# WDD 330 Personal Project

This document serves as your final course assessment.

## **Introduction**

**Name**: Mauro Villalba

**Video Link**: <https://github.com/villalba-mauro/wdd330-final-project/blob/main/video-final-project.mp4>

**Application Link**: <https://moviesmv.netlify.app/>

## **Course Outcomes**

The following are the course outcomes of WDD 330:

1. Become more efficient at applying your innate curiosity and creativity.
2. Become more dexterous at exploring your environment.
3. Become a person who enjoys helping and learning from others.
4. Use a divide and conquer approach to design solutions for programming problems.
5. Finding and troubleshooting bugs you and others will have in the code you write.
6. Developing and debugging HTML, CSS, and JavaScript programs that use medium complexity web technologies.

To complete this course, you need to demonstrate your skill in these areas. Outcomes #1-5 demonstrate your personal development and are most easily shown through self-assessment and sharing experiences. Outcome #6 demonstrates your programming skill and is shown through code and experience in projects.

## **Skill Development Outcome**

*Developing and debugging HTML, CSS, and JavaScript programs that use medium complexity web technologies*.

This outcome is demonstrated by your skill in the following learning objectives:

|  |  |  |
| --- | --- | --- |
| **Objective** | **%** | **Description** |
| JavaScript | 25% | Robust programming logic is demonstrated.  For example, validating the screen data, looping through an array of JSON data to display to the screen, creating and using events, changing element styles with JS, changing element classes to use different CSS rules. |
| Third-party APIs | 15% | APIs are used effectively, including APIs that provide rich JSON data. |
| JSON | 15% | Demonstrate skill processing JSON data to dynamically update the website. |
| CSS | 15% | Appropriate use of Transforms and Transitions. For example: Add round the edges to DIV, add shadows. enlarge an input field on focus, and shrink it on blur, Add borders. CSS should subtly add style to a page. |
| Events | 15% | Use events to enhance the user experience. For example, increase the size of the input field on focus or add a shadow. React to a button click. Initialized the page with data once the onload event triggers. |
| Local Storage | 5% | Local storage is used effectively. |

These learning objectives are rated on the following scale:

|  |  |
| --- | --- |
| **Rating** | **Description** |
| Unsatisfactory | Very little if any work was shown in this area. |
| Developing | The learning objective was shown in very basic ways. |
| Proficient | Effective use of the learning objective was shown in multiple places. |
| Mastery | Extensive use of the learning objective was shown in non-trivial ways in many places in the code. |

For each learning objective, discuss how the topic was used in your application. List several examples of places where the topics are demonstrated.

The following is an example of what is expected:

|  |  |  |
| --- | --- | --- |
| **Learning Objective** | **Description** | **Where can this be seen in your application?** |
| CSS | *I spent a lot of time choosing colors that would complement each other.*  *I used CSS to make the input field bigger when it received the focus and to shrink it when it lost focus.* | *This can be seen on the home screen for each input field.* |
| *Images are enlarged on hover.* | *The recipe detail pages have this effect.* |
| The search results have alternating colors for the rows for readability. | See the home page after a search is successfully run. |

In the following table:

1. Describe how the topics are used.

Have someone test your links to make sure they are accessible by the grader. These links will be to your final personal project.

Feel free to add more rows to this table if needed.

|  |  |  |
| --- | --- | --- |
| **Learning Objective** | **Description** | **Where can this be seen in your final personal project application?** |
| JavaScript | *I implemented robust programming logic to handle all aspects of the movie application.* | This can be seen in the app.js file where I use fetch to get popular movie data from the API and then dynamically generate HTML elements to display each movie on the home page. |
| I developed functions to manage navigation between pages and display specific details for each movie. | In details.js, the cargarDetallesPelicula() function retrieves data from the API based on the movie ID and dynamically updates the interface with all relevant information. |
| I implemented validation and error handling for cases where data is not available. | The mostrarError() function in details.js handles cases where the API doesn't return correct information or there are connection problems. |
| Third-party APIs | I integrated TheMovieDB API to get updated movie data. | In app.js, I use fetch to connect with TheMovieDB API and get popular movies using the URL <https://api.themoviedb.org/3/movie/popular>. |
| *I implemented multiple API endpoints to retrieve different types of data.* | In credits.js, I use the /credits endpoint to get cast and crew information. In details.js, I use the /similar endpoint to show similar movies. |
| *I used API parameters to customize requests according to user needs.* | In app.js, I use page parameters to implement result pagination, allowing navigation through multiple pages of popular movies. |
| JSON | I processed complex JSON data with multiple levels of nesting. | In details.js, I extract and process nested JSON data to display movie genres, release dates, and other detailed information. |
| I transformed JSON data into interactive visual elements. | In mostrarPeliculasSimilares() from details.js, I convert the JSON data of similar movies into an interactive movie grid. |
| I implemented local storage of JSON data for favorites functionality. | In favorites.js, I save and retrieve favorite movies as JSON objects in localStorage, allowing user preferences to persist between sessions. |
| CSS | *I developed a responsive design with transitions and transformations.* | In animation.css, I implemented the jello-vertical animation that applies to images when clicked. |
| *I used CSS animations to enhance user experience.* | In the configurarAnimacionScroll() function from animation.js, I apply animations to actor elements when they enter the viewport. |
| I implemented hover effects with smooth transitions. | In details.css, the .pelicula-similar elements have a scale transition on hover, creating a smooth zoom effect. |
| Events | I implemented event-based navigation for a fluid user experience. | In app.js, I use addEventListener for the "Previous" and "Next" buttons for pagination. |
| I used events to trigger animations on interactive elements. | In animation.js, I add click events to similar movie images that trigger the jello-vertical animation before navigating to the details page. |
| I implemented viewport entry detection for scroll animations. | I use IntersectionObserver in animation.js to detect when actor elements enter the viewport and trigger animations. |
| Local Storage | I developed a complete favorites management system using localStorage. | In favorites.js, I implemented agregarAFavoritos() and eliminarDeFavoritos() functions that manage the list of favorite movies in localStorage. |
| *I implemented existence verification to avoid duplicates in favorites.* | In the agregarAFavoritos() function, I check if a movie already exists in the favorites list before adding it, avoiding duplicates. |
| I allow users to manage their favorites with visual notifications. | The mostrarNotificacion() function in favorites.js provides visual feedback when the user adds or removes movies from favorites. |