# 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 |  |  |
|  |  |
|  |  |

**Personal Project Outcomes**

**Introduction**

* **Name**: Ojobor Favour
* **Video Link Youtube**: https://youtu.be/PgAZcBv4yi4?si=9J\_0XYejuiHrS4Ug
* **Working Application Link**: https://yeva04.github.io/fitness-tracker/ or Nettlify: ttps://fitness-tracker.netlify.app/
* **GitHub Source URL**: https://github.com/Yeva04/fitness-tracker/
* **Trello Board URL**: https://trello.com/b/lbJs3sNn/fitness-tracker

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Learning Objective** | **Description** | |  |  | | --- | --- | |  | **Where can this be seen in your application?** | |
| **1** | JavaScript | I used robust programming logic to validate user input in the login and signup forms, ensuring fields like username, email, and height are required and within valid ranges. I looped through arrays of JSON data from localStorage to display user goals and workout entries on the dashboard. I created events to handle form submissions and dynamically changed element styles, such as toggling the visibility of the signup modal.  Rating: mastery | This can be seen in the login page where form validation prevents submission with invalid data. The dashboard page shows looping through JSON for displaying goals. The modal toggle changes styles on click events. |
| **2** | Third-party APIs | I integrated the Nutritionix API to fetch calorie and protein data for food items entered by the user, parsing the rich JSON response to update the nutrition list. I also used the Strava API to retrieve activity data like steps and calories burned, handling the OAuth flow to authenticate and fetch user-specific information.  Rating: mastery | This can be seen in the Nutrition Tracker section where entering a food name fetches and displays calorie/protein data. The Dashboard shows Strava data fetched and added to the chart after authentication. |
| **3** | JSON | I processed JSON data from localStorage to load and update user workouts, nutrition entries, and goals, dynamically rendering lists and charts on the dashboard. I parsed JSON responses from the Nutritionix API to extract calorie and protein values for display. I handled JSON for Strava API responses to aggregate steps and calories burned.  Rating: mastery | This can be seen in the Workout Log where JSON from localStorage is parsed to render the entries list. The Nutrition Tracker parses JSON from the API for calorie/protein. The Dashboard aggregates JSON for the line chart. |
| **4** | CSS | I used CSS transforms to scale the profile icon on hover for a subtle interactive effect. I applied transitions to the signup modal for smooth fade-in and slide-up animations when opening/closing. I rounded edges on list items and buttons with border-radius, and added shadows to sections for depth.  Rating: mastery | This can be seen on the login page where the signup modal transitions in/out. The dashboard list items have rounded edges and shadows. The profile icon scales on hover with transform. |
| **5** | Events | I used events to enhance UX by triggering the signup modal on click of the "Sign Up" link. I reacted to button clicks in the navigation to switch sections dynamically. I initialized the page with data on the onload event to load user-specific information from localStorage.  Rating: mastery | This can be seen on the welcome page where clicking "Sign Up" triggers the modal event. The dashboard navigation buttons react to clicks to show/hide sections. The page onload event loads data in the dashboard. |
| **6** | Local Storage | I used local storage to save and retrieve user data, such as profile details and current user session. I stored workout and nutrition entries to persist across sessions. I managed goals and dashboard data for quick access without server calls.  Rating: mastery | This can be seen in the login process were local storage checks for existing users. The workout and nutrition lists load from local storage on page reload. The dashboard chart uses local storage data for rendering. |