Instructions:

There are two different assignments – JSON-Merge and MovieWiki. If you scroll further, you will be able to read the problem statements for each. There are both functional and non-functional requirements listed out in each.

As a bare minimum, we expect the functional and non-functional requirements to be met. Any additional features on top of that will fetch you additional bonus points.

You are free to select either one – or both, as per your interest. This assignment should not take more than a couple of hours. We encourage you to submit it at the earliest.

Submission:

- Submit a link to the source code, ideally committed to GitHub.
- Code accompanied by thorough unit tests is typically a mark of quality work.
- Ideally, your utility will work on most operating systems. If this is not the case, please specify which OSes are supported.

JSON-Merge

Write a program that can merge a series of files containing JSON array of Objects into a single file containing one JSON object.

Example:

Suppose there are 3 files - data1.json, data2.json, data3.json.

Let's say data1.json contains -

```
{"strikers": [
      { "name": "Alexis Sanchez", "club": "Manchester United" },
      { "name": "Robin van Persie", "club": "Feyenoord" }
] }
```

data2.json contains -

```
{"strikers": [
      { "name": "Nicolas Pepe", "club": "Arsenal" }
] }
```

data3.json contains -

```
{"strikers": [
      { "name": "Gonzalo Higuain", "club": "Napoli" },
      { "name": "Sunil Chettri", "club": "Bengaluru FC" }
] }
```

A merge of these 3 files will generate a file with the following data.
merge1.json -

```
{"strikers": [
      { "name": "Alexis Sanchez", "club": "Manchester United" },
      { "name": "Robin van Persie", "club": "Feyenoord" },
      { "name": "Nicolas Pepe", "club": "Arsenal" },
      { "name": "Gonzalo Higuain", "club": "Napoli" },
      { "name": "Sunil Chettri", "club": "Bengaluru FC" }
] }
```

Functional requirements:

1. Accept the following parameters as an input –
   a. **Folder Path:** The folder where all the JSON files are stored.
   b. **Input File Base Name:** The common prefix all file names share. In our example above, this prefix was *data*.
   c. **Output File Base Name:** The prefix for the merged file name generated by the merge utility.
   d. **Max File Size:** The maximum file size (in bytes) that each merged file is allowed to be.
2. The utility will read all files in the Folder Path that begin with the Input File Base Name, and process them in increasing order of the number added as a suffix to each file (1,2,3,....,12,13,...).
3. The utility will ensure that the output files are named using the Output File Base Name as a prefix, and a counter as a suffix.
4. Merged files will never be greater than Max File Size.

5. Each output file will contain a proper JSON array.
6. Bonus points for making the solution generic so any kind of JSON array can be merged. (e.g.: The root key "strikers" becomes "employees". The objects in the array carry fields like "name", "id", "designation", etc.)
7. Bonus points for supporting non-English characters.

Non-functional requirements:

1. The algorithmic complexity of the merge will be as small as possible. Please capture the algorithmic complexity of your solution in the README file.
2. The merged files will be as large as possible, without exceeding Max File Size.

Languages:

Ideally, we would not restrict you from working on a language of your choice. However, it would be preferable if you stick with one of these:

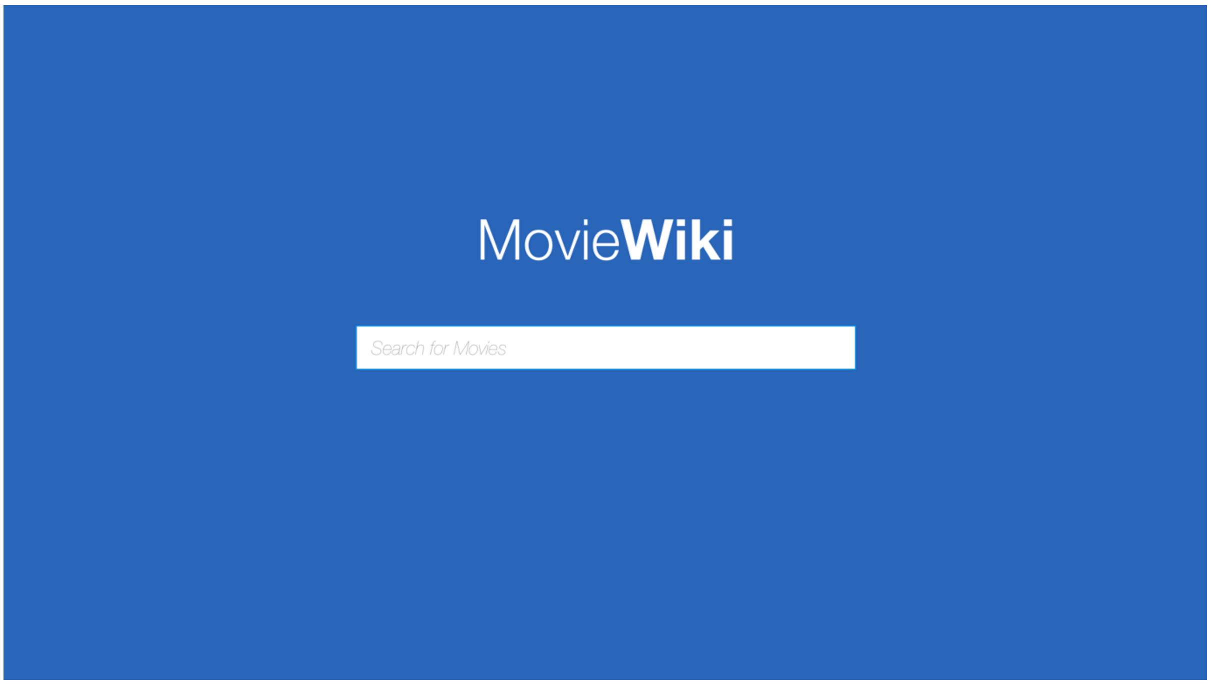
- NodeJS
- Java
- Python
- GoLang
- Ruby
- C/C++

MovieWiki

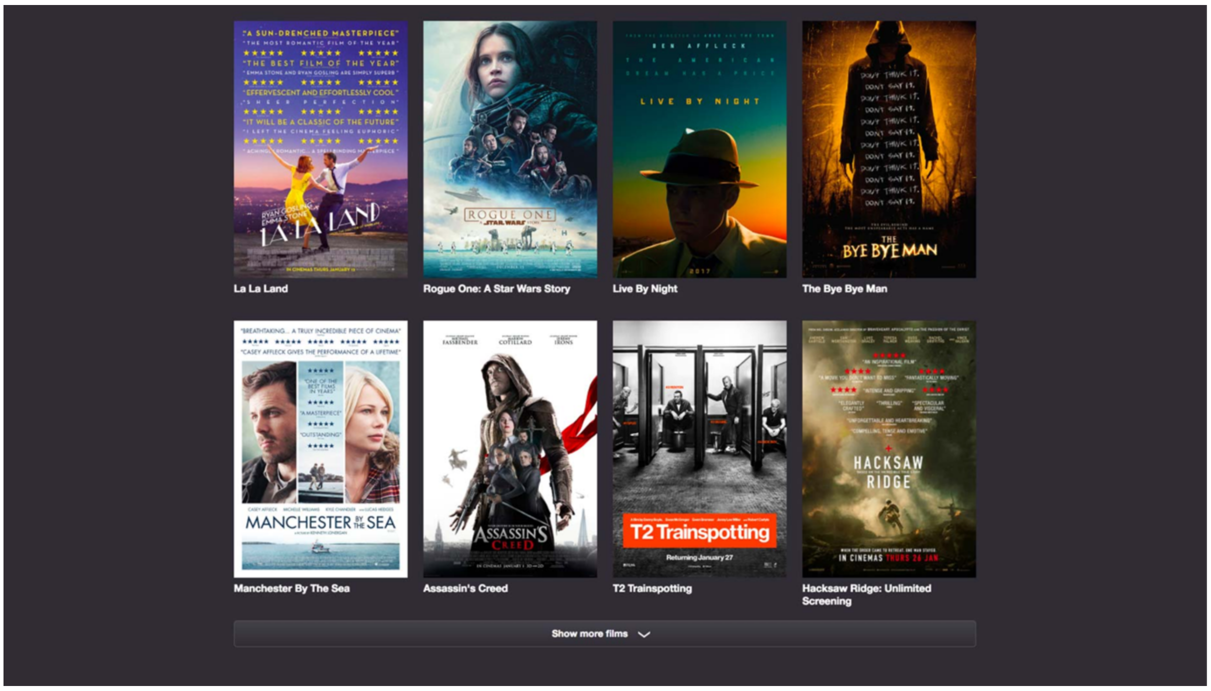
Build a web application to enable users to search for any movie and view details about it.

Functional requirements:

1. The first page should have a search bar to search for any movie. Use APIs from http://www.omdbapi.com/.



2. On search, it should list the movies that match the search query in a grid form with posters. This page will be paginated. A *show more* button will be visible to show more results if present.



3. On clicking a movie poster, one should be redirected to a movie information page to show details about the movie. This will also contain the full plot. You can be creative in designing this page.
4. The details page should have a link to go back to the previous page, where the search results were displayed thus storing the previous state.

Non-functional requirements:

1. The web app should be written in HTML, JS, and CSS. We would like you to build the app **without** using any web frameworks (e.g, React, Angular, Vue, Jquery etc).
2. The pages should be responsive and should adapt nicely to any screen size.
3. Avoid using bootstrap. Use CSS media queries instead.
4. API calls should be made using the fetch API (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API).
5. Clean code and comments are highly appreciated.
6. Bonus points for good error handling.
7. Bonus points for adding interesting features as well.