**Design:**

Overview – To find the frequency of a word in a lab entry and list down similar words

**Endpoint** – “localhost:8080/labnotebook/entry/search”

**Type** – POST

**Request design**

id – To store the id of the lab notebook entry that needs to be searched

search\_text - To store the word that is to be searched

***Sample Request :***

*{*

*“id” : ” 186428a4-b5b8-4750-acea-28f531d05b40”,*

*“search\_text” : ”word”*

*}*

**Response Design**

word\_frequency – To store the frequency of the exactly matching words

similar\_words – To store the list of words which has a levenshtein distance 1 with the given word

status – To Store the success or failure status of the API

status\_code – To store the API status code

message – To Store the response message based on the success and failure status

***Sample Response :***

*{*

*“word\_frequency” : 2 ,*

*“similar\_words” : [“wor”, “Word”],*

*“status”: “Success”,*

*“status\_code” : 200 ,*

*“message” : “Word frequency and similar words have been fetched !”*

*}*

**Pseudocode**

* Get the id of the lab entry from the request body
* Find the lab entry by querying the database with the Id
* If database connection fails then handle the exception and return an Internal Server error 500 in the response
* If request body is not complete or empty throw an exception and response of status 400 and stop excecution
* Get the lab entry from the queried data
* Split the text into a list of words
* Loop through each word
* calculate the Levenshtien distance between each word from the entry and the word from the request
* if levenshtien distance equals 0 then an exact match is found and the frequency is incremented by 1
* if levenstien distance is 1 then the word is added to the list of similar word
* Return the success response with code 200

**External Libraries used**

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-text</artifactId>

<version>1.10.0</version>

</dependency>

The above dependency is added to find the levenshtien distance

LevenshteinDistance.getDefaultInstance().apply(1st word, 2nd word) // Returns the levenshtien distance after comparing the two words