CS2003 Practical 2 Report

24th October 2023

Tutor: Steven McQuistin

220019540

Overview

In this practical, I was tasked with creating the following:

- Creating a node.js program which will take in a CSV file of 100 IMDB movies and convert the data into a .json file.
- Coding a web application which will read in the json file and create a queue data structure, with a placeholder at the initial load of the webpage.
- Designing and implementing solutions that will allow the user to add a movie to the queue and remove it
- Adding style and changing the appearance of the webpage using CSS.

I had to make use of multiple systems and languages, including node.js to handle conversion of the file, and ensure that I create a well-designed and intuitive web page while implementing every requirement needed for the application.

Design

HTML Webpage Content

The HTML in the web application contains the layout of the page, alongside some javascript that will dynamically generate other parts of the webpage, such as the movie queue.

I decided to implement a simple, yet intuitive design, with a movie queue horizontally centered at the top of the web page displaying all the movies in the queue, and then the relevant statistics under the queue interface.

Styling and Appearance using CSS

For my webpage, I wanted a clean and simple look, so that users could clearly see the page without obstructions such as a clashing / unclear colour palette or badly placed text. I stuck to one font family, monospace, which gives a more uniform and almost 'vintage movie production' look. Colour coding the buttons to blue and red for adding and removing respectively allows the user to associate colours with the actions of the buttons.

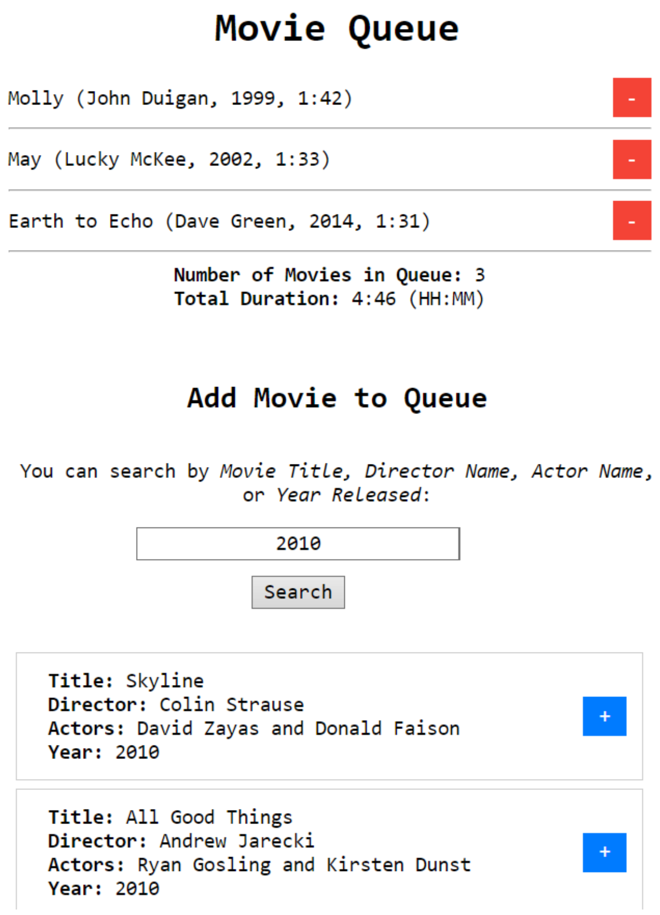
Adding and Removing Movies from the Queue

One of the biggest tasks of this practical was designing and implementing your own idea for adding a movie to the queue and removing a movie from the queue. I decided an intuitive and easy-to-understand method of adding movies from a substantial list of movies available was to implement a search function that allows users to find a movie based on the title, director, actors, or year released.

In my program search.js, I created a function that works with the searching interface in the HTML to find any movie, then display all matches in a list underneath the search box. Every single result found had a button positioned beside it, allowing users to add the movie to the queue. Each movie that was added was automatically added to the end of the queue, following the data structure rules.

For removing movies from the queue, I added the removing counterpart of the add button from the search results beside every item found in the movie queue. This made it very simple for users to take out any movie from the queue without having to perform any difficult operations or entering any text.

Here is an example of my search result interface alongside a movie queue, displaying both the adding and removing buttons:



Movie Queue

Molly (John Duigan, 1999, 1:42) -

May (Lucky McKee, 2002, 1:33) -

Earth to Echo (Dave Green, 2014, 1:31) -

Number of Movies in Queue: 3
Total Duration: 4:46 (HH:MM)

Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name,* or *Year Released:*

2010

Search

Title: Skyline
Director: Colin Strause
Actors: David Zayas and Donald Faison
Year: 2010 +

Title: All Good Things
Director: Andrew Jarecki
Actors: Ryan Gosling and Kirsten Dunst
Year: 2010 +

Figure 1: The placeholder queue and search results, including buttons to remove and add movies

Implementation

I had to separate my code into different parts to handle individual tasks and modules within the web application:

- Convert.js: the node.js program which converts the CSV file into a readable JSON file.
- Queue.js: Handling the queue data structure and dynamically generating the queue interface on the webpage.
- Search.js: my own design and implementation to allow users to add movies to the queue: creating an interface which allows users to search up movies and add them to the queue.

Converting the CSV file to a JSON file

In the program convert.js, I created a method which can take in the file path of any csv file, and then convert the information into a JSON file.

Creating the movie queue using JSON

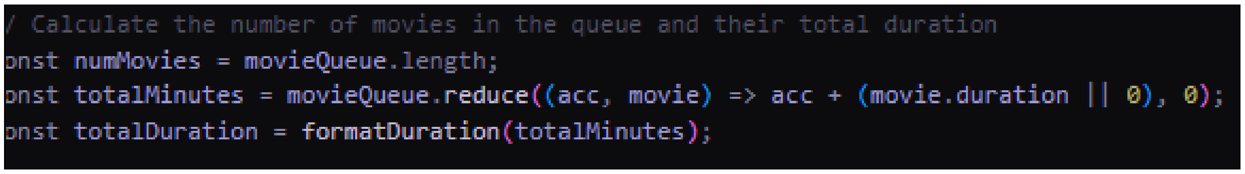
I used the http fetch protocol to get the file information from the CSV file and implement it to the program. This only works when the website is served on a web server.

Creating the Movie Queue interface

To make the webpage look complete when it's loaded, I used the first 3 movies in the movie data to include in the placeholder movie queue that appears when the user sees the page for the first time.

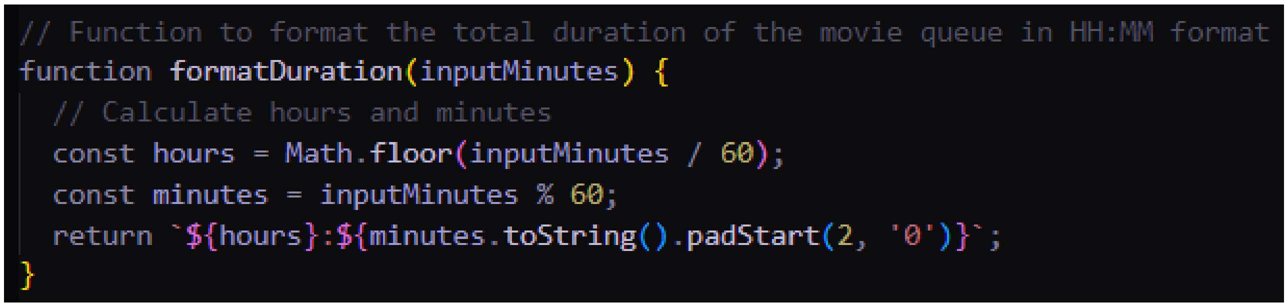
Alongside displaying the dynamically-generated movie queue using javascript, I also had to include statistics pertaining to the queue. I had to display the total number of movies and the total duration of the entire queue, formatted into hours and minutes.

For this to work, I used an accumulator method to get the total amount of minutes in the whole queue (Figure 1). I then created a function which takes in the number of minutes and formats it into a HH:MM output, which is then sent to be dynamically generated on the webpage (Figure 3).



```
/ Calculate the number of movies in the queue and their total duration
onst numMovies = movieQueue.length;
onst totalMinutes = movieQueue.reduce((acc, movie) => acc + (movie.duration || 0), 0);
onst totalDuration = formatDuration(totalMinutes);
```

Figure 2: Using the accumulator method to find the total duration of the queue in minutes



```
// Function to format the total duration of the movie queue in HH:MM format
function formatDuration(inputMinutes) {
  // Calculate hours and minutes
  const hours = Math.floor(inputMinutes / 60);
  const minutes = inputMinutes % 60;
  return `${hours}:${minutes.toString().padStart(2, '0')}`;
}
```

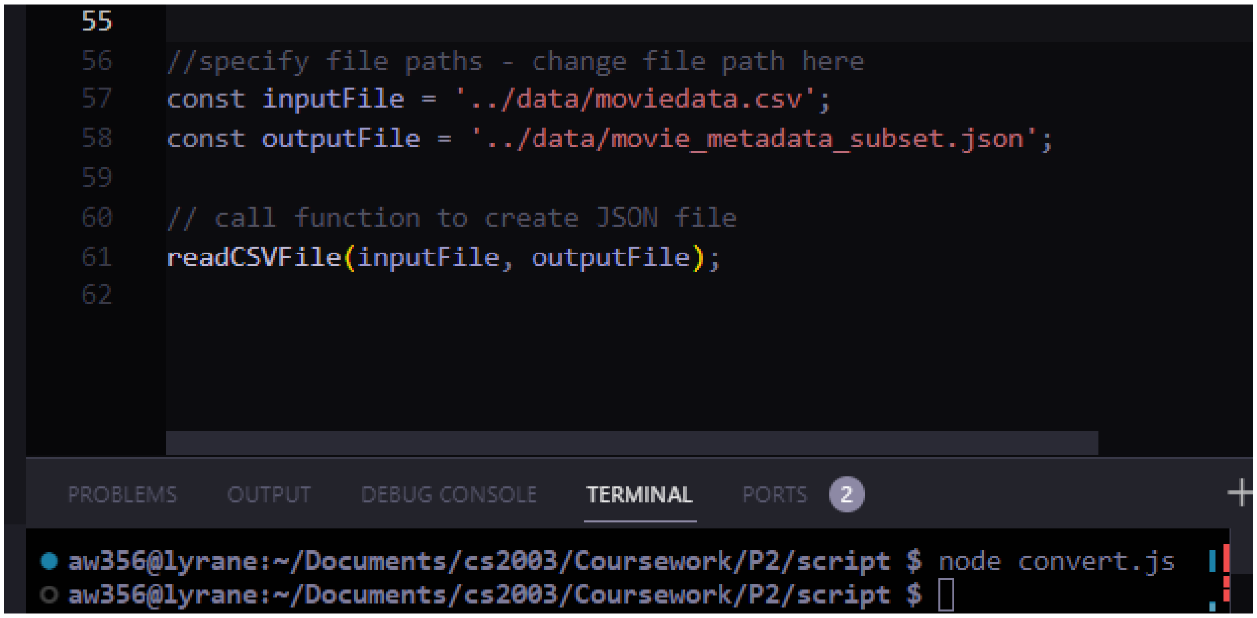
Figure 3: Function to format the total minutes into HH:MM

Testing

I had to rigorously test the functions of my entire web application to ensure that everything works as appropriate. I prepared a few test cases for each javascript program:

Convert.js

Test Case	Pre-Conditions	Expected Outcome	Actual Outcome
Using a different file to convert	Changing the file path to a different csv file	The csv file will be formatted into JSON and outputted correctly.	The file was successfully converted (Figure 4)



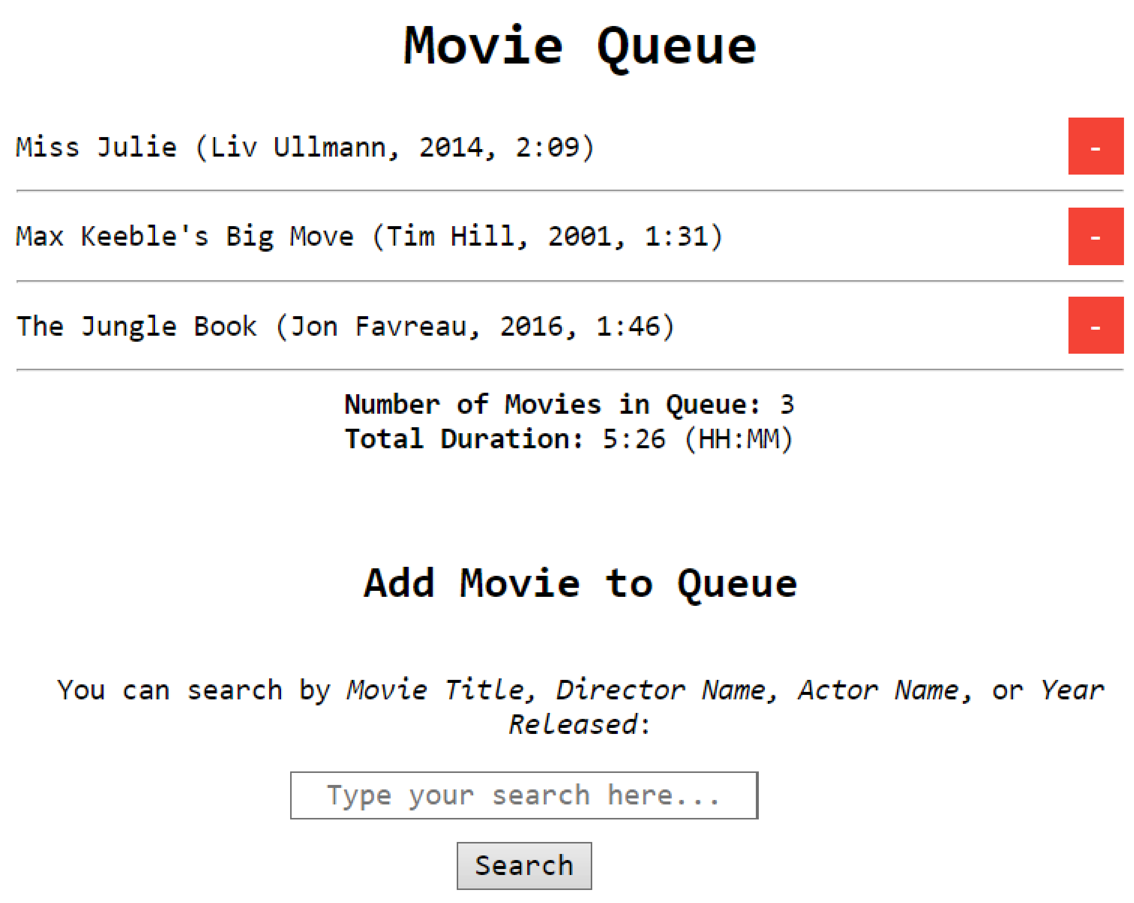
```

//specify file paths - change file path here
const inputFile = '../data/moviedata.csv';
const outputFile = '../data/movie_metadata_subset.json';

// call function to create JSON file
readCSVFile(inputFile, outputFile);

```

```
● aw356@lyrane:~/Documents/cs2003/Coursework/P2/script $ node convert.js
○ aw356@lyrane:~/Documents/cs2003/Coursework/P2/script $ 
```


Movie Queue

Miss Julie (Liv Ullmann, 2014, 2:09)

Max Keeble's Big Move (Tim Hill, 2001, 1:31)

The Jungle Book (Jon Favreau, 2016, 1:46)

Number of Movies in Queue: 3
Total Duration: 5:26 (HH:MM)

Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name,* or *Year Released*:

Figure 4: I changed the csv file to have different movies at the top, and changed the file name

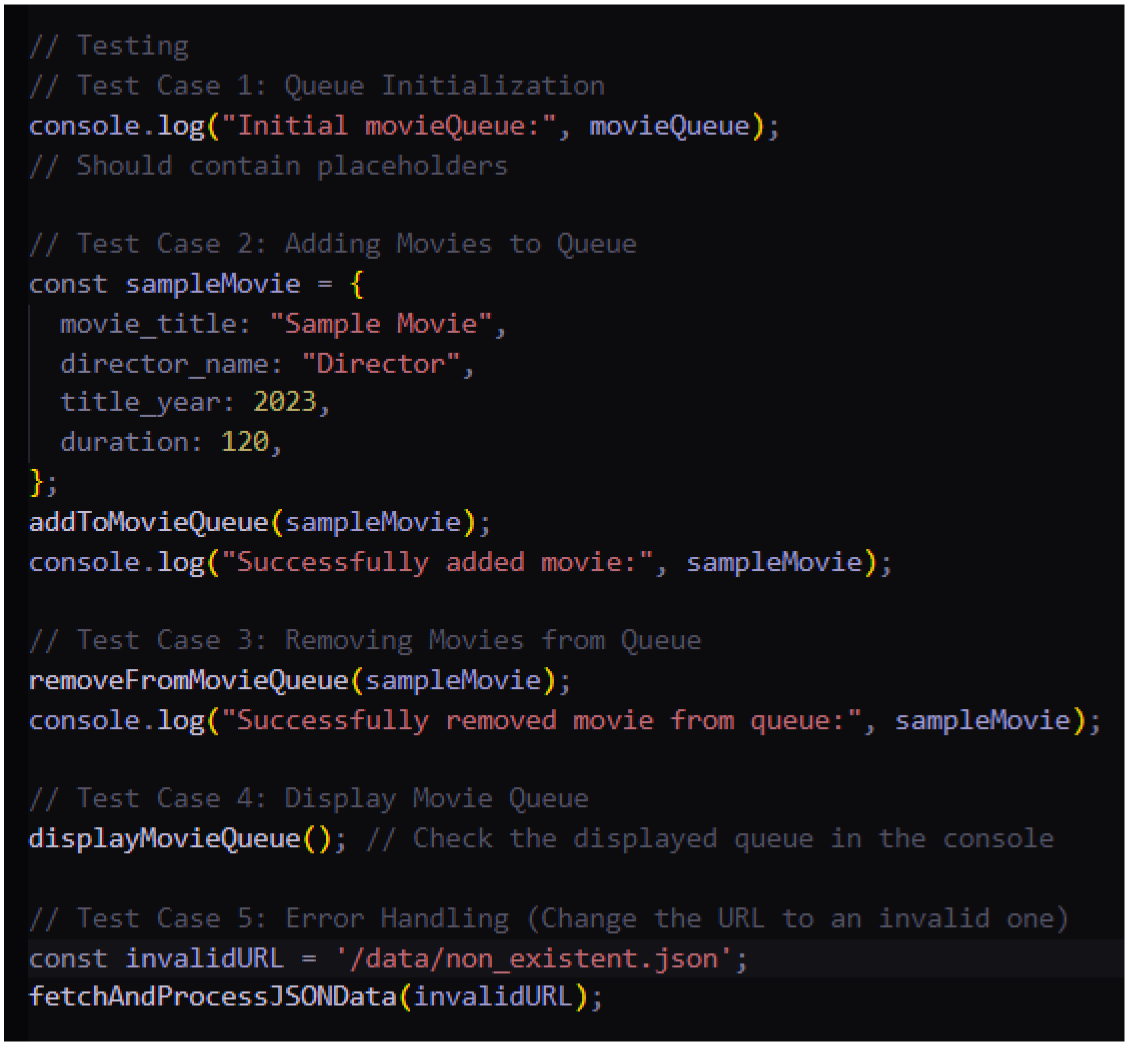
In the node.js program convert.js, I made the assumption that the does not have to provide the csv data file path using command-line arguments, so the file path is accessed and changed internally within the program.

Queue.js

Test Case	**Pre-Conditions**	**Expected Outcome**	**Actual Outcome**
Testing adding a sample movie to the queue	Create a variable movie with its own values and add to movieQueue	The movie will be added to the queue and will be able to be searched up	The movie is added successfully.
Removing sample	Calling the remove	The movie was	The movie was

movie from the queue	function on the sample movie that was added in an earlier test case	successfully removed.	removed from the queue.
Display the initial movie queue	Print the contents of movieQueue in console when website loads	3 placeholder movies from the top of the list are displayed	The placeholders are displayed correctly.
Fetch a json file from a non-existent file path	Call the fetch function with a url that does not exist	An appropriate error message will display and no action will be taken.	The error message was displayed successfully.

Here is the code and the console results for all the test cases:



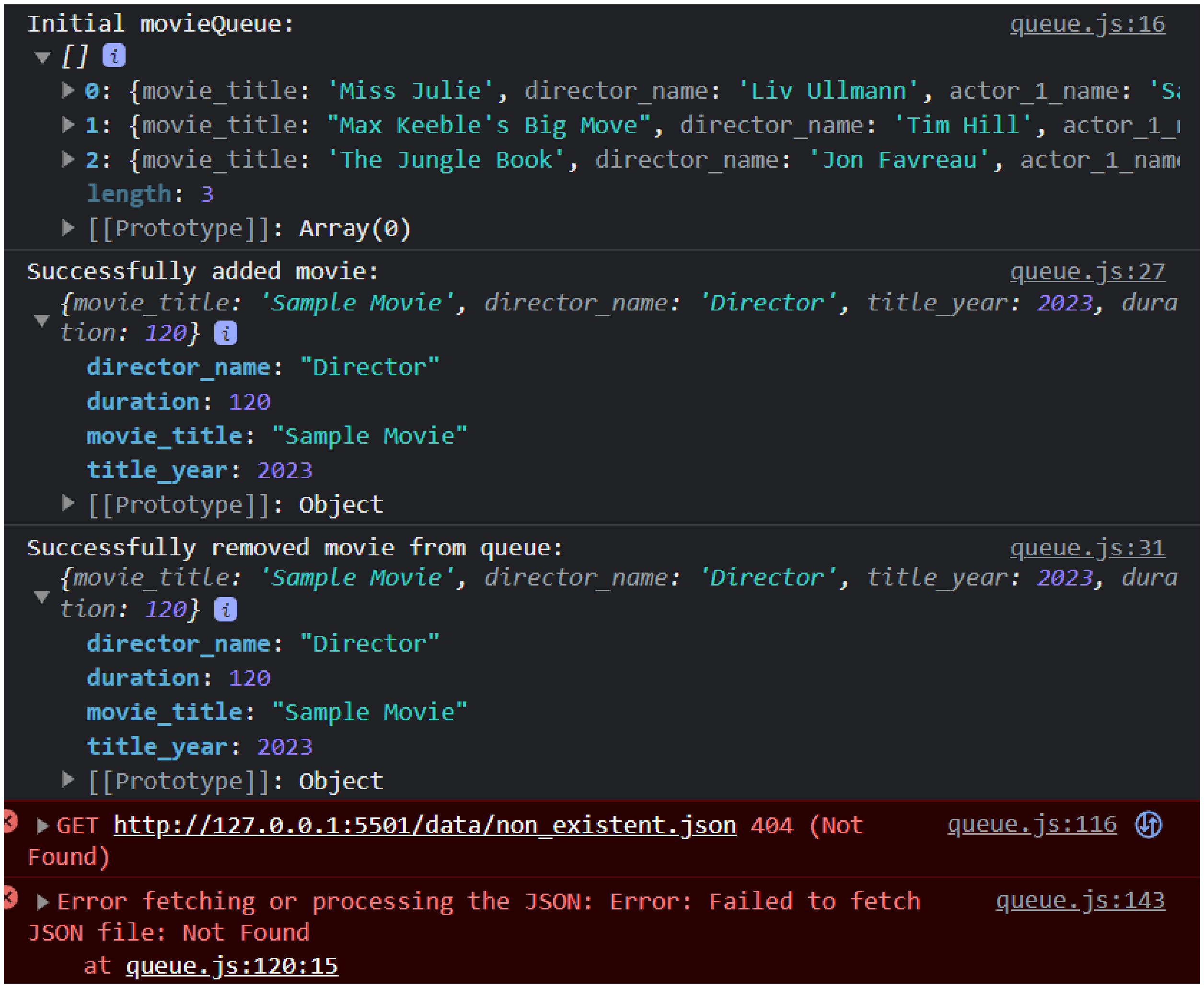
```
// Testing
// Test Case 1: Queue Initialization
console.log("Initial movieQueue:", movieQueue);
// Should contain placeholders

// Test Case 2: Adding Movies to Queue
const sampleMovie = {
  movie_title: "Sample Movie",
  director_name: "Director",
  title_year: 2023,
  duration: 120,
};
addToMovieQueue(sampleMovie);
console.log("Successfully added movie:", sampleMovie);

// Test Case 3: Removing Movies from Queue
removeFromMovieQueue(sampleMovie);
console.log("Successfully removed movie from queue:", sampleMovie);

// Test Case 4: Display Movie Queue
displayMovieQueue(); // Check the displayed queue in the console

// Test Case 5: Error Handling (Change the URL to an invalid one)
const invalidURL = '/data/non_existent.json';
fetchAndProcessJSONData(invalidURL);
```

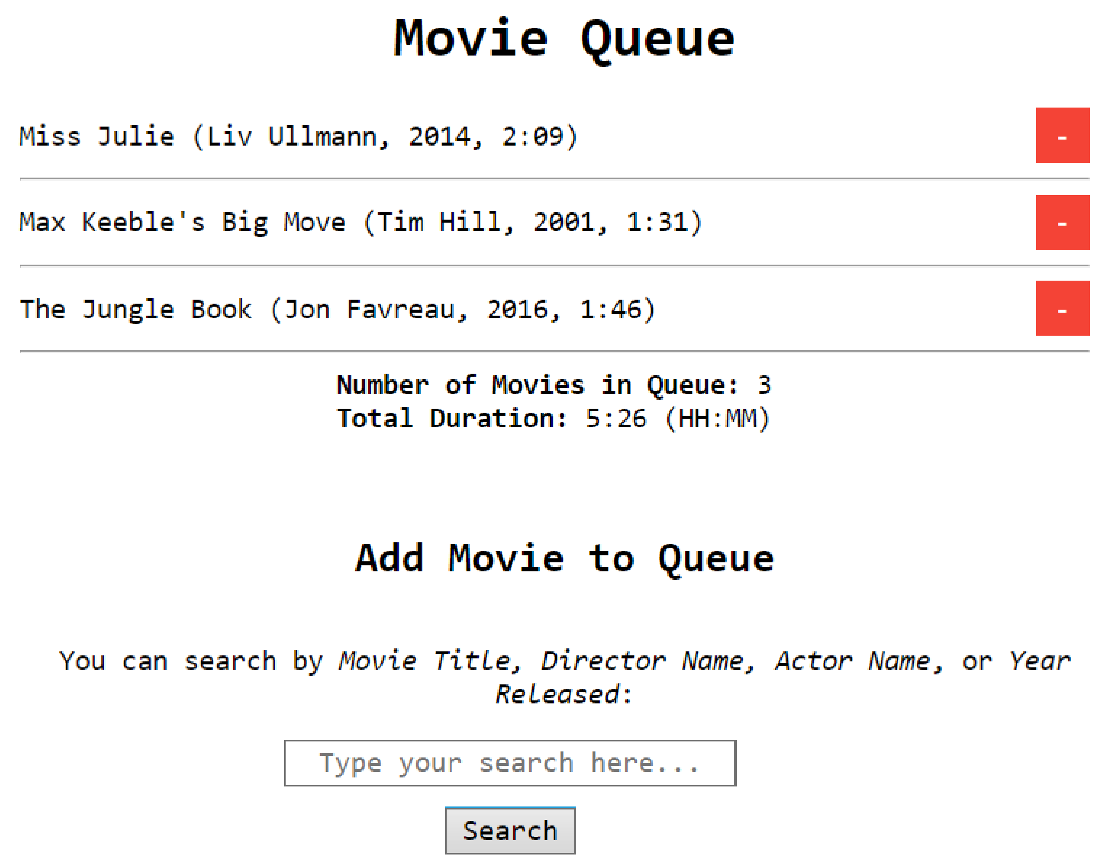


```
Initial movieQueue:                                                          queue.js:16
▼ [] i
  ▶ 0: {movie_title: 'Miss Julie', director_name: 'Liv Ullmann', actor_1_name: 'S
  ▶ 1: {movie_title: "Max Keeble's Big Move", director_name: 'Tim Hill', actor_1_
  ▶ 2: {movie_title: 'The Jungle Book', director_name: 'Jon Favreau', actor_1_nam
    length: 3
  ▶ [[Prototype]]: Array(0)
Successfully added movie:                                                    queue.js:27
  {movie_title: 'Sample Movie', director_name: 'Director', title_year: 2023, dura
▼ tion: 120} i
    director_name: "Director"
    duration: 120
    movie_title: "Sample Movie"
    title_year: 2023
  ▶ [[Prototype]]: Object
Successfully removed movie from queue:                                       queue.js:31
  {movie_title: 'Sample Movie', director_name: 'Director', title_year: 2023, dura
▼ tion: 120} i
    director_name: "Director"
    duration: 120
    movie_title: "Sample Movie"
    title_year: 2023
  ▶ [[Prototype]]: Object
 ▶ GET http://127.0.0.1:5501/data/non_existent.json 404 (Not          queue.js:116
Found)
 ▶ Error fetching or processing the JSON: Error: Failed to fetch       queue.js:143
JSON file: Not Found
    at queue.js:120:15
```

Search.js

Test Case	Pre-Conditions	Expected Outcome	Actual Outcome
Searching with a null string input	Not typing anything in the search input and performing the search	The search will not go through and nothing will be displayed	No error message or result was displayed, the program rejected the null input (Figure 1)
Normal Movie Search	Search by all categories for a movie that exists in the data.	The movie will be displayed with the ability to add it to the queue	The movie was correctly displayed and there was an option to add it to the queue.

Exceptional Case: Search for a movie that does not exist	Type a non-existing movie in the search bar and press click	No movie will be displayed and an error message will appear.	No movies were displayed and a suitable error message was shown.
Adding a movie to the movie queue	Pressing the add button beside a search result	The movie is added successfully and displays on the page.	The movie is added successfully.



Movie Queue

Miss Julie (Liv Ullmann, 2014, 2:09)

Max Keeble's Big Move (Tim Hill, 2001, 1:31)

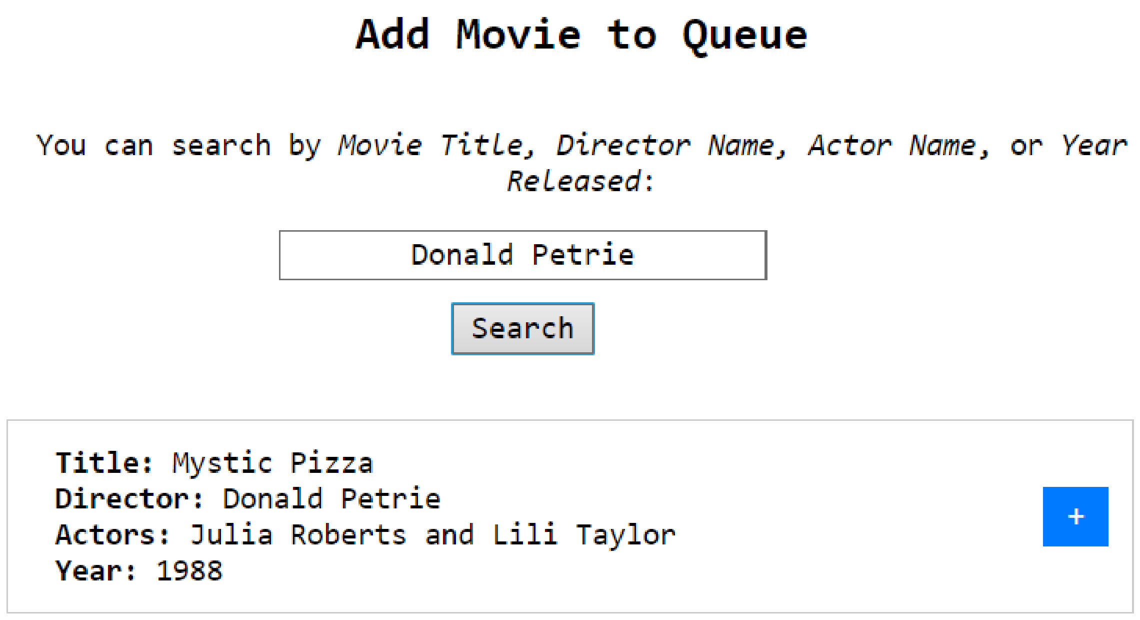
The Jungle Book (Jon Favreau, 2016, 1:46)

Number of Movies in Queue: 3
Total Duration: 5:26 (HH:MM)

Add Movie to Queue

You can search by *Movie Title*, *Director Name*, *Actor Name*, or *Year Released*:

Figure 1: no movie is displayed with a null input

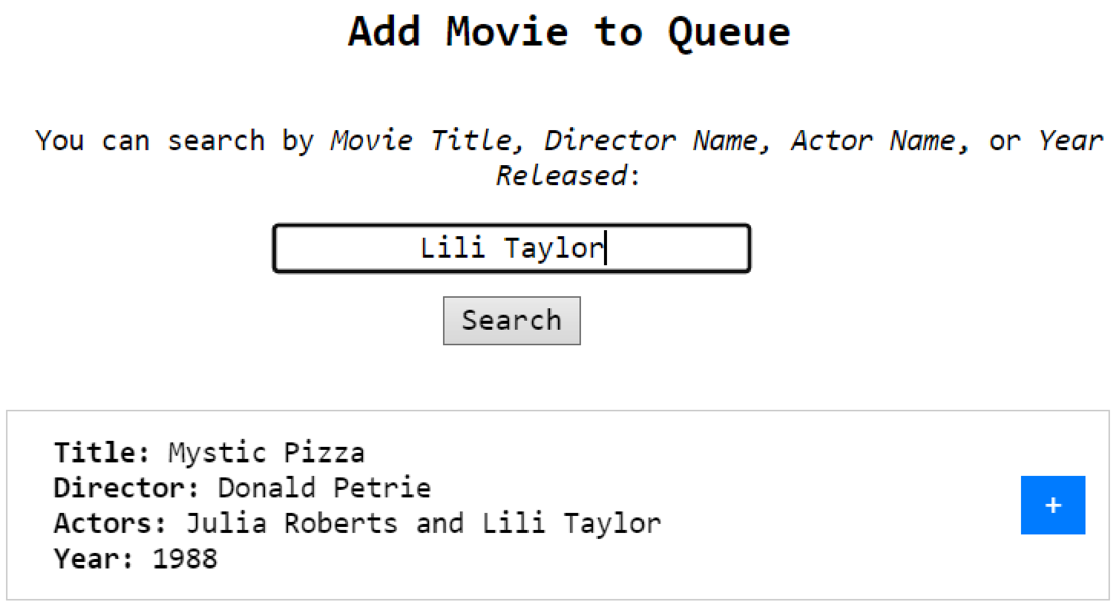


Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name*, or *Year Released*:

Donald Petrie

Title: Mystic Pizza
Director: Donald Petrie
Actors: Julia Roberts and Lili Taylor
Year: 1988

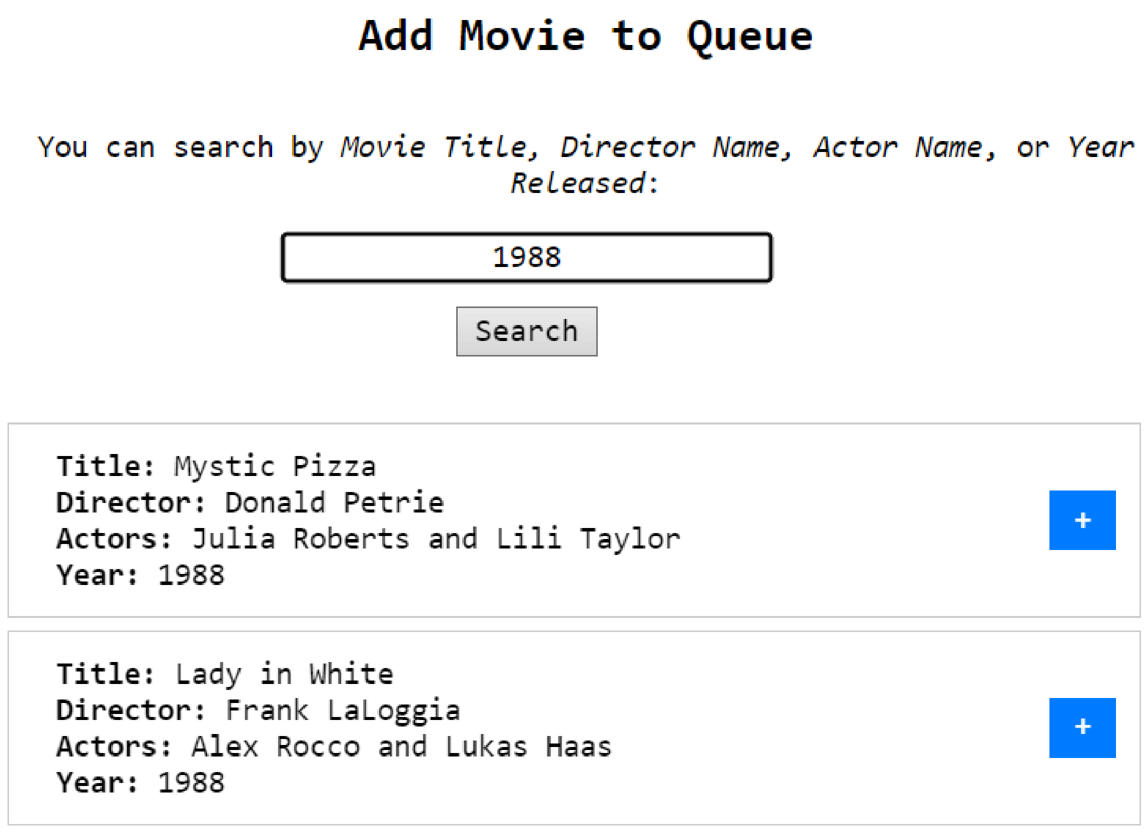


Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name*, or *Year Released*:

Lili Taylor

Title: Mystic Pizza
Director: Donald Petrie
Actors: Julia Roberts and Lili Taylor
Year: 1988

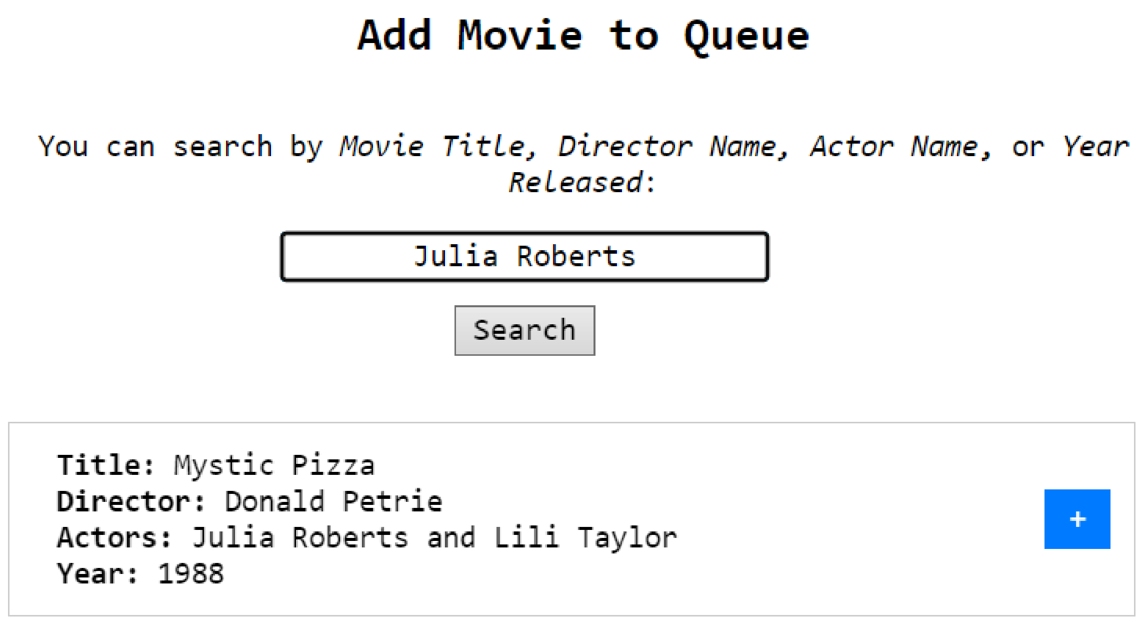

Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name,* or *Year Released*:

1988

Title: Mystic Pizza
Director: Donald Petrie
Actors: Julia Roberts and Lili Taylor
Year: 1988

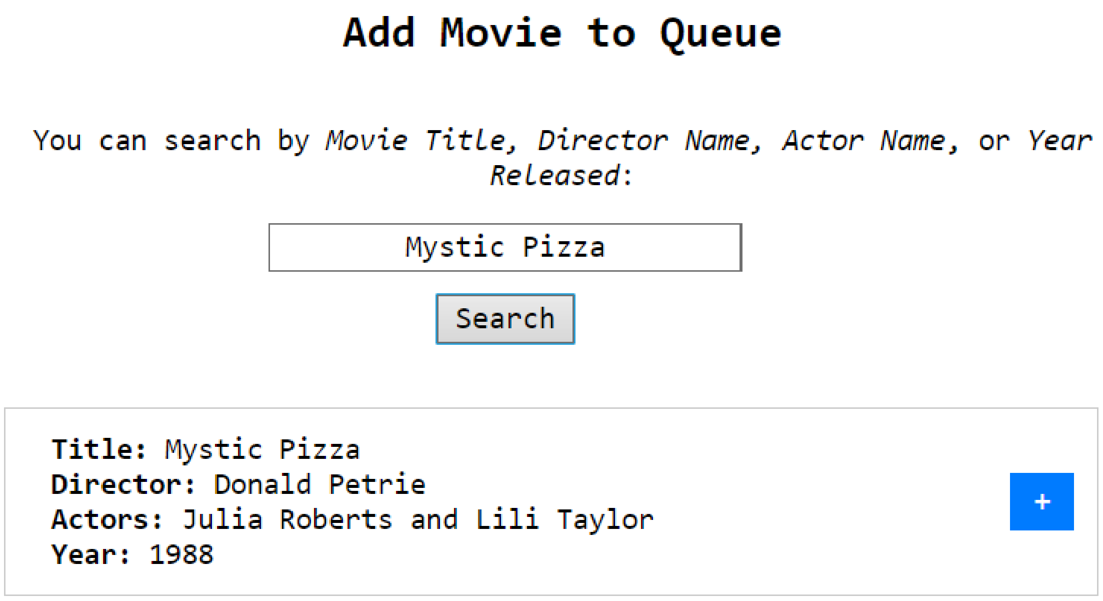
Title: Lady in White
Director: Frank LaLoggia
Actors: Alex Rocco and Lukas Haas
Year: 1988


Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name,* or *Year Released*:

Julia Roberts

Title: Mystic Pizza
Director: Donald Petrie
Actors: Julia Roberts and Lili Taylor
Year: 1988

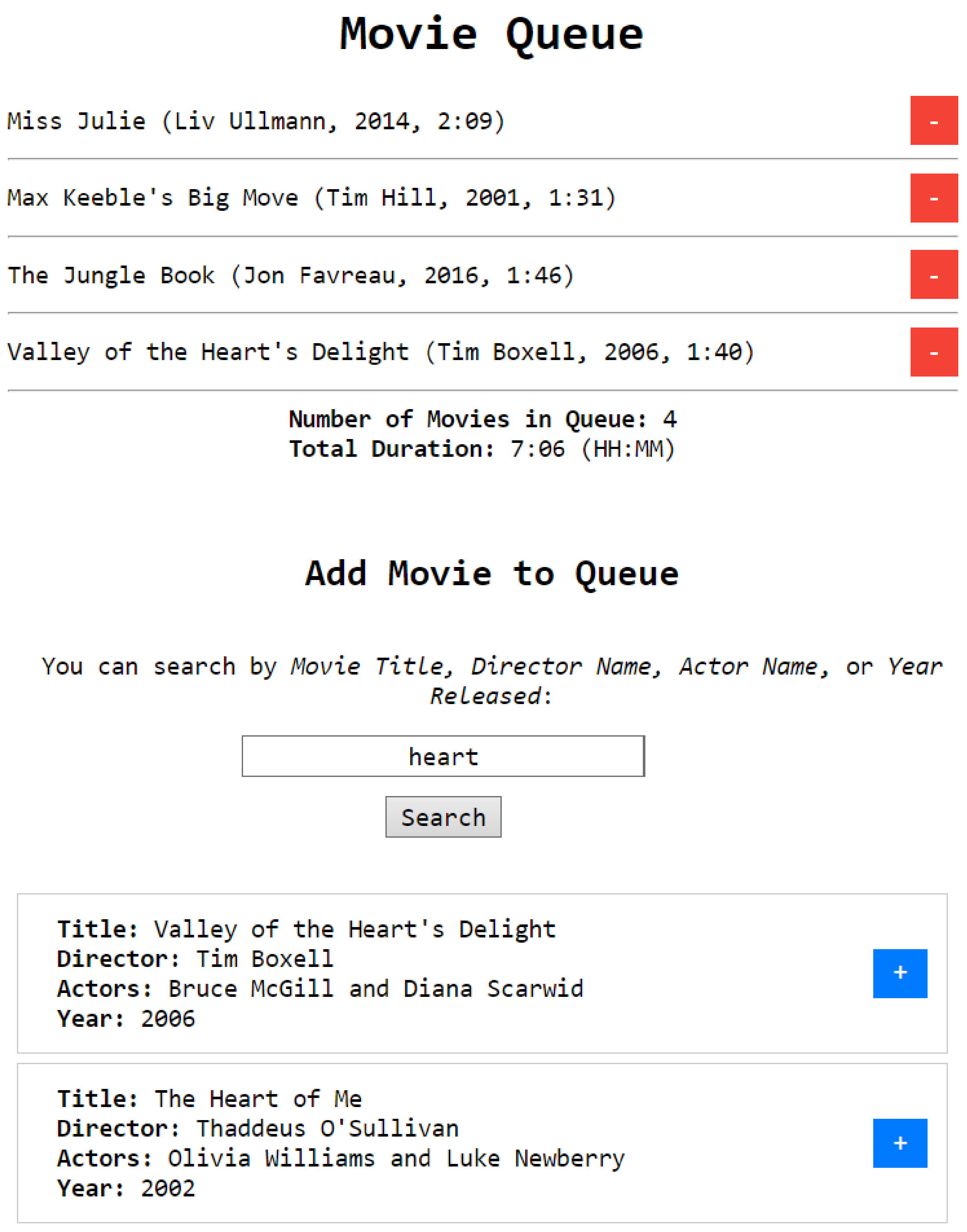

Add Movie to Queue

You can search by *Movie Title, Director Name, Actor Name,* or *Year Released:*

Mystic Pizza

Title: Mystic Pizza
Director: Donald Petrie
Actors: Julia Roberts and Lili Taylor
Year: 1988

Figure 2: all searches on all categories are successful.


Movie Queue

Miss Julie (Liv Ullmann, 2014, 2:09)

Max Keeble's Big Move (Tim Hill, 2001, 1:31)

The Jungle Book (Jon Favreau, 2016, 1:46)

Valley of the Heart's Delight (Tim Boxell, 2006, 1:40)

Number of Movies in Queue: 4
Total Duration: 7:06 (HH:MM)

Add Movie to Queue

You can search by *Movie Title*, *Director Name*, *Actor Name*, or *Year Released*:

heart

Search

Title: Valley of the Heart's Delight
Director: Tim Boxell
Actors: Bruce McGill and Diana Scarwid
Year: 2006

Title: The Heart of Me
Director: Thaddeus O'Sullivan
Actors: Olivia Williams and Luke Newberry
Year: 2002

Figure 3: a movie is added successfully to the queue.

Evaluation

I believe my program has met all the requirements that have been ruled out within the specification. There were some parts that I felt I could have changed, such as keeping the css naming conventions consistent within all my ids and classes, but the program works correctly and is able to add and remove movies in a queue, whilst also containing an intuitive design to search up and add movies.

Conclusion

I thoroughly enjoyed this practical and expanding upon the material that was taught to me during the lectures and course material. Web development has been a new and interesting branch of software development that I was eager to try and wasn’t disappointed by at all. If given more time on this project, I would’ve expanded upon the styling and user layout even further, perhaps adding more functionality and features whilst also improving upon the look and feel of the website.