

IFQ716 Advanced Web Development – Assignment1

Sulhee Choi 11513560

INTRODUCTION

In this project, a REST API has been created using Node.js. The API includes the following endpoints.

- GET/movies/search/title : Searches movie data based on the title.
- GET/movies/data/imdbID : Retrieves movie data based on the ImdbID.
- GET/posters/imdbID : Retrieves poster of movie using the ImdbID
- POST/posters/add/imdbID : Upload a new poster image for a movie using the ImdbID.

For the last GET endpoint, external movie API resources were utilised to obtain movie posters.

A client.html page has been created to demonstrate the user interface.

TECHNICAL DESCRIPTION OF THE APPLICATION

The server is built using Node.js and the client interface is demonstrated through a client.html page. The application utilises a '.env' file, where the OMDb API key and the streaming API key are stored securely. The server-side code is organised in the file, 'app.js', which serves as the main entry point for the application.

- **moviesByTitle(movieTitle)** : This function fetches movie data from the OMDb API based on a specified title from the user input and returns the data.
- **movieByImdbID(imdbID)** : This function fetches movie data from OMDb API with the specified imdbID from the user input, combines the data with streaming information from the streaming info API and returns the combined data.
- **Movie(res,input,type)** : This function handles the logic for serving movie data based on 'type' of user input. If the 'type' is a movie title, it calls moviesByTitle function and if the 'type' is an imdbID it calls movieByImdbID function. It also sets the correct response headers depending on the data result.
- **getPoster(imdbID)** : This function fetches the movies poster from the OMDb API based on a specified ImdbID and returns an 'ArrayBuffer' containing the image data if the fetch is successful.
- **addPoster(req,imdbID)** : This function parses the incoming request object containing form data, using 'formidable' module. It reads and writes the file data using 'fs' module and saves the file to the designated folder using 'path' module. The function returns a success or error message based on the output.

- **poster(req,res,input,type)** : This function handles the logic for serving movie posters based on 'type' of purpose – if the type is search, getPoster function is called and if the type is upload, addPoster function is called. It also sets the correct response headers depending on the data result.
- **Routing(req,res)** : This function determines the requested URL and method and routes incoming requests to the appropriate function based on the URL path and HTTP method.

One of the technical issues I have encountered was related to uploading poster images in the application. With the live server, the client page would get refreshed as soon as a file has been uploaded, which would result in no network information of requests or responses. This issue could be resolved either by appointing the final path of the image file to a different directory, not in the same one where the project files are, or by opening the client page through http-server.

Another technical issue I had was related to analysing the data from external API. When there is no streaming information available, the streaming info API would return either a message of 'Not found' or an empty object, which made it challenging to detect this condition straightforwardly. Also, while fetching movie poster, it would return an image of 'server error' rather than a text message when there is no poster image available. This issue was particularly challenging because the code needs to recognise it as an error condition rather than a valid response. These issues could be resolved by implementing error handling while monitoring returned data with different inputs.

REFERENCES

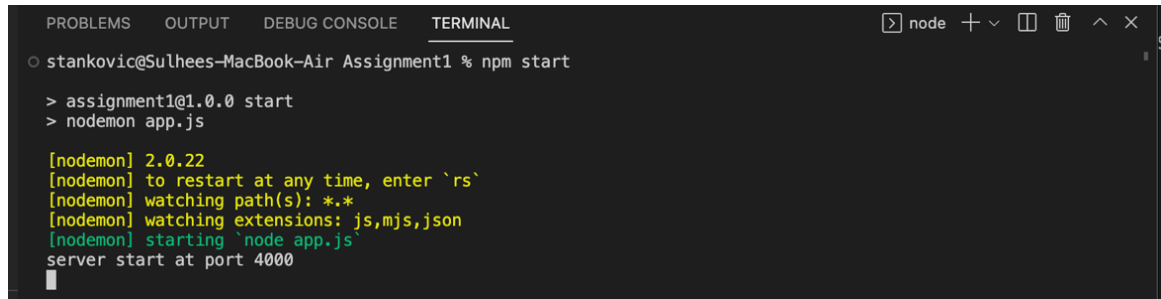
Brian Fritz. OMDb API.(2023). Retrieved from <http://www.omdbapi.com>

Movie of the Night. Streaming Availability API.(2023). Retrieved from <https://rapidapi.com/movie-of-the-night-movie-of-the-night-default/api/streaming-availability>

Npm.(2003). Retrieved from <https://www.npmjs.com/package/formidable>

APPENDIX

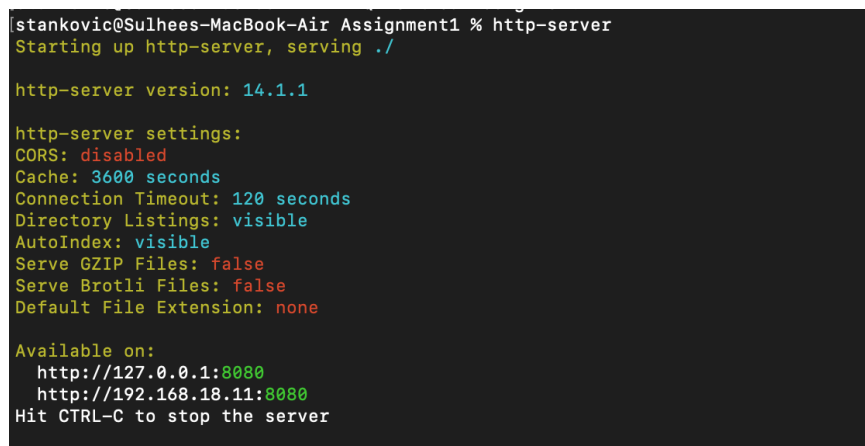
1. Open 'Assignment1' folder in VS Code and type 'npm start' in the terminal to start the server. 'server start at port 4000' should be logged in terminal.



```
stankovic@Sulhees-MacBook-Air Assignment1 % npm start
> assignment1@1.0.0 start
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
server start at port 4000
```

2. Open a new terminal and make sure you are in the 'Assignment1' directory, and type 'http-server'.



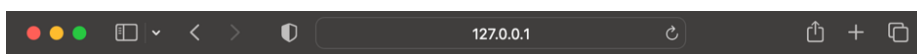
```
stankovic@Sulhees-MacBook-Air Assignment1 % http-server
Starting up http-server, serving ./

http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://127.0.0.1:8080
  http://192.168.18.11:8080
Hit CTRL-C to stop the server
```

3. Open a web browser and type <http://127.0.0.1:8080> in the address bar. Click 'client.html' link.

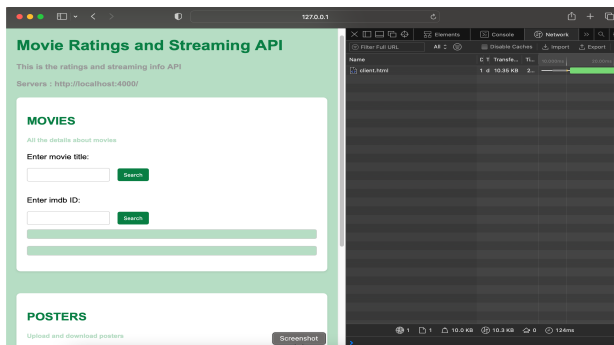


Index of /

 (drwxr-xr-x) 26-May-2023 21:07	node_modules/
 (drwxr-xr-x) 31-May-2023 12:53	posters/
 (-rw-r--r--) 27-May-2023 20:25	6.0k .DS_Store
 (-rw-r--r--) 19-May-2023 14:02	94B .env
 (-rw-r--r--) 31-May-2023 13:54	8.7k app.js
 (-rw-r--r--) 31-May-2023 12:49	9.7k client.html
 (-rw-r--r--) 26-May-2023 21:07	37.3k package-lock.json
 (-rw-r--r--) 26-May-2023 21:07	389B package.json

Node.js v18.13.0/ [http-server](#) server running @ 127.0.0.1:8080

4. Open the network panel from Develop tool to check the status code, response header and the returned data.



- Type in a movie title in the first input box in the MOVIES section, and click 'search'. The following data and the status code will be returned according to the inputs.

input	Status code	Data returned
'the dark knight'	200	Movie data
'random input'	200(valid input)	Error : movie not found!
Empty input	400	Message : You must supply a title.

- Type in an imdbID in the second input box in the MOVIES section, and click 'search'. The following data and the status code will be returned according to the inputs.

input	Status code	Data returned
tt0468569	200	Movie data and streaming info
tt0011234	403	Movie data
tttt	403(valid input)	Error : Movie not found. Invalid input.
Empty input	400	Message : You must supply a title.

- Type in an imdbID in the first input box in the POSTERS section, and click 'search'. The following data and the status code will be returned according to the inputs.

input	Status code	Data returned
tt0468569	200	Movie poster of 'the dark knight'
tttt	500 (OK)	Message : No poster image found.
Empty input	400	Message : You must supply a title.

- Type in an imdbID in the second input box in POSTERS section and click 'choose file' button to select an image file and then click 'Upload' button. The image file can be found in 'posters' folder in Assignment1 directory if upload is successful.

File name input	File input	Status code	Data returned
Any valid text	Any image.jpg	200	Message: Poster (image.jpg) uploaded successfully
Any valid text	No input	400	Message : No poster file found
Empty input	Any image.jpg	400	Message : No poster ID provided
Empty input	No input	400	Message : No poster file found and no poster ID provided

