EthicaData

Design and Development Documentation

CMP4310 Website Design and Development
Baber Khan (Student ID 17125325)

GitHub: https://github.com/baberlabs/ethicadata.git
Live Demo: https://ethicadata.netlify.app (if the link does not work, check GitHub for updates)

1. Project Overview

EthicaData is a static, standards-compliant website that explores principles and practices of responsible technology use and ethical data mining. The site prioritises semantic structure, accessibility, and clarity of information over visual novelty or excessive interactivity.

The website was designed and implemented using vanilla HTML, CSS, and JavaScript. No external frameworks or libraries were used. This approach was chosen to demonstrate a clear understanding of core web technologies and modern web standards, in line with the learning outcomes of the module.

2. Accessibility and Semantic Design Decisions

Accessibility considerations informed the structure and implementation of all pages within the website.

Semantic HTML5 elements including `<header>`, `<nav>`, `<main>`, `<section>`, `<address>`, and `<footer>`, are used consistently to provide meaningful document structure for both visual users and assistive technologies. Where native HTML semantics were sufficient, additional ARIA roles were intentionally avoided in order to prevent redundancy and over-annotation.

This approach aligns with W3C guidance, which recommends prioritising native HTML semantics before introducing ARIA attributes.

2.1 Use of Alternative Text and ARIA Attributes

At several points within the site, image `alt` attributes are intentionally left empty. This decision was made in two specific scenarios:

1. Where images are purely decorative and do not convey information beyond visual styling.

2. Where an accessible name has already been provided at a higher level, such as when an `aria-label` is applied to a parent anchor element.

In these cases, omitting alternative text improves the experience for screen reader users by preventing redundant or unnecessary announcements. This approach follows current accessibility best practices and avoids over-describing content.

ARIA attributes are used sparingly and only where they provide meaningful state or relationship information. For example, the navigation menu toggle uses `aria-expanded`, which is dynamically updated using JavaScript to accurately reflect the open or closed state of the menu.

3. Folder Structure and File Naming

Although the module guidance suggested folder names such as "Scripts" and "Styles" using capitalised naming, lowercase folder and file names were used throughout the project (for example, `styles` and `scripts`). This decision reflects common industry conventions and improves consistency across operating systems that distinguish between case-sensitive paths.

4. Metadata and Document Semantics

Basic metadata, including page-specific description and authorship information, was included in the `head` section of each HTML document. This improves document semantics, professional completeness, and clarity of purpose without introducing unnecessary search-engine-specific configuration.

5. Use of Relative Links

All internal links use relative paths, as required by the assessment brief. External resources, such as references to official organisational websites, use absolute URLs where relative addressing is not possible.

6. Skip to Main Content

A "Skip to main content" link was implemented using a standard anchor element. CSS was used to visually hide the link by default and reveal it when focused via keyboard navigation (Figure 6.1). This improves accessibility for keyboard and screen reader users by allowing them to bypass repeated navigation content.

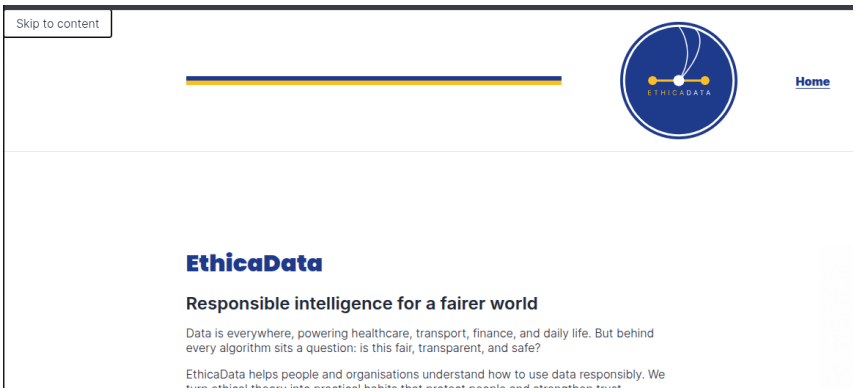


Figure 6.1

7. Image Optimisation

All images were reviewed for appropriate format, size, and purpose.

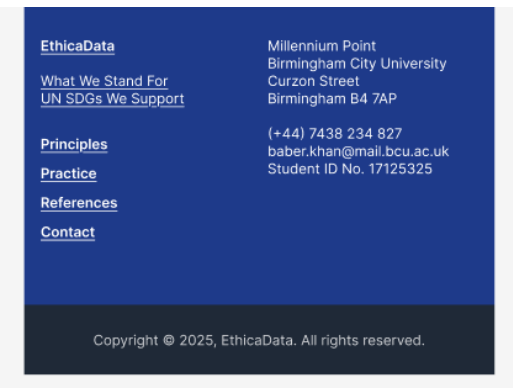
Icons, logos, and illustrative graphics were implemented using SVG files to ensure minimal file sizes and clarity across different screen resolutions. The hero image is a flat illustrative graphic provided as a raster image. As the original vector source was not available, the image was retained as a PNG to preserve sharp edges and colour accuracy. The image was resized to closely match its maximum rendered size in CSS and compressed to approximately 318 KB to balance visual quality and performance.

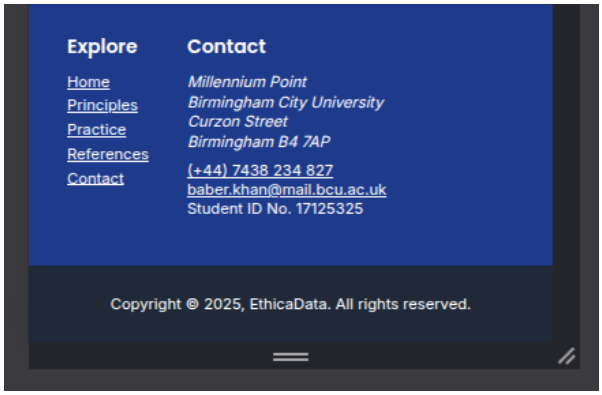
With the exception of the site logo and the hero image on the home page, **lazy loading** was applied to all other images using the `loading="lazy"` attribute. This reduces unnecessary network requests during initial page load while preserving above-the-fold content visibility.

8. Design Changes from Figma to Development

Several design changes were introduced during implementation following usability feedback and further research:

- A subtle border was added beneath the site header to improve visual separation and readability when content scrolls beneath it.
- Clear section headings were added to the footer navigation ("Explore") and contact information ("Contact") to improve scanability and semantic clarity.
- Footer navigation was simplified and re-oriented for better user experience (left: Figma, right: live).





- The active state of the primary call-to-action button was adjusted to align visually with the secondary button style, ensuring consistency across interaction states.
- United Nations Sustainable Development Goal (SDG) icons [in the homepage] were converted into hyperlinks pointing to their respective official UN pages, providing contextual depth without adding additional explanatory text.
- Tabular data [in the Practice page] was retained using native HTML table elements across all screen sizes. On smaller viewports, horizontal scrolling was enabled rather than restructuring the table using non-semantic layouts and CSS hacks. This decision prioritises accessibility and preserves table semantics. A contextual hint is displayed only when horizontal scrolling is required i.e. viewport is not in desktop mode.

- The references section was redesigned from a card-based layout to a simple list format. This reflects academic conventions and improves readability.
- A dedicated `thank-you.html` page was introduced to provide clear feedback following contact form submission, rather than returning users silently to the same page.

9. JavaScript Functionality

9.1 Navigation Menu Toggle

JavaScript was used to implement a responsive navigation menu for smaller screen sizes. When the menu button is activated, a CSS class is toggled to animate the icon state, while the `hidden` attribute is added or removed from the navigation element to control visibility.

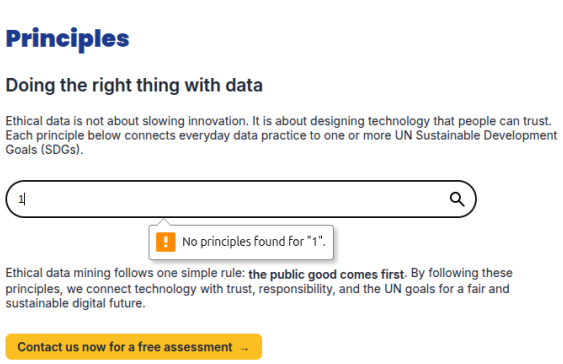
The `aria-expanded` attribute is updated programmatically to ensure that assistive technologies receive accurate information about the menu's current state. This approach ensures both visual and accessibility consistency without relying on external libraries.

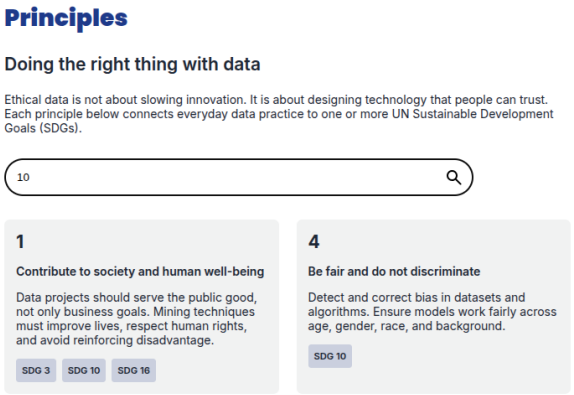
9.2 Client-Side Search and Filtering

A client-side search feature was implemented on the Principles page to allow users to filter principles by Sustainable Development Goal (SDG) number or by keywords in the principle title.

The search operates in real time as the user types and updates the visibility of existing principle cards without altering the document structure. All principle content remains present in the HTML by default and JavaScript is used only to control filtering behaviour.

When no matching results are found, native browser form validation is used to provide clear and accessible feedback to the user.





10. User Testing

Informal user testing was conducted with four peers and one tutor on both mobile and desktop devices. All participants were able to locate key sections of the site without guidance. One participant noted that the header

could benefit from greater visual distinction when scrolling. In response, a subtle bottom border was added to the header, which resolved the issue and improved overall readability.

11. WebKit Warning in CSS Validation Report

118	Due to their dynamic nature, CSS variables are currently not statically checked
831	`-webkit-overflow-scrolling` is a vendor extension
951	Due to their dynamic nature, CSS variables are currently not statically checked

The `-webkit-overflow-scrolling: touch` is included as a vendor-specific enhancement to enable momentum scrolling on iOS Safari. The property is safely ignored by non-WebKit browsers and does not affect standards compliance or functionality elsewhere.

12. CSS Layers and Nesting

During development, the stylesheet initially made use of **CSS Cascade Layers (`@layers`)** and **CSS nesting** to improve structure, maintainability, logical grouping of component styles. These features are part of modern CSS specifications and are supported by current evergreen browsers, which is why the website rendered and functioned correctly in practice.

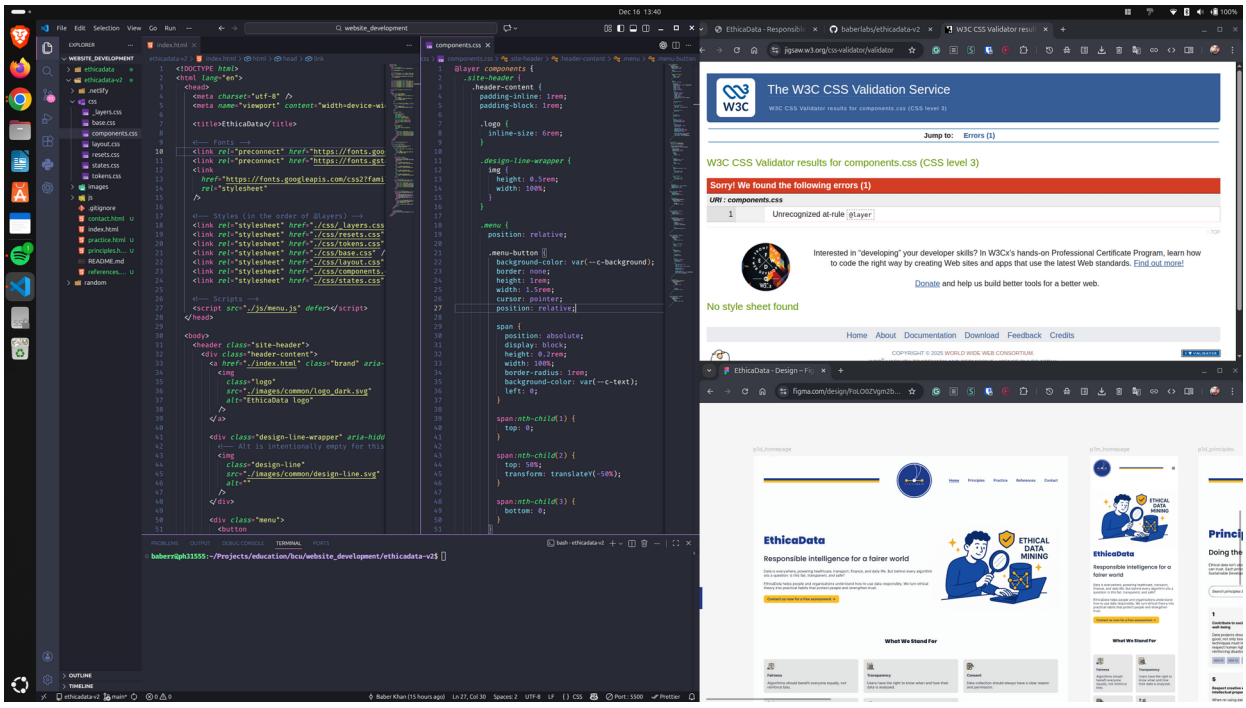


Figure 12.1

However, when validating the stylesheets using the **W3C CSS Validation Service,** both `@layer` rules and nested CSS syntax were reported as errors. The validator did not recognise these constructs and treated the affected files as syntactically invalid, even when the underlying CSS was otherwise correct. This resulted in misleading validation output suggesting that large portions of the stylesheet were faulty.

Notably, removing cascade layers while retaining nesting did not resolve the issue (Fig 12.2). The validator continued to flag parsing errors, despite the site remaining fully functional in modern browsers. This indicates that the validation service lags behind the current CSS feature adoption and does not fully support newer language constructs.

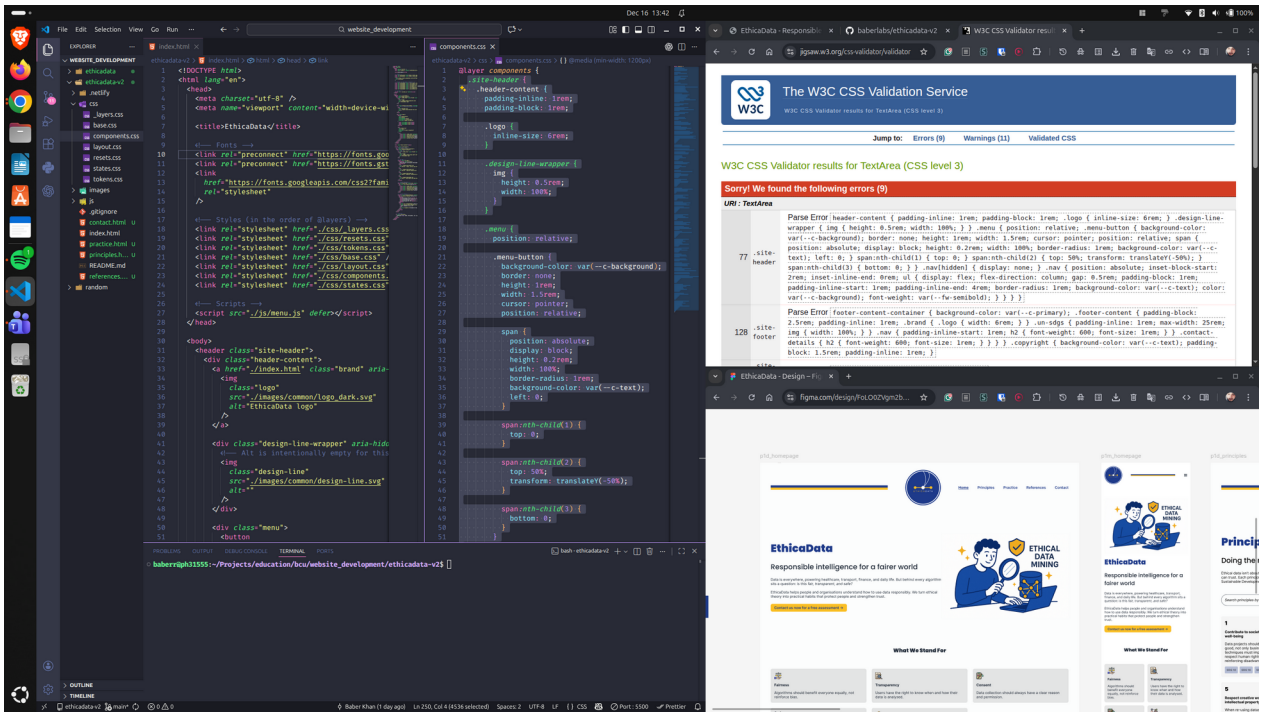


Figure 12.2

For a production environment, this would not present a practical issue, as browser support is the primary concern. However, for academic coursework, where standards validation and demonstrable correctness form part of the assessment criteria, it was necessary to prioritise validator compatibility.

As a result, the CSS was deliberately refactored from a layered and nested structure into fully flattened, standards-recognised CSS, ensuring that stylesheets validate successfully without errors. This change does not affect the visual output or functionality of the site but ensures compliance with formal validation tools.