1.

Then we check each word in the input list and check if it is present in the hash map:

1. If the word is present in the hash map, the count is retrieved and added to the list of maps.
2. If the word is not present in the hash map, the word and value 0 are added to the list of maps.

Then I have used a utility class “FilterUtil.java” to filter out the bean properties from being serialized using FilterProvider and PropertyFilter, converted to json values using MappingJacksonValue.

**Task 2: To retrieve the top {N} most frequently used words in the text file**

To achieve this, I have taken a Rest controller “WordController.java” in which I declared a get request “topNWordsCount()”. This request calls a service “WordsService.java” which in turn calls a repository “WordRepository.java” in which there is an initializer which is being used by 2 methods namely “wordSearchCount()” and “WordsAndTheirCounts()” and call the “WordsAndTheirCounts()” method.

Initially, in the initializer, I have separated the words in the text file using delimiter and converted them to lowercase. I have taken a hash map. Here I check for two conditions:

1. If the word is not present, in this case the word is added to the hash map with count as 1.
2. Else the count of the word in the hash map increments by 1.

Then I have checked the number of top N words requested by the user (for instance top 5 words or top 10 words). If the number of words is

1. Less than or equal to zero return null
2. Otherwise, the hash map that holds all the words and their count from the text file is being sorted (in descending order of their frequency) into a tree set using the comparator. All the sorted words from the set are appended into a Linked Hash Map until the count reaches N-1 so that only top N words and their counts are retrieved.

Then I have used a utility class “CsvUtil.java” to convert the map data to csv format and prints the words and their counts with the delimiter “|”.

In order to provide security, I have taken a java class “Security.java” where the user is being authenticated with the following credentials:

1. username: “admin”
2. password: “password”

I also enabled the basic HTTP authentication and disabled CSRF (Cross-Site Request Forgery).

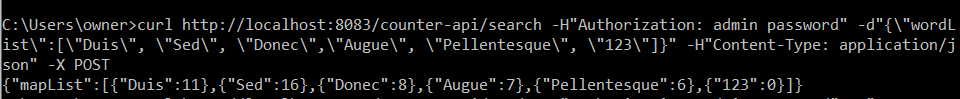
**Execution Steps:**

For executing the application, run it as a java application and give the following commands in the curl command to see the expected results. I am using localhost with port: 8083, username: admin, password: password.

**Command for Task 1:**

curl http://localhost:8083/counter-api/search -H "Authorization: admin password" -d "{\"wordList\":[\"Duis\", \"Sed\", \"Donec\",\"Augue\", \"Pellentesque\", \"123\"]}" -H "Content-Type: application/json" -X POST

**Output for Task 1:**



**Command for Task 2:**

curl http://localhost:8083/counter-api/top/5 -H "Authorization: admin password" -H "Accept: text/csv"

**Output for Task 2:**

