Portfolio 1 - Course Assignment

Cross Course Project – Rainydays

Project summary

Rainydays was my first larger project, combining first Design, then HTML and CSS, and later JavaScript (split into three separate course assignments) to build an e-commerce website for a fictional company selling premium rain jackets. I focused on building a complete user experience, from browsing and filtering products to adding items to the cart and completing checkout. This project taught me how to structure multi-page sites, write semantic HTML, and maintain clean, reusable styles.

Teacher feedback

In the initial HTML & CSS submission, I received positive feedback on responsiveness, image optimization, clear navigation, and accessibility. One improvement suggested was adjusting the layout of call-to-action buttons on very small screens. I resolved this issue in the JavaScript submission by restructuring the layout using flexbox, resulting in proper stacking on narrow screens.

In the JavaScript assignment, all functionality passed, and my instructor praised the use of error handling, a custom modal, and organized code.

Improvements made during this course assignment

Looking back with more experience, here are some improvements I made during this course assignment:

Updated the README.md file:

The original README.md file only included the JavaScript 1 Course Assignment Brief. As part of preparing Rainydays for Portfolio 1, I moved the brief to a separate brief.md file and created a detailed new README.md. It now includes a project overview, goals from each

course brief, setup instructions, features, refactoring summary, and future improvement ideas. This makes the project more professional and easier to understand for potential repository visitors.

HTML:

- Improved alt text on images to make them more descriptive and helpful to screen readers.
- Added appropriate aria-labels to enhance accessibility.

CSS:

- Moved repetitive styles like font-sizes to the global styles.css file.
- Merged smaller CSS files into one shared file (about.css, confirmation.css, and information.css was merged into pages.css) for better maintainability.
- Improved spacing, layout, and responsiveness for a more modern design.
- Reorganized header and footer CSS into separate header.css and footer.css files to make styling more modular.
- Improved spacing and responsiveness in the header using media queries.
- Smoothed the menu animation using opacity, transform, and visibility transitions.
- Improved the footer layout and styling by using a responsive grid layout and aligning content more cleanly across mobile, tablet, and desktop breakpoints.

JavaScript:

- Rebuilt the hamburger menu using JavaScript instead of CSS-only for better interactivity and accessibility (keyboard support, overlay click-to-close, resize handling).
- Added a scroll to top button (using JavaScript) to improve UX on long pages, with smooth scrolling and visibility toggling after 300px.
- Refactored product filtering logic.

  Originally, I used three separate JavaScript files (mens.js, womens.js and script.js) to fetch and display the products by gender. Each of these files contained very similar logic for fetching products, filtering by gender, and displaying them. This led to a lot of repeated code, which made the code inefficient and hard to maintain.

  To improve this, I replaced all three files with a single reusable script, productfilter.js and added a data-gender attribute in the two HTML files where only one genders products are displayed (women.html and men.html) and doesn't have filtering options (like index.html and jackets.html). This handles gender filtering dynamically based on either the selected dropdown value, or a data-gender value set on the product container. The result is cleaner, more efficient, and easier to maintain code. If we ever

need to change how products are fetched or displayed, we now only have to update one file instead of three.

These changes not only improve usability and design but also demonstrate my progress in writing modular, accessible, and effective code. It helped make the Rainydays site more accessible, maintainable, and user-friendly across devices. Even though there are still changes I would have liked to make with more time, I'm proud of how far this project has come since the first version, and feel it now reflects a more polished and professional frontend development approach.

Here are some more improvements that could have been done to Rainydays:

- Folder and file structure could be more organized.
- Use product.id instead of title to identify cart items more reliably.
- Provide better visual feedback on cart updates, such as animations or highlights.
- Avoid re-rendering the full cart when updating quantity in checkout.js.
- Add a shared constants file for reusable values like API URLs or currency.
- Implement stronger error handling and fallback behavior for corrupted or missing localStorage.

Summary

Rainydays was a turning point for me as a frontend development student. It helped me build confidence in using real API data, creating dynamic interfaces, and improving accessibility. The improvements I made afterward reflect my progress as a developer, especially in writing modular, maintainable code and thinking critically about user experience and structure. Rainydays was the first major project I built and has grown significantly through each course module. By updating the HTML, CSS, and JavaScript files and completing a full README.md, I now feel the project meets the Portfolio 1 requirements. This version reflects both the original assignments and my growth as a frontend developer.

Semester Project 1 – Community Science Museum

Project summary

This was my first project exam and my first time following a full client-style brief from start to finish. The goal was to design and build a modern, accessible, responsive website for a fictional science museum aimed at children and families. I completed all the required pages using only HTML and CSS, without any frameworks or JavaScript. The project focused on

delivering a well-structured, visually appealing, and accessible multi-page site based on a provided sitemap and content brief.

Teacher feedback

For Semester Project 1, I received very positive feedback. My HTML was described as neat, semantic, error-free, and fully responsive, with strong accessibility and image optimization. The CSS received similar praise for being clean, organized, and following DRY principles. One small improvement suggested was to reuse common font size classes more efficiently across different elements, but overall, the separation of styles and structure was well executed.

Improvements made during this course assignment

In preparation for Portfolio 1, I reviewed and enhanced the code for Semester Project 1 with a focus on semantic HTML, accessibility, CSS structure, and maintainability.

Created a README.md file:

I wrote a complete README.md file, including a project overview, built-with list, page breakdown, setup instructions, goals, and useful links. This makes the project more professional and easier to understand for anyone visiting the repository.

HTML:

- Text and title consistency: All page titles, navigation links, and headings were updated to use Title Case (e.g., “Visit Us”, “Special Events”) for consistency.
- Semantic structure: Pages now use more meaningful HTML elements like <section>, <main>, and appropriate <h> tags to better reflect the page structure and improve screen reader support.
- ARIA labels: ARIA attributes were added to key areas such as <main>, <form>, and embedded maps to improve accessibility for assistive technologies.
- Contact form: Improved the form by adding a disabled and pre-selected “Select a topic” option to the <select> dropdown.
- Image optimization: Added lazy loading on images.
- Layout improvements: Added content wrappers with max width on multiple pages for better readability.
- Date formatting: Updated the date format on Privacy Policy and Terms and Conditions pages to international format (“May 12, 2024”).

CSS:

- Modular structure: Moved header and footer styles into separate header.css and footer.css files.

- Centralized typography: Consolidated font sizing for headings, paragraphs, lists, buttons, and small text in styles.css.
- Modern font sizing: Reduced font sizes slightly for a cleaner, more modern look.
- Global styles: Moved shared styles (e.g., map, lists) to global styles.css.
- Removed redundant breakpoints across all CSS files.
- Media query cleanup: Removed redundant breakpoints and simplified queries across all CSS files for consistency and easier maintenance.
- Code cleanup: Removed duplicated or unused code, added missing CSS variables, clarified comments, and fixed bugs and typos.

Here are some improvements that could have been done if JavaScript allowed:

- Interactive navigation menu: A responsive hamburger menu with toggle functionality.
- Form validation: Instant client-side validation for the contact form.
- Form submission feedback: Visual confirmation messages or animations after form submission instead of redirecting to a separate page.
- Image carousel for home page: JavaScript could be used to build an automatic or manual image carousel to showcase featured exhibitions, events etc.
- Scroll to top button: A floating button to help users quickly navigate back to the top on longer pages.
- Dynamic opening hours or countdown timer: Real-time updates for today's opening hours or a countdown to upcoming events.
- Dark mode: A theme switch with localStorage support for users to choose between light and dark modes.

Summary

Semester Project 1 was my first full-scale project exam, and it gave me a solid foundation in responsive design, semantic HTML, and accessibility best practices. The improvements I made for Portfolio 1 reflect how much I've grown as a frontend developer. By cleaning up the HTML, refactoring the CSS, and improving structure and readability, I now feel this site represents a more professional and maintainable project.

Although JavaScript was not part of this project, I laid a clean foundation for future interactivity. These enhancements show my ability to revisit and refine a project to meet higher standards and reflect industry best practices. I'm proud of how far this project has evolved.

Project Exam 1 – The Tasty Table

Project summary

For my Project Exam 1, I designed and developed The Tasty Table, a fully responsive food blog web application built using HTML, CSS, and JavaScript. The goal was to build a front-end user interface for an existing blog API, allowing users to view blog posts and enabling the blog owner to create, edit, and delete content through a secure login system. The project followed a full development workflow, including planning, UI design, API integration, and deployment.

Teacher feedback

The feedback for Project Exam 1 was very positive overall. The design system stood out, especially the detailed style guide, which was praised for its thoroughness. One small design suggestion was to consider moving the logo to the top-left corner on mobile. The planning process was described as excellent, with every requirement fully met.

Code quality was also highly appreciated. The HTML, CSS, and JavaScript were described as well-documented and error-free. The use of CSS variables and DRY principles was praised. However, a few improvements were suggested to push the project even further. I was advised to avoid declaring styling attributes like width and height directly in the <img> tags. I initially did this to improve performance by preventing larger images from being loaded unnecessarily.

Some JavaScript areas could also benefit from more reusable functions, like extracting repeated DOM manipulation logic into utility functions, and handling login errors with a separate helper. Using createElement() instead of .innerHTML was recommended for better performance, and storing the API base URL in a constants file would help keep the code cleaner and easier to maintain. There was also a reminder to use the URL constructor for the share button to keep things more modern and efficient.

Improvements made during this course assignment

Throughout this project, I made several improvements based on the teacher feedback to meet the technical requirements, improve performance, and follow best practices.

Updated the README.md file:

Added a preview image, a summary of improvements made during the Portfolio 1 Course Assignment, and a section outlining potential future enhancements. These updates make the project README more complete and better aligned with the other projects in the portfolio.

HTML:

- Removed width and height attributes from <img> elements to follow best practices.

CSS:

- Moved logo to the top-left corner in mobile view, based on the teacher's design suggestion. This aligns better with standard mobile UI patterns and improves brand visibility.

JavaScript:

- Replaced .innerHTML with createElement() in the spinner logic for improved performance.
- Moved spinner function from common.js to a separate utility file (uiUtils.js) to improve modularity and reuse.
- Created formUtils.js with a reusable showError() function for displaying error messages. This utility is now used across all form scripts (create.js, edit.js, login.js, and register.js)
- Refactored form field population in edit.js into a separate populateFormFields() function. This improves clarity and makes the code easier to maintain.
- Updated the share button logic to use URL constructor for cleaner and more reliable URL handling.
- Moved API_BASE into a new constants.js file and imported it wherever needed, improving consistency and centralizing configuration.

Here are some improvements that could have been done:

Based on my current knowledge of HTML, CSS, and JavaScript, there aren't many more improvements I could identify beyond what was mentioned in the teacher's feedback. However, here are some ideas for potential future enhancements:

- Add filtering of blog posts using tags:
  Since the Create Post form already includes tags, it would be useful in the future to allow users to filter posts by categories like "Lunch", "Dinner", or "Dessert". This would improve navigation and user experience on the blog feed page, especially if the blog grows with more recipe posts.
- Refactor all .innerHTML to createElement():
  I replaced .innerHTML with createElement() in the loading spinner logic, as suggested in the feedback. I decided not to refactor the rest of the .innerHTML usage due to time constraints and because the performance benefits are minimal for a small site like this. However, I understand that createElement() is generally safer and more performant, and I plan to explore using it more consistently in future projects.

Summary

Project Exam 1 gave me the opportunity to apply everything I've learned so far in project planning, design, HTML, CSS, and JavaScript by building a complete, responsive blog web application connected to a real API. I planned and designed a unique food blog concept, implemented full CRUD functionality for the blog owner, and created a user-friendly layout for both desktop, tablet, and mobile users. I received helpful teacher feedback and was able to make targeted improvements to the code structure, performance, and maintainability. This project helped me gain confidence in working with APIs, structuring JavaScript using reusable utilities, and following best practices in modern front-end development. This is the most challenging frontend project I've ever worked on, and I'm proud of the result and look forward to applying what I've learned in future projects.