I would like to start off by going over my overall process for finding a solution to the assessment. My first goal was understanding the assignment that was given, with that being said my first task was to implement a function that searches for a specific word within scanned book content. This function needed to return matching lines based on the provided search term.

The next step to understanding the problem was to identify the need to navigate through each book and its content to perform the search operation. I had to recognize the importance of matching the search term against the text content. With this in mind, I decided to use array loops like 'forEach()' to navigate through the scanned book content. I then implemented the function using iterative programming approaches to search for the given term within each line of text.

My next hurdle was figuring out how I wanted to design the given function. I started off by defining the function named 'findSearchTermInBooks', then implemented logic that would accept a search term and scanned text objects as inputs. I then needed to initialize an empty 'result' object to hold search outcomes.

Lastly, I needed to conduct a variety of unit tests to confirm the logic that I was trying to implement. The types of tests I used for this assignment was a positive test (to confirm when matches were found), negative tests (to verify scenarios when no matches were expected), and a case-sensitive test (to ensure accurate handling of case variations in the search term). The following are breakdowns of actual tests that were used in the assessment:

1. The first unit test I implemented checks if something doesn't occur or doesn't match the expected condition. To begin my test, I created a variable that would store the results returned to me from calling the function 'findSearchTermInBooks' with the parameters of "nonexistent" and 'twentyLeaguesIn' object. Next, I created a conditional statement (lines 124-128) that will check if the length of the 'Results' array is equal to 0. This will confirm that no matching results were found for the search term "nonexistent." Following this step in logging the test results, if the condition is true or no results are found I will see a message indicating that no results were found for the "non-existent" term. Otherwise, if results are found, I will receive a message along with the expected and received results counts for further debugging opportunities.

2. For my second unit test I decided to test a scenario that would check for case sensitivities. To start, I invoked the function 'findSearchTermInBooks' with the search term "The" and the book object 'twentyLeaguesIn' as parameters. The function will run a search based on that search term and book content. Lines 131-137 evaluate the result returned by the function 'findSearchTermInBooks'. If the length of the 'Results' array in the returned object is zero, this would indicate that no matches were found for the case-sensitive search term "The", and the function would print out 'PASS'. Alternatively, if there was a result that got returned, the test would output 'FAIL' along with details about the expected and received number of results for further debugging purposes.

3. For the third unit test I wanted to create a scenario that would compare the title of a book with the search term used. The overview for this test compares the length of the 'Results' array and checks if the first object in both 'Results' arrays ('expected' and 'received') match by ISBN, page, and line numbers. To breakdown the code I am using 'twentyLeaguesIn' as an example of scanned content from the book "Twenty Thousand Leagues Under the Sea" and the example output object 'twentyLeaguesOut' that represents the expected output object after searching for the term 'the' within that book content. Line 138 implements the 'findSearchTermInBooks' function with the

search term “the” and the scanned content of the book. In lines 140-159 I am comparing the received results from the search function with the expected results (‘twentyLeaguesOut’). I start by checking that the length of the results array match and then I compare those details to the first object in each result array. If the lengths match and the object details match, the unit test will pass. If the lengths don’t match or the object details don’t match, it will log a fail message along with the expected and received details for further debugging opportunities.

I am really proud of getting back into JavaScript. Before starting the test, I spent time refreshing my memory on how to write code in JavaScript. The steps I took to prepare myself for this challenge were simple ones. I went online and started watching tutorials that went over the best practices, proper code construction, and basic problem-solving techniques. In addition to watching videos, I worked on a few exercises that required me to use conditional statements as well as different types of loops to solve the problems. I am grateful for the chance to challenge myself and by doing so I was able to accomplish a goal. The trickiest part was navigating through the scanned text to find the right matches for the search term. It required careful looping through each piece of text to check for matches accurately. Once I felt confident that the logic was sound, I was able to use that understanding to finish the rest of the assessment.