**Movie App**

Write an application that has 3 pages total:

1. Page One is the Home page thats shows on app launch (localhost:4200/):
   1. This is the starting page the user sees in the browser.
   2. It says the name of the app and also what the app does below it.
   3. Has a link/button to navigate to Search Movies Page (see below).
2. Page Two is Search Movies (localhost:4200/search):
   1. Has a text box and search button to search for movies.
   2. Using entered text and clicking button, it fetches movies JSON results (it will be first 10) via a web call to omdbapi.com (Open Movie Database).
      1. REST GET call to search for movies by title: “http://www.omdbapi.com/?apikey=91c3e55a&s=some movie search here”.
   3. Display each movie result in a list on page after call is done.
      1. Each movie result shows title, plot, and poster.
      2. There is a link or button on each movie result that will navigate to the Movie Details Page and pass the imdbID in the url (see below).
   4. NEW Show the read-only, user rating for each movie result.
   5. NEW Add pagination to the search results to see the next page of 10, if any.
3. Page Three is a Movie Details (localhost:4200/movie/{id}):
   1. This page shows the information for a single movie.
   2. Will fetch a single movie JSON result from “<http://www.omdbapi.com>” using
   3. Styled how you want, show on page:
      1. Title
      2. imdbID
      3. released
      4. runtime
      5. genre
      6. Metascore
      7. imdbRating
      8. plot
      9. poster
      10. NEW User Rating
   4. NEW Page has an input on the page to give it a user rating.
      1. Will allow the user to give a rating of 1 to 10, only integers.
      2. The ratings will be updated and stored in a new RatingService class which will store them in memory.
4. NEW Make all buttons use Bootstrap 3 styling (via ngx-bootstrap preferably).

Code Design Requirements/T:

* There should be one main Angular Module for the startup (AppModule).
* There should be 3 Angular Feature Modules for each Page (HomeModule, SearchModule, DetailsModule).
  + Each Feature Module should have a <feature>.routing.module.ts to map its proper app URL.
    - Home is “/”.
    - Movie Search is “/search”.
    - Movie Details is “/movie/{id}”.
* Make a MovieService class that contains two methods to make http GET calls to <http://www.omdbapi.com>.
  + One searches for movies and returns a list of Movie objects (Movie[]).
  + One gets one movie by imdbID and returns one Movie object (Movie).
  + NEW Inject RatingService into this service so that a user rating can be read and set on each Movie object inside of the method that gets one Movie.
* Make a Movie class to store all your properties to show for each movie result. Will have properties:
  + - title: string
    - imdbID: string
    - released: string
    - runtime: string
    - genre: string
    - metascore: number
    - imdbRating: string
    - plot: string
    - posterUri: string
    - NEW userRating: number
  + These properties will be set by you inside the MovieService class from the matching JSON properties after a GET call is made.
  + View the JSON response to see the structure and property names to set into your Movie class.
* For the movie search results list, use \*ngFor with a custom MovieResult Component you make to show each movie result.
  + Pass movie properties for each component inside the \*ngFor loop to show via @Input class properties in MovieResult Component.
  + NEW When the rating is changed, use an EventEmitter to send the new rating value to the Parent component which can use RatingService to set the rating for that movie.
* NEW Add additional libraries via NPM.
  + Adding both on an Angular CLI project guide here:
    - <https://github.com/valor-software/ngx-bootstrap/blob/development/docs/getting-started/ng-cli.md>
  + bootstrap 3 (style framework)
    - npm install bootstrap@3.3.7 –save
  + ngx-bootstrap (Angular version for bootstrap components)
    - npm install ngx-bootstrap –save
* NEW Use this control for the user rating input and display:
  + <https://valor-software.com/ngx-bootstrap/#/rating>
* NEW Make a RatingService class that will have two methods:
  + One to get the current rating for a movie by id (imdbID).
  + One to save a rating for a movie by id (imdbID).
  + This data will be stored temporarily in the service and will be gone when the browser is refreshed or closed.
  + This rating-per-movie info can be stored locally as a property in RatingService via an array or object for fast access for a movie id.
* NEW Use this control for pagination of search results:
  + <https://valor-software.com/ngx-bootstrap/#/pagination>
  + This control will allow the user to request a different page of (10) results for the current search term.
  + MovieService search method must use “page” query parameter in URI, given from page clicked from pagination control.
    - Ex. [http://www.omdbapi.com/?apikey=91c3e55a&s=star%20wars**&page=3**](http://www.omdbapi.com/?apikey=91c3e55a&s=star%20wars&page=3)
  + Pagination control needs passed “totalResults” property from search JSON response.

Web call response example:

Example GET Url: “http://www.omdbapi.com/?apikey=91c3e55a&s=star%20wars” returns:

{

"Search": [{

"Title": "Star Wars: Episode IV - A New Hope",

"Year": "1977",

"imdbID": "tt0076759",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BNzVlY2MwMjktM2E4OS00Y2Y3LWE3ZjctYzhkZGM3YzA1ZWM2XkEyXkFqcGdeQXVyNzkwMjQ5NzM@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: Episode V

- The Empire Strikes Back",

"Year": "1980",

"imdbID": "tt0080684",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BYmU1NDRjNDgtMzhiMi00NjZmLTg5NGItZDNiZjU5NTU4OTE0XkEyXkFqcGdeQXVyNzkwMjQ5NzM@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: Episode

VI - Return of the Jedi",

"Year": "1983",

"imdbID": "tt0086190",

"Type": "movie",

"Poster": "https://images-na.ssl-images-

amazon.com/images/M/MV5BOWZlMjFiYzgtMTUzNC00Y2IzLTk1NTMtZmNhMTczNTk0ODk1XkEyXkFqcGdeQXVyNTAyODkwOQ@@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: The

Force Awakens",

"Year": "2015",

"imdbID": "tt2488496",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BOTAzODEzNDAzMl5BMl5BanBnXkFtZTgwMDU1MTgzNzE@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: Episode I - The Phantom

Menace",

"Year": "1999",

"imdbID": "tt0120915",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BYTRhNjcwNWQtMGJmMi00NmQyLWE2YzItODVmMTdjNWI0ZDA2XkEyXkFqcGdeQXVyNTAyODkwOQ@@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: Episode

III - Revenge of the Sith",

"Year": "2005",

"imdbID": "tt0121766",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BNTc4MTc3NTQ5OF5BMl5BanBnXkFtZTcwOTg0NjI4NA@@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: Episode II - Attack of the

Clones",

"Year": "2002",

"imdbID": "tt0121765",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BOWNkZmVjODAtNTFlYy00NTQwLWJhY2UtMmFmZTkyOWJmZjZiL2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyNDUzOTQ5MjY@.\_V1\_SX300.jpg"

},

{

"Title": "Rogue One: A Star Wars Story",

"Year": "2016",

"imdbID": "tt3748528",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BMjEwMzMxODIzOV5BMl5BanBnXkFtZTgwNzg3OTAzMDI@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: The Last

Jedi",

"Year": "2017",

"imdbID": "tt2527336",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BMjQ1MzcxNjg4N15BMl5BanBnXkFtZTgwNzgwMjY4MzI@.\_V1\_SX300.jpg"

},

{

"Title": "Star Wars: The Clone

Wars",

"Year": "2008",

"imdbID": "tt1185834",

"Type": "movie",

"Poster": "https://ia.media-

imdb.com/images/M/MV5BMTI1MDIwMTczOV5BMl5BanBnXkFtZTcwNTI4MDE3MQ@@.\_V1\_SX300.jpg"

}],

"totalResults": "418",

"Response": "True"

}