# WDD 330 Personal Project

This document serves as your final course assessment.

## **Introduction**

**Name**: [Tiago Neves Sousa]

**Video Link**: [<https://youtu.be/aqnjf_VVHV0>]

**Application Link**: [<https://tiagorockman.github.io/wdd330/pages/search.html>]

**Trello Link: [**[**https://trello.com/b/7NnmuGOR**](https://trello.com/b/7NnmuGOR)**]**

## **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 | JavaScript was used extensively to control UI behavior, navigation, filtering, and dynamic rendering. | In search.js: Filtering universities via performSearch(), rendering cards with renderUniversities(), using createUniversityCard() to inject HTML dynamically. Also, navigation functions like navigateToHome() and navigateToSearch() in base.js. |
| Third-party APIs | I have used third-party API to fetch JSON university data from a remote backend server and also to load Map University location. | In search.js, the API URL is https://college-api-wxz6.onrender.com/api/colleges, used in initialLodaData() and searchWithFilters(). |
| In the Modal Popup of University information details I have added a Google Map API Integration to load University’s location. |
| JSON | JSON is parsed and used to populate the university cards dynamically on the page. | After fetching, JSON data is stored in universities and processed in renderUniversities() and populateModalContent(). Fields like tuition, ranking, and eligibility flags are parsed from JSON and used in rendering. |
| CSS | |  | | --- | |  |  |  | | --- | | CSS Transitions and Transforms are used to enhance UI interactivity and aesthetics. | | nput fields use :focus border transitions (base.css). The university cards have hover shadows, rounded corners, responsive layout, and color badges like .bg-green-100, .rounded-lg, .hover\:shadow-lg. The modal and search bar are styled with transitions. |
| Events | Multiple DOM events enhance interactivity. | In base.js: Event listeners for clicks, form submission, and keyboard input (e.g., pressing "Enter" on search). In search.js: Events are used for filtering changes, button clicks, and modal triggers. The toast system in base.js also uses events for dismissing messages. |
| Local Storage | Local storage retains search values and theme preferences. | - localStorage.setItem('searchValue', searchValue.value) in findUniversities() (base.js) - localStorage.setItem('theme', ...) in toggleTheme() - Reads theme preference on page load in initializeApp() |