Thursday, December 7, 2023

Dear United States Digital Corps,

In this memorandum I describe my thought process for developing our 2024 Software Engineering Project-Based Assessment. I designed, implemented, and tested a function that searches for a word in the scanned content of a number of books. I designed our function by developing the below System Description, System Diagram, Use-Case Description, and Design Of Interface. I implemented our function by developing the below Flow Chart and writing JavaScript in `book_search.js`. I tested our function by describing my strategy for writing tests below and writing tests in `book_search.js`.

System Description for *Function That Searches For A Word In An Array Of Excerpts Of Books*

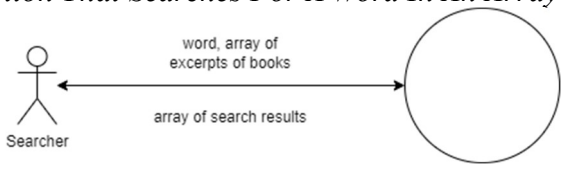
Context
Electronic books are valuable in part because they are searchable. Kindle For PC is a software application that allows users to read electronic books and search electronic books for words.

Opportunity That Function Will Address
Our function may be used by a new system that allows users to read electronic books and search electronic books for words.

What Function Will Do
Our function will receive from a searcher a word and an array of excerpts of books. Our function will provide an array of search results.

Iterations Of Development
At the end of a first iteration, our function will be as described in *What Function Will Do* and *Design Of Interface* below.

System Diagram for *Function That Searches For A Word In An Array Of Excerpts Of Books*



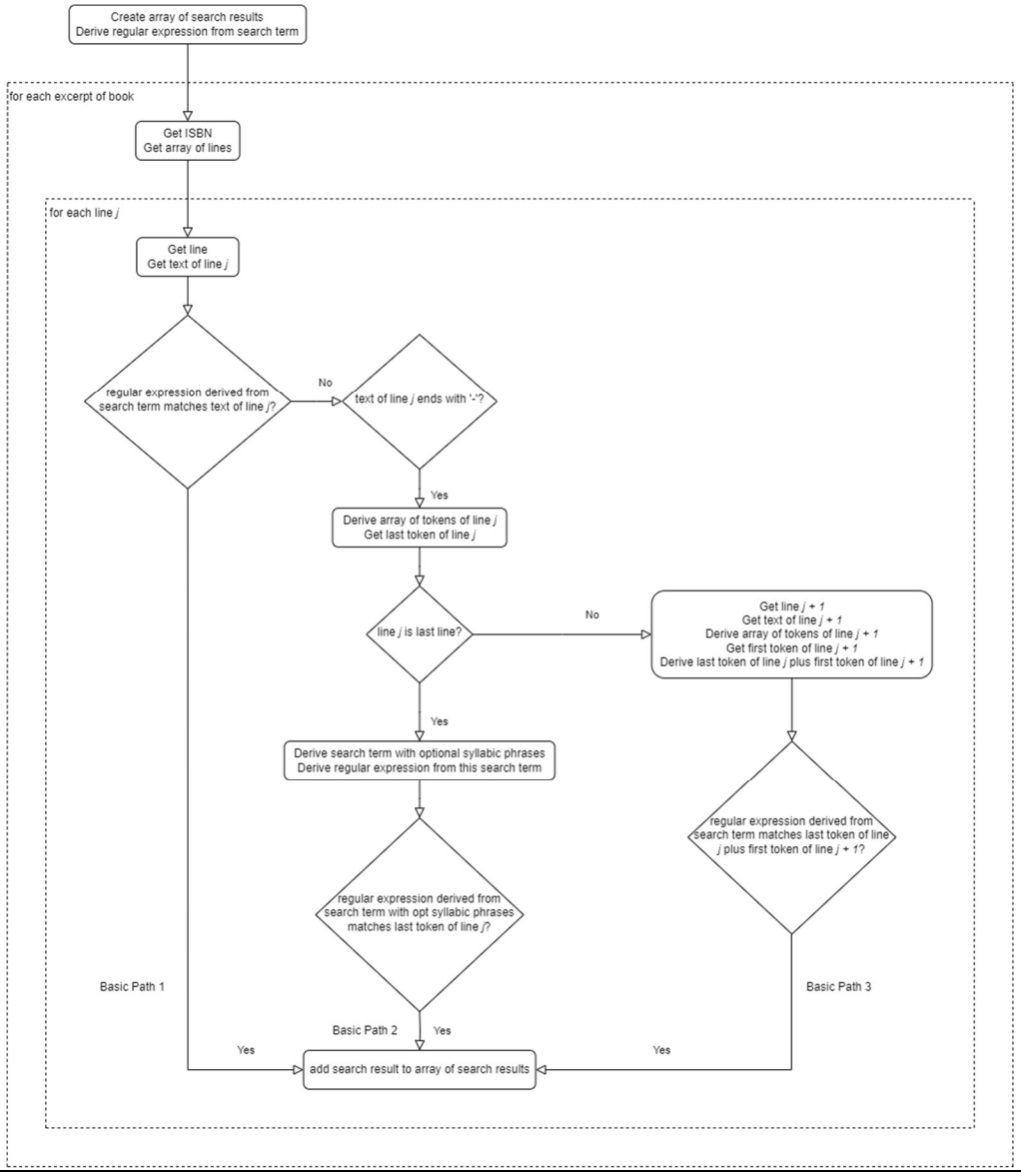
Use-Case Description for *Searcher Requests Search Results*

Searcher Requests Search Results is a use case related to our function. In this use case, a searcher provides a word and an array of excerpts of books and receives an array of search results. See below interface between a searcher and our function.

Design Of Interface between *Searcher And*
First Iteration Of Function That Searches For A Word In An Array Of Excerpts Of Books

Our function will receive a word. A word contains one or more letters and contains either an optional / soft hyphen or a required hyphen after each syllable. A word may contain one or more non-adjacent apostrophes.
Our function will receive an array of excerpts of books. An excerpt of books contains a title, an International Standard Book Number (ISBN), and an array of lines. A line contains the number of the page of the line in a book, the number of the line on the page with the line, and the text of the line.
Our function will provide an array of search results. A search result contains an ISBN, a number of a page with the line indicated by a number of a line, and the number of a line with text containing the provided word or the beginning of the word. If a word is split by a hyphen across two lines, a search result for the former line will be provided.

Flow Chart for
First Iteration Of Function That Searches For A Word In An Array Of Excerpts Of Books



As illustrated in our Flow Chart above, our function has three basic paths. In all paths, an array of search results is created and a regular expression is derived from a search term. For each excerpt of a book, the ISBN of the book and an array of lines is gotten. For each line *j*, a line and the text of the line are gotten and the following events occur. In path 1, if the regular expression derived from the search term matches the text of line *j*, a search result is added to our array of search results. In paths 2 and 3, if the text of line *j* ends with a hyphen, an array of tokens of line *j* is derived, the last token of line *j* is gotten, and whether line *j* is the last line is determined. In path 2, if line *j* is the last line, a regular expression is derived from a version of our search term with optional syllabic phrases. If this regular expression [e.g., `mo-?(ther-(in-(law)?)?)?`] matches the last token of line *j* (e.g., `mo-` or `mother-` or `mother-in-`), a search result is added and a warning is logged. In path 3, if line *j* is not the last line, line *j + 1* and the text of line *j + 1* are gotten, an array of tokens of line *j + 1* is derived, the first token of line *j + 1* is gotten, and the concatenation of the last token of line *j* and the first token of line *j + 1* is derived. If the regular expression derived from our search matches the concatenation, a search result is added.

In our test suite, I defined multiple input arrays of excerpts of books, defined multiple expected output results, and comparing expected output results to the outputs of our function. I performed basic-path testing of our function. Basic-path testing is testing in which each instruction in tested software is executed at least once. Tests 1, 2, 8, and 11 test path 1. Tests 4, 6, 13, 14, and 15 test path 2. Tests 3, 5, and 7 test path 3. Tests 9 and 10 test the root path. Test 12 tests a subfunction that creates a version of our search term with optional syllabic phrases. From another perspective, I wrote positive tests that returned matches, negative tests that did not return matches, and case-sensitive tests that returned different matches based on whether our search term was capitalized. Tests 1, 2, 3, 4, 6, 7, 8, 11, 13, 14, and 15 are positive. Tests 5, 9, and 10 are negative. Case sensitivity is tested especially by tests 1 and 11. From another perspective, I followed the Pareto principle that states that results can often be increased most rapidly by applying efforts in a concentrated area. Tests 12, 13, 14, and 15 test our subfunction and use of this function. See the Chrome developer console for detailed descriptions of tests.

If I were given more time to work on this problem, I would seek to make the test suite more robust by adding tests of the behavior of our function that searches for a word in an array of excerpts of books when objects deviating from our definitions of word, array of excerpts of books, excerpt of book, and line are provided. I would add a test of our function when a book contains no lines. I would add tests of our subfunction for words with different combinations of syllables, hyphens, and optional hyphens and for non-word strings.

I am grateful extremely for this opportunity to apply to serve the U.S. Digital Corps and your agencies as a Software Engineer. I was most challenged in developing path 2 due to the difficulty of matching a derivative of a word to a syllabic phrase of that word at the end of an excerpt when both may involve hyphens and optional hyphens. I am most proud of following a relatively full engineering process in this document and in coding and testing.

Best regards,

Tom Lever