Feature Documentation for Lab Research application

Description: The Application is a Spring Boot Application, which has following features.

Specs and libraries: Lombok, Controller Advice, Slf4j, Custom Exception, Spring data Validator.

Steps to Run:

- You can run the application on port **8081** from either command line or from any framework using the main class **LabResearchApplication.class.** Log file(lab-research.log) is generated in same folder.
- You can run maven clean install command and run the application using jar named '*lab-research-0.0.1-SNAPSHOT.jar'* inside target folder, you can start the application with command, '*java –jar lab-research-0.0.1-SNAPSHOT.jar* '
- The application doesn't require any vm arguments.

Feature 1: Determine frequency of word occurrence in notebook Entry

- When the words are entered through the API, they are stored in a map of their frequency, in essence the map of string (word) and count (frequency).
- When queried, if the word is present, we'll return the frequency or else return 0
- Just for the viewing purpose the words are stored in another list and are returned through the notebook entry API, which can be removed if not needed.
- It was also possible to use the frequency map to display the words and reduce the space complexity, but as the words are associated with a research, they might have a meaning if placed in the same order as entered.

Feature 2: Determine similar words in notebook Entry

- We need to check the remove, replace or add character count required for a word to convert into another word
- We'll iterate through the frequency map and check for every word for similarity
- Points to check for similarity
    - If the difference between the lengths of two words is more than 2 then they aren't similar
    - Initialize Levenshtein distance variable to zero
    - Take two pointers for both words and check if the character at every pointer matches or not
    - For match: increment both the pointers
    - For mismatch:
        - Check if the Levenshtein distance variable is already 1 then return false
        - Increment the pointers as based on the word's length
        - Increment the Levenshtein distance variable
    - At last, increment the Levenshtein distance variable if the character of any word hasn't been left.
    - Return true if Levenshtein distance variable is equal to 1

Sub Feature: Add words to notebook

- If the notebook entry is new notebook, then the existing frequency map and the actual notebook word list is cleared.
- Else just put the words in the existing list and add to the frequency map

API Documentation

Root Endpoint: /notebook/api

1. Post entry to Notebook API: This API enters the words to the notebook

   Endpoint: /entry

   Http method: POST

   Sample Curl: curl -X POST \ http://localhost:8081/notebook/api/entry \ -H 'content-type: application/json' \ -d '{ "entries" : "Word Word Word word", "isNewBook" : "true" }'

   Sample Request Model:

{
"entries" : "Word Word ",
"isNewBook" : "true"
}
	entries (String): contains words to be inserted space separated
	isNewBook (Boolean): contains true if the entries are for new book

   Sample Response:

[“Word”, “Word”]	List of String : words in the notebook

   Error codes: 400 Bad Request, 200 OK (can use custom error codes)

2. Get Frequency and Similar Words: This API returns the frequency of the given word and similar words

   Endpoint: /frequency-and-similar/{requestedWord}

   Http method: GET

   Sample Curl: curl -X GET \ http://localhost:8081/notebook/api/frequency-and-similar/word

   Sample Response:

{
"frequency" : 1,
"similarWords" : [“Word”, “wor”]
}
	frequency (Long): contains the requested word’s frequency
	similarWords(List of String): contains similar words in the notebook

   Error codes: 400 Bad Request, 200 OK (can use custom error codes)

3. View Notebook: This API returns the list of words in the notebook (can be converted to space separated words as per requirement)

   Endpoint: /view

   Http method: GET

   Sample Curl: curl -X GET \ http://localhost:8081/notebook/api/view

   Sample Response:

[“Word”, “Word”]	List of String: words in the notebook

Future Enhancements:

- Swagger can be used for documentation
- Logger file appender can be configured
- Custom Error codes can be used and Global exceptional handler can be configured accordingly
- Notebook API can be enhanced in case of multiple notebooks and can use database for the same
- The calculation of frequency count can be done asynchronously using thread executor to improve API latency