

Step-by-Step Guide

Building a 3-Page HTML & CSS Website

*A Worked Walkthrough for ACCA4023 - Introduction to Web and Database Concepts
Covers Task 1 (Website) and Task 2 (Report) - GA36a Level 4 Assessment Specification*

Written for complete beginners - no prior coding experience assumed

Tools used: Notepad++, a web browser, and free web-based validators

Important Note on Using This Guide

 This guide is a teaching resource. The sample website ('GreenLeaf Community Garden') and the sample report text at the end are worked EXAMPLES to show you the structure, standard and depth expected - they are not to be copied into your own submission. Your module specification states that Generative AI must NOT be used to write your submitted text or code (see the GA36a AI-use table). Use this guide to learn the method, then design, code and write your own original website and report about your own chosen topic (e.g. a hobby, a band, a town, or a fictitious organisation of your choice).

Step 1: Set Up Your Project Folders Before You Write Any Code

Good organisation from the very first minute will save you hours of frustration later - broken image links and 'file not found' errors are almost always caused by messy folders. Follow these steps exactly.

1.1 Create your main project folder

1. Decide on a permanent location for your work - for example your Documents folder or a dedicated folder on your PC's hard drive. Avoid saving only to a USB stick, a temporary Downloads folder, or a cloud-sync folder that might change its path.
2. Create a new folder and name it after your website with no spaces, e.g. MyWebsite or GreenLeaf-Community-Garden. Avoid spaces and special characters in folder and file names - browsers can misread them.
3. Inside that folder, create two more folders:
 - images - will hold every picture you use on your site
 - css - will hold your single stylesheet file

 Write down (or take a screenshot of) the full path to this folder right now, e.g. C:\Users\YourName\Documents\MyWebsite. You will need to find this exact folder again in Step 6 to compress it into a .zip file for submission. Losing track of where you saved your project is one of the most common reasons students run out of time before a deadline.

1.2 The folder structure you are aiming for

```
MyWebsite/  
  index.html  
  about.html  
  contact.html  
  css/  
    style.css  
  images/  
    photo1.jpg  
    photo2.jpg
```

Teaching point:

Web pages link to images and stylesheets using relative paths (a path relative to the html file's own location, e.g. images/photo1.jpg) rather than absolute paths (a full path such as C:\Users\YourName\Documents\...). Relative paths are what make your website 'stand-alone' - they still work correctly no matter which PC the folder is opened on, because the images and css folders always sit right next to the html files. If you use an absolute path, the links will break the moment you move or zip the folder.

Step 2: Tools You Will Need

Tool	Purpose
Notepad++ (free)	A plain text code editor used to type your HTML and CSS by hand. Do not use Microsoft Word - it adds hidden formatting that breaks web pages.
A web browser (Chrome, Edge or Firefox)	Used to preview your pages by double-clicking index.html, and to test responsive design using the browser's device toolbar.
W3C Markup Validator - validator.w3.org	Checks your HTML for errors (Task 2, Implementation section).
W3C CSS Validator - jigsaw.w3.org/css-validator	Checks your CSS for errors (Task 2, Implementation section).

 **Reminder from the assessment specification: you must hand-code your website. Design packages that generate the code for you (e.g. Adobe Dreamweaver's WYSIWYG designer) are not permitted.**

Step 3: The Scenario-Based Activity

To make the task concrete, work through this brief exactly as if it were a real client instruction. A complete worked example - a fictitious community garden called 'GreenLeaf Community Garden' - is used throughout this guide and provided to you as a finished sample website. Use it to see the method, then apply the same method to your own chosen topic.

3.1 The brief (worked example)

“A small local charity, GreenLeaf Community Garden, needs a simple, friendly website so that local residents can find out what the garden offers, learn its story, and find out how to get involved. The site must work on a desktop computer and on a phone, and must be easy for a first-time visitor to navigate in three clicks or fewer.”

3.2 Choose your own topic

The specification allows any of the following - pick one that genuinely interests you, since you will be writing 2,000 words about it:

- An organisation of your choice (real or fictitious) - a café, a charity, a small business
- A personal interest or hobby - a sport, a craft, a game
- A favourite band or artist
- A town or city you know well

3.3 Plan your three pages before you type anything

For the GreenLeaf example the three pages are: Home (index.html), About Us (about.html) and Get Involved (contact.html). On paper or in a simple diagram, sketch:

- What each page is for, in one sentence
- What content each page needs (text, one image, one list, one table)
- How the navigation menu will link the three pages together

Step 4: Build Your Home Page (index.html)

Open Notepad++, create a new file, and save it immediately as index.html inside your main project folder (not inside images or css). The file MUST be named index.html - this is the file a browser looks for automatically when someone opens your website folder or hosts it online.

4.1 The document head - line by line

Type the following at the very top of the file. Each line is explained in the table beneath it.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="A one-sentence summary of your page, shown in Google search
results.">
<meta name="keywords" content="3-5 words that describe your site">
<meta name="author" content="Your Name">
<title>Your Site Name | Home</title>
<link rel="stylesheet" href="css/style.css">
</head>

```

Line	What it does	Why it matters (teaching point)
1	<!DOCTYPE html>	Tells the browser this page uses the current HTML5 standard, so it renders consistently.
2	<html lang="en">	Declares the page language. Screen readers use this to choose the correct pronunciation rules - a WCAG 2.1 requirement (Success Criterion 3.1.1).
4	<meta charset="UTF-8">	Sets the character encoding so accents, currency symbols and special characters display correctly instead of as garbled text.
5	viewport meta tag	Essential for responsive design. Without it, mobile browsers shrink the whole page to fit and text becomes unreadably small.
6	description meta tag	The single most important on-page SEO element after the title - search engines often display it as the snippet under your link in results.
7	keywords meta tag	Historically used for SEO; most modern search engines now ignore it, but it is still commonly taught and can support internal site search tools - include it, but do not rely on it for ranking.
9	<title>	Shown in the browser tab, in bookmarks, and as the clickable blue link in search results - a major on-page SEO signal. Give every page a unique, descriptive title.
10	<link rel="stylesheet">	Connects the page to your one shared CSS file so all three pages look consistent.

4.2 The header and navigation

```

<body>
  <header>
    <h1>Your Site Name</h1>
    <p class="tagline">A short tagline</p>
    <nav>
      <ul class="nav-list">
        <li><a href="index.html" aria-current="page">Home</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Get Involved</a></li>
      </ul>
    </nav>
  </header>

```

Teaching points:

- Use exactly one <h1> per page - it should be your site or page name. Search engines and screen-reader users rely on a logical heading order (h1, then h2, then h3) to understand your page's structure (WCAG 1.3.1).
- A <nav> list of links is marked up as an unordered list () of links, not just plain text with spaces - this gives assistive technology a 'landmark' region and an item count, e.g. 'navigation, list of 3 items'.
- Use relative links (href="about.html") rather than a full web address. This is what makes the site work when opened directly from a folder, with no web server.
- aria-current="page" tells assistive technology and search engines which page in the menu is the current one - a small but genuine accessibility improvement (WCAG 2.4.8, Location).

4.3 Main content - lists, an image, and a table

```
<main>
  <section id="welcome">
    <h2>Welcome</h2>
    
    <p>An introductory paragraph about your site goes here.</p>
  </section>

  <section id="whats-on">
    <h2>What We Offer</h2>
    <ul>
      <li>An unordered list item - use for items with no set order</li>
    </ul>
    <h3>How to Join in Three Steps</h3>
    <ol>
      <li>An ordered list item - use for sequential steps</li>
    </ol>
  </section>

  <section id="opening-times">
    <h2>Opening Times</h2>
    <table>
      <caption>Garden opening hours</caption>
      <thead>
        <tr><th scope="col">Day</th><th scope="col">Open</th><th scope="col">Close</th></tr>
      </thead>
      <tbody>
        <tr><td>Monday - Friday</td><td>9:00 am</td><td>5:00 pm</td></tr>
      </tbody>
    </table>
  </section>
</main>
```

Teaching points:

- alt text on every image (WCAG 1.1.1) - describe what is in the photo in a short sentence. This is read aloud by screen readers and displayed if the image fails to load. Never leave alt="" unless the image is purely decorative.
- width and height attributes on reserve the correct space on the page before the image has finished loading, which prevents the rest of the page jumping around (a recognised usability and Core Web Vitals issue).
- Choose vs based on meaning, not appearance: an ordered list implies sequence or ranking; an unordered list does not. This is a genuine accessibility distinction, not just a style choice.
- A real <table> with <caption>, <thead> and scope="col" on header cells links each data cell to its column heading for screen-reader users (WCAG 1.3.1) - never use a table purely to lay out a page.

4.4 Embedding a YouTube video

Many students are unsure how to add a video without hosting the video file themselves. The correct, copyright-safe method is to embed the official YouTube player using an <iframe>. Follow these steps:

1. On YouTube, open the video you want to embed.
2. Click the Share button beneath the video, then choose Embed.
3. YouTube shows you a ready-made <iframe> code snippet. You only need to reuse the web address inside its src="..." attribute.
4. In your own HTML, wrap the iframe in a container div so it can be resized responsively (see the CSS in Step 5), and add your own descriptive title attribute:

```
<div class="video-wrapper">
  <iframe
    src="https://www.youtube.com/embed/VIDEO_ID_HERE"
```

```

        title="A short description of the video for screen readers"
        frameborder="0"
        allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-
picture"
        allowfullscreen>
    </iframe>
</div>

```

Teaching point:

Embedding (linking to the video that still lives on YouTube's own servers) is legitimate and does not raise copyright issues, because you are not copying or redistributing the video file itself - you are simply displaying YouTube's own player. Downloading a copy of someone else's video and uploading that file to your own site would not be acceptable.

4.5 Footer

```

<footer>
    <p>&copy; 2026 Your Site Name. All rights reserved.</p>
    <p>Your address | <a href="mailto:you@example.com">you@example.com</a></p>
</footer>
</body>
</html>

```

4.6 Build about.html and contact.html the same way

Copy the whole of index.html, save it as about.html, then:

- Update the <title>
- Move aria-current="page" onto the About link
- Replace the content inside <main> with content for that page
- Repeat for contact.html

Keep the <header>, <nav> and <footer> identical (word-for-word) across all three pages - a consistent, predictable menu on every page is a core usability principle.

Step 5: Build Your Stylesheet (css/style.css)

Create one new file called style.css and save it inside your css folder. Every page links to this single file, so a change here updates the look of the whole site at once - this is the main advantage of separating content (HTML) from presentation (CSS).

 **This assignment asks for your own design - do not use CSS Grid or a framework such as Bootstrap. The layout below is built entirely from the CSS box model and Flexbox, which you write and control yourself.**

5.1 Reset, base styles and colour scheme

```

* { margin: 0; padding: 0; box-sizing: border-box; }

body {
    font-family: 'Segoe UI', Verdana, Arial, sans-serif;
    color: #2b2b2b;
    background-color: #f5f7f0;
    line-height: 1.6;
}

```

Teaching points:

- box-sizing: border-box makes padding and borders count inside an element's declared width, which avoids confusing layout maths for beginners.

- Choose two or three colours that suit your topic (here, greens for a garden) and reuse them consistently - a recognised look-and-feel usability principle.

5.2 Flexbox navigation (your own layout, no framework)

```
.nav-list {
  list-style: none;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 1.5rem;
}
```

display: flex lays the menu items out in a row and lets them wrap onto a new line automatically on a narrow screen, without you writing a single media query for this element.

5.3 Table, list and card styling

Style your sections as simple 'cards' with a border and padding, style your table headers with a background colour, and use zebra-stripping (alternating row colours) to make wide tables easier to scan - a well-documented usability technique for improving row-tracking in data tables.

5.4 A responsive video container

```
.video-wrapper {
  position: relative;
  padding-bottom: 56.25%; /* 16:9 aspect ratio */
  height: 0;
  overflow: hidden;
}
.video-wrapper iframe {
  position: absolute; top: 0; left: 0;
  width: 100%; height: 100%; border: 0;
}
```

This well-known CSS technique keeps the embedded video at a correct 16:9 shape at any screen width, instead of being stretched or squashed.

5.5 Responsive breakpoints - testing different screen sizes

```
@media screen and (max-width: 768px) {
  /* rules for tablets and small laptops */
}

@media screen and (max-width: 480px) {
  .nav-list { flex-direction: column; }
  /* rules for phones */
}
```

Teaching points:

- A media query applies a block of CSS only when the browser window matches the stated condition - here, 'when the screen is 768px wide or narrower'.
- To test this without an actual phone: in your browser press F12 (or right-click > Inspect), then click the small phone/tablet icon (Toggle Device Toolbar) to preview common device widths, and also try simply dragging the edge of your browser window narrower.
- Take a screenshot of your site at a wide (desktop) width and a narrow (mobile) width for your Task 2 Implementation section, to provide evidence of responsive design.

Step 6: Make the Site Stand-Alone and Submit It

6.1 Test that it is truly stand-alone

1. Close Notepad++ and your browser.
2. Open your project folder and double-click index.html directly (do not open it through Notepad++'s 'preview' feature).
3. Click every navigation link and confirm all three pages open correctly, all images appear, and the video plays.
4. Copy the whole folder to a different location (or a different PC, or a USB stick) and repeat the test. If a link or image breaks, you have used an absolute path somewhere - fix it to a relative path.

6.2 Create the submission .zip file

1. Right-click your main project folder (the one containing index.html, css and images).
2. Choose Send to > Compressed (zipped) folder (Windows) or Compress (Mac).
3. Rename the resulting .zip file sensibly, e.g. YourName_Website.zip.
4. Before submitting: extract your own zip file into a brand-new folder and double-click index.html one more time to prove the zip itself works.

 **Upload the .zip file to the separate website upload link on Moodle - NOT the Turnitin link, which is for your Word report only.**

Step 7: Validate Your HTML and CSS

Validation is an automated check of your code against the official web standards published by the W3C (the organisation that defines HTML and CSS). It is a required piece of evidence for your Implementation section.

7.1 Validating your home page's HTML

1. Go to <http://validator.w3.org/>
2. Choose the 'Validate by File Upload' tab, click Choose File, and select your index.html. (Alternatively use 'Validate by Direct Input' and paste your code.)
3. Click Check.
4. Take a screenshot of the results page for your report.

Why we run this validation (teaching point):

A browser will often display a page even when the underlying code has mistakes, because browsers are built to guess what you meant and 'fail gracefully'. This is convenient, but it hides problems such as an unclosed tag, a missing alt attribute, or two elements with the same id - mistakes that can break your layout unpredictably on a different browser, confuse assistive technology, or lower a search engine's trust in the page. Validation is an objective, standards-based quality check that is independent of any one browser's guesswork.

What to expect as output:

A clean page will state something like: "This document was successfully checked as HTML5. No errors or warnings to show." If errors are listed instead, each one names the line number and gives a description (e.g. 'an img element must have an alt attribute'). For your report: briefly explain how you corrected each type of error you encountered. If several errors are simply repeats of the same mistake, describe it once in general terms rather than listing every repetition. If you cannot resolve an error, briefly state what you believe is causing it.

7.2 Validating your main CSS

1. Go to <http://jigsaw.w3.org/css-validator/>
2. Choose 'By file upload' and select your css/style.css (or 'By direct input' and paste it).
3. Leave the profile set to the current CSS level, then click Check.
4. Take a screenshot of the results for your report.

What to expect as output:

A clean stylesheet produces the message: “Congratulations! No Error Found.” You may still see Warnings (shown separately from Errors) - these are not failures, but notes such as a colour being defined more than once; briefly mention them if present, but they do not need to be 'fixed' in the way an Error does.

Step 8: Research Usability, Accessibility and SEO (Evaluation Section)

The Evaluation section of your report asks you to research and define usability, accessibility and search engine optimisation, and to support your discussion of each term with at least one example from a peer-reviewed journal or conference paper.

8.1 How to find a genuine peer-reviewed source

1. Start at your university library's search portal, or use Google Scholar (scholar.google.com), IEEE Xplore, or the ACM Digital Library.
2. Search using plain terms, e.g. “website usability evaluation”, “WCAG accessibility evaluation website”, “on-page SEO ranking factors”.
3. Read the abstract, note the authors, year, title, journal name and (if available) the DOI - you will need all of this for your IEEE-style reference list.

8.2 Worked examples of real peer-reviewed sources (one per term)

These are genuine, real studies you can look up to see what a suitable source looks like. You are still expected to search for and select your own sources that best relate to your own website.

Usability example:

A 2021 study published in the Journal of Medical Internet Research adapted a usability-scoring methodology and applied it to a sample of United States academic medical centre websites, then used the results to recommend specific areas for improvement. This is a good model for how a usability discussion moves from a definition, to a method, to concrete recommendations - exactly what the Evaluation section of your own report should also do for your website.

Accessibility example:

A study published through the ACM's Digital Government: Research and Practice journal evaluated the landing pages of thirteen educational websites using six separate WCAG evaluation tools endorsed by the W3C, and reported the range of accessibility issues each tool identified. This illustrates a useful point for your own accessibility evaluation: different automated tools can disagree, so it is good practice to check your findings with more than one tool where possible, and to relate what you find back to the specific WCAG 2.1 principle it affects.

SEO example:

Peer-reviewed papers on search engine optimisation commonly describe on-page optimisation as the practice of improving individual pages through elements such as meta tags, headings, URLs and keyword placement, in order to improve a page's visibility and relevance to a search engine. Search your library database for a current peer-reviewed SEO paper and use it to support your own definition, then go on to document, with your own screenshots, exactly where you applied similar techniques in your own website (e.g. your <title>, your meta description, and your use of heading levels).

8.3 Teaching point - why one example study per term is required

The requirement is not just to define a term from memory, but to show that your understanding is grounded in the wider academic field. Summarising a real study's method and key finding in your own words (never copy-pasting) demonstrates engagement with literature (worth 20% of your marks) rather than relying purely on your own opinion or a general website.

Step 9: Evaluate Your Own Completed Website

9.1 Usefulness - utility, usability and content

Write a short, honest evaluation of your own finished site under three headings:

- **Utility (functionality):** Does every link work? Does the video play? Does the site do what it set out to do?

- Usability (look and feel, ease of use): Is the navigation obvious? Is text easy to read against the background? Could a first-time visitor find what they need in three clicks or fewer?
- Content (information): Is the information accurate, sufficient and well-organised for the target audience you described in your Requirements section?

Finish with two or three specific, realistic recommendations for future improvement (e.g. “add a search box”, “add a booking form”, “add more photographs of the interior”) - vague statements such as “make it better” will not score well.

9.2 Search Engine Optimisation - document your own techniques

List the specific on-page SEO techniques you actually used, with a screenshot or the exact line of code as evidence for each one, for example:

Technique	Where applied in your code
Descriptive, unique <title> on every page	e.g. line 9 of index.html
Meta description summarising the page	e.g. line 6 of index.html
Logical heading structure (h1 > h2 > h3)	e.g. lines 15, 28, 46 of index.html
Descriptive alt text on every image	e.g. line 29 of index.html
Meaningful, human-readable file and folder names	e.g. images/garden-hero.jpg, not images/img1.jpg

9.3 Accessibility - WCAG 2.1 and the four POUR principles

WCAG 2.1 organises accessibility around four principles, often remembered by the acronym POUR. For each principle, briefly explain what it means and give one genuine example from your own website.

Principle	What it means	Example from a website
Perceivable	Information must be presented in ways all users can perceive, e.g. with alternatives for non-text content.	alt text on every ; sufficient colour contrast between text and background.
Operable	All functionality must be usable via keyboard, with enough time and no content known to cause seizures.	Navigation links are reachable and activatable using only the Tab and Enter keys.
Understandable	Text is readable, pages behave predictably, and users are helped to avoid and correct mistakes.	An identical navigation menu appears in the same place and order on every page.
Robust	Content must work reliably with current and future browsers and assistive technologies.	Valid, semantic HTML (confirmed by the W3C validator) that uses correct elements such as <nav>, <table> and .

Run at least one accessibility tool against your home page (e.g. the WAVE browser extension or the free contrastchecker.com tool for your colour choices) and include a screenshot of the result, briefly relating any issues raised back to the relevant POUR principle above.

9.4 Comparing two live websites

Choose two real, recognisable websites that do a similar job (e.g. two supermarket sites, two university sites, or two news sites) and compare them directly against the criteria the specification sets out. A blank version of the comparison table you need to complete is shown below - fill it in using your own live testing of the two sites you choose.

Area	What to compare
Usability	Layout, navigation, readability and overall user experience
Accessibility	Run one tool (e.g. WAVE or axe) against both sites and relate the findings to WCAG 2.1
SEO	Compare meta tags, page titles, image alt attributes and heading structure (view each site's page source)
Mobile Friendliness	Compare how each site displays using your browser's device toolbar or an actual phone
Conclusion	A reasoned judgement on which site performs better overall, and why

Step 10: Sample Task 2 Report Write-Up (Model Structure)

 This is a condensed **MODEL** showing the structure, tone and level of detail expected in a Task 2 report, based on the GreenLeaf Community Garden sample website. It is illustrative, not something to copy: your own report must be about your own website, written in your own words, with your own screenshots, your own validation results and your own research. Word counts of individual sample paragraphs below are shortened for space - your own sections should be developed in full to reach the required word count.

10.1 Title Page

HTML & CSS Web Development - Specification Report

Prepared by: [Student Name], [Student ID]

Programme: CertHE Computing Skills for Workplace

Module: ACCA4023 - Introduction to Web and Database Concepts

Date: [Submission date]

10.2 Table of Contents

- 1. Introduction
- 2. Requirements
- 3. Implementation
- 4. Evaluation
- 5. References
- 6. Appendices - HTML and CSS Listings

10.3 Introduction (Abstract)

This report documents the design, implementation and evaluation of a three-page website built for GreenLeaf Community Garden, a fictitious local charity. It describes the target audience and purpose of the site, presents screenshots of the completed pages, reports the results of HTML and CSS validation, and evaluates the site's usability, accessibility and search engine optimisation with reference to relevant academic literature and the WCAG 2.1 guidelines. The report concludes with a comparison of two live websites and a set of recommendations for future development of the GreenLeaf site.

10.4 Requirements

GreenLeaf Community Garden is a small, volunteer-run charity that offers plot rentals, free beginner workshops and family events. The website's purpose is to help local residents quickly understand what the garden offers and how to get involved. The primary target audience is local adults aged roughly 25-65 with little or no prior gardening experience, together with families looking for weekend activities; the site therefore prioritises plain language, a short navigation menu, and clear calls to action over dense text.

10.5 Implementation

[Insert full-page screenshots of index.html, about.html and contact.html here, taken using the Windows Snipping Tool or Snip & Sketch. If a page is long, a cropped screenshot showing a representative amount of content is acceptable.]

HTML Validation (Home Page)

The home page (index.html) was validated at validator.w3.org using the direct-input method. The first check returned two errors: a missing alt attribute on the hero image, and a duplicate id value used on two different <section> elements. The alt attribute was added directly, and the duplicate id was renamed to make each section identifier unique. Re-running the check produced the message: "This document was successfully checked as HTML5. No errors or warnings to show." [Insert screenshot of this result.]

CSS Validation (Main Stylesheet)

The stylesheet (css/style.css) was validated at jigsaw.w3.org/css-validator. The result returned no errors and the message: “Congratulations! No Error Found.” One informational warning noted that a colour value was declared more than once across different selectors; as this was an intentional design choice rather than a mistake, no change was made. [Insert screenshot of this result.]

10.6 Evaluation

a) Usability, Accessibility and SEO - Definitions and Research

Usability describes how effectively, efficiently and satisfactorily a user can achieve their goals on a website. A 2021 study in the Journal of Medical Internet Research applied a structured usability-scoring method to a sample of academic medical centre websites and used the results to recommend specific, practical improvements - demonstrating that usability evaluation should move beyond opinion towards a reproducible method and concrete recommendations, an approach this report also follows in section (b) below.

Accessibility describes the extent to which a website can be used by people with a wide range of abilities and disabilities, and is formally defined for the web through the W3C's WCAG guidelines. A study published through the ACM's Digital Government: Research and Practice journal evaluated thirteen educational websites using six different automated WCAG evaluation tools and found that the tools did not always agree with one another - a useful reminder that automated accessibility tools are a starting point, not a complete guarantee, and that findings should be cross-checked and related back to specific WCAG criteria.

Search engine optimisation (SEO) describes techniques used to improve a website's visibility and ranking in search engine results. Peer-reviewed literature commonly separates on-page techniques - such as meta tags, heading structure, URL design and keyword placement within a page's own content - from off-page techniques such as backlinks from other sites; this report focuses only on the on-page techniques relevant to Task 1, as documented in section (b) below.

b) Evaluation of the GreenLeaf Website

Usefulness: the site's utility is high - every navigation link, image and the embedded video function correctly. Usability is generally strong: the three-item menu is identical on every page and a first-time visitor can reach the plot prices in two clicks. One weakness identified during testing is that the plot price table on the Get Involved page becomes cramped on very small phone screens; a recommendation for future development is to allow this table to scroll horizontally, or to reformat it as stacked cards, below the 480px breakpoint.

SEO: each page has a unique, descriptive <title> and meta description; the home page's <title> reads ‘GreenLeaf Community Garden | Home’, combining the brand name with the page purpose. Heading levels follow a logical h1-h2-h3 order, and every image carries descriptive alt text (e.g. ‘Volunteers tending raised vegetable beds...’) which also supports discovery through image search.

Accessibility: with reference to WCAG 2.1, the site addresses Perceivable through alt text on all images and adequate colour contrast between the dark green header and white text; Operable through a keyboard-navigable menu; Understandable through a consistent, predictable navigation structure repeated on every page; and Robust through valid, semantic HTML confirmed by the W3C validator. Running the WAVE browser extension against the home page returned zero errors and one alert relating to redundant link text, which was resolved by making each footer link's visible text more descriptive.

c) Comparison of Two Live Websites

[This section should compare two real websites you have tested yourself, e.g. two council websites or two small charity websites similar in purpose to your own site. An illustrative, general example structure is shown below - replace the content with findings from your own live testing.]

Area	Site A	Site B
Usability	Findings from your own testing	Findings from your own testing
Accessibility	Result from WAVE/axe + WCAG reference	Result from WAVE/axe + WCAG reference
SEO	Titles, meta tags, alt text, headings observed	Titles, meta tags, alt text, headings observed
Mobile Friendliness	Observation using device toolbar	Observation using device toolbar

Conclusion: [State, with reasons drawn from the rows above, which site you judge to perform better overall.]

10.7 References (IEEE format - sample entries)

[1] W3C, “Web Content Accessibility Guidelines (WCAG) 2.1,” World Wide Web Consortium, 2018.

[2] Author(s), “An Analysis of US Academic Medical Center Websites: Usability Study,” Journal of Medical Internet Research, 2021.

[3] Author(s), “Accessibility Analysis of Educational Websites Using WCAG 2.0,” Digital Government: Research and Practice, Association for Computing Machinery.

[4] Author(s), “A Brief Study on Search Engine Optimization (SEO) Strategies,” [Journal name], 2023.

 **These are shortened placeholders to show IEEE format and ordering. For your own submission, replace these with the full, correctly formatted references for the sources you actually used - check your university's IEEE referencing guide for the exact punctuation of author names, volume and page numbers.**

10.8 Appendices - HTML and CSS Listings

[Paste the complete text of index.html, about.html, contact.html and style.css here, in a monospace font such as Consolas, each clearly labelled with the file name as a heading. Appendices do not count towards your 2,000-word count.]

Final Submission Checklist

- index.html, about.html, contact.html, css/style.css and an images folder all sit inside one main project folder
- All links use relative paths and the site works when opened from a copied/moved folder
- Home page HTML validated with a clean or explained result (screenshot saved)
- Main CSS validated with a clean or explained result (screenshot saved)
- Report contains Title Page, Table of Contents, Introduction, Requirements, Implementation (with screenshots), Evaluation (definitions with peer-reviewed examples, self-evaluation, WCAG 2.1 discussion, two-site comparison), References (IEEE) and Appendices (full code listings)
- Project folder compressed into a single .zip file and test-extracted to confirm it still works
- Report uploaded via the Turnitin link; .zip uploaded via the separate website upload link (not Turnitin)