Search Engine Challenge:

General Instructions:

- **Please use a github repository to submit your code**
- **Use smaller commits with proper messages showing how you proceeded with the development**
- **There are two tasks in this challenge. Submitted repository should have solutions to both tasks clearly distinguishable.**
- **Task specific expectations and notes are outlined below each task description. Please read them carefully before submitting.**
- **Please do not use libraries for search related functionality, auto complete behaviour or style management**

1. Write a utility to search summaries (~ 60 min)

Unibuddy wants to build a service that allows students to search through coursebooks summaries which would make picking and buying a coursebook, a much better experience for students.

First we need to develop a local version of the system. Our bot scraped a website with book summaries, and stored them in *data.json* in the *summaries* array. The summaries array is a small data example for local development. You should assume that the real service will have ~10^6 summaries.

Your goal is to code a search utility function/class that given a search query, searches the book summaries and returns the *K* most relevant ones. A search engine query is the set of keywords that users will type in order to find a relevant document. You are allowed to use only basic language (python/javascript) functionality.

The api of the search engine should be as follows:

```
Input: The input should be a user query of type string and number of
items to return
           query (string): eg. 'is your problems'
           K (integer): eg. 3

Output: List of K relevant summaries sorted according to order of
```

```
relevance given a query. A summary is a dictionary that follows the
schema: {'summary': string, 'id': integer}
            summaries: eg. [
                  {'summary':'The Book in Three Sentences: Practicing
meditation and mindfulness will make you at least 10 percent
happier....', 'id':0},
                  {'summary':'The Book in Three Sentences: Finding
something important and meaningful in your life is the most productive
use of...', 'id':48},
                  {'summary':'The Book in Three Sentences: Everything in
life is an invention. If you choose to look at your life in a new
way...', 'id':7}
            ]
```

Expectations for the task:

1. Readable, testable, modular and well-commented code
2. Use of proper data structures for making your code reusable, optimal and efficient
3. Optimal and/or accurate search results
4. Unit-tests with required mocks
5. Clarity on challenges you faced in your implementation and thoughts on areas of your code that you would go back to and improve on

NOTE 1: We are not making a server or a frontend component at this stage
NOTE 2: Relevance of match is to be defined by candidate. Can be percentage match and/or number of instances of partial match etc.

2. Write a frontend application to find and select books (~ 60 min)

We need to make the search functionality available as a frontend component, so users can find coursebooks using keywords from summaries.

Use the previously built search utility for searching summaries, and authors and titles of all books from *data.json*, to make a **responsive ReactJS SPA**.

The SPA will have two parts:

1. **Form**: A form containing an autocomplete. This autocomplete would take "*input search string*", and "*number of suggestions to display*", as as mandatory arguments (corresponding to the arguments of the search engine made previously).
When the user types an input string into the input part of the autocomplete component, a list of suggested **book titles** is displayed below. Calculating the suggested summaries would be done using the **search util** made earlier. **Titles** for these calculated summary suggestions can be found in the *data.json*. These **titles** are used to **display** the suggestions in the autocomplete as the user looks for a search string.
When the user **selects a title** from the suggestions, and submits the form, form is cleared and selected book is **appended** to a list below the form.

2. **List**: A list of cards displaying the books selected by the user. Each card should display:
    - Title (available in *data.json*)
    - Author (available in *data.json*)
    - Short Summary (substring of the summary ending in "...")

Preview the sample output here

Expectations for the task:

1. Well defined structure to your front end components
2. Thorough Unit-tests added with proper mocks
3. Accessibility in styles, responsive design and overall good looking auto-complete behaviour
4. Clarity on challenges you faced in your implementation and thoughts on areas of your code that you would go back to and improve on

NOTE 1: There is a 1:1 match between titles and summary objects, ie. the first title would correspond to the first summary, second title corresponds to the second summary and so on.

NOTE 2: There is a 1:1 match between authors and summary objects, ie. the first author would correspond to the first summary, second author corresponds to the second summary and so on.

NOTE 3: Please do not use any design framework like Material-UI or bootstrap for styling

NOTE 4: Example output is only base level expectation in terms of design. Please improve on that.