Nlyte Software Coding Task: Library

- **Role: Senior Full Stack Developer**

To assist us in understanding the technical experience of candidates, we ask that you *fill in the blanks* of the simple WebAPI and ES6 project that accompanies this document. This will involve creating the architecture and implementation of both Frontend and Backend projects.

The WebAPI project uses .NET Framework and C# and iisexpress to minimise dependencies on your software environment. The UI code is written using ES6, to avoid the need to compile Typescript.

Please do not add any further external dependencies to your solution.

- The overall purpose of the BackEnd application is as follows:

    1. To expose, via WebAPI, the titles of a number of books, as “GET /api/books”.

        The books are stored as UTF8 text files in the Resources folder.

    2. To allow a caller to retrieve the most common 10 words within an individual book.

        The words are returned as a list of strings, from “GET /api/books/{id}”.

        A word is defined as a sequence of non-whitespace, non-punctuation, non-special characters, we discard words shorter than 5 letters.

        Case-matching is insensitive, and the top 10 words should be returned in Capital Case (e.g. “Word”)

    3. The caller should then be able to send a string (min 3 characters), and receive back a list of all words which begin with that string from the specified book, e.g. “GET /api/books/1?query=Wha”.

        Case-matching is insensitive.

    The logic encapsulated by these calls should be supported and verified by at least one unit test.

    NOTE: Candidates should prioritise good architecture, then performance over memory. Where possible, data should be ‘lazy loaded’ once, then retained for future requests.

- The UI calls the WebAPI API detailed above:

    1. To retrieve the book titles, these are displayed to the user as HTML DOM elements added to the document tree.

    2. When the user clicks on a book title, we retrieve the most common 10 words in that book and display them, and the count of those words’ usage.

    3. The user can also type in a string (min 3 characters) into a search box. The UI will request all words which start with that prefix and display those in the same manner as the “top 10” above.

    The candidate should provide their own HTML and CSS structure for the UI, but please avoid adding more dependencies or image assets. We have provided an index.html and an app.js file as a starting point, though expect candidates to add further classes as required by their architecture.

The code you send will be assessed on architecture, clean coding and how you’ve approached what you’ve attempted. Obviously, the more that a candidate attempts, the more we have to judge on, but we are aware that many candidates may have limited time, and we would prefer a few things done well, than a larger number done badly. Please mention any known issues in your mail.

Please also consider what further improvements and enhancements you would have made to the solution if time was not a factor. This, and any questions arising from your code sample will provide the starting point for any further interview.