CS2003 Practical 2 Report

Tutor: Dr. Tristan Henderson

Matriculation ID: 230030861

2023-10-25

Overview

In Practical 2, our assignment was to develop a simple web application that allows users to create queues of movies to watch, utilizing a movie dataset provided in CSV format. This assignment involved two parts, data cleaning and processing, and the development of a web application using HTML, CSS, and JavaScript. For the data cleaning and processing, I created a Node.js program called convert.js, which converts a CSV file containing movie data into JSON format. For development of the web application, I generated a list of all movies from the cleaned JSON data. The application allows users to create and manage a movie queue in an ordered list. Users can add or remove movies from the queue, and the application displays movie details, including the name, director, year, and duration, in the required format. I also implemented a range of UI enhancements, which included introducing a curated color palette, adding a search bar, refining button elements, and using many other CSS elements to elevate the overall visual appeal of the application. Nevertheless, the application is designed to be adaptable to different datasets. I adhered closely to the given guidelines and conducted comprehensive testing and documentation.

Design

The Node.js script, convert.js, reads data from a CSV file and converts it to a JSON file. For its design, I read from the path "./data/movie_metadata_subset.csv", splitting lines and headers to create JSON data. It uses a simple custom CSV parsing function to handle the CSV data. The JSON data is then written to a file at path "./data/movie_metadata_subset.json" in an appropriately structured format.

The JavaScript file, script.js, is responsible for the client-side functionality of the web application. It fetches movie data from the JSON file, stores it in an array, and allows users to interact with it. The script features functions for loading movie data, filtering movies, adding/removing them to/from a queue, and calculating total movie duration.
Upon fetching and storing the movie data, the script provides search functionality, dynamically updating the movie list based on search queries. It

also enables users to add and remove movies from a queue, with real-time updates on the queue's status, including the total number of movies and their duration.

The Cascading Style Sheets file,"styles.css," is a major component of the web application's design, responsible for most visuals. It provides unique styles for different elements and sections of the webpage, including custom headers, paragraphs, columns, lists, and containers. With the custom color palette, it creates an engaging visual experience. The flexible containers and columns, with specific styles like backgrounds, borders, and box shadows, ensure an attractive layout for movie lists and queues. Buttons like ".addBtn" and ".removeBtn" also offer distinct styles and hover effects, providing users with clear visual cues.

The Hypertext Markup Language file, index.html, creates the structure for the web page. It includes references to the CSS and JavaScript files and builds the layout of the application from head to toe. The layout, featuring two columns within a "movie-container," optimizes the presentation of content. The left column showcases the list of all available movies, complete with a dynamic count. Meanwhile, the right column focuses on the movie queue. It also utilizes the 'Nunito' font family and icons from the Font Awesome library to enhance the visual appearance of the page.

These design decisions were made to create a user-friendly, visually appealing, and easily maintainable web application.

Testing

What is being tested?	**Pre-conditions**	**Expected Outcome**	**Actual Outcome**
Fetch movie data from the JSON file	The JSON file containing movie data is accessible and correctly formatted	The web application loads movie data from the JSON file.	The web application loads movie data from the JSON file as expected.
JSON file is not found	The JSON file is intentionally missing	Error message and web page loads without movies	The web application displays an error message and loads without movies as expected.
Movie details are displayed	The web application is	Movie details (Name, Director,	Movie details (Name, Director,

correctly in all movies	loaded and functional. Movie data is successfully loaded and displayed.	Year, Duration) are displayed correctly in all movies.	Year, Duration) are displayed correctly in all movies.
Search bar accurately filters movies based on user input in real time	The web application is loaded and functional. Movie data is successfully loaded and displayed.	Movies are filtered and updated in real-time as the user types in the search bar.	The movies are accurately filtered and updated in real-time as expected.
Movies can be added to the queue by clicking the "Add" button	The web application is loaded and functional. Movie data is successfully loaded and displayed.	Movies are added to the queue when the "Add" button is clicked.	Movies are added to the queue when the "Add" button is clicked as expected.
Movie count in the queue increases after adding a movie	The web application is loaded and functional. Movie data is successfully loaded and displayed. At least one movie is available to add to the queue.	The count of movies in the queue increases by one after adding a movie.	The count of movies in the queue increases by one after adding a movie as expected.
Movie details are displayed correctly in the queue	The web application is loaded and functional. Movie data is	Movie details (Name, Director, Year, Duration) are displayed correctly in the	The movie details are displayed correctly in the queue

	successfully loaded and displayed. At least one movie is added to the queue.	queue.	
Remove movies from the queue using the "Remove" button	The web application is loaded and functional. Movie data is successfully loaded and displayed. At least one movie is added to the queue.	Movies are removed from the queue using the "Remove" button.	Movies are removed from the queue using the "Remove" button.
Movie count in the queue decreases after removing a movie	The web application is loaded and functional. Movie data is successfully loaded and displayed. At least one movie is added to the queue. At least one movie is removed from the queue.	The count of movies in the queue decreases by one after removing a movie.	The count of movies in the queue decreases by one after removing a movie.
Movie order is maintained after adding/removing from the queue	The web application is loaded and functional. Movie data is successfully loaded and displayed. At least one movie is added to the	The added movie goes to the end and the removed movie shifts the list upward	The added movie goes to the end and the removed movie shifts the list upward

	queue. At least one movie is removed from the queue.		
Accuracy of the total duration calculation in the queue	The web application is loaded and functional. Movie data is successfully loaded and displayed.At least one movie is added to the queue with a duration.	The total duration of movies in the queue is accurately calculated in HH:MM format.	The total duration of movies in the queue is accurately calculated in HH:MM format.
Font Awesome icons and Nunito font are displayed correctly	The web application is loaded and functional. Movie data is successfully loaded and displayed	Both are displayed correctly	Both are displayed correctly
More movies in CSV File	The web application is loaded and functional. A CSV file with a larger dataset than the current one is available and correctly formatted.	The web application successfully handles a larger dataset from the CSV file.	The web application successfully handles a larger dataset from the CSV file.
No movies in CSV file	The web application is loaded and functional. A CSV file with	The web application successfully handles a dataset without	The web application successfully handles a dataset without

	no dataset is available and correctly formatted.	movies from the CSV file.	movies from the CSV file.
Very long movie title	The web application is loaded and functional. A CSV file with a long movie title is included and correctly formatted.	The web application successfully handles a long movie title.	The web application successfully handles a long movie title.
No movie title	The web application is loaded and functional. A CSV file with no movie title is included and correctly formatted.	The web application successfully handles no movie title.	The web application successfully handles no movie title.

Evaluation

My submission aligns with the specified project requirements and objectives. It demonstrates a well-implemented web application that enables users to manage movie queues effectively. The initial task of converting the CSV dataset to a JSON file was executed successfully. The convert.js script accurately processed the data, generating the JSON file. The web application exceeded the defined requirements for functionality. It successfully allows users to create and manage movie queues. The search bar accurately filters movies in real time, making it user-friendly. The ability to add, remove movies from the queue, and maintain the order of movies worked as intended. The page presents movie details accurately, even for movies with exceptionally long titles. Font Awesome icons and the Nunito font are displayed correctly, enhancing the visual appeal. Error handling for scenarios such as a missing JSON file or inaccessible data was implemented. Overall, the script for data conversion functions as intended, and the web application's user interface and functionality contribute to an engaging user experience. This submission successfully accomplishes the core project goals and more.

Conclusion

I have successfully achieved the core objectives. I effectively converted the CSV dataset to a JSON file, read and presented the data, and implemented the queue. The application itself exceeds the specified requirements, providing a more user-friendly experience. I have a large amount of experience working with UI in Swift, but learning how to do it all over with a different language was quite challenging. If I had more time, real-world user testing would have been valuable. Additionally, more user-friendly features would surely enhance the application.

Bibliography

"Font Awesome." Font Awesome, *https://fontawesome.com*.

"Nunito." Google Fonts, *https://fonts.google.com/specimen/Nunito*.

"CSS - Cascading Style Sheets." Mozilla Developer Network, *https://developer.mozilla.org/en-US/docs/Web/CSS*.

"HTML (Hypertext Markup Language)." Mozilla Developer Network, *https://developer.mozilla.org/en-US/docs/Web/HTML*.

"JavaScript." Mozilla Developer Network, *https://developer.mozilla.org/en-US/docs/Web/JavaScript*.

"Tips for loading JSON." St Andrews University, *https://studres.cs.st-andrews.ac.uk/CS2003/Coursework/CS2003-P2-movie-queue/tips_for_loading_json.txt*.