Ty MItchell
Final Project: Product Website

Product Documentation

Overview:

RateGames is a website that allows you to write and share about video games with others. Users can do this through either rating a video game and writing a review about it, or creating a list that serves as a compilation of games that share a similar theme. If users find a review or list created by another, they can like it to show their support.

The pages I built were a homepage, reviews page, and a lists page.

The homepage explains the purpose of the website and the features that it provides. It also shows the most active and popular users as a website, along with a review or list that is featured that day. This featured review and list is randomized, and changes every time someone visits the website. There is also a section at the bottom of the page that shows the 5 most popular games at the moment. The main purpose of this page is to give the user information about how the website works, and what users and games are currently popular in the community.

The reviews page allows the user to search up reviews written by other users, and to write a review about a game themselves. Users can write a review containing their rating and a comment about what they thought about a game. Before they post the review users can also preview what it will look like to other users when posted. The reviews page also shows some of the popular user reviews at this time, allowing users to like the reviews that they found interesting or valuable. It also contains some videos of popular reviews written by gaming news outlets and a table of recently posted reviews. The purpose of this page is to allow users to share their own reviews and to see the reviews of others.

The lists page allows users to create lists of games that fit a certain theme. This could be their favorite games, or games that share a similar aesthetic. Similar to the reviews page, the lists page also allows users to preview their list to see what it will look like to other users once posted. Users can also scroll down to see some of the most popular lists that have been posted recently. The purpose of this page is to allow users to share their own lists, as well as to view lists created by other users in the community.

The core message and experience I designed for users is a place where people can share their thoughts with a community of people who care about analysing video games. This is done through the ability to post reviews and lists, as well as the ability to view the posts of others. The like feature for reviews allows users to interact with other community members as well as provide feedback.

Coding Approach and Technical Decisions:

I organized my HTML files into 3 separate files, one for each page on the website. These pages all started with a header which set the title of the website as it appears in the browser, as well as link the CSS style sheet. The header also set up search optimization and a script to link the google analytics script to track how many users are using the site. The body of the HTML was quite different from each page as it sets up the unique layout for that individual page. The only similarity was that the top of each page contained the same navbar and footer. The body content of each page is all organized into divs to section off the different pieces of content, and each page has the necessary JavaScript files linked at the bottom of the body tag.

I only had one CSS file which was used to style the entire website. I organized the file into a couple different sections for general purpose style, styling for individual pages, and styling for page responsiveness. The general purpose styling included several reusable patterns such as ones to make a content box float left or right, one to center a header in the middle of the screen, and one to style buttons to fit with the general theme.

I created 5 JavaScript files which each focused on a different aspect of my website. The first JavaScript file I created was called popular_review.js, which displayed the top 5 users on the website homepage. This was a much simpler file that just utilized DOM methods to get elements and alter their text and values. The second JavaScript file I created was one that displayed a featured review and featured list on the homepage. With this script I optimized my code by making a similar function to pick a random element from a list of 5 values instead of using a separate function for reviews and lists. Along with this function I also created a reusable function which would simply pick a random value given a set range.

The next JavaScript file I created was one which would create a preview of your review before you post it given you click the preview button. This file also provided some error handling features on forms where the form will tell you how long the message should be. To layout this file I first created the functions which would use DOM methods to get the correct elements and updated them. I then used an event handler which would activate these methods given the right conditions. I chose event handlers because they are a more modern approach and very easy to code and understand.

The fourth JavaScript file I created was one which displayed the most popular recent reviews on the reviews page, as well as giving the ability for the user to like the review. To do this I used objects to create the reviews to be displayed. This allowed me to create a function that could display a review on screen by taking in a review object and a DOM node of where the review is supposed to go. This allowed me to optimize my code, as I could create a list of 3 reviews, and 3 DOM nodes, and loop through each of them while calling the function. I also manually added 3 event listeners which would change the like count of their respective review. Looking back I could have optimized this slightly by creating a loop which took in each like button and added an event listener to each of them.

The final JavaScript file I created was very similar in function to the review preview file, except this time with lists. Before a list is posted users can click the preview button which allows them to see what it will look like to other users. This is done by getting the values from the form elements and displaying them in a different format. The way I did this was almost the exact same as the review preview file, so I was able to use a lot of the same concepts with a couple tweaks here and there.

Course Concepts Integration:

I incorporated design concepts by first picking the SDLC plan that best fit what I was trying to build given my skillset. I went with an agile approach as I had never used HTML, CSS, or JavaScript before. Since I was not familiar with these languages, I wanted to see their capabilities before deciding on the details. I also created a general sitemap with an AI tool which I generally followed but didn't have time to fully do everything on it. Another design approach I used was to create wireframes to help me plan out pages of my site. I followed these initially, but as the course went on there were more and more things I wanted to add to each page so the current version differs a lot from what is in my initial wireframes.

HTML course concepts were integrated throughout each of my HTML files. Every file started with a head tag which contained the metadata of the website, while the body tag contained all of the content. I also incorporated multiple different tags in my website to create header, paragraphs, lists, tables, and forms. CSS course concepts were incorporated through creating content boxes with padding and margins. I also used the float attribute to make a left column and right column on each of my pages. Responsive design was implemented through media queries for smaller screen sizes. If the screen would go under 1300 pixels, the columns would collapse into block elements to fit with the new screen size. Under 800 pixels tables would start to get smaller so they don't go off screen and the navbar would collapse into block elements. Under 600 pixels font sizes would increase, as well as images being set to scale with the smaller screen.

JavaScript course concepts were first integrated through the use of dynamic updates to the screen. I used a random number generator to pick a random featured review and list to show up on screen each time the page was loaded or refreshed. Interactive features and event listeners were incorporated through the preview review and preview list features where users could click the preview button to see what their post will look like to other users. The like button on the reviews page is also an example of event listeners causing dynamic updates on the page as the like count increases every time the button is clicked. Form handling was incorporated on the reviews page in the text field where you can enter in a name for your review. When it goes into focus a message pops up letting you know the max length, and if it exceeds the max length when it goes out of focus an error message pops up.

Accessibility was implemented through the use of alt text on images, which gave the image a clear description for screen readers. I also made sure that the color palette for my website was contrasting, with a dark background and light colored text. Every form on the website also has a form label so that visually impaired users can use screen readers to understand what is supposed to be entered into each form. Every interactable element is also navigational with the tab key. I incorporated clear navigation by using CSS to give different effects for links you have visited, not visited, or hovered over. I also put a grey box over the link to the page you are currently on to let users know where they are. Each page also included a general structure with an explanation on what the purpose of the page is, as well as content boxes with the actual features of the page.

Challenges and Problem Solving:

One of the most significant challenges for me was designing my CSS file to help me create a layout for my website. I struggled a lot with figuring out how to create a left and right column setup on each page. Specifically the aspect I struggled most with was figuring out how the float property worked. What worked for me was setting the left column to float left, the right column to float right, and creating a horizontal line tag with a class that clears the float. This way the left and right columns could be set up, and the float right property would be cleared immediately once the content of the columns was finished. This problem gave me a lot of insight into how CSS worked. At first it was very frustrating because I had a clear vision for what my page was supposed to look like but I lacked the coding experience necessary to implement. Through trial and error and learning more about CSS, I was able to become a lot more comfortable with how to create a page layout.

Another CSS related challenge I had was how to implement media queries to make my site more responsive and adapt to smaller screen sizes. The two aspects that gave me the most problems were resizing the tables and getting the navbar to display as block elements moving downwards. For the tables, the solution I found was to create an extra media query where under 800 pixels the table would shrink and the padding and margin would be removed to make extra space. At under 600 pixels the table would shrink more to create a more gradual shrink effect. For the navbar I initially just used one header with each link separated by spaces. To fix the problem I first had to make each link an individual header tag and separate them using margins rather than spaces. I then created a class to make each of them display inline, and set it to display as a block element when under 800 pixels. This kept it looking the same as it did on a full screen size, but now all these headers collapse into one column when the screen size gets low enough. These challenges taught me how to utilize media queries and how to get elements to display inline rather than as block elements.

Another significant challenge I ran into was how to optimize the code for my JavaScript file that picked a random list and review to feature on the homepage. What I came up with was creating a function that took in a list and a random number and picked that element from that list using an if statement. I was still unfamiliar with JavaScript at the time, so looking at this problem I realized I could have done it much more efficiently than what I came up with, especially utilizing objects. This challenge did teach me a lot about the general syntax of JavaScript, and now I feel a lot more confident with how to write better JavaScript code.

One more challenge I ran into was getting values from the post review form for my preview review feature. I struggled a lot with using the right DOM methods to get the values from the review posting form. Specifically it was difficult getting the right rating value as I used a radio button for users to decide the rating. To solve this I ended up looping through each radio button to see if it was selected or not. If it was selected I set the number of stars equal to that value. This taught me a lot about how DOM methods work, and how to extract values from DOM nodes.

Strength and Areas For Improvement:

The part of my project that I'm the most proud of is the interactable and dynamic elements I created using JavaScript. I think these elements make the website feel so much more alive. I really like how when you enter the homepage every time a random review or list shows up. It gives the website more of a community element and shows how users are influencing the appearance of the site. I also really like the preview list and preview review feature. They are very simple but it's nice seeing your input being displayed on a site. This is also shown in the like feature, as the page updates the like count each time you press the button.

I'm also quite proud of the changes I made to the CSS and general styling. During our peer reviews I received some feedback on the CSS elements of my page. Using this feedback I revamped the styling of a lot of elements, particularly in the tables and forms. For the tables I gave them a lot more padding, a different outline, and a blue border. This made it look a lot more interesting and more up to par compared to the styling of the other elements. For the forms I gave them the same background color as the main page but gave it a different outline that differentiated it more from the rest of the website. I also changed the border styling on the buttons to be much simpler, as for some reason the default styling has a 3d type effect to it. This made my website look much more professional and put together, and I really like how it looks now.

I think the part that needs the most work is adding content for the lists page. It feels a little weaker compared to the content of the other pages. I tried to remedy this by adding a preview list feature similar to the review page, but it still looks a little empty. I think given more time I would add a couple example lists from users that you can like. These lists would also include an image of the game for each entry as well. I also think more work could be used on how the website adapts to smaller screen sizes too. Right now it does work and nothing is cut off, but I feel like if I had more time I could make the changes more gradual and not as noticeable between screen sizes. Some changes I could make would be to add a couple more media queries to gradually shrink elements as the screen size gets smaller, making the difference less jarring.

If I had more time, the first thing I would do would be to make a profile page. This is the only page in my wireframe that I didn't end up getting to. This is mainly because I struggled with coming up with how to implement it without having more backend work such as a login or signup feature. I think a potential idea would be to create a form for entering a user name and displaying that on the screen while the user is on the page. Overall though I would really like to learn how to implement some form of a backend to go along with the website. This would allow for your reviews and lists to actually be saved to your account after you post it, which would make the site feel more interactive.