# WDD 330 Personal Project

This document serves as your final course assessment.

## **Introduction**

**Name**: Ojobor Favour

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

**Working Application Link**: [Insert your link here]

**GitHub Source URL**: [Insert your URL here]

**Trello Board URL**: [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 |  |  |
|  |  |
|  |  |
| Third-party APIs |  |  |
|  |  |
|  |  |
| JSON |  |  |
|  |  |
|  |  |
| CSS |  |  |
|  |  |
|  |  |
| Events |  |  |
|  |  |
|  |  |
| Local Storage |  |  |
|  |  |
|  |  |

**Section 2: Personal Project Outcomes**

**Introduction**

* **Name**: Ojobor Favour
* **Video Link**: [Pending; record demo of http://localhost:5173]
* **Working Application Link**: [Pending; deploy: https://yeva04.github.io/wdd330/]
* **GitHub Source URL**: https://github.com/Yeva04/wdd330
* **Trello Board URL**: [Pending; create at https://trello.com]

| **Learning Objective** | **Description** | **Where can this be seen in your final personal project application?** |
| --- | --- | --- |
| JavaScript | Mastery | I implemented dynamic product loading in product.js and ProductDetails.mjs, with working cart functionality (src/js/product.js, src/js/ProductDetails.mjs, src/js/cart.js). |
| Third-party APIs | Proficient | I used ProductData.mjs as a mock API to fetch tents.json for dynamic pages (src/js/ProductData.mjs, src/product\_pages/index.html). |
| JSON | Mastery | I parsed tents.json for product details and updated index.html links to match IDs (src/json/tents.json, src/index.html). |
| CSS | Mastery | I used transforms and transitions for product card hovers (src/css/style.css). |
| Events | Mastery | I implemented click events for “Add to Cart” in ProductDetails.mjs and cart rendering in cart.js (src/js/ProductDetails.mjs, src/js/cart.js). |
| Local Storage | Mastery | I stored and displayed cart items in so-cart using setLocalStorage and getLocalStorage (src/js/ProductDetails.mjs, src/js/cart.js, src/js/utils.mjs). |

week 3

| **Learning Objective** | **Description** | **Where can this be seen in your final personal project application?** |
| --- | --- | --- |
| **JavaScript** | Enhanced ability to manipulate DOM elements, handle dynamic content loading, and implement client-side routing using query parameters. Learned to debug and optimize JavaScript code for deployment on static hosting platforms like GitHub Pages. | - Dynamic rendering of product details and cart items based on ?product= query parameters in ProductDetails.mjs and Cart.mjs. - Client-side routing implemented in ProductList.mjs to update the page without redirects. - Error handling and console logging for troubleshooting (e.g., getParam and product initialization). |
| **Third-party APIs** | Gained experience in integrating and troubleshooting third-party tools, such as Vite, to manage builds and asset handling, including addressing deprecation warnings (e.g., CJS Node API). | - Use of Vite for building and previewing the application, ensuring assets like images are correctly served (e.g., vite.config.js with legacy plugin). - Dependency management with npm install to resolve plugin issues. |
| **JSON** | Improved skills in working with JSON data for product information, including fetching and parsing data from tents.json to populate product and cart pages. | - Data fetching and rendering from public/json/tents.json in ProductData.mjs for product details and cart items. - Mapping JSON data to HTML elements in ProductList.mjs and Cart.mjs. |
| **CSS** | Strengthened ability to style responsive layouts and ensure visual consistency across product lists, details, and cart pages, adapting to deployment constraints. | - Consistent styling of product cards, details, and cart items across index.html, product\_pages/index.html, and cart/index.html. - Adjusted image display to handle 404 fallbacks gracefully. |
| **Events** | Developed proficiency in handling DOM events, such as click events for product selection and cart item removal, enhancing user interaction. | - Click event on product links in ProductList.mjs to update query parameters. - Remove button event listeners in Cart.mjs to delete items from the cart. |
| **Local Storage** | Mastered using localStorage to persist cart data across page reloads and sessions, ensuring a seamless shopping experience. | - Storing and retrieving cart items in localStorage using setLocalStorage and getLocalStorage in Cart.mjs and ProductDetails.mjs. - Dynamic cart updates reflected in the UI. |