U.S. DIGITAL CORPS PROJECT ASSESSMENT

DOCUMENTATION/EXPLANATION

JACOB HEWITSON

I appreciate your consideration of my application. I am very excited about the prospect of working with the US Digital Corps and hope you will find this project satisfactory. I will provide the required documentation and explanation for my thought process below.

Although it was not part of the project requirements, I created a UI for the app so that the assessor would have an easier time running the app. Even so, I ensured that the Javascript file logged all results to the console to ensure the project meets the requirements. The files should be sent in this same zip file, but for reference the app is live at *https://hewitson-j.github.io/hewitson-j-jacobhewitson-usdc-2024/*.

- Process/Decision-Making:
    1. The requirement of this app was to make a Javascript application which would use a function use a search term or phrase to search for a matching ISBN, page, and line of a book. While it was not required, I felt it would be best to also create a simple UI where a user would be able to put in their search term and which would output the results of their search to the screen. I felt this would best reflect an actual use case for our organization's users, but this can be easily removed and the Javascript code modified if necessary.

       I started by creating the test data based on our real data to work off and ensure the function would be created correctly. Each book entry in the test data consists of a title, ISBN, and content. Content consists of zero, one, or multiple entries containing page numbers, line numbers, and the text of the line. The test data

       Using this information I created a function *findSearchTermInBooks* which accepts the test data as well as the search term. The function starts by creating a results object which contains the initial search term and matching results. I felt it was best so that the later iterative loops would be able to have somewhere to store the information. Once I had the structure set up for the results, I created a nested loop, where each book entry would be iterated through checking each of its content entries for possible matches. If there was a match, the book's ISBN and the content entry's page and line numbers would be created as an entry and appended to the matching results array in the results object. If there was no match, then the loop would pass onto the next book and its contents until all

items had been iterated through.

Once the iteration ended, the results would be logged to the browser's console as per the project requirements. I included DOM Manipulators so that with the additional UI users would be able to interact with the search bar and the results would also appear on the screen.

I created several cases to test the function, including positive cases, negative cases, case-sensitive cases, and full phrases. I also set up the search bar in the UI to not accept blank search queries or queries with only whitespace, so coverage for these cases was already taken care of.

2. Testing and Iteration

1. My strategy for writing tests was to think of all the different ways a user would use the query function. As this is a string function, there was no need to prohibit special characters as they might exist in the book's entries. I was able to think of the following cases:

   - A user inputs a term found in one or multiple of the entries.
   - A user inputs a term not found in any of the entries.
   - A user inputs a term found in one or more entries but the entries are case-sensitive.
   - A user inputs a full phrase instead of just a word.
   - A user inputs all whitespace.
   - A user leaves the input blank.

   I wanted to provide coverage for all of these cases. Since I also created a UI for this project, I made it so that input couldn't be left blank or with only spaces. Once I ensured this, I followed the project requirements in creating unit tests to provide coverage for the remaining four cases.

   Given more time I would have improved my test suite by using a testing library such as Jest or Mocha to automate these tests. I would have also included the tests in another file to modularize my code and avoid confusion.
2. Of the app as a whole, I'm most proud of that the user is able to input any string and see the results of their search in a user-friendly way. Of the logic itself I'm most proud of the nested loop I used in the function. Usually, one would have to use a for (let I = 0; I < arr1.length; I++) loop and then add another nested loop and iterate through each object which could end up being confusing and verbose. I feel that using a for-of loop

made it a lot more understandable and drastically reduced the number of lines I would have had to use the other way.

3. In truth, I didn't feel the problem was overly difficult. I would say the hardest part was making sure I was able to pull the data from both the book and the content of the book and making sure it showed correctly as stated in the project requirements.
4. Even though it was not necessary I decided that both for me and a potential assessor it would be more clear and easier to understand if I created a simple UI. In the search bar I made sure that before a user could search they would need to make sure the search wasn't blank and there wasn't just whitespace. That took away the need of me to create tests for these edge cases. Apart from this, I was able to think of the edge cases from 2-1 and make sure that there would be appropriate results for a found term, a not-found term, a found term in a different case, and a full phrase.