# WDD 330 Personal Project

This document serves as your final course assessment.

## **Introduction**

**Name**: Didier Alexander Virguez

**Video Link**: [Insert your video link here]

**Application Link**: [Insert your URL here]

## **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 | Used JavaScript for adding interactivity to the product listing, handling events like adding/removing items to the cart, and dynamically displaying data from JSON. | JavaScript is used to handle product display and cart updates. For example, when an item is added to the cart, it triggers a JavaScript event that stores the product in localStorage. When removing items, JavaScript dynamically updates the cart and re-renders it. |
| I used JavaScript to reuse code efficiently | The loadHeaderFooter() function is called in app.js to inject consistent header/footer HTML (from /partials/header.html and /partials/footer.html) into every page. Verified by inspecting the #header and #footer elements in the DOM. |
|  |  |
| Third-party APIs | Integrated a third-party API to fetch product data and display it on the website. | The product data is fetched using an API call to a JSON file, which is processed and displayed using JavaScript. This is seen on the homepage where products are displayed dynamically based on the data returned from the API. |
|  |  |
|  |  |
| JSON | |  | | --- | |  |  |  | | --- | | Used JSON to store and process product data, as well as the cart state. | | JSON is used to structure the data from the API and to save the cart information in localStorage. The cart data is retrieved from local storage and displayed on the cart page. |
|  |  |
|  |  |
| CSS | I spend a lot of time giving good styles to the product categories because I wanted the home page of the app to look really good. It’s help me to undertand the importance of Id in HTML elements. | To see the results, it’s on the main page of the app. |
| I used pseudo-class to give better visualization when the user want to complete the registration | Focus Management: Input field styling in register.css with :focus pseudo-class |
|  |  |
| Events | |  | | --- | |  |  |  | | --- | | Utilized JavaScript events to improve user experience, such as click events, hover effects, and focus events. | | For instance, when a user clicks "Add to Cart," it triggers a JavaScript function that adds the product to the cart and stores it in local storage. On hover, the products enlarge for better visibility, enhancing user interaction. |
| I implemented comprensive event handling to create the hamburguer menu in small views | Hamburger Menu: Mobile menu toggles in utils.js using click events |
|  |  |
| Local Storage | Used local storage to persist the cart data even after the user leaves or refreshes the page. | When an item is added to the cart, it's stored in localStorage and retrieved when the cart page is visited. The user can leave the site and return later, and the items will still be in their cart. The cart is rendered dynamically from the data stored in localStorage. |
| Created helper functions (saveToLocalStorage, getFromLocalStorage) to serialize/deserialize data for persistent storage (e.g., register.js). | When users add a preference Gym then the boton register save to the localStorage (in gyms.js) preserves the gym selected. On page reload, getFromLocalStorage("selectedGym") restores the data |
|  |  |