Project Exam 1

Alexander Barrett

Word count 2428

Planning

For this task I used Planyways, a Trello plugin to layout my project plan on a calendar for an easy overview on my progress. With an outline of my weeks as follows;

1. Research, design, and prototype user testing.
2. WordPress setup, research for plugins and building basic webpages plus their components.
3. Wrapping up webpage creations, adding blog content and user testing.
4. Modifications based on feedback, adding extra features and final user testing phase.
5. Wrapping up testing, final checks on code and report writing.

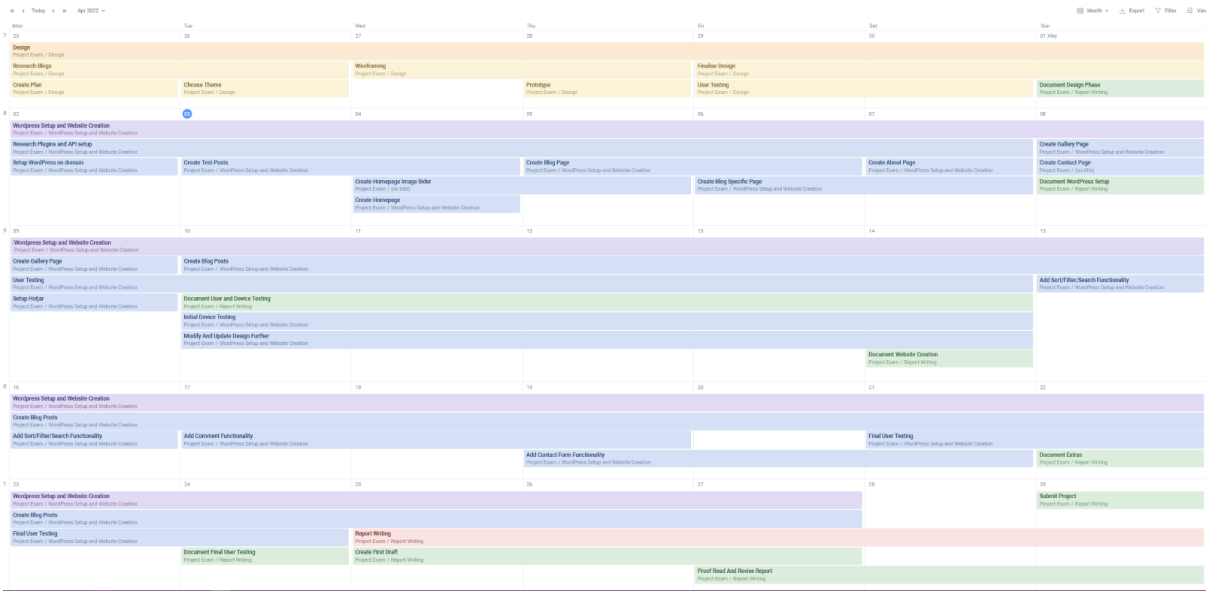

Figure 1. Rough plan for project.

Design

I spent a lot of time looking at blogs initially, I then read an article on First Site Guide titled “What is a Blog?” to get a basic outline of what I needed. I decided on making a light-hearted blog about my dog. My audience was going to be family, friends and anyone that likes dogs wanting a casual read.

This time I prepared my low-fidelity wireframes in XD laying out blocks on a page and re-arranging them. This sped up my design process in making medium fidelity wireframes as I already had the pages laid out.

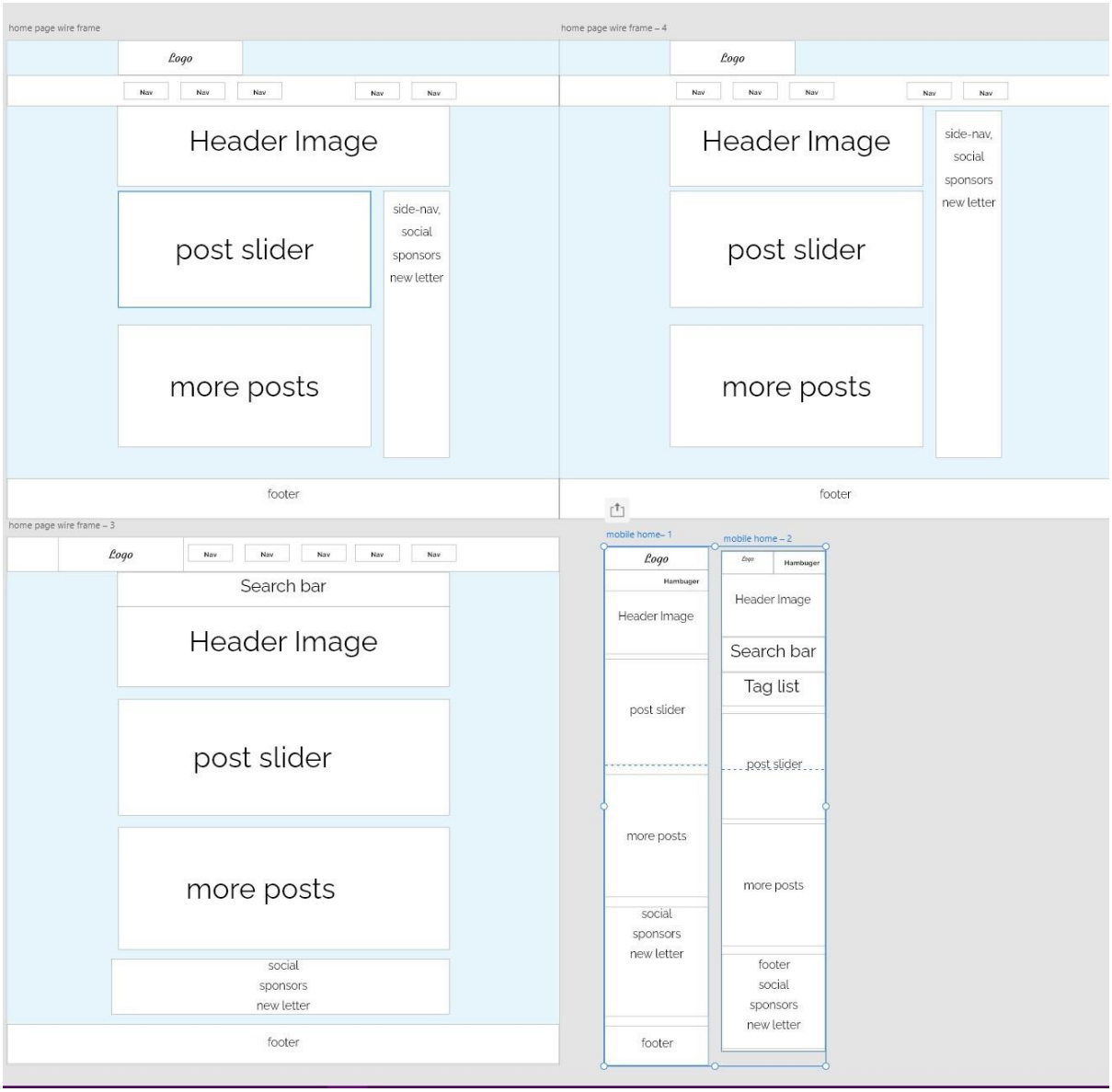

Figure 2. Homepage wireframes.

Once I picked a theme for my blog, I looked at colour pallets on Adobe Color's trends section till I found one that suited my theme, then altered it slightly in the colour wheel. I looked at popular and trending on google fonts and then took a selection and compared them on a page to pick out the ones I liked this time opting for a handwritten heading style for the h1 and h2. I then quickly created a style tile from a previous projects template. I then began increasing the wireframe fidelity, creating template pages with all the main components for each design, so I just needed to connect the main components up, to speed up the prototyping.

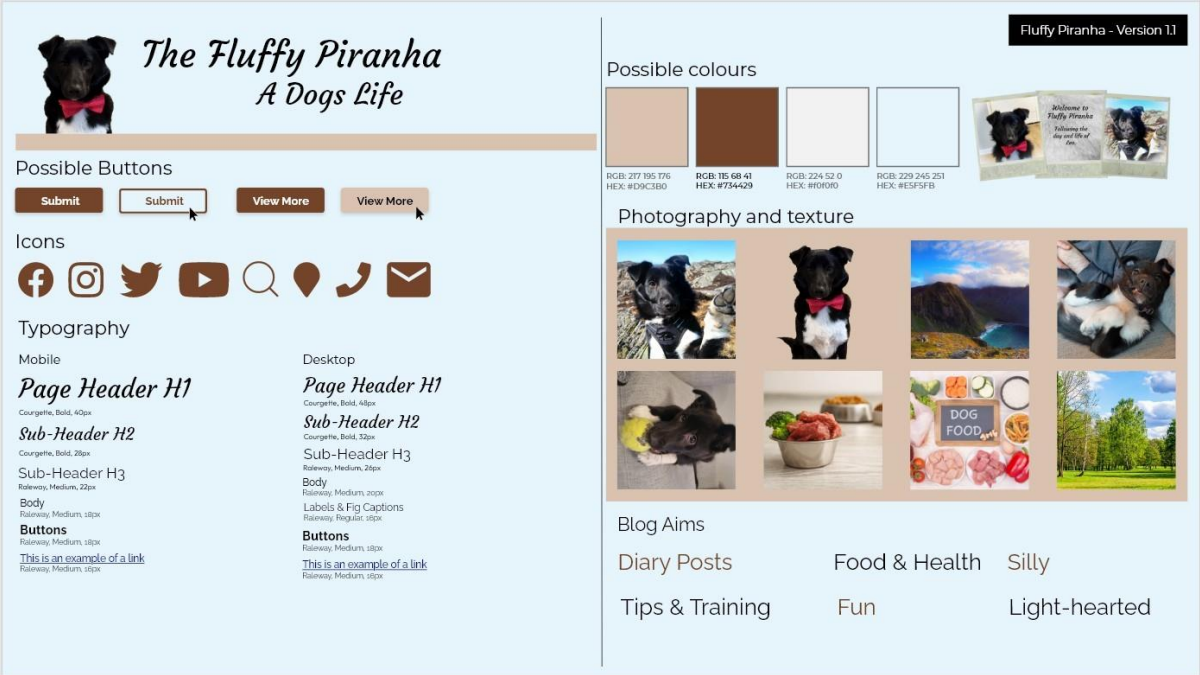

Figure 3. Last version of my style tile.

I struggled to balance the amount of information to give the user about the posts. As well as not putting enough emphasis on the featured image for the posts. User testing of the prototypes highlighted these problems. The summary of my user feedback;

- Pages too busy and needed content to be more space spaced out.
- Bigger post images to draw them to specific articles.
- Side nav bar felt too imposing.
- Banner image on the homepage should be something more Leo specific.
- Nav bar doesn't need the extra options as they go to the same page.
- Option to search as part of the main navigation.

My initial XD prototypes used for testing;
Mobile design version 1
Desktop design version 1

From this feedback, I reduced the clutter on the home and posts pages. A major struggle I had was trying to find an image good enough to use as a banner sadly most of my photos are from camera phones so quality varied greatly and didn't expand well. I instead opted to combine multiple good images in a polaroid picture montage. I increased the picture prominence for each of the posts, and added a little author image that was suggested. I moved the side nav to the bottom of the page, and the search function to the main nav bar. I made a second desktop prototype and gave it to some of my previous testers.
Desktop design version 2

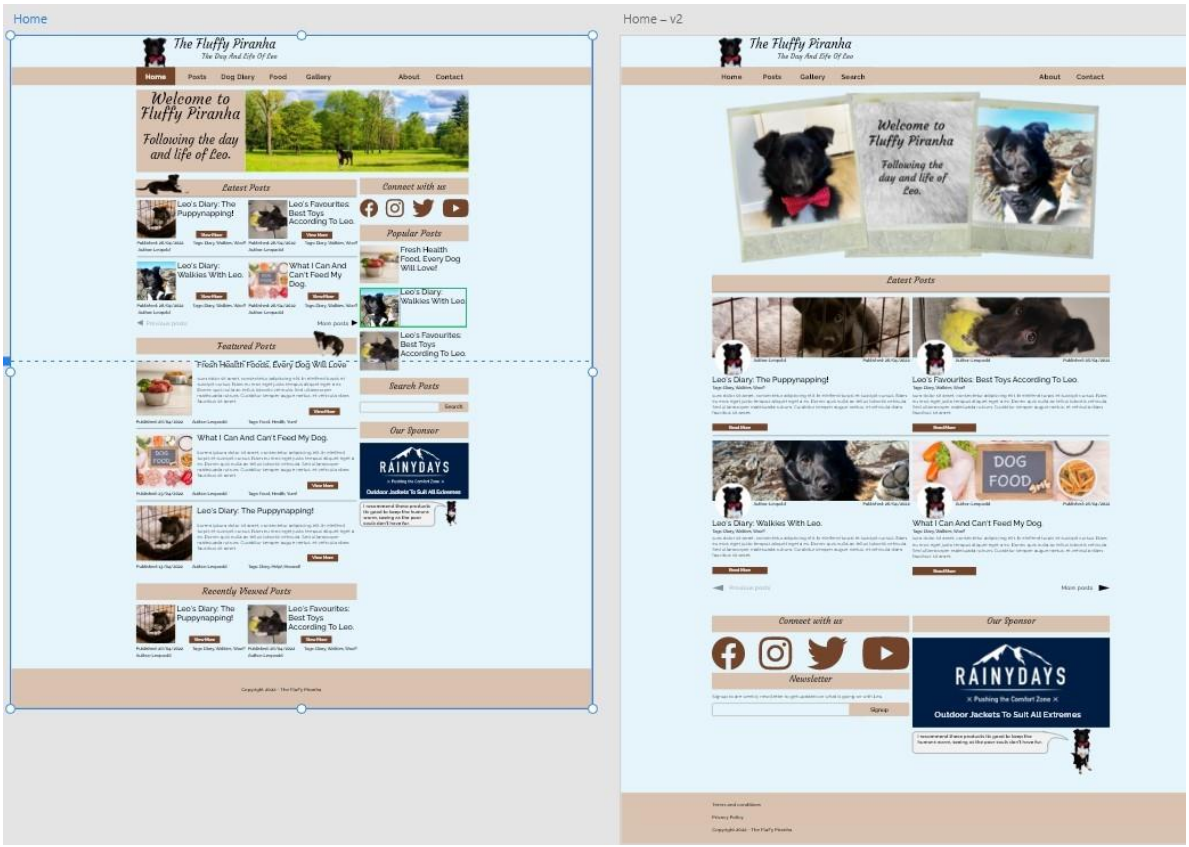

Figure 4. Homepage Design version 1 vs version 2 first of many changes.

What would you do differently next time

In future projects I will do more regular user testing as I progress through the designing phases to better highlight problems and refine the pages before moving to higher fidelity. I would also use different testers in each phase, as many said the second prototype was much better but didn't elaborate much on any new problems, instead comparing its improvements to the previous design. I think I will allocate more time to my design phase for user testing and modifications based on feedback in future projects.

Technical

Setting up the HTML pages, and laying them out with CSS went smoothly. I created a main stylesheet and a separate one for each page, I tried to separate my CSS in to logical sections to make editing my styles easier. With my JS initially I had a file for each page, with reusable functions and constants being imported into them. Later creating a JS file with all the functionality for the navigation and footer, calling it in addition to each pages main JS file.

Getting the data from my API was easy enough most problems came with configuring it to give me what I wanted without additional fetches. I did have some problem with inline styles being injected with my rendered content from pages, but fixed that by using regex after reading a Stack Overflow thread on it.

My aim was to get all the content from WordPress, and make it easy for someone to update that content. I found this a little constraining in what I could realistically achieve on the pages without creating problems for the user or spending a lot of time trying to learn to write my own PHP for WordPress. I managed to get almost all the content off WordPress with the aid of plugins. Posting comments to WordPress was easy enough, once I setup an application key using basic authentication, but I struggled initially to connect my contact form to WordPress through the CF7 plugin. As it didn't accept the object I created for the body, after looking around on Stack Overflow I found a solution in needing to create a formData object and appending the form data to that.

For my slider I looked at some examples notably a video from the Web Dev Simplified on creating a Netflix slider for some inspiration. I then tried to make my own based on the image slider I used in my semester project, adding JS, it took many forms as it evolved through user testing (older versions in the website folder of report documentation). I had some problems injecting a varied number of posts into the slider, getting it to move correctly and then be responsive. I eventually opted for a max size of 20 posts and kept note of the transform percentage, and max allowed percentage. I then used an event listener to detect resize and adjust the slider position near the start and end as well as manage the buttons. I also added code to deal with post amounts under 20 as I was unsure if I would take the time to add 20 or more posts. Once this was done creating an image slider for the gallery page went smoothly, with the aid of a WordPress plugin to allow me to add categories to images and then filter my media call with those categories.

For my posts page I modified the fetch URL for the posts based on query strings, and fetched category data to generate the filter options. For search query data fetches, some details were lacking so I performed a second call with the returned IDs to get all relevant data. I then added an event listener to my filter and sort selectors to modify the URL, fetch and generate the related content.

For the display more button I had a variable for the number of displayed posts, set to display 10 posts initially or the current data length if less and enabled the display more button if there were more result. When the display more button is pressed a function would increase the displayed post variable by 10 or how every many were left to be displayed and add the extra posts disabling the button if there were no more results.

My first image modal code had a problem with the image closing the modal as well as the background. I looked at some examples, and redesigned the modal to grab the src and alt from the target image, using it to create an image in a modal container and expanding it with a separate container for the background set to close the modal.

For my contact page, I validated the inputs as the user typed into them using an event listener to detecting when the user typed, with a series of if else statements to determine which input was being typed into. I then added an orange border to the input as the user typed which turned green when the content met validation, adding an event listener in the process that would turn the input red and display a validation message if the input lost focus before meeting validation. Once all inputs met validation, I enabled the submit button to posted the form, with an extra validation check just in case.

I browser tested as I added code, fixing problems as they occurred, I also borrowed a Mac as having the ability to bug fix on safari's console directly helped a lot. From my user testing, feedback and Hotjar data I created a summary of the main problems.

- Still a bit too cluttered with a need for greater spacing and hierarchy emphasis.
- Text in banner image was not good on all screens.
- Headings look like buttons.
- Slider was too big, not clear it was a slider, too long on mobile.
- The social section too prominent and icons too big.
- Feedback on signup input, cursor pointer on buttons and on images for modal.
- Bigger text, more line spacing, and separation of images from text.

Most of the feedback was easily fixed, just making a responsive slider took some work.

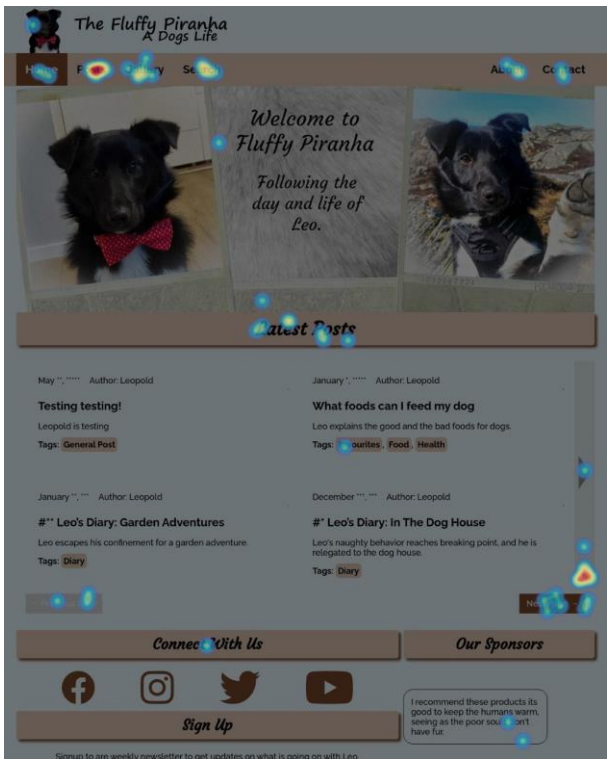

Figure 5. heat map for home page users clicked on heading.

What would you do differently next time

As with the design phase I need to user test earlier in my development I often want to produce a near finished product before showing my work to others, rather than a rough prototype of the site. I also need to be better at creating single purpose functions.

WCAG guidelines, content management and SEO

Content Management WordPress

I initially attempted to modify the responses and replace WordPress block content classes with my own, I had some initial success but ended up corrupting the output from the API and causing errors as I change multiple outputs. Not being overly confident at this point in PHP, I changed my focus to finding plugins that could get me the outputs I wanted. Through

trial and error, I eventually found a selection of plugins that worked together. As well as some additional plugins that would allow me to create a better user experience for the blogger.

Adding content to WordPress once setup was easy, a user can simply fill out require ACF entries fields and use the built-in block editor with my custom blocks for images and text boxes to add posts. I created pages for the home, about and contact, a user could update content in these and it would be reflected on the website. The homepage featured image would change the banner image, and content would be displayed below in an empty container, about page displayed the block editor content, and contact page filled out the additional details section. I also add a custom post type for sponsors, with ACF fields, so a user could add and remove sponsors easily. For my images I tried to keep the image sizes as low as possible without them looking pixelated when expanded.

Figure 6. Custom image block for WordPress

For posting comments I used WordPress's basic authentication with an app key after watching a video by LearnWebCode, on "WordPress REST API Authentication". I created a subscriber account on my WordPress with permissions to only post and then created an app key for the account using the username and key in my post requests. For the contact form I use CF7s endpoints to post to a contact form template, and used their Flamingo plugin to create an inbox for the user to see.

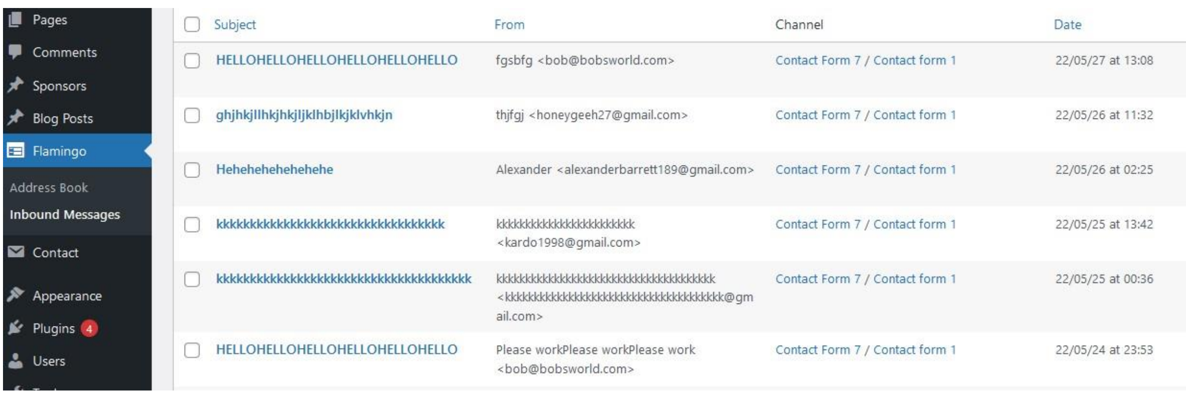

Figure 7. My Flamingo mail box with very important mail.

WCAG and SEO

I tested my colour pallet in a contrast check to ensure there wouldn't be any problems going forward adjusting to get a balance between what looked good and was easy to read.

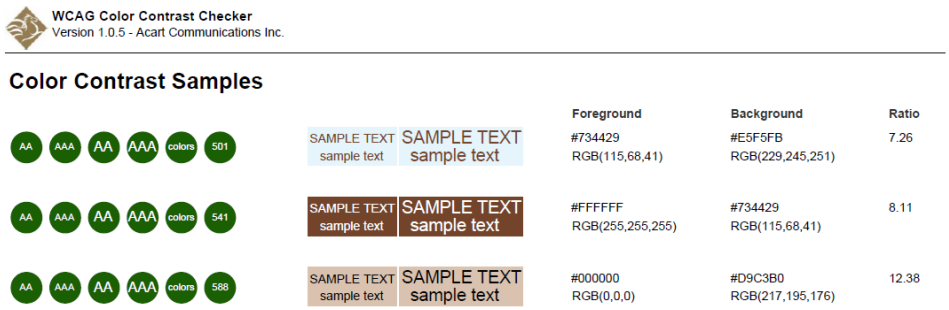

Figure 5. Colour contrasts checks.

I made sure every html page had their title, meta description and h1 heading hardcoded even if this would be overwritten by the API fetch data later. Using the validators from W3, WebAIM's wave tool and chromes lighthouse to check for errors in code and accessibility. I also flicked through my blog specific posts to check all the image alts came through.

I used some icons from font awesome and was pleased to see that their kit allowed me to add a title attribute to an icon for a screen reader. I added a skip to main content link at the top of all my pages. I added some visually hidden labels to the menu button and search input for a screen reader too after reading an article on WebAIM "Invisible Content Just for Screen Reader Users".

For search icon I had to make extra addition during my keyboard testing due to it not being button elements. I had to add a tab index to it, and then needed to add some JS to allow a keyboard user to interact. Reading an article on Dev, "When role="button" is not enough", to learn how to achieve this. I also move the search icon in the nav list order, so the next thing you tabbed to would be the input and submit button. I then tabbed through the page and found slider allowed the user to tab through all the posts without the need for the buttons so added a negative tab index to them, I then added a negative tab index to the images on my large posts to avoid the double tabbing through each item.

What would you do differently next time

I think in future project I will consider the use of icons as buttons carefully considering the extra work needed to make them accessible. I was lucky with my slider design this time but when designing sliders in the future I should consider how to make them easily accessible, and it is debatable as to if a user would want tab through 20 posts to get to the next section so maybe a skip link should be added. In the case of WordPress getting more familiar with its inner workings and building my confidence in using PHP would make things easier.

References

Image modal

Dev Ed, “Fluid Image Pop Up JavaScript Tutorial”, 26 Jun 2020, https://www.youtube.com/watch?v=4SQXOA8Z-lo [accessed May 2022]

Dev, “Create an Image modal with JavaScript!”, Saleh Mubashar, 31 Dec 2021, https://dev.to/salehmubashar/create-an-image-modal-with-javascript-2lf3 [accessed May - 2022]

W3 Schools, “How TO - Modal Images”, https://www.w3schools.com/howto/howto_css_modal_images.asp [accessed May - 2022]

Post Slider Inspirations

Web Dev Simplified, “Can I Create Netflix’s Video Carousel UI?”, 12 Apr 2022, Can I Create Netflix’s Video Carousel UI? [accessed May - 2022]

LogRocket, “Build an image carousel from scratch with vanilla JavaScript”, April 12, 2022 https://blog.logrocket.com/build-image-carousel-from-scratch-vanilla-javascript/, [accessed May - 2022]

W3 Schools, “How To JS Slideshow” https://www.w3schools.com/howto/howto_js_slideshow.asp, [accessed May - 2022]

APPCODE, “15 CSS Image Slider Examples and Code”, September 12 2021. https://appcode.app/15-css-image-slider-examples-and-code/ https://appcode.app/ [accessed: Dec – 2021]

From article I focused on the code from these 4 examples for inspiration;
Dudley Storey, “HTML CSS Driven Responsive Image Slider”, https://codepen.io/dudleystorey/pen/kFoGw

Joshua Hibbert, “Pure CSS Featured Image Slider”, https://codepen.io/joshnh/pen/KwilB
Sandra Vos, “Image Slider CSS Only”, https://codepen.io/sandra90/pen/xwMGgB

Mayur Birle, “Pure CSS3 Responsive Slider”, https://codepen.io/mayurbirle/pen/eEevBZ

Chris Coyier, “CSS-Only Carousel”, Jan 10 2020. https://css-tricks.com/css-only-carousel/ [accessed: Dec – 2021]

Contacts Form

Stackoverflow, “What Parameter Contact Form 7 using JSON to sent using API”, Jun 24, 2019, https://stackoverflow.com/questions/56731006/what-parameter-contact-form-7-using-json-to-sent-using-api/61956314#61956314 [accessed: May - 2022]

Comment Posting

WordPress REST API Authentication: Application Passwords
LearnWebCode, “WordPress REST API Authentication: Application Passwords”, 30 Mar 2022,
https://www.youtube.com/watch?v=e_thybKPKHc, [accessed: May - 2022]

Removing Inline styles
Remove style tag from rendered content
https://stackoverflow.com/questions/52212030/js-how-to-remove-style-tags-and-their-content-from-an-html-string-using-regula

Web accessibility research

WebAIM, “Invisible Content Just for Screen Reader Users”, Last updated: Sep 25, 2020.
https://webaim.org/techniques/css/invisiblecontent/ [accessed: Mar – 2022]

Benjamin Johnson, “Accessible icon buttons”, Sep 02, 2020
https://benjaminjohnson.me/accessible-icon-buttons [accessed: May - 2022]

Dev, “When role="button" is not enough”, 18 Sept 2020, https://dev.to/tylerjdev/when-role-button-is-not-enough-dac [accessed: May - 2022]

CSS-Tricks, “KeyboardEvent Value (keyCodes, metaKey, etc)”, Updated on Aug 14, 2019
https://css-tricks.com/snippets/javascript/javascript-keycodes/ [accessed: May - 2022]

W3 Schools, “How TO - Trigger Button Click on Enter”
https://www.w3schools.com/howto/howto_js_trigger_button_enter.asp [accessed: May - 2022]

Research and Inspiration

First Site Guide, “What is a Blog? – Definition of Terms Blog, Blogging, and Blogger”, Updated: March 24th, 2022, https://firstsiteguide.com/what-is-blog/ , [accessed: April, 2022]

Feed Spot, “100 Best Dog Blogs” May 09, 2022, https://blog.feedspot.com/dog_blogs/, [accessed: April 2022]

First Site Guide, “50+ Best Examples of Popular Blogs in 2022”, Jan 10th 2022,
https://firstsiteguide.com/examples-of-blogs/ [accessed: April 2022]

Passion WP, “50+ Most Popular Blogs in 2022 Across 10 Categories”
https://passionwp.com/most-popular-blogs/ [accessed: April 2022]

Tools, Plugins and Resources

Adobe colour wheel tool
https://color.adobe.com/create/color-wheel

Contrast Checkers
https://contrastchecker.com/
https://webaim.org/resources/contrastchecker/

Waves web accessibility evaluation tool
https://wave.webaim.org/

W3 validators for HTML and CSS
https://validator.w3.org/nu/
https://jigsaw.w3.org/css-validator/

Wordpress Plugin
Genesis Custom Blocks, to code my own blocks.
https://wordpress.org/plugins/genesis-custom-blocks/
Custom Post Types, add custom posts.
https://wordpress.org/plugins/custom-post-type-ui/
Advanced-custom-fields, add custome data entries to posts and pages.
https://wordpress.org/plugins/advanced-custom-fields/
Add categories to images, allowing filtering on media by category.
https://wordpress.org/plugins/wp-media-library-categories/
Custom Rest API Generator, allowing me to get custom taxonomies, categories, and featured images.
https://wordpress.org/plugins/wp-custom-rest-api-generator/
Contact form 7, used for posting contact for to wordpress.
https://wordpress.org/plugins/contact-form-7/
Flamingo, creates an inbox for user to see contact form feedback.
https://wordpress.org/plugins/flamingo/
Headless mode, removes front end for wp.
https://wordpress.org/plugins/headless-mode/