# WDD 330 Personal Project

This document serves as your final course assessment.

## **Introduction**

**Name**: Camila Moya Casanova

**Video Link**: <https://www.youtube.com/watch?v=rmj8p-5areg>

**Application Link**: <https://camilamoyac.github.io/wdd330/project/>

**Source code**: <https://github.com/camilamoyac/wdd330/tree/main/project>   
**Trello board:** <https://trello.com/b/iL0sEirQ/book-playlist-project>

## **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 used event listeners for the search button and favorite toggles to interact with the UI dynamically. | Clicking the search button triggers book searches. |
| Implemented asynchronous API calls with fetch and async/await to retrieve book and emotion data. | Fetching book data from Google Books API and mood data from Twinword API. |
| Controlled DOM manipulation to render book cards and modals dynamically with user interactions. | Clicking a book card flips it and clicking “Get Mood” opens the playlist modal. |
| Third-party APIs | Integrated Google Books API to fetch fiction book data. | Search results show real-time data from Google Books. |
| Used Twinword Emotion Analysis API to analyze the mood of book descriptions. | Mood detection on book descriptions uses Twinword’s API. |
| Embedded Spotify playlists matching the detected moods dynamically. | Playlist modal embeds Spotify playlists corresponding to detected moods. |
| JSON | Parsed JSON responses from both Google Books and Twinword APIs. | API responses parsed in fetchBooks and analyzeMood functions. |
| Extracted nested book info such as authors, ratings, and genres from API data. | Book card rendering uses JSON fields like volumeInfo.authors. |
| Stored favorites as JSON strings in localStorage. | Favorites saved/retrieved using JSON.stringify and JSON.parse. |
| CSS | Used transforms and transitions for smooth card flip animations. | Book cards flip on click with a smooth rotateY transition. |
| Styled inputs to enlarge on focus and shrink on blur for better UX. | Search input expands on focus to improve usability. |
| Added shadows and rounded corners on cards and buttons for visual polish. | Cards and modal use shadows and rounded borders throughout the UI. |
| Events | Attached click events to search, toggle favorites, and card buttons. | Search button triggers book fetch. |
| Used window.load event to initialize default content. | Page loads default bestsellers automatically. |
| Modal close events listen for clicks outside content or on close button. | Modal closes on clicking outside or on the X button. |
| Local Storage | Favorites are saved to localStorage to persist user choices. | Clicking “Save to Favorites” adds a book to localStorage. |
| On page load, favorites are loaded and rendered from storage. | Favorite panel displays stored favorites when toggled. |
| Duplicate favorites prevented by checking stored data before saving. | Alerts notify if a book is already in favorites. |