Personal Project Assessment

This document serves as your final course assessment.

Introduction

Name: Caitlin Evans

Video Link: Youtube Video / https://youtu.be/iUm1O2n0cB4

Application Link: Cloak and Ember

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.

Project Purpose (from my project proposal)

I've always loved Harry Potter—not just the magic, but the deeper themes about love, resilience, loss, loyalty, and friendship. These stories have stuck with me since I was a kid and still mean a lot to me now as an adult. As a data nerd and a lifelong fan, I saw this project as a fun way to blend both passions.

I have been a software engineer for around 6 years but I am currently transitioning from front-end specifically into data science. I've been working as a Business Data Analyst for the last 9 months and have started looking at everything—including this JSON—through a more analytical lens. This project is my way of stretching both sides of my skillset: clean, accessible front-end UI *and* meaningful data organization. I decided I wanted this project to the a SPA derived from JS. I only have one HTML page as my index and I use JS and injected HTML to create my page changes.

My goal is to make something whimsical, accessible, and technically robust using data from multiple sources. I want this to be fun for users (especially other HP nerds), but also thoughtful—designed with sound structure, animation, interactivity, and deep exploration of the relationships in the data itself.

Learning Objective Category	Principle or Growth Area	**Description**	Technical Implementation (what did I do)	**Where can this be seen in your final personal project application?**	Challenges overcome
JavaScript					
	JavaScript Modularization	Highly scalable and professional	folder structure file planning and layout based on directory hierarchy and organization	folder structure	sometimes a file might fit in more than on parent catefory and just deterining which would make the most sense future

					proofing and if another developer would be looking for it.
	Navigation Management	Single Page Application Logic	- Implemented route-based rendering - Created dynamic content loading - Developed history management	Category navigation and page transitions	- Managing browser history -Creating smooth transitions - Handling different content types - handling different data APIs where the category was similar but had overlapping details or missing details in comparison
	Dynamic Image Fallback Strategy / Intelligent Resource Handling	Implement a hierarchical image selection mechanism for character representations. I Set the image provided in the API to display, if no image, then created default house image, check house associated with character then assign the house image, if no image, and no house listed, use default open book svg	- Created multi-tier image source selection - Used conditional logic for image resolution - Implemented fallback mechanism with prioritized sources	Character cards and modal views showing character images	- Handling missing image data - Creating a consistent visual experience - Managing different API image structures
	Conditional Rendering / Dynamic UI Generation	Create adaptive UI based on data availability	- Implemented null/undefined checks - Created fallback rendering strategies - Developed flexible content generation	Each Card type dynanmically renders the data differently based on what data is available. Even if one character has more data from the API, it doesnt break another	- Handling incomplete data sets - Creating graceful degradation - Maintaining consistent UI

				character card with less, and the character card and character modal are different than the spell card and spell modal, which is different from the potion card and potion modal etc.	
	Dynamic Data Merging in `hpApi.js`	Merge characters from HP API and Potter DB in hpApi.js	- Practiced advanced array manipulation - Learned object merging techniques - Developed robust data transformation skills - Used `Array.prototype.find()` for matching - Implemented object spreading - Created type-safe data transformation	Characters page showing merged character details from HP and Potter DB APIs	- Handling inconsistent API data structures - Ensuring data integrity across sources - Managing different data schemas
	Conditional Rendering in `cards.js` / `modal.js`	Create adaptable card and modal rendering based on data type	- Implemented type-based rendering functions - Used template literals for dynamic HTML - Created flexible content mappingCards and modal views for characters, spells, quotes- Managing multiple content types - Creating **reusable** rendering logic	Cards and modal views for characters, spells, quotes	- Managing multiple content types - Creating reusable rendering logic - Maintaining clean, readable code
	Event-Driven Modal Interaction	Develop comprehensive modal interaction system	- Created custom event dispatching - Implemented flexible modal	Modal pop-ups across different categories	- Managing complex event flows - Ensuring

			content loading - Developed dynamic interaction handlers		smooth user interactions - Creating flexible event systems
	Type-Safe Data Transformation / Advanced Object Manipulation	Transform and normalize data from different sources	- Implemented complex object mapping - Created type inference mechanisms - Developed robust data normalization	Character and spell details pages	- Maintaining data consistency - Handling missing or inconsistent data - Creating flexible transformation logic
	Random Fact Generator / Dynamic Content Discovery	Create a feature that retrieves and displays a random item from various categories	- Implemented comprehensive data aggregation - Developed random selection algorithm - Created flexible modal rendering for different item types	"Accio a Random Fact File" button on navigation	- Handling multiple data types - Ensuring consistent rendering across categorie - Creating a truly random selection mechanism
Third-party APIs					
HP-API (https://hp-api.onrender.com/)	Comprehensive Data Retrieval	Fetch and process character data from the Harry Potter API	- Implemented multiple character fetch methods - Created comprehensive data transformation - Developed robust error handling for API calls	Characters and spells page showing detailed character and spell information	- Handling inconsistent API data structures - Managing different character attributes - Ensuring data consistency
PotterDB (https://potterdb.com/) (API Doc: https://docs.potterdb.com/)	Multi-Source Data Merging	Integrate additional character and magical entity data from Potter DB API	- Developed advanced data merging algorithms - Implemented unique item detection - Created comprehensive fallback mechanisms	Characters and spells page showing detailed character and spell information	- Resolving data conflicts - Ensuring data consistency - Managing multiple data sources

Wizard World API (https://publicapis.io/wizard-world-api) (https://wizard-world-api.herokuapp.com/swagger/index.html)	Diverse Data Source Integration	Incorporate additional magical world information for potions / elixers / ingredients / other spells • Elixirs • Houses • Ingredients • Spells • Wizards	- Implemented flexible API data fetching - Created comprehensive error handling - Developed data normalization techniques	Other category details for potions / elixers, ingredients, additional spell data etc.	- Handling different API data structures - Managing partial data availability - Creating unified data presentation
Quotes Api / Rapid Api (https://rapidapi.com/hannanel100/api/harry-potter-quotes)	Resilient Data Fetching Strategy	Fetch quotes from RapidAPI with local JSON backup	- Implemented sequential API fetch methods - Created comprehensive error handling - Developed local JSON fallback mechanism	Quotes section with dynamic quote display	- Managing API reliability issues - Ensuring consistent quote availability - Creating seamless data presentation
	Asynchronous Data Fetching / Promise Handling	Implement complex, concurrent API data retrieval	- Used `Promise.allSettled()` for multiple API calls - Created comprehensive error handling - Implemented loading state management	Initial page load with multiple API sources	- Managing multiple asynchronous calls - Handling partial API failures - Providing seamless user experience
JSON					
	JSON Fallback Mechanism / Data Persistence	Implement a local JSON backup strategy for quotes when external API fails Quotes API form RAPID API was broken in the playground before implementation but I really wanted a quotes section..	- implemented the broken API knowing it would fail.. - Created structured local JSON data storage - Developed fallback parsing mechanism - Implemented flexible data retrieval strategy	Quotes section with local data backup when RapidAPI fails	- Ensuring data structure consistency - Managing parsing edge cases - Creating seamless API alternative

CSS **(I would have preferred SASS/SCSS but since the JS had to be vanilla I was unsure if that was allowed...)**					
	Dynamic Color Palette System /Advanced Color Management	Implemented comprehensive root-level color variables	- Created extensive color palette variables - Developed flexible theming approach - Implemented consistent color application	Entire application color scheme	- Maintaining design consistency - Creating flexible color management - Ensuring readability across components
	Flexible Spacing System / Responsive Typography and Spacing	Developed a comprehensive spacing and typography scale for different device sizes (common set up for base website implementation)	- Implemented root-level spacing variables - Created adaptive font size system - Developed consistent spacing methodology	Typography and layout spacing throughout the app	- Creating scalable design system - Ensuring consistency across components - Developing flexible layout approach
	/ Advanced Selector Techniques	Utilized advanced CSS selectors for complex styling	- Implemented attribute selectors - Created sophisticated pseudo-class styling - Developed context-aware styling methods	Card interactions, modal styling	- Managing complex styling scenarios - Creating reusable style patterns - Improving code maintainability
	Media Query Strategy / Adaptive Layout Techniques	Developed comprehensive responsive design approach	- Implemented mobile-first media queries - Created flexible grid systems - Developed adaptive layout techniques	Responsive layout across different devices	- Handling multiple screen sizes - Maintaining design integrity - Ensuring consistent user experience
	Advanced Layout Management / Grid and Flexbox uses	Created adaptive grid layout for content	- Implemented `repeat()` and `minmax()` functions - Developed responsive grid techniques - Created flexible	Cards grid, modal layouts	- Managing complex layout scenarios - Ensuring content adaptability - Creating

			content containers - used flexbox within the grid structure where needed		performance-efficient layouts
	Typography Scaling /adaptive typography desivn	Developed responsive typography system	- Implemented `clamp()` for font sizes - Created viewport-based scaling - Developed accessible text sizing	Adaptive Text sizing across different devices	- Maintaining readability - Ensuring cross-device consistency - Creating accessible text presentations
	Interaction Animations	Created meaningful, performance-optimized animations ** see animations row for details)	- Developed custom transition effects - Implemented hover and interaction animations - Created performance-optimized keyframe animations	- Loading animation (page and data for modal) - basic card animations effecting all cards on hover / focus - specific animations on spells cards / modal and potions cards and modal - special animation on patronus page - special animation on the random fact button	- Maintaining performance - Creating meaningful animations - Ensuring cross-browser compatibility
	Transformation Techniques	Implement sophisticated UI transformation strategies	- Developed 3D transformation techniques - Created complex transition sequences - Implemented performance-optimized transforms	Card hover effects, modal animations	- Managing complex animation scenarios - Ensuring smooth user interactions - Creating visually appealing transformations

					-A11Y
	State-Based Animations	Create animations based on user interaction states	- Implemented state-driven animations - Developed dynamic class-based animations - Created interactive animation sequences	Favorite button, modal interactions	- Managing animation complexity - Ensuring intuitive user feedback -Creating meaningful interaction animations
Events					
Navigation Event Handling	Single Page Application Routing	Implemented flexible category navigation system	- Created click event listeners for category cards - Developed dynamic content rendering - Implemented browser history management	Category navigation and page transitions	- Managing browser history - Creating smooth transitions - Handling different content types -naming syntax... I created the patronus page and was stuck on why it was not rendering the data and was deep debugging the data when I realized I had the navigation as `Patronus` and it was expecting `patronus` .....
Modal Interaction Event System	Advanced Event Management	Create comprehensive modal interaction mechanism	- Implemented custom event dispatching - Developed flexible modal content loading - Created dynamic event listeners for modal interactions	Modal pop-ups across different categories	- Managing complex event flows - Ensuring smooth user interactions - Creating flexible event systems
Favorites Toggle Event	Persistent User Interaction	Develop dynamic favorites addition and	- Created event listeners for favorite buttons - Implemented	Favorite button in modals and cards	- Maintaining consistent favorite state - Ensuring

		removal mechanism	state management for favorites - Developed localStorage interaction		smooth UI updates - Managing user interaction feedback
Toast Notification Events	User Feedback Mechanism	Develop dynamic error and success notification system	- Implemented custom toast creation - Created dynamic message rendering - Developed auto-dismiss functionality	Favorites Error and success notifications	- Creating non-intrusive notifications - Ensuring consistent user feedback - Managing multiple notification types - A11Y
Local Storage					
	Favorites / User Interaction Persistent Storage	Implemented robust localStorage mechanism for saving user favorites	- Created type-safe storage methods - Developed comprehensive favorites management - Implemented add/remove functionality with error handling	Favorites page and favorite toggle in modals	- Ensuring data integrity - Managing storage limitation - Creating seamless user interaction
Accessibility					
	Keyboard Navigation: Modal Interaction	Implemented comprehensive keyboard navigation for modal components	- Added focus trapping within modal - Implemented Escape key to close modal - Created keyboard-accessible close button	Modal interactions across application Without implemented focus, a common bug with modals is keyboard trapping users or the fous can change from inside the modal to outside the modal..	- Ensuring complete keyboard accessibility - Managing focus states - Creating intuitive keyboard interactions
	Color Contrast / Inclusive Design	Developed a color system with robust accessibility considerations by starting with	- Implemented CSS variables with accessible color contrast - the app already	Entire application color scheme also implemented	- Maintaining design aesthetics - Ensuring readability

		high contrast color pallete to begin with	comes in a `high-contrast mode` - Ensured WCAG 2.1 color contrast ratios	`prefers-reduced-motion` , `forced-colors: active` etc...	- Creating inclusive color system
Animations					
Loading animation	for initial page load or loading of data is a wand with bubbles. (page load says accio magical knowlege)				
Navigation buttons	general transform / translate on Y for animation movement				
All cards have a loading transition	all cards have a transition loads onto page / card fade-in				
Card Base Hover Effects:	All cards have a left and right angeled transitional animation, and images zoom, each card has an icon in the right corner denoting the category type that is angled as hover active.				
character card	are basic animations listed above				
Special Hover Effects:					
	spell card shave an added dynamic border glow, this glow matches the spell cast color listed on the card.	**spell Modal** has a wand that same wand from loading minus the bubbles and infinite loop for left to right wave. (if I had more time the tip could have changed color dynamically to match card glow)			

	Plants and Elixers: Cards have a bubble effect on hover. These bubbles are spans that I have set to be random in size and drifts to have movement	Plants and Elixers: Modals has no additinal animations			
Accio Random Fact File Button	glows and pulsates while on hover for 2 seconds infinitely				
Patronus list	has a float animation even if the user isn't hovering.				

Project Conclusion

Cloak and Ember is a fully dynamic, accessible, and responsive single-page web application built with HTML, CSS, and vanilla JavaScript.

The application brings the wizarding world to life by dynamically fetching, merging, and displaying characters, spells, potions, and quotes from multiple APIs. It features dynamic navigation, filtering, favorites management with local storage, responsive animations, and accessible, mobile-first design.

A strong focus on **accessibility** underpins the entire user experience, with features like reduced motion support, forced-color scheme respect, ARIA labeling, and accessible notifications. In case of network failures, the application gracefully falls back to local JSON files, ensuring uninterrupted user experience and shows my commitment to creating engaging, inclusive, and resilient web experiences.