Sip List London Master Project

Arianna Kern

11 December 2025

Abstract

My aim in creating the website Sip List London was to provide a single domain where people can find and access multiple wine bars in East London. While developing the site, I visited numerous wine bars across various neighbourhoods in East London. I documented my time at each bar with photography and storytelling that captured its ambiance. I built my site on WordPress, using HTML5 for the structure and CSS for the styling. I focused on creating a layout that felt clean, intuitive, and approachable, allowing all users to quickly find the type of wine bar they are interested in. The finished product is a site that was built with a mobile-first approach and used progressive enhancement so it works well on any device. I wanted the site to feel like a helpful guide for anyone exploring East London, making it easier to discover new places and feel confident about where to go.

1. Introduction

Context of your Masters project/Purpose of the website

Upon moving to London, I had been working in multiple wineries in the Willamette Valley in Oregon. I was excited to try new European wines, and more specifically British wines. I quickly found that the wine bar scene in East London was quite prominent. Although wine bars are popular in East London, there was no way to find or locate them all. In Oregon, it was quite easy because all the wineries are in one area, and you can drive or walk down the street, and you will simply bump into the next one. Whereas here, they are spread throughout the city. I took this on as a challenge. I knew many like me were interested in visiting wine bars, but did not know where they were or how to tell them apart. The purpose of my site was to create a space where the idea of 'bumping into the next wine bar', similar to my experience in Oregon, became possible in a much bigger city. This personal experience became the foundation for my project, informing both the research and design decisions behind creating a user-friendly digital tool for discovering wine bars across East London.
I realised early on that this website was about more than just mapping locations–it was about understanding how people choose spaces. Discovering what is going on within the bar beyond the wine, what is unique and appealing about each space. By visiting each bar in person, I began paying attention to atmosphere, layout, and the types of people who spent time there. These observations helped me think critically about how each bar would be uniquely presented on the website, ensuring that users could understand the experience that they could expect. This research shaped the direction of my content strategy.

Why WordPress (industry relevance, accessibility, scalability)

I chose to use WordPress for my website as I was able to use the skills I have already learned in CSS and HTML and apply them in a new format. I also wanted to get familiar with WordPress as it is an industry standard and something that I can carry with me into my future career. Future-focused, I would like to work on the design and accessibility of websites, and I believe working in WordPress would be the most relevant way to expand my skill set. I initially created sketches of my desired layout in ProCreate, but then quickly moved to Visual Studio Code, where I could create my initial layouts.The foundations of my website were built in both HTML and CSS, breaking through what worked and didn't work in my preliminary sketches and colour codes. After creating the baseline pages of my website, I moved over to WordPress.

What you aimed to learn

I aimed to learn how to use a completely new CMS that I could carry with me into my future career. Entering this project, I understood HTML and CSS, but did not know how to translate that into CMS. I chose WordPress as I knew it would benefit me long term, especially as it is used widely across the industry. Alongside this transition, I wanted to incorporate SEO and learn when, where, and how to add those elements effectively. I understood their importance before starting, but this project gave me the space to explore them in a more practical and intentional way. Strengthening my skills in structuring

content, adding metadata, and improving search visibility felt essential as I am interested in moving into User Experience design. UX includes visual layout, accessibility, information, and interactivity. So, combining CMS knowledge with SEO was a meaningful step towards UX design.
As I continued building my site, I became more aware of how each plugin affected different aspects of the site itself. Going into building my site, I always kept in mind to keep a limited number of plugins–only the necessary ones. Knowing that themes, plugins, and navigation patterns will affect the overall usability of the site. I found myself thinking more critically about how users move through a website and how to make that journey more intuitive. Making sure every move and piece of the site felt purposeful, functional, and aligned with my long-term goals in digital design.

Overview of the structure of the thesis

This thesis is divided into seven sections. Section one introduces the concept and motivation behind Sip List London. Section two discusses initial technical decisions and layouts. Section three describes my typography choices and design logic. Section four details converting static code to WordPress and the development of understanding themes vs. templates. Section five is a critique of working and troubleshooting with Elementor. Section six presents plugins, performance and SEO. Section seven is a reflection on what I would have done differently if I were to do this project again. Section Eight is my conclusion.

2. Initial Technical Decisions

2.1 Choosing JPEGs and Image Optimisation

Why JPEGs for web (lightweight, fast loading)

JPEGs use lossy compression, which significantly reduces file size compared to PNGs, making them ideal for photographs where slight quality loss isn't noticeable. Throughout my website, I used JPEGs for all images except my illustrations. I wanted to create a fast-loading site, as many people will be using this website on the go while looking for wine bars in London. This makes pages load faster, especially on mobile devices or slower connections. Since I designed this website mobile-first, I focused heavily on fast-loading options. Especially when it came to images and illustrations, as my website contains many of them, leading to longer loading times. JPEGs and PNGs are widely considered the standards for browsers and are universally supported, ensuring consistent display for all users.

Comparison to PNG / WebP

While PNGs offer lossless compression and are better for graphics with sharp edges or text, they create much larger file sizes for photographs. WebP offers better compression than both JPEG and PNG while maintaining quality, but I chose not to use it due to limited platform and device compatibility. As my site is meant for both London locals and tourists, I want my website to be accessible to everyone. Images are one of the most important assets of my website, as they showcase the interior and ambience of each wine bar, which is important for everyone to see when visiting the site. For my limited number of illustrations, I used PNGs to preserve their crisp quality since the performance impact was minimal.

How image compression impacted performance

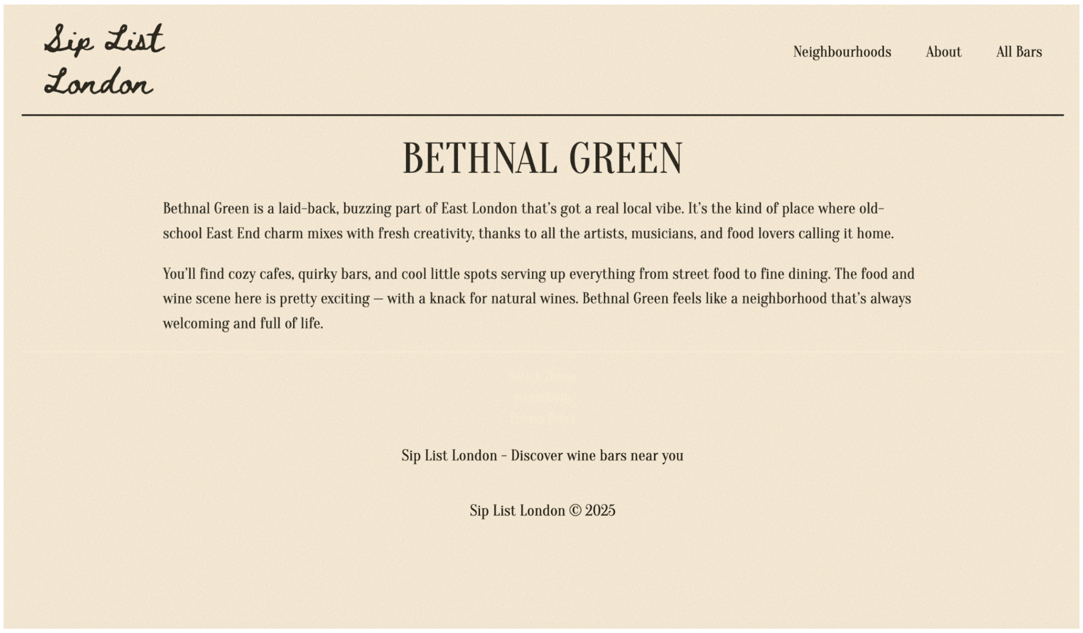
By compressing my wine bar photos as JPEGs, I reduced loading times without visible quality degradation to users. While WebP may have offered an even better compression ratio and smaller file sizes, the performance and quality gains were substantial with JPEGs. My photos still looked professional and atmospheric, loading quickly enough to keep users engaged across all platforms with JPEG compression. This was particularly important as many users might be loading my site on cellular data rather than WiFi while exploring East London. Additionally, using JPEGs instead of WebPs meant I did not need to create fallback options, as it ensured consistent performance across all devices. The trade-off between file sizes was very clear for this type of website and the ways and locations it would be accessed. Smaller file sizes from JPEG compression mean the site is more accessible to users with limited data plans or slower internet connections. Faster image loading also benefits users of assistive technologies, as they don't have to wait as long for content to appear and be read by screen readers. In some areas of East London, patchy coverage is standard, and JPEG compression ensures that users with poor connections still have full access to site information without images failing to load.

Accessibility considerations

In my accessibility consideration, I worked to keep every user in mind. My design choices focused on colour contrast, mobile-first design, image optimisation, and performance considerations. I achieved WCAG AAA compliance with a 14.17:1 contrast ratio. This was highly important to me as I was testing and deciding on my final colours and typefaces. When working on layout, I designed mobile-first, as that is the standard way most people visit websites, and it is easiest for a layout to go from mobile to desktop. I ensured that touch targets were appropriately sized and my hamburger menu was clear and uncluttered. All of these performance considerations ensured that my site works across all devices.

2.2 Designing the HTML/CSS Layout First

Your initial static prototype



Sip List London

Neighbourhoods About All Bars

BETHNAL GREEN

Bethnal Green is a laid-back, buzzing part of East London that's got a real local vibe. It's the kind of place where old-school East End charm mixes with fresh creativity, thanks to all the artists, musicians, and food lovers calling it home.

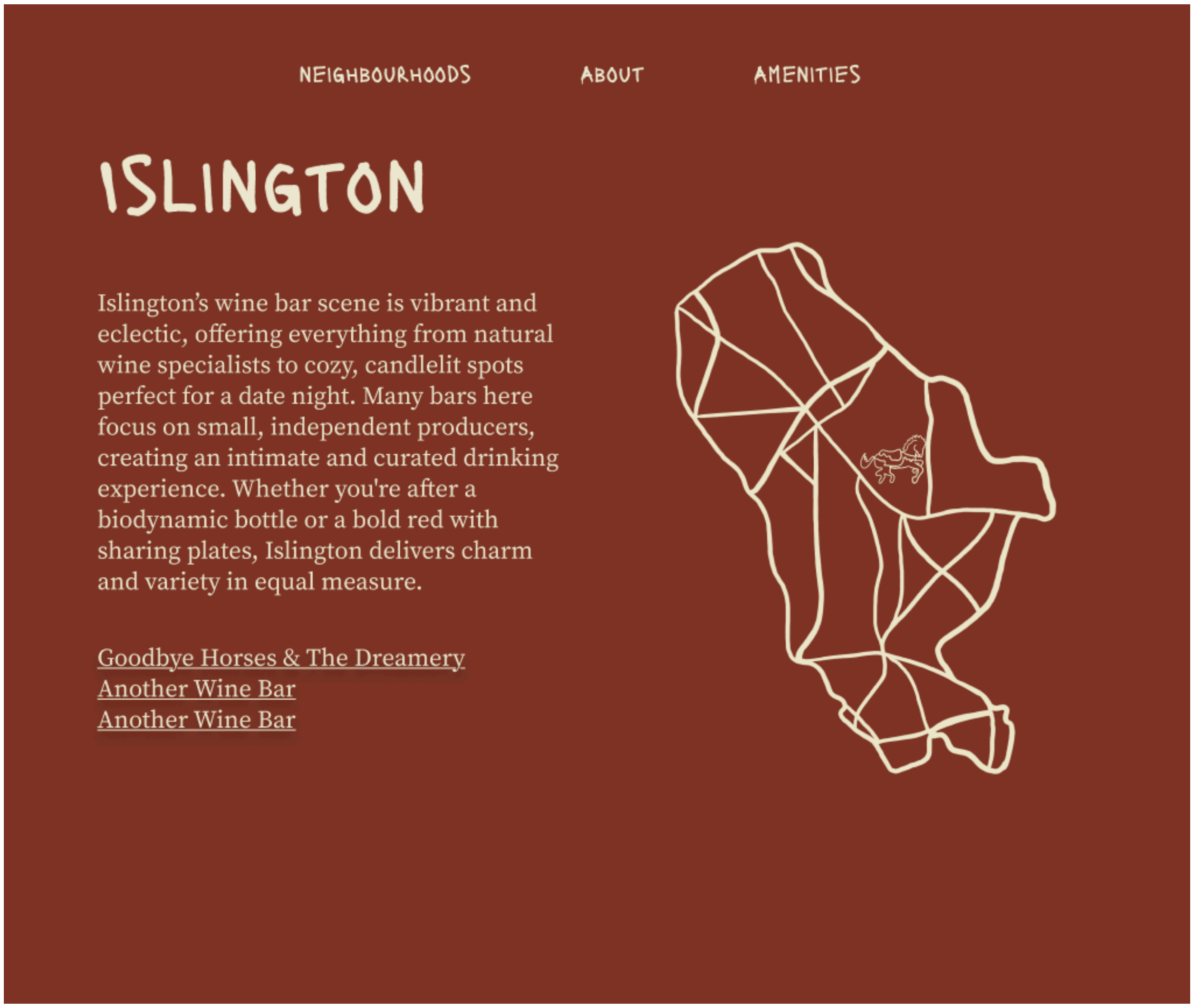
You'll find cozy cafes, quirky bars, and cool little spots serving up everything from street food to fine dining. The food and wine scene here is pretty exciting — with a knack for natural wines. Bethnal Green feels like a neighborhood that's always welcoming and full of life.

Sip List London - Discover wine bars near you

Sip List London © 2025

SIP LIST
LONDON

I followed my initial static prototypes that I mocked up in HTML/CSS layout. These layouts, as pictured above developed a lot from my original sketches in procreate and Adobe illustrator that I later incorporated back into my design. When I made my first design in Illustrator I created two templates. I created one template of a neighbourhood homepage and one bar page. While the theme of these pages carried through many things changed in design, based on feedback I



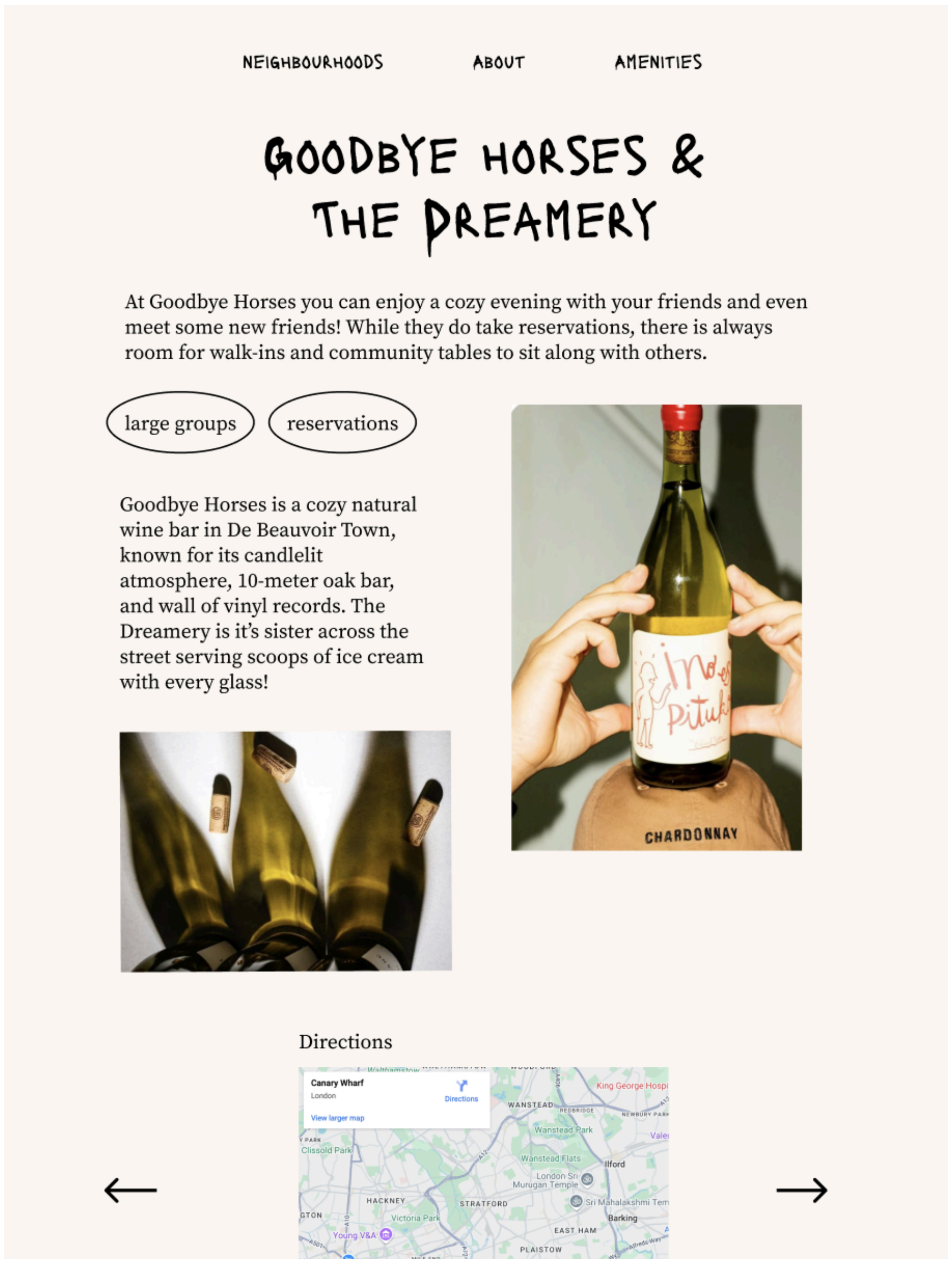
ISLINGTON

Islington's wine bar scene is vibrant and eclectic, offering everything from natural wine specialists to cozy, candlelit spots perfect for a date night. Many bars here focus on small, independent producers, creating an intimate and curated drinking experience. Whether you're after a biodynamic bottle or a bold red with sharing plates, Islington delivers charm and variety in equal measure.

Goodbye Horses & The Dreamery
Another Wine Bar
Another Wine Bar

received in my class critiques.

Example of Neighbourhood page



GOODBYE HORSES & THE DREAMERY

At Goodbye Horses you can enjoy a cozy evening with your friends and even meet some new friends! While they do take reservations, there is always room for walk-ins and community tables to sit along with others.

Goodbye Horses is a cozy natural wine bar in De Beauvoir Town, known for its candlelit atmosphere, 10-meter oak bar, and wall of vinyl records. The Dreamery is it's sister across the street serving scoops of ice cream with every glass!

Example of Wine Bar Page

The feedback I received on these two screens and presentation were that of:

> *"The visual direction is not clear (to me) now, I am not sure about that heading font. I have a feeling it won't quite fit the overall feel for the website and could be tricky to read. I am not sure which direction this design is going in"*
>
> *"I really liked all the graphics you showed, all could work well as a direction for the visual design. I cannot say which one will suit your project best - but I think you do need to make a choice whether to go with more traditional etching style graphics, or modern and simple/minimal line"*
>
> *"Your earth tones are lovely, very cosy. I also enjoyed seeing your illustrated map, I think this really jives with your magazine-esque concept you've talked to. I echo the comments about the line weight. However, I also question the use of icons to signal the different bars (even if clickable), because you put the burden on the user to figure out which is which. Numbering might be more clear and helpful, BUT you could maybe re-use the logos in another way, e.g. on the bar listing pages themselves to give the bars their own visual identity."*

With this feedback I received on *Tusk* and similar commentary in class provided during class critiques, enabled me to refine and elevate my initial design concept. My revisions focused on enhancing accessibility and ensuring it could be used by a greater audience. Although the design began with simply choosing a typeface and colour palette that I liked, following WCAG accessibility tests and in-class evaluations I came to understand that they were not functional.

During a review of my work a week before submission, a classmate redirected my attention to my original neighbourhood layout, which I had presented in the final class critique. She mentioned that integrating the text alongside the illustration could help the user clearly perceive the map's intended interactive use, and, beyond that, give the page more structure.

How coding from scratch helped clarify design/Benefits before moving to WordPress

Moving from my original sketches to coding in HTML/CSS helped me clarify my designs by showing me how they could logically function on a website. Writing in HTML5 structure from scratch had me think about semantic markup and logical page hierarchy before visual design. At the same time, hand-coding CSS made me understand the mechanics of layout. I understood box models, positioning, and flexbox going into this project, but I understand when and where to use them—for example, finding ways to use a flexbox in a desktop layout but not in a mobile layout for the header to display correctly. With CSS, I already had a lot of my design laid out, so when I moved it into my WordPress CMS, changes were easy to identify in the labelled CSS. Creating layouts from scratch helped me recognise which design elements were structural vs. decorative, guiding my WordPress page-building decisions. This made it simpler to add styles to either my style.css or Additional CSS in WordPress, as I could quickly recognise conflicting rules between the two locations.

3. Typography & Design Logic

3.1 Using theme builder + additional CSS

Using theme builder + additional CSS

While working in WordPress's theme customiser, I quickly learned how different styling layers interacted. I used the Customizer's Additional CSS section for simple and targeted changes that would override the base theme without editing core files. This was an adjustment, as I was used to having only one CSS file, style.css. For broader structural changes site-wide, I worked directly in the child theme's style.css file, which gave me better organisation for my foundational styles, such as typography, colours, and layout. The distinction between these two locations became clear through practice and through using WordPress. I was quite confused as to what the purpose of Additional CSS was at first. Additional CSS was great for minimal tweaks, padding adjustments, hover states, or small mobile layout structural issues. It is loaded within the theme's main stylesheet, so it naturally has a higher specificity. Meanwhile, style.css holds my core design system. It holds my colour palette #F9EDD6 and #231F15 and typography rules. While it took me a while to understand the difference between style.css and Additional CSS, once I did, I discovered just how helpful Additional CSS is.
Understanding how WordPress prioritises styles was crucial. The cascade flows from theme defaults, to style.css, to Additional CSS, and finally to any inline styles. Knowing this hierarchy meant I could strategically place rules where they'd have the intended effect. If I had grasped this cascade earlier within my first WordPress endeavour, there might be fewer unnecessary !important declarations throughout my code. When Elementor was in the mix of my design, it added another layer of complexity with its own style rules. It required even more forceful overrides in Additional CSS to maintain design consistency.

Ensuring fallback cursive font

Typography played a significant role in establishing Sip List London's approachable, personal aesthetic. Originally, I had designed a logo specific to Sip List London, but in some of my class critiques, I received feedback that it did not fit the wine aesthetic, and as I changed my other types and colour palette, I decided to change my logo as well. Prisca and David both mentioned that wine bottle labels often feature cursive writing, and suggested following the theme of my site's logo. I chose the cursive font Beth Ellen for my logo to evoke the handwritten logo appeal, the same as a nice bottle of wine.The serif typeface used throughout the whole of my site is Oranienbaum; it is used in all of the headers and body text on my website. Both of these fonts are downloaded from Google.
If my logo font fails, I implemented a font fallback option font-family: 'PrimaryFont', 'Georgia', cursive;. This CSS rule tells the browser to attempt loading my primary custom font first, then fall back to Georgia if all else fails. While Georgia is not cursive, it maintains a decorative quality that preserves the intended tone better than falling back to generic sans-serif fonts.
If my body font fails, I implemented a similar fallback flow. I chose Times New Roman as my fallback font in case it fails. Which, unlike Georgia to Beth Ellen, is quite similar to my chosen serif typeface, Oranienbaum.

Accessibility considerations

Accessibility shaped every design decision for Sip List London. Beyond the WCAG AAA-compliant 14.17:1 contrast ratio between my dark brown text (#231F15) and light beige background (#F9EDD6), I considered how different users would interact with the site. The high contrast colour palette was not just about meeting standards - it ensured readability. There are many contexts in which readability is necessary. Some users may access the site in bright outdoor conditions, making visual clarity essential not only for individuals viewing the interface in sunlight but also for those with visual impairments.
While cursive and serif typefaces are not always considered legible. I ensured that they were not just being used for decorative purposes. The cursive font stays solely in the logo, so that it is large enough to be legible and only used minimally. And the body font is a legible serif that is set at a large enough pixel size that the user can view it and tap it on their phone easily and clearly. I avoided using colour alone to convey information - if something was clickable or important, it was also indicated through size, weight, or position. And the options on my site that are clickable on my interactive map are also easy to find in other locations of the site that are inaccessible to the user.
The mobile-first approach was a decision driven by accessibility. The hamburger menu provides a clear, uncluttered navigation that does not overwhelm on small screens. Image compression ensured the site loaded quickly for users on slow connections or limited data plans. Alt text for images described not just what was visible, but the atmosphere I wanted to convey. Helping screen reader users understand what each photo represented.

Challenges getting it consistent across Elementor + theme

Inconsistency stemmed from how Elementor injects styles.
Inconsistency stemmed from Elementor injecting its own highly specific styles that worked to override my theme CSS. This caused my style.css rules to be ignored. To avoid or counteract this, I had to double down on Elementor CSS and Additional CSS, using specific selectors and many !important rules throughout my code. This made my code extraordinarily heavy and messy.
Mobile responsiveness added another layer of complexity - Elementor's responsive controls sometimes contradicted my CSS media queries, causing layouts to break unpredictably. Eventually, these frustrations built up and led me to dead ends in creating the site that I wanted to build. It ultimately led me to abandon Elementor entirely and rebuild using WordPress's standard block editor.
While this meant recreating pages, it gave me complete control over the CSS. The block editor kept my theme's styles intact while still offering flexibility through Additional CSS. Learning by using Elementor is a simpler, more basic approach with direct code control that produces more reliable, maintainable results.

4. Converting Static Code to WordPress

4.1 Understanding Themes vs. Templates

How WordPress's architecture works

Every page in static HTML/CSS existed as a complete file, and what I had coded was exactly what was displayed. Then moving that template and theme to WordPress was a major shift in learning to edit pages, knowing my theme was still intact, while I could not see it as my edits were being made. It was storing it, and I could test it by previewing, but some images or even my text didn't look the same, which was unsettling at first. This actually turned out to be helpful because it led me to focus on the content first. Knowing that my theme was already there, but I did not have to focus on it as I was inputting my already prepared content.

Your learning curve

The transition from my original layout to WordPress CMS was daunting because I was unfamiliar with WordPress, but it quickly became very engaging and easy to understand. WordPress was most confusing to me in the initial stages, as the site kept inviting me to use Elementor for various projects. I did engage with Elementor at the beginning because it seemed simple and useful, but it ended up being the complete opposite. Elementor was causing structure placement issues and multiple items to lag. It ended up creating more difficulties than it was worth, but it presented itself as a simpler layout. I quickly dropped the pages I had created with Elementor and found ease in creating my own layout and injecting my code into WordPress CMS.

4.2 Linking Themes to Pages and Posts

Template hierarchy and themes

The WordPress system uses PHP files to control how different types of content display. The theme I added to the website is the framework and backdrop. This framework is the template that contains my header, footer, individual pages, and more. When moving from static code to WordPress, I had to be cautious that my personal HTML structure would now be split across multiple template files (header.php, footer.php, page.php, etc.) and that they would all load together.

Issues with theme conflicts

I did not seem to encounter many issues with my theme. I started with a Vanilla theme layout and added all my personal themes to its basic structure. In the beginning, it was smooth sailing, and it worked beautifully. It was only after time that I had issues with my theme, and those issues were more specifically

battles of specificity. The two main issues I ran into were with transitioning from mobile to desktop and image sizing on different pages. My theme stayed consistent, but sometimes that itself was the issue.
As my pages grew, they tended to move images in ways that I did not intend. If they were organised in blocks, they would turn into columns or stretch. The responsive behaviour worsened these issues. An image that looked perfectly normal in the editor and again on mobile would morph into something else on the desktop. It began stacking items oddly. While this is something I ran into with static HTML/CSS as well, I had to learn how to approach the issue with which block types were moving and which ones would actually respect my layout intentions rather than imposing their own responsive logic.

4.3 Using Parent/Child Page Structure

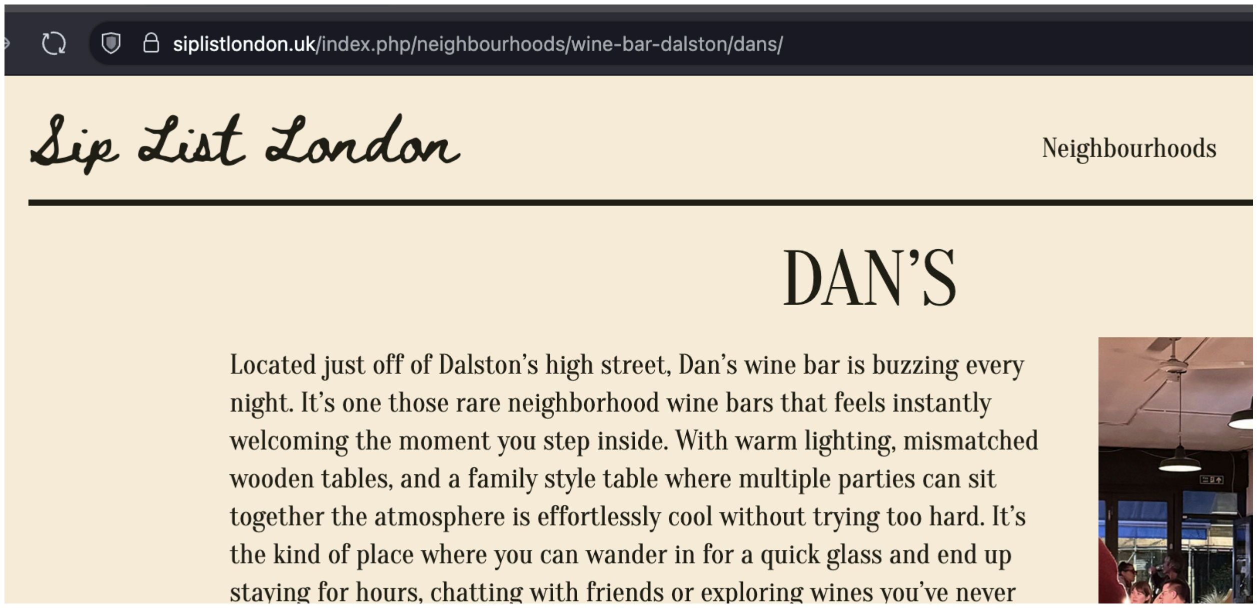
SEO advantages

The parent/child structure significantly improved the site's search engine optimisation. By organising each neighbourhood under 'neighbourhood' and each bar under the correct neighbourhood, I was able to steer Search engines like Google to prioritise my site as it has a clear, logical hierarchy. By nesting neighbourhood pages under a Neighbourhoods parent, I created a clear topical structure that signals to search engines: 'This site is about neighbourhoods in East London, and here are specific areas within that topic.'
The URL structure reflects this hierarchy automatically. Instead of flat URLs like siplistlondon.uk/dalston and siplistlondon.uk/hackney; I included the neighbourhood in the URL. The parent/child relationship was visible through the URLs and created titles like siplistlondon.uk/neighbourhoods/dalston and siplistlondon.uk/neighbourhoods/hackney .These descriptive and hierarchical URLs are more SEO-friendly because they provide context. Both users and search engines immediately understand that Dalston is a neighbourhood category, not a simple standalone page. This structure also allows me to rank for broader search terms such as 'East London neighbourhoods wine bars' through the parent page. Breadcrumb navigation, which many WordPress themes automatically generate from the page hierarchy, further boosts SEO. Breadcrumbs appear as clickable paths like 'Home > Neighbourhoods > Dalston' and give search engines additional signals about site structure. This is something I used to my advantage when building my site to boost SEO.

Navigation clarity

From a user perspective, the parent/child structure made navigation intuitive and logical. Visitors arriving on the site can immediately understand how content is organised - there's a clear, simple path from all neighbourhoods to specific neighbourhood pages to bars in those neighbourhoods. This mental model matches how people naturally think about exploring an area. Therefore, when exploring a website about exploring an area, it makes sense to follow the same mental model.

Neighbourhoods structure


siplistlondon.uk/index.php/neighbourhoods/wine-bar-dalston/dans/

Sip List London

Neighbourhoods

DAN'S

Located just off of Dalston's high street, Dan's wine bar is buzzing every night. It's one those rare neighborhood wine bars that feels instantly welcoming the moment you step inside. With warm lighting, mismatched wooden tables, and a family style table where multiple parties can sit together the atmosphere is effortlessly cool without trying too hard. It's the kind of place where you can wander in for a quick glass and end up staying for hours, chatting with friends or exploring wines you've never

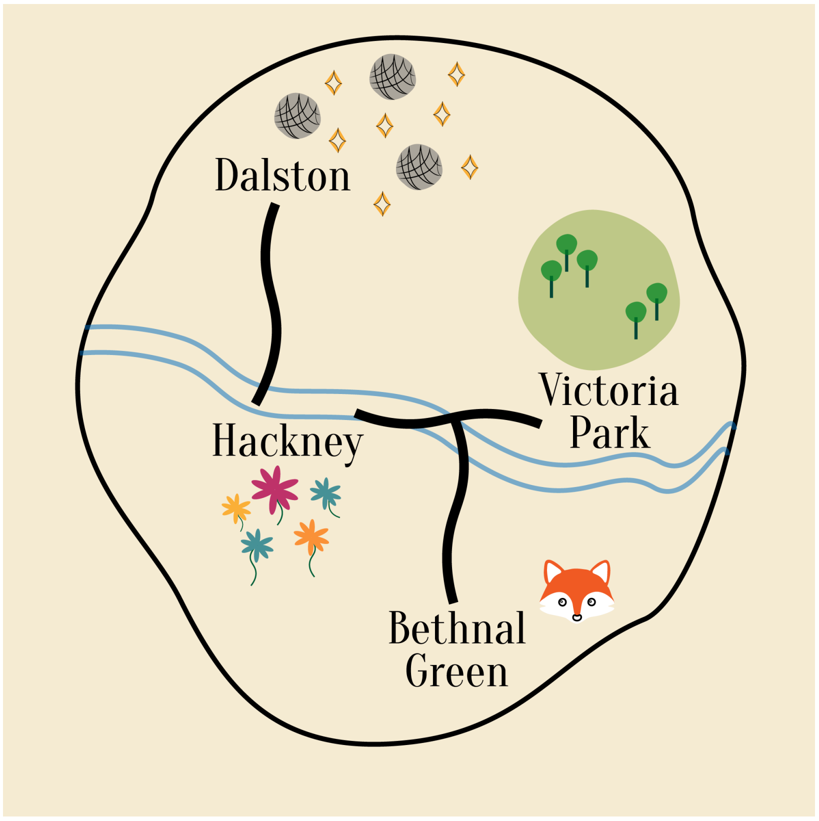
6.1 Yoast SEO

Constant iterative updates to achieve the best optimisation

I achieved a perfect 100 SEO score in Lighthouse metrics, though it didn't come immediately. I did testing and refinement throughout the development process of my site. Each time adding new content, updating pages, or making structural changes, ensuring the parent/child structure was built correctly. I was constantly checking in with Yoast SEO to identify what needed improvement within my site. As I added meta tags, slugs, and alt text, I made sure to follow Yoast's guidelines and confirm that I was passing its tests.

How you used readability/SEO scoring

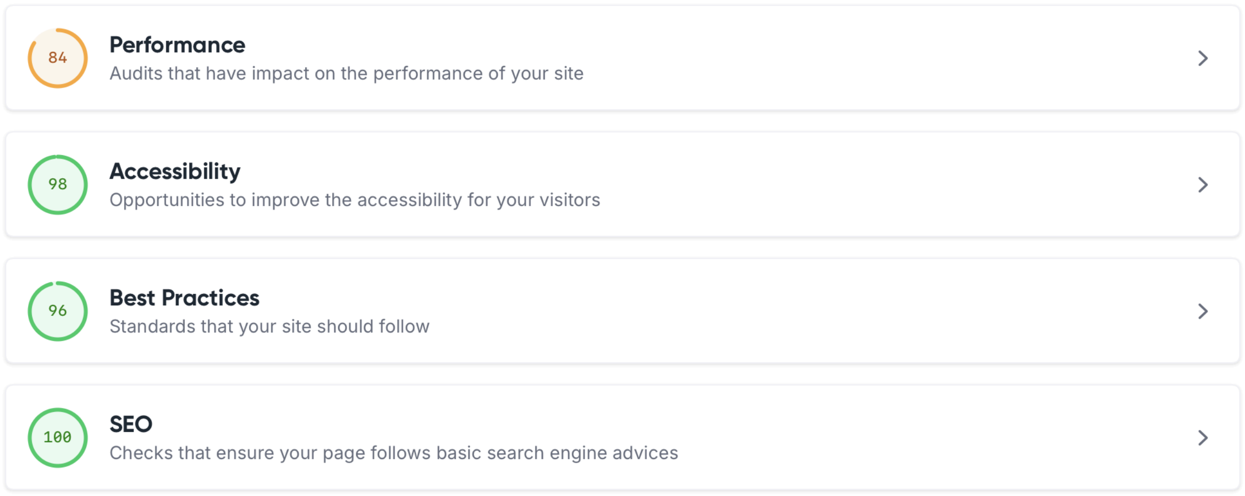
Yoast's SEO analysis evaluated each page against specific criteria: focus keywords usage, title tag optimisation, internal linking, and content length.
This was my first time working with Yoast, and I found the plugin extremely beneficial and simple to understand. The plugin used a traffic light system - red for critical issues, orange for improvements needed, green for optimised elements.The visual feedback from these lights and smiley faces made it easy to identify what needed attention. For example, when writing about Dalston wine bars, Yoast would flag if my focus keyword "Dalston wine bars" didn't appear in the first paragraph, or if my meta description exceeded 160 characters. It was clear that if Dalston was the header, Dalston should also be my Keyword. And once I learned these simple facts, it became a habit throughout the rest of my site. However, there were some portions where I stepped outside of the alt text and SEO recommendations.



On my homepage, I have an interactive illustration with a fairly lengthy alt text. It states:
'Playful illustrated map of East London showing Dalston at the top with disco balls, Hackney on the left with flowers, Victoria Park on the right with trees, and Bethnal Green at the bottom with a fox. A river runs horizontally across the map with simple line roads connecting the labelled areas. Click around the map to navigate between neighbourhoods and local bars.'
While I understand that this is longer than most meta descriptions are meant to be, I determined that this is a unique case: as it is not just a visual item but an interactive one with buttons, it deserves a longer description for low-vision and audio users.

Sitemap configuration

The combination of Yoast's real-time scoring and Lighthouse's audits created a great optimisation workflow. Yoast provided me with immediate feedback, while Lighthouse validated my technical implementation with overall metrics. This helped me achieve a perfect 100 SEO score. Almost all of my other scores were just as high. I achieved a 98 in accessibility, a 96 in best practices, and an 84 in performance.
I am ecstatic about all of these scores, except the 84; however, the 84 makes sense to me given the design decisions I made. As previously mentioned, I created an image-heavy website. Due to this and the density of all my images, loading times were a bit slower and, as a result, the performance score was lower.


84 **Performance**
Audits that have impact on the performance of your site

98 **Accessibility**
Opportunities to improve the accessibility for your visitors

96 **Best Practices**
Standards that your site should follow

100 **SEO**
Checks that ensure your page follows basic search engine advices

6.2 WP Memory Usage Plugin

Why install it

I downloaded the WP Memory Usage Plugin before I started building the Sip List London on WordPress. As my website grew in complexity, with various neighbourhoods, integrated Google maps, and custom CSS, I wanted to avoid my site feeling sluggish. I used Lighthouse to track front-end performance and metrics of my website, and WP helped me diagnose the background performance issues on my site. As WordPress runs on PHP and consumes server memory, it could cause inefficient use of certain processes or bottleneck performance. Using WP helped me ensure that my code was optimised from front to back.

What it revealed about performance

The plugin immediately revealed that certain pages were consuming significantly more memory than others. My neighbourhood pages with embedded Google Maps and multiple high-resolution images explained why those pages sometimes loaded slowly - the server was working overtime to process them compared to simpler pages like the About page.
Peak memory usage occurred when loading pages with complex layouts or when using extra or unnecessary plugins likeElementor. Loading and displaying all wine bar locations on a single interactive map required more server processing than showing them one at a time on separate pages. Ultimately, WP Memory usage transformed performance optimisation from guesswork into targeted problem-solving, allowing me to focus my efforts where they'd have a greater impact on my site's purpose.

6.3 Server Configuration

PHP version and Hosting constraints

PHP is the programming language that powers WordPress, and the PHP version running on the server significantly impacts both performance and security. It was important in building my website that I use an up-to-date PHP version that was compatible with WordPress features and plugins. By maintaining the latest version of PHP, I avoided compatibility issues and benefited from improved memory management and faster database queries. This was noticeable in the speed of saving pages or updating files with large amounts of content on them.
Storage limits were another consideration. While my hosting plan provided adequate space for the current site, high-resolution photography for each wine bar consumed storage quickly. I had to balance image quality with storage constraints, compressing photos as JPEGs while maintaining visual appeal.
Bandwidth limits also mattered - if the site received unexpected traffic (for example, being shared on social media), exceeding monthly bandwidth could result in additional charges, but as this is a student project and a small-scale wine bar guide, that is unlikely to be the case.
But if the site grew much larger - say, covering 20 neighbourhoods across all of London with hundreds of wine bars, thousands of photos, and much higher visitor numbers - shared hosting wouldn't be enough. The site would need more server power, memory, and control over settings.

7. Reflection: What I Would Do Differently

7.1 Maybe Not Use Elementor

Why

Elementor initially seemed like the ideal solution for building Sip List London. A straightforward visual page builder that had drag-and-drop sections, columns, and widgets, allowing simplicity without needing to write additional code. The interface looked intuitive. For someone transitioning from static HTML/CSS to WordPress, Elementor felt like a bridge between hand-coding and content management systems. It allowed me to have visual control over the layout while WordPress handled the backend design.
However, the frustrations quickly outweighed the benefits. Elementor generated its own CSS with highly specific selectors. It kept full control and gave me little wiggle room to create my own layout and styles as promised. Constantly overriding my crafted theme customisations. A simple heading that looked perfect on standard WordPress pages would suddenly have different font sizes, colours, or spacing when placed inside an Elementor section. What should have been a site-wide style controlled by a single CSS rule became a repetitive task of configuring each element individually.
The plugin also introduced performance overhead. Elementor loads its own CSS and JavaScript files on every page, even for simple content that doesn't need advanced layout features. This added unnecessary bloat, slowing page load times - ironic for a tool meant to streamline development. The visual editor itself was resource-intensive, sometimes making the WordPress admin dashboard sluggish, particularly on pages with many sections or embedded media.
More fundamentally, Elementor created a dependency. Content built with Elementor was locked into Elementor's proprietary format. If I ever wanted to switch themes or stop using the plugin, migrating content would be difficult - Elementor stores page layouts in shortcodes and custom data structures that don't translate cleanly to standard WordPress blocks.

What would the alternative workflow be

Abandoning Elementor meant rebuilding pages using WordPress's native block editor. Rather than fighting against Elementor's inline styles, I could create a clean hierarchy: base styles in style.css for site-wide consistency, and targeted overrides in Additional CSS. It allowed me creative freedom when specific pages needed unique styling.
This workflow was more manual but offered complete control. When I needed custom styling beyond what blocks offered natively, I added CSS classes directly to blocks and styled them in Additional CSS. This alternative workflow of abandoning Elementor also meant better performance. Without Elementor's overhead, pages loaded faster with fewer HTTP requests and smaller file sizes. My content was also safe from future Elementor updates, so that if WordPress's block system evolved, my content would evolve with it rather than being tied to a third-party plugin's lifecycle.

What did you learned from the frustrations

The Elementor experience taught me that convenience tools aren't always worth their trade-offs. Visual builders promise speed and ease, but they also often sacrifice control, performance, and maintainability. The time I "saved" using Elementor's drag-and-drop interface was lost to debugging CSS conflicts and fighting against the plugin's opinionated styling. It ended up costing me much more time trying to solve problems and fighting Elementor rather than writing some of my own CSS.
I preferred writing a few extra lines of CSS if it meant having a predictable, portable codebase. This mindset shaped not just how I approached Sip List London, but how I'd approach future projects - choosing simplicity and control over complexity and convenience, even when the latter seems easier at first glance.

7.2 What Worked Well

Hybrid coding approach

The most successful aspect of building Sip List London was adopting a hybrid approach that combined WordPress's content management capabilities with custom hand-coded CSS. Although I started my process by relying and then fighting against the automatic page builders, I used WordPress for what it does best - organising content, managing pages, and handling dynamic elements like navigation menus - while maintaining direct control over design through custom CSS in style.css and Additional CSS.
When layout issues arose, like the header pushing content to the left edge on mobile, I could diagnose and fix them directly in CSS rather than navigating through multiple plugin settings or visual builder interfaces.
The hybrid approach also meant my skills from coding the site from scratch weren't wasted when transitioning to WordPress. Understanding how CSS rules cascade and how to structure layouts translates directly into solving WordPress challenges. I wasn't learning an entirely new system; I was applying existing knowledge within a different framework. This made problem-solving intuitive and efficient.

Template system

Instead of creating each Dalston, Hackney, Victoria Park, and Bethnal Green page from scratch with duplicated code, I was able to use a template built within WordPress and provided a shared structure that all neighbourhood pages inherited.Header, footer, and navigation remained consistent automatically, while the content area could vary based on each neighbourhood's specific wine bars and characteristics. This was only my second time using PHP, and it was so seamless to use the header and footer. It was so nice to know that all of my elements would show up the same on every single page, without me having to copy and paste them individually each time. It felt safer and more efficient as I knew there was consistency with PHP.

The parent/child page relationship worked seamlessly within this template system. If I needed to adjust how neighbourhood pages displayed, I only needed to modify the template once rather than every page individually. This efficiency was crucial for maintaining the site as it grew.

WordPress's template hierarchy also allowed strategic customisation. When I wanted the homepage to display differently from neighbourhood pages - perhaps with a larger map or different layout emphasis - I could create a custom template specifically for the homepage while leaving other pages to use the default template. This meant I could standardise where consistency mattered and customise where differentiation was necessary, all without duplicating code.

Manual page structure

While WordPress offered automatic layout options, manually structuring pages using the block editor combined with custom CSS gave me precise control over how content appeared. Rather than accepting default block styling, I intentionally chose which blocks to use and how to arrange them.

For neighbourhood pages, this meant establishing a consistent manual structure: neighbourhood introduction paragraph, embedded map showing wine bar locations, followed by individual wine bar sections with a few images showcasing the vibe, a short description, and ambience notes. This consistent structure wasn't imposed by a rigid template - it was a deliberate pattern I maintained across pages, which made the site predictable for users while remaining flexible if a particular neighbourhood needed a different approach.

Manual page structure also meant understanding the relationship between blocks and CSS. When I needed some full-width images for a greater atmospheric impact, I did not use them on every single wine bar page, as they could come across as overwhelming. When text needed specific spacing around wine bar descriptions, I manually adjusted padding through custom CSS rather than relying on WordPress's spacing presets. This hands-on approach meant every design decision was intentional, resulting in a polished, cohesive site rather than a collection of default WordPress styling with occasional customisations.

I knew exactly how every page was structured, where every style rule lived, and how changes would cascade through the site. This clarity made future updates straightforward and ensured the site remained true to the original design vision rather than drifting toward generic WordPress aesthetics. It gives me the freedom to add more bars and potentially neighbourhoods in the future.

7.3 What I Would Improve

Starting with a child theme

One of my biggest regrets was not implementing a child theme from the beginning. I made customisations directly to the Vanilla theme's style.css, which worked initially but created complications when the theme received updates. Theme updates can and could overwrite custom CSS, meaning hours of styling work could potentially be lost if I wasn't careful to back up files before updating. A child theme would have solved this by creating a separate stylesheet that inherits the parent theme's functionality while keeping my customisations safe and update-proof. There were also extra pieces of information within the theme that I did not end up including in my own site, which were important to remove as they backed up my site and were not vital code. Again, this felt a safer option as this was all a new CMS to me. But next time, I will be able to jump in without my own layout from the start.
Beyond preservation, child themes offer better organisation. All custom code - CSS, PHP template modifications, custom functions - lives in one dedicated folder rather than being mixed with the parent theme's files. This separation makes troubleshooting easier because I'd know exactly where my code ended versus where the theme's original code began.

Using WebP

In hindsight, choosing WebP for image formats would have significantly improved performance. While JPEG compression reduces file sizes compared to PNG, WebP offers superior compression to JPEG while maintaining comparable, if not equal, quality. For a site centred around wine bar photography with dozens of images, those savings would compound into noticeably faster load times, especially on mobile devices or slower connections.
The main reason I avoided WebP was concern about older browser compatibility, but modern browser support for WebP is nearly universal - Chrome, Firefox, Safari, and Edge all support it. I could have implemented WebP with JPEG fallbacks for the tiny percentage of users on outdated browsers, ensuring everyone saw images while most users benefited from faster loading. The performance gains, particularly for users on limited data plans exploring East London on their phones, would have been worth the extra setup effort.

Setting the design system earlier

I should have established a comprehensive design system before building any pages. Instead, I made design decisions organically as I developed each section - choosing spacing values, defining heading sizes, determining colour usage - which led to inconsistencies that required retroactive cleanup.
This systematic approach would have sped up development, too. Instead of deciding "how much padding should this section have?" each time, I'd simply apply a predefined spacing value. The site would feel more cohesive and have a greater visual rhythm connecting all pages.

Performance mindset from the start

I should have run Lighthouse audits from the first homepage draft, establishing performance and target metrics for load time and file sizes. Instead, I focused on getting features working first and worrying about speed later, which meant working backwards at times, which took more time than necessary.

8. Conclusion

Building Sip List London transformed my understanding of web development from theoretical knowledge into a practical, applied skill. What began as a static HTML and CSS site evolved into a fully functional WordPress CMS, taking me through the complete cycle of a web project - from initial design and hand-coding, through the challenges of CMS, to optimisation and professional layout. This journey revealed that professional web development isn't about mastering a single technology, but about understanding when and how to strategically apply different tools based on project needs. Not understanding a new system immediately is normal, and figuring it out on your own is part of the process.
The process of coding from scratch before moving to WordPress proved invaluable. Going into WordPress with all of my content done was invaluable as well. Having all my reports and photography ready to import was important so that working within a new system was the only process I needed to understand, not editing images and writing statements at the same time.
While static sites offer complete control, they become impractical as content grows - updating wine bar information across multiple pages would require editing HTML files repeatedly. WordPress's separation of content and presentation taught me that I can update my site without touching code. For a project meant to grow and evolve, this flexibility is game-changing. The CMS structure also enabled features like parent/child page hierarchies and automatic sitemaps that would require custom JavaScript implementation on a static site.
However, the migration also taught me that CMS platforms introduce complexity.
The technical skills gained - CSS debugging, responsive design, SEO optimisation, server configuration, performance monitoring - are directly transferable to future work. Whether building client websites, contributing to larger teams, or creating personal projects, I now have practical experience with industry-standard tools like WordPress and Lighthouse metrics. More importantly, I've developed problem-solving methodologies: systematic debugging through browser DevTools, iterative optimisation based on performance data, and researching solutions when encountering unfamiliar issues.
Beyond technical competencies, this project professionalised my approach to web development. I learned to prioritise accessibility from the start, achieving WCAG AAA compliance not as an afterthought but as a fundamental requirement. I discovered the importance of performance design systems and documentation. The frustrations with Elementor taught me to critically evaluate tools rather than adopting them uncritically.
Sip List London stands as both a functional product and a learning piece. It's a genuinely useful guide for exploring East London's wine bars, but also a comprehensive record of my development journey - every CSS rule, every WordPress element, every performance optimisation represents a problem solved and a concept internalised. The site proves I can take a project from initial concept through design, development, deployment, and optimisation, adapting my approach as challenges arise. This experience, with all its successes and setbacks, has prepared me not just to build websites, but to think like a

professional developer who balances both technical excellence with practical constraints; who balances user needs with performance requirements; and who has creative vision with maintainable code.

References

Bruno. london.com. (n.d.). https://www.bruno-london.com/

Claude.ai. (n.d.). https://claude.ai/

Dans Wine. Dan's. (n.d.). https://www.dans.wine/

The Dreamery. Ice cream and wine. (n.d.). https://www.dreamery.london/

Get yoast seo premium. Yoast. (2025, September 24). https://yoast.com/get-yoast-seo-premium/?gad_source=1&gad_campaignid=22402907903&gbraid=0AAAAADFXXRGnEcYrSQP7HWxlCeDJomvTb&gclid=Cj0KCQiA9OnJBhD-ARIsAPV51xOBBVKt3WyCZnHWjuWZhromG2QHxz5dKAcOkZrk0H3yr7gWbzawBlAaApOcEALw_wcB

Global Lighthouse Scores. Lighthouse Metrics. (n.d.). https://lighthouse-metrics.com/lighthouse/checks/f5d7f97e-869d-4b8b-a3f6-8d029d9f0406

Goodbye horses. Goodbye Horses. (n.d.). https://www.goodbyehorses.london/

Grammarly. (n.d.). https://app.grammarly.com/

Hector's London. (n.d.). https://hectorslondon.co.uk/

Renegade london wine: Premium, sustainable wines crafted in London. Renegade Urban Winery. (n.d.). https://www.renegadelondonwine.com/

Sager + Wilde. (n.d.). https://www.sagerandwilde.com/

Top cuvée. Top Cuvée. (n.d.). https://www.topcuvee.com/