Assessment 4: HTML CSS DESIGN and Java Script

Surya Nair

Student Number: 11840178

Charles Sturt University

ITC293

Table of Contents

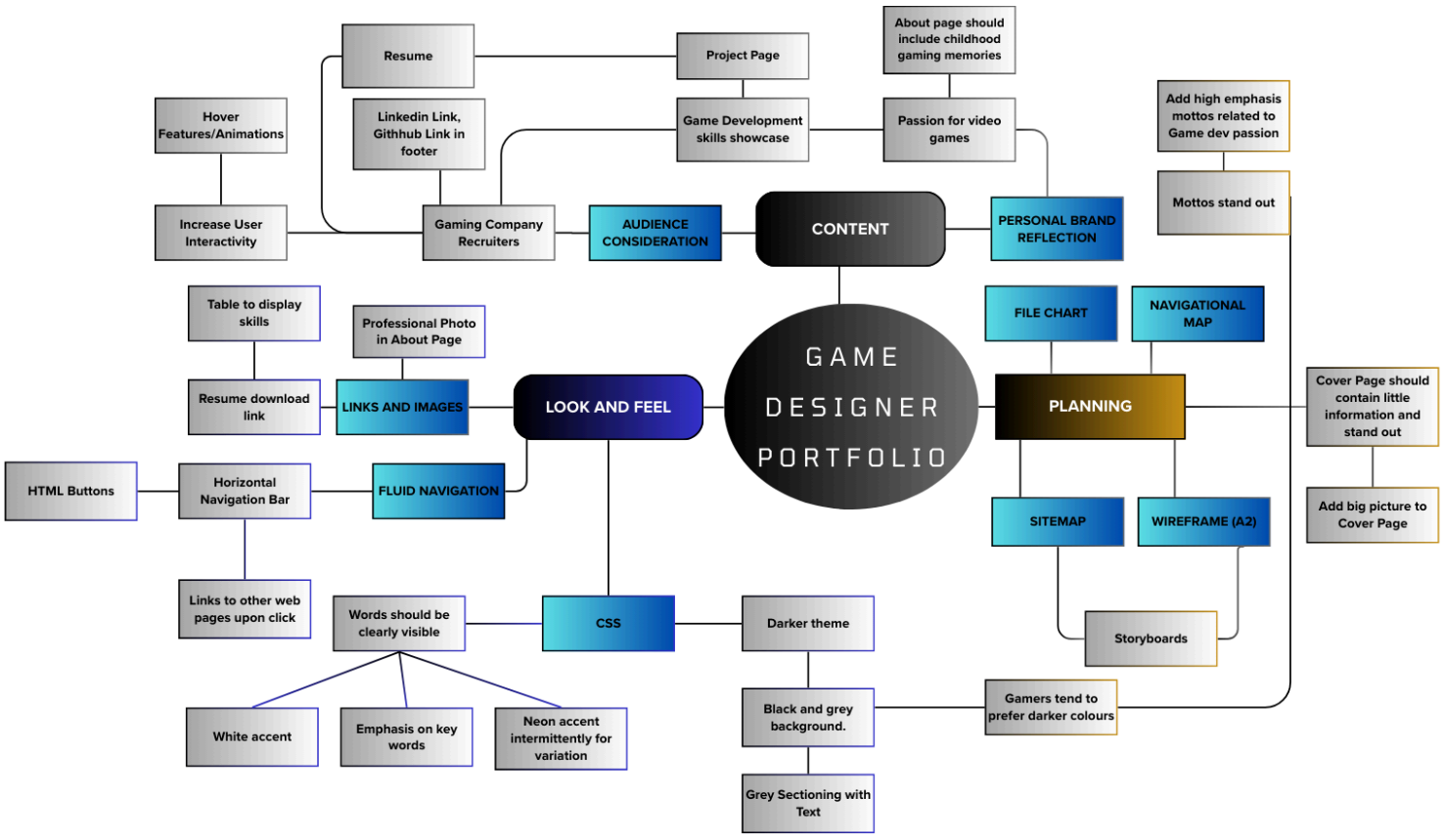
Introduction

This assignment is a continuation of A2 and A3 and will improve upon various elements of web design as well as incorporating Javascript functionality. It aims to address the new A4 requirements and implement feedback received for the previous assessments. For context, the website to be developed is a portfolio for a game developer attempting to be recruited by a fictitious online video game company. The developer in question is myself although the information used in the website is a fictitious representation and therefore not accurate to reality.

Preliminary Design

This section will outline the initial concept and idea generation behind the website creation in general and efforts to include every required HTML and CSS element while meeting the standards for professional design and improving on previous attempts. Note that changes have been made since this stage.

Concept Map:



The preliminary design aimed to create a professional and highly stylized online portfolio that would be appealing to the target audience- recruiters, hiring managers, and fellow developers within the video game industry. The core concept was to move away from a generic, corporate feel and instead embrace a "cyberpunk gamer" aesthetic. This theme was chosen to immediately convey a passion for gaming and a familiarity with modern, tech-focused visual language, making the portfolio memorable and directly relevant to the target industry. The design prioritizes visual engagement through animations and a strong color scheme to capture and hold the user's attention.

Initially, A2 and A3 went for a greyish look with neon blue accent which was a rather weak attempt at conveying a “gamer” aesthetic. The grey text and background made it rather dull with inappropriate contrast as stated in the feedback to A3. In an effort to revamp the site, the design was overhauled with now improved knowledge of CSS and a more intermediate understanding of web design. The overhaul started by trialing various colour combinations until there was a standout- which happened to emerge as vibrant green on black. This is in line with the Xbox series colour theme and Nvidia’s general look and feel. Ideally, this would incite a feeling of familiarity amongst gamers; giving users an immediately strong first impression whilst presenting the main idea of the site.

The requirements for the site have been updated to include 4 new “skills” pages of the name which pertain to the relevant industry (online video games). The most crucial skills for a game developer would be gameplay design, level design as well as the ability to take advantage of artificial intelligence. AI Programming is up and coming in the game development sphere and is widely used to create models for Non-Playable-Characters (NPCs) and to determine their course of action. Then as required by A4, there will be a web skills page for the developer to show that they have a broader skillset and are capable of developing media to support the brand of the game and its company. For the sake of simplicity, these pages will contain a few demonstrative projects with images and brief descriptions with an emphasis on skills used and value created.

A HTML5 form will be made using HTML and its functionality enabled using JavaScript with the goal of promoting a newsletter by an online video game company. Alongside, the website itself - the form and its output data will be hosted on live servers to ensure easy access for users and to showcase the form’s data processing. The website will further include a contact page using similar code to the newsletter form in order to enable recruiters, and other interested users to directly contact the developer.

Aside from this, the main ideas behind the footer and the Resume page will stay consistent with A2 and A3 although there are design adjustments made.

Interface Design

The interface is built on a dark theme to evoke a sense of a futuristic, digital space. The primary color palette consists of a glossy, fading black gradient background overlaid with a vibrant, neon green for all interactive elements, headings, and accents. This creates a high-contrast, visually striking look reminiscent of computer terminals, heads-up displays (HUDs), and cyberpunk media (neon glows). Great care has been taken to ensure an efficient , responsive yet pleasing design to create a positive user experience though most of this will be discussed in the “Usage of CSS” section.

Typography: The standard Sans-Seriff is used throughout giving to its simplistic, elegant and highly readable form.

Layout: The layout is designed to be clean and spacious, using a single-column format for content within a full-width container. Sections are clearly delineated by stylized container boxes with subtle borders and background glows, ensuring information is well-organized and easy to digest.

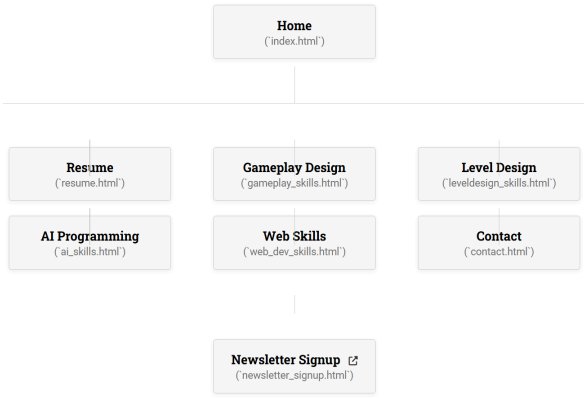
CSS Text & Colour Variables Plan

Description	Specification	Purpose
Primary Font	Sans-Seriff	Serves as the main font for the website
Primary Text	#ffffff (pure white)	Highly contrasting to black background and green accent. Is the main colour of text within the web page.
Secondary Text	#dddddd (light grey)	Used for less important text to create a subtle visual hierarchy
Primary Accent/Heading Text	#39ff14 (neon green)	Adds to the Gamer aesthetic, stands out well from white and black. Used for heading text, border, shadows and links/icon glow effect.
Glitch Accent 1	#ff00de (Magenta)	Used in the glitch animation effect on titles as a high-contrast accent color to enhance the "digital error" feel.
Glitch Accent 2	#00ffff (Cyan)	Used as a secondary accent in the glitch animation to complement the magenta.
Background	#0a0a14/#000000 (black, near black)	Serves as a clean dark backdrop for the entire site, the linear gradient fade to lighter black,grey and eventually white adds to an immersive site experience.

Main aspects of User Interface: As lightly touched on before, the User Interface will involve forms, links, buttons, vertical scroll (kept to a minimum) and glowing icons upon hover. The navigation bar is central to how Users move around the site and as such will be discussed next.

Navigational Design

This section goes over the process and result of designing navigation functionality in the website. Below is a map containing the main outline of how users will navigate along the site. The required HTML5 form (newsletter) will be available from a link in the Web_skills page. This is the only page that is not directly accessible from every other webpage via the navigation bar. The downloadable Resume pdf is only accessible in the “Resume” Page. Note, file names are in the form SN_A4_(insert name).html however my name has been omitted from these diagrams for simplicity’s sake. The Html used to create a functioning navigation bar is rather simple with design complexity mostly resting on the CSS styling side of development. However, here is the code for its creation (copied across all primary webpages).



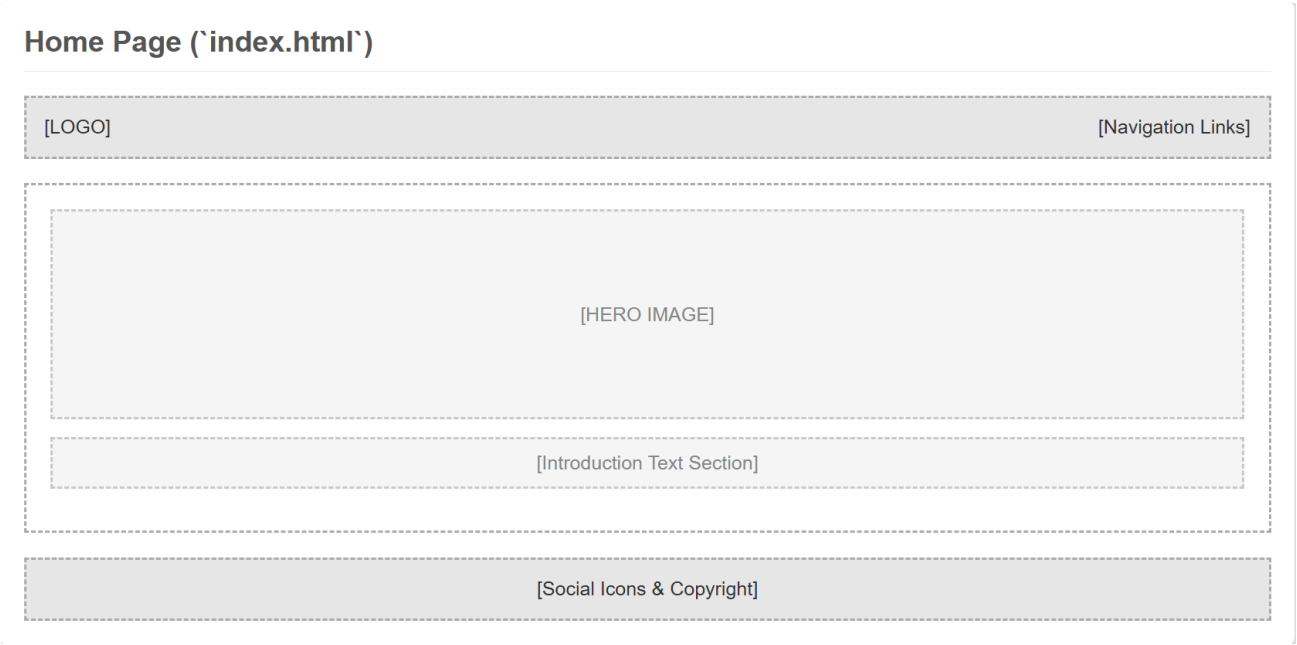
All primary pages are accessible from the fixed navigation bar on any other primary page.

```
<!-- Standard Container class allows for consistent styling -->
<div class="container">
    <header>
        <div class="logo-container">
            <h1 class="logo-text" data-text="SURYA NAIR">SURYA NAIR</h1> <!-- The 'data-text' attribute is used by CSS for the glitch effect. -->
            <p class="sub-logo-text">Game Developer</p>
        </div>
        <nav>
            <ul>
                <!-- Links from the index page point directly to other HTML files -->
                <li><a href="index.html" data-page="home">Home</a></li>
                <li><a href="Pages/resume.html" data-page="resume">Resume</a></li>
                <li><a href="Pages/gameplay_skills.html" data-page="gameplay_skills">Gameplay Design</a></li>
                <li><a href="Pages/leveldesign_skills.html" data-page="leveldesign_skills">Level Design</a></li>
                <li><a href="Pages/ai_skills.html" data-page="ai_skills">AI Programming</a></li>
                <li><a href="Pages/web_dev_skills.html" data-page="web_dev_skills">Web Skills</a></li>
                <li><a href="Pages/contact.html" data-page="contact">Contact</a></li>
            </ul>
        </nav>
    </header>
```

Storyboards - Content

The finalized ideas for each web page, with a storyboard explaining its purpose, visuals, user engagement and content in more depth are presented here.

Title: Homepage - Index.html **Screen ID: 1.0**



Key Elements:

Fixed Header: Contains the "SURYA NAIR" logo, sub-logo, and the primary navigation bar.

Hero Section: A large, full-width image representing a game project. Overlaid with a "Welcome, Player 1" title and a "View Project" button.

Introduction Section: A container below the hero section with a brief professional summary.

Footer: Contains social media links and copyright information.

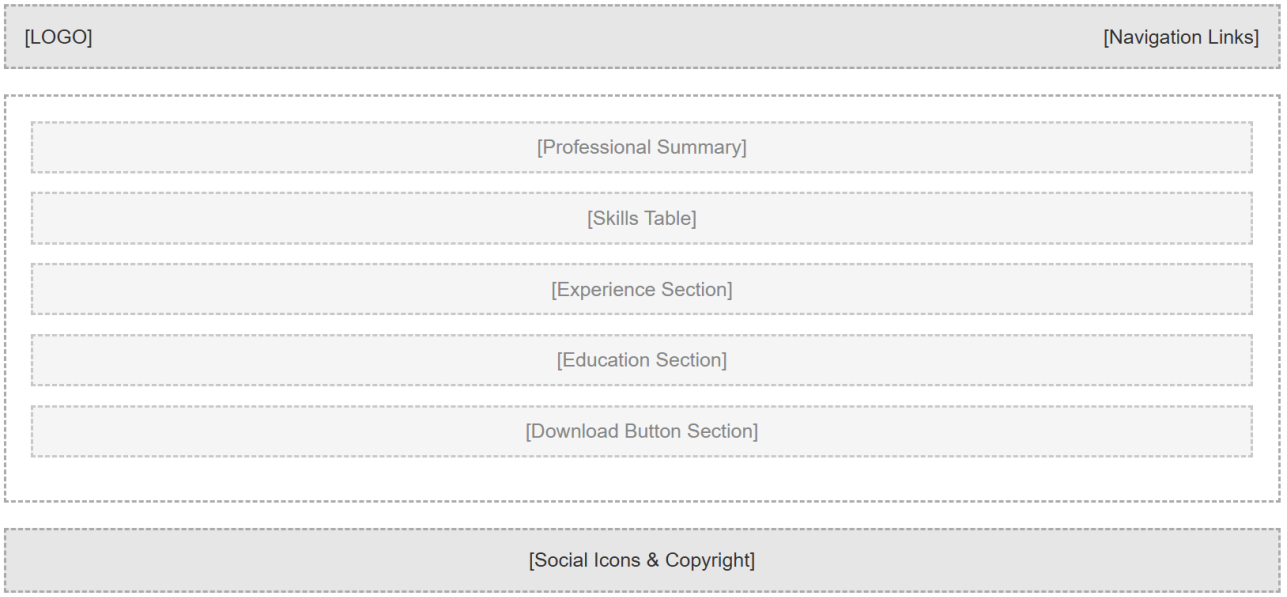
Dark Linear Gradient Background for visual variety and readability.

User Flow & Interaction:

1. On Load: The page fades in.
2. First View: The user is immediately presented with the visually striking hero section with game visuals in the background
3. Interaction:
    - "View Project" button sends users to gameplay page
    - Hovering over the navigation links triggers the "cyber-bracket" animation.
    - Scrolling down smoothly reveals the introduction section below the hero image.
    - Green glow when hovering over icons
4. Navigation: The user can click any link in the navigation bar to move to another page.

Title: Resume - SN_A4_resume.html **Screen ID: 2.0**

Resume Page (`resume.html`)



Objective: To present professional qualifications, skills, and experience in a clear, organized, and easily digestible format, while also providing a downloadable version.

Key Elements:

- Header & Footer: Consistent with the rest of the site.
- Main Content:
  - Professional Summary section.
  - A stylized skills table with categories (Technical, Software, Design).
  - Experience section with job titles, companies, and key responsibilities.
  - Education section.
  - A final section with a prominent "Download PDF" button.

User Flow & Interaction:

1. Arrival: The user clicks "Resume" from another page. The page fades in, and the "Resume" link in the header is now active with the bracket styling.
2. Scanning: The user can easily scan the content, which is broken into logical chunks inside styled containers. The table format for skills allows for quick assessment of capabilities.
3. Action: The user can click the "Download PDF" button, which will trigger a browser download for the resume.pdf file.

Title: Gameplay Design - SN_A4_gameplay_skills.html **Screen ID: 3.0**

Objective: To showcase ability to design and implement engaging core mechanics, player systems, and feedback loops. The focus is on demonstrating a deep understanding of what makes a game "feel" good to play, bridging the gap between a design document and a functional, satisfying experience.

Key Elements:

- **Header & Footer:** Consistent with the site's overall design.
- **Introductory Paragraph:** A design philosophy statement focusing on player agency, intuitive controls, and rewarding game loops. It establishes Surya's approach to gameplay as a conversation between the player and the game.
- **Project Article 1 (Cybernetic Echo):** Details the creation of the momentum-based movement and hybrid combat systems. The description goes beyond listing features, explaining *why* certain decisions were made (e.g., using "weapon-swap cancels" to promote high-skill expression).
- **Project Article 2 (Aetherion's Gate):** Details the design of the spell fusion system and risk/reward mechanics. This demonstrates versatility by showcasing skills in a different genre (top-down roguelike) and a focus on emergent gameplay systems.
- **Interactive Images:** Each article features a high-quality, cropped image

User Flow & Interaction:

1. **Arrival:** The user navigates from the main menu. The page fades in, with the "Gameplay Design" link now active.
2. **Conceptual Understanding:** The user reads the design philosophy, establishing a baseline understanding of developers' approach to creating player-centric mechanics.
3. **Detailed Examination:** The user examines the "Cybernetic Echo" project to understand how abstract concepts like "game feel" are translated into technical implementations. They can then expand the associated image to see a detailed, full-screen view of the fluid action.
4. **Versatility Check:** The user moves to the "Aetherion's Gate" project, recognizing varying skills and genres with unique design challenges (action vs. systems design).

Title: Level Design - SN_A4_leveldesign_skills..html **Screen ID: 4.0**

Objective: To demonstrate proficiency in crafting compelling and purposeful game spaces. This page showcases the ability to guide the player, facilitate interesting gameplay scenarios, and tell a story through the environment itself.

Key Elements:

- Header & Footer: Consistent branding and navigation.
- Introductory Paragraph: A philosophy statement about the importance of player flow, strategic sightlines, environmental storytelling, and balancing authored spaces with procedural systems.
- Project Article 1 ('Aetherion District'): Details the creation of a competitive multiplayer map, focusing on the intentional design of verticality and strategic chokepoints.
- Project Article 2 ('The Sunken Vault'): Details the design of a procedurally generated single-player dungeon. This highlights a different skill set: creating modular room templates, scripting unique encounters.
- Interactive Images: One image shows a top-down, annotated map layout for 'Aetherion District'. The other shows an atmospheric, in-game screenshot of 'The Sunken Vault' to highlight its theme.

User Flow & Interaction:

1. Arrival: The user arrives on the page, and the "Level Design" link is active.
2. Design Philosophy: The user reads the introduction to understand methodical approach to world-building.
3. Analysis of Authored Space: The user analyzes the 'Neon District' project, focusing on how the layout directly serves the needs of competitive gameplay. They click the image to expand the top-down map for a closer look.
4. Analysis of Procedural Space: The user then examines 'The Sunken Vault' project, demonstrating an understanding of a different, more systemic approach to level creation.

Title: AI Programming- SN_A4_ai_skills..html **Screen ID: 5.0**

Objective: To prove technical expertise and design sensibility in creating believable, challenging, and optimized non-player characters (NPCs) that enhance the game world.

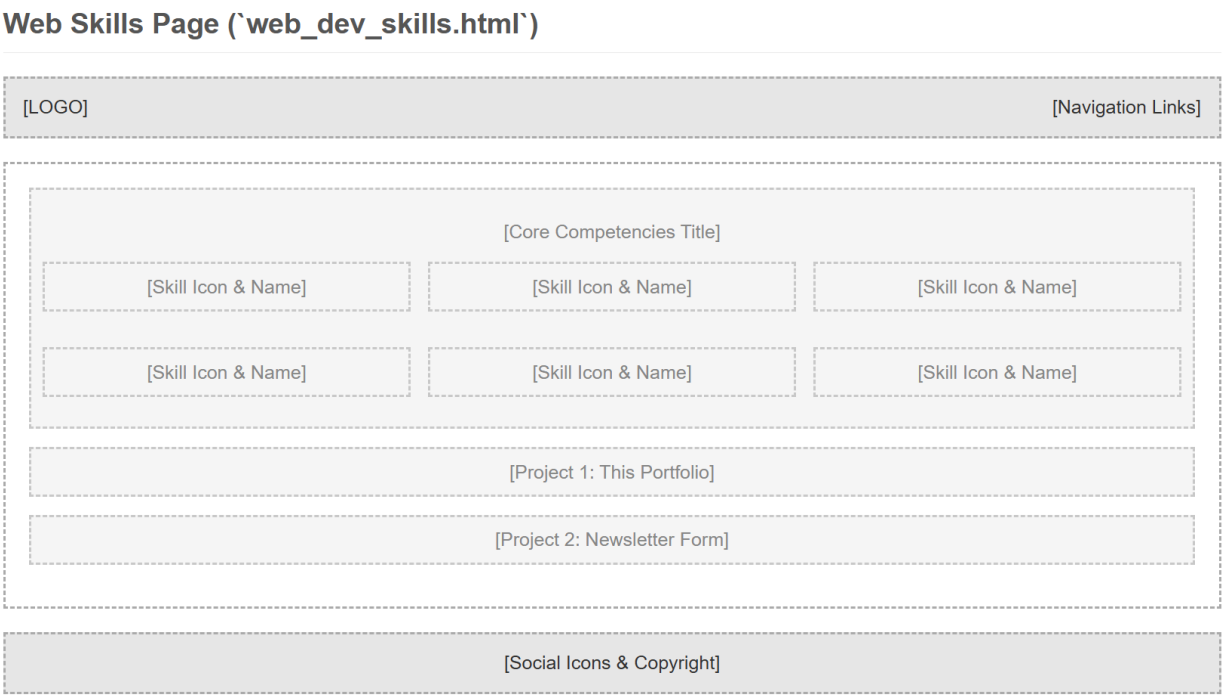
Key Elements:

- Header & Footer: Consistent.
- Introductory Paragraph: A philosophy on AI as a tool to create interesting challenges and enhance player immersion, rather than just creating simple "enemies."
- Project Article 1 (Dynamic Squad Tactics): Details the use of complex Behavior Trees for coordinated enemy squads, mentioning specific behaviors like cover usage, suppressing fire, and flanking maneuvers.
- Project Article 2 (Creature Ecosystem): Details the use of more performant Finite State Machines (FSMs) for simpler creature interactions, including a faction system that allows for emergent AI vs. AI combat.
- Interactive Images: Screenshot of player interacting with NPC.

User Flow & Interaction:

1. Arrival: The user lands on the page with "AI Programming" active.
2. Approach to AI: The user reads the intro to understand that Surya's focus is on creating "smart" and believable AI, not just functional cannon fodder.
3. Optimized AI Example: The user then views the "Creature Ecosystem" project, understanding how different AI techniques (FSMs) can be applied to different problems, demonstrating an awareness of performance optimization and context-appropriate design.

Title: Web Development - SN_A4_web_dev_skills.html **Screen ID: 6.0**



Objective: To showcase versatility and demonstrate foundational knowledge of front-end web development.

Key Elements:

- Header & Footer: Consistent.
- Skills Grid: A visually appealing grid of "skill cards," each with an icon and title (HTML5, CSS3, etc.).
- Project Articles: Detailed descriptions of web-based projects, including the portfolio itself and the newsletter form.
- "Open Form" Button: A stylized button that links to the newsletter signup page.

User Flow & Interaction:

1. Arrival: Page fades in with "Web Skills" active in the navigation.
2. Scanning: The user can quickly get an overview of competencies from the skills grid. Hovering over a card makes it lift up slightly.
3. Action: Reading the description of the newsletter form project, the user can click the "Open Signup Form" button, which opens newsletter_signup.html in a new browser tab.

Title: Contact Page - SN_A4_contact.html **Screen ID: 7.0**

Objective: To provide a simple and effective way for potential employers or collaborators to get in touch.

Key Elements:

- Header & Footer: Consistent.
- Contact Form: A fully styled form with all required fields (First/Last Name, Email, Age, Interests, Message).
- Submit/Reset Buttons.
- Validation Modal: A hidden pop-up for displaying error messages.

User Flow & Interaction:

1. Arrival: Page fades in with "Contact" active in the navigation.
2. Form Entry: The user fills out the form fields.
3. Validation:
    - If the user clicks "Submit" without filling in a required field (e.g., First Name), the validateNameFields function in script.js is triggered.
    - The form's submission is stopped (event.preventDefault()).
    - The custom validation modal appears with an error message like "First Name is a required field."
    - The user closes the modal and corrects the error.
4. Submission: Once all fields are valid, clicking "Submit" successfully sends the data to the form's action URL.

Organization of Information - Concept

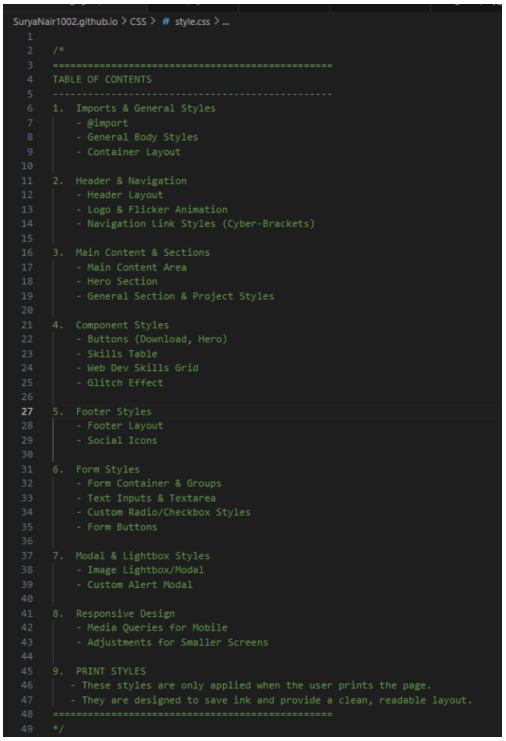
The information architecture is linear and straightforward, mimicking a standard professional portfolio. The navigation bar acts as the primary organizational tool, with links ordered logically from a general overview (Home) to specific skills (Gameplay, Level, AI, Web), and finally to actionable items (Contact).

This structure allows a visitor, such as a recruiter, to follow a clear path. They can get a first impression from the Home page, dive into specific skill sets that interest them, verify credentials on the Resume page, and then easily make contact. Each project within the skills pages is presented in its own <article> tag, creating a modular and organized flow of information.

Information is presented in mixed forms such as lists, tables, images and paragraphs with appropriate headings. This will ensure users are more engaged than just reading a brick wall of text. Paragraphs have been kept as short and concise as possible for this reason - with heavy detail omitted and (perhaps) can be incorporated in new sections of the website in the future for the enthusiasts who would like to explore further.

Usage of CSS

This section provides a detailed breakdown of CSS implementation, covering theming, layout, animations, and best practices. The entire visual presentation of the website is controlled by a single, external stylesheet named style.css. This approach follows the core web development principle of "Separation of Concerns," where the structure (HTML) is kept separate from the presentation (CSS). Using an external stylesheet ensures that the design is consistent across all pages, and allows for efficient, site-wide updates; a change made in this one file will be reflected on every page of the portfolio. The external sheet is given a table of contents as seen below in order to improve readability and efficiency.



1. Imports & General Styles

This initial section of the stylesheet sets the foundation for the entire site's look and feel.

- @import: To keep the HTML clean and centralize dependencies, the CSS file itself handles the importing of external resources. Placing these at the very top of the file is a best practice, ensuring the fonts and Icons are available results in a faster load time.

- General Body Styles: The body selector establishes site-wide defaults. The text color is set to pure white (#ffffff) for high

contrast against the dark background. A key decision was to set the background to a linear-gradient that fades, which gives the page a more dynamic, "glossy screen" feel than a flat color. A padding-top is also applied to the body to create space for the fixed header, preventing content from being hidden underneath it on page load.

- Container Layout: The main .container class acts as a wrapper for all page content. It is configured as a flex container (display: flex) with a vertical direction (flex-direction: column). The most crucial part of this setup is giving the <main> content area flex-grow: 1, which forces it to expand and fill any available vertical space. This creates a "sticky footer," solving the common layout problem where the footer would float halfway up the page on pages with little content.

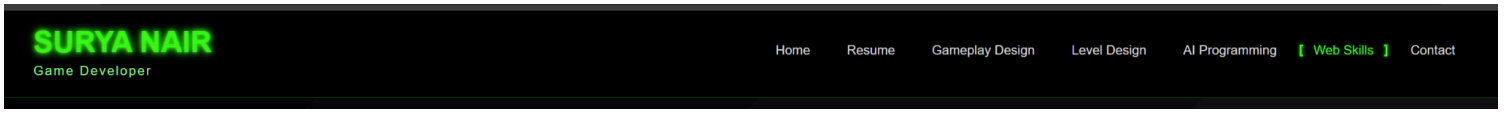
2. Header & Navigation

This section styles the persistent navigation elements that appear at the top of every page.

- Header Layout: The <header> element is given position: fixed with a high z-index (100). This removes it from the normal document flow and fixes it to the top of the viewport, ensuring it remains visible as the user scrolls. It has a solid black background (#000000) and a backdrop-filter: blur(5px) to create a distinct, semi-translucent "frosted glass" effect that separates it from the content scrolling beneath.

- Logo: The main logo text is styled with the neon green accent color. The controlled-flicker animation is a multi-step @keyframes rule that rapidly changes the opacity and text-shadow. This creates a more complex and believable "dying neon" effect rather than a simple blink. The animation is applied via a class added by JavaScript on page load, and by the CSS :hover pseudo-class for direct user interaction.

- Navigation Link Styles (Cyber-Brackets): The navigation links were styled to be interactive without causing layout shifts. The final "cyber-bracket" design uses the ::before and ::after pseudo-elements. These are positioned absolutely and initially moved off-screen using transform: translateX(). On hover, or when the link has an .active class, their transform is set back to translateX(0), creating a smooth slide-in animation that doesn't affect the position or spacing of any other elements.



3. Main Content & Sections

This section defines the look of the main content containers.

- Main Content Area: The <main> tag itself is given vertical padding to ensure space between the header/footer and the content within. As mentioned in the layout section, its flex-grow: 1 property is essential for the overall page structure.

- Hero Section: The .hero-section on the homepage is designed to be a large, attention-grabbing banner. It uses a background image with background-size: cover to ensure the image always fills the section without being distorted. An inset box-shadow is used to darken the edges, which helps draw the user's eye to the centered text.

- General Section & Project Styles: A consistent style is applied to all <section> and .project containers. They share a semi-transparent background, a subtle black border, and a faint box-shadow that makes them appear to float above the page background. This creates a clear, organized, and modular structure for presenting information.

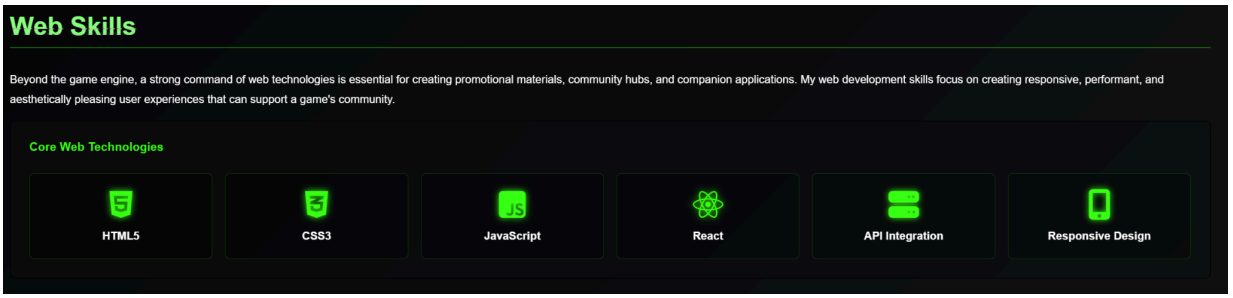
4. Component Styles

This section details reusable components that appear in multiple places.

- Buttons (Download, Hero): The main stylized buttons share a common class (.download-button or .hero-link). They feature a neon green border and text, with a glowing box-shadow. On hover, the background fills with the neon green color and the text becomes black, providing clear and satisfying user feedback.

- Skills Table: The table on the resume page is styled with thin, neon green borders. The table headers (<th>) have a semi-transparent green background to distinguish them from the data cells (<td>), which helps with scannability.

- Web Dev Skills Grid: This component on the web skills page uses display: grid with a responsive column layout (repeat(auto-fit, minmax(150px, 1fr))). This allows the grid items (skill cards) to automatically wrap and resize to fit the available screen width.



Skills Icons with Green glow

- Glitch Effect: The .glitch class is applied to all main <h2> headings. It uses ::before and ::after pseudo-elements to create two duplicates of the heading text. These duplicates are then shifted slightly using clip-path and given contrasting text-shadow colors. An infinite CSS animation rapidly changes the clip-path, creating the illusion of a digital glitch or distortion.

5. Footer Styles

The footer is styled to be a distinct, full-width block at the bottom of the page.

- Footer Layout: The <footer> has a solid black background, matching the header. A key challenge was making it span the full width of the viewport while its parent (.container) has side padding. This was solved by setting its width to 100vw (100% of the viewport width) and then using a negative margin-left to counteract the padding of the parent container.

- Social Icons: The social media links are styled to be large and easily clickable. The icons themselves are sourced from Font Awesome, and their color changes to the theme's neon green on hover.

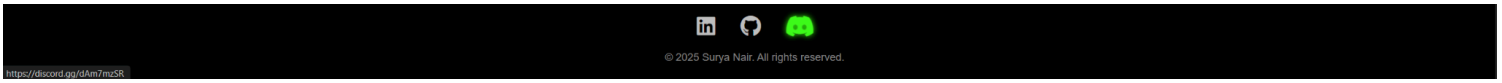


Image showing footer (cursor is hovering over “Discord” media icon)

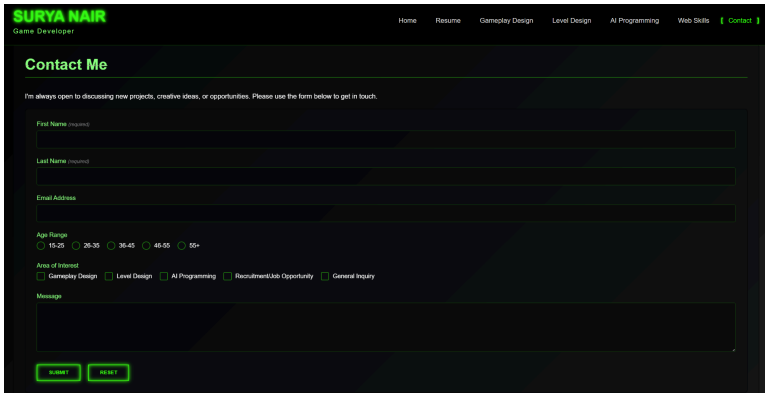
6. Form Styles

The forms on the contact and newsletter pages are heavily styled to match the site's theme, avoiding default browser styles.

- Form Container & Groups: The entire form is wrapped in a .form-container or .contact-methods div, which gives it the same floating, stylized box appearance as other content sections. Each label-input pair is wrapped in a .form-group for consistent spacing.

- Text Inputs & Textarea: All text fields have a semi-transparent background and a thin neon border. On :focus, the border color becomes solid and a glowing box-shadow appears, providing clear feedback to the user that they are actively editing that field.

- Custom Radio/Checkbox Styles: The default browser radio buttons and checkboxes are hidden using display: none. They are replaced with custom-styled ::before pseudo-elements on the <label>. When the hidden input is :checked, the adjacent sibling selector (+) is used to change the background color of the custom element, indicating selection.



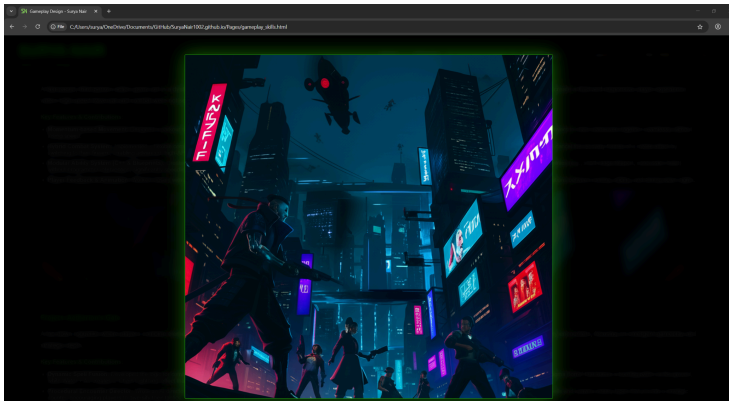
Contact Page (form) stylized using standard website aesthetic.

7. Modal & Lightbox Styles

The site uses two types of modals, both styled for a consistent experience.

- Image Lightbox/Modal: The .image-modal-overlay is a position: fixed container that covers the entire screen with a semi-transparent black background. The backdrop-filter: blur(8px) property creates a modern "frosted glass" effect over the content behind it. When the modal becomes visible, the image inside scales up slightly for a smooth transition.

- Custom Alert Modal: Used for form validation and welcome messages, this modal shares a similar overlay style. The content box is designed to match the aesthetic of the other page containers, ensuring that even system messages feel integrated with the site's theme.



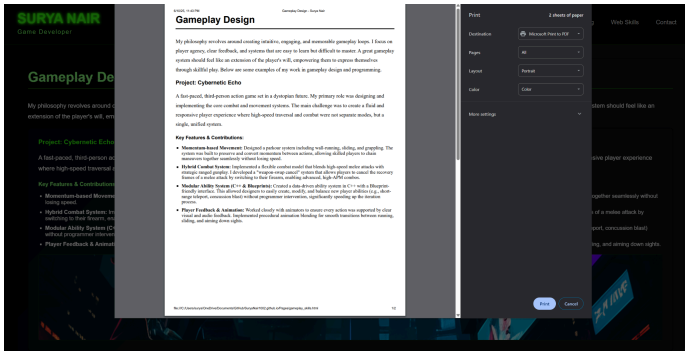
An example Image Lightbox Modal on Gameplay Design Page.

8. Responsive Design

This section of the stylesheet uses @media (max-width: 768px) to apply specific style overrides for screens smaller than 768px, such as tablets and mobile phones. The most significant changes are to the header and navigation, which stack vertically to be more touch-friendly. (Note: This section is currently commented out for testing purposes).

9. Print Styles

This final section uses @media print to define styles that are only applied when a user attempts to print a page. These styles are purely functional, designed to create a clean, readable, and ink-saving document. They remove all backgrounds, colors, and non-essential elements like the header, footer, and navigation, and format the text in a simple, high-contrast, black-on-white layout.



An example of Printing friendly webpage (Gameplay Design)

10. Favicon

After long consideration, a favicon was chosen that represented my initials (SN) in a modern tech themed neon green font serving as the best representation of the website. It is implemented manually in every webpage in the head section.

HTML Form and JavaScript

Game Dev Weekly Form Data: **Form data**
Form is available on website in "Web_skills" page

This section provides the creation process, technical implementation, and client-side scripting for the forms used in the Surya Nair portfolio website.

1. HTML Structure and Semantics

The foundation of any web form is its HTML structure. The forms on this website were built with a focus on semantic correctness, accessibility, and ease of styling.

- The <form> Element: This is the container for all form-related inputs.
  - id: Each form is given a unique id (contact-form, signup-form). This is crucial for the external JavaScript file to target the correct form and attach event listeners without ambiguity.
  - action: This attribute specifies the URL where the collected data should be sent upon submission. While the forms point to a test endpoint, this was designed to be replaced with a Google Apps Script URL for live data storage.
  - method="post": The POST method is used for all submissions. This is a web standard and a security best practice, as it sends the form data in the body of the HTTP request, rather than exposing it in the URL like the GET method would.
  - onsubmit: This attribute is used on the newsletter form to call the handleSignupSubmit(event) JavaScript function. It acts as a direct trigger for a more advanced, asynchronous submission process.

- Labels and Inputs: A core accessibility principle is to explicitly link every form input with its corresponding <label>.
  - The for attribute of a <label> is matched with the id of its associated <input>. This creates a connection that allows users of assistive technologies (like screen readers) to understand the purpose of each field. It also provides a better user experience for everyone, as clicking on the label will focus the cursor on the input field.

- Input Types: Appropriate HTML5 input types were used for each field to leverage built-in browser functionality and improve user experience, especially on mobile devices.
  - type="text": Used for the First Name and Last Name fields.
  - type="email": Used for the email address field.
  - type="radio": Used for the "Age Range" selection. All radio buttons in a group share the same name attribute (ageRange), which is what ensures that only one can be selected at a time.
  - type="checkbox": Used for the "Area of Interest" selection.

- Structural divs: For styling and organizational purposes, inputs are wrapped in elements with specific classes (.form-group, .radio-group, .checkbox-group). This allows CSS to easily target and style the elements consistently and create the desired horizontal layout for the radio and checkbox options.

The GameDev Weekly Newsletter signup form.

2. JavaScript Implementation (Behavior and Interactivity)

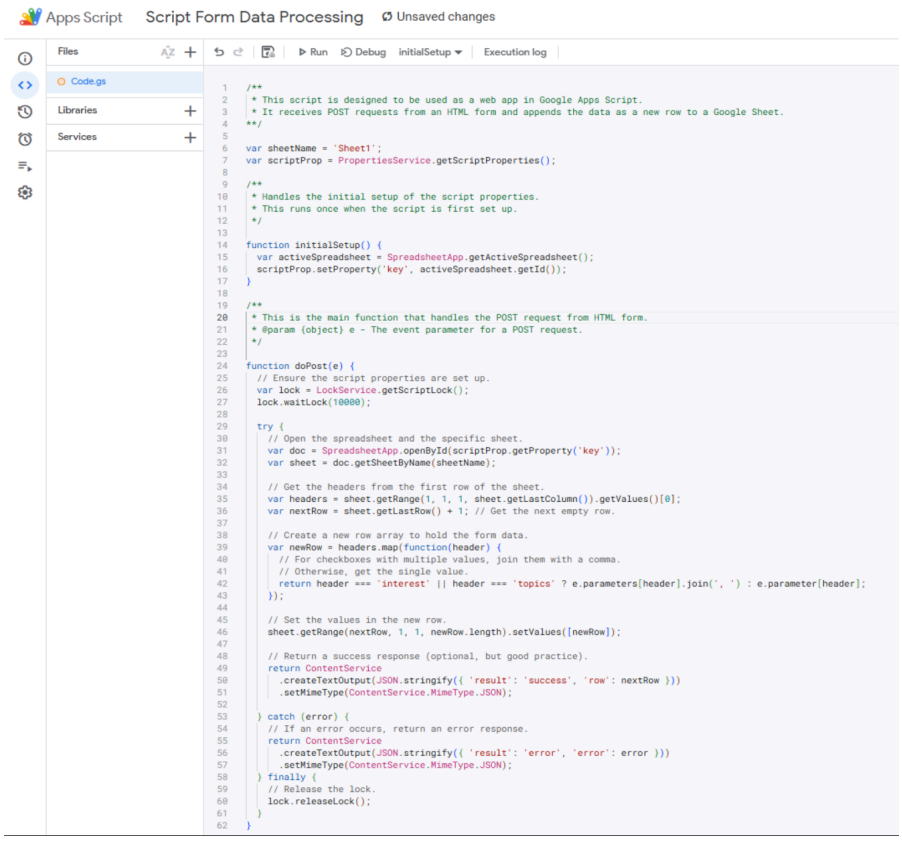
All client-side interactivity is managed by a single external file, script.js. This adheres to the "Separation of Concerns" principle.

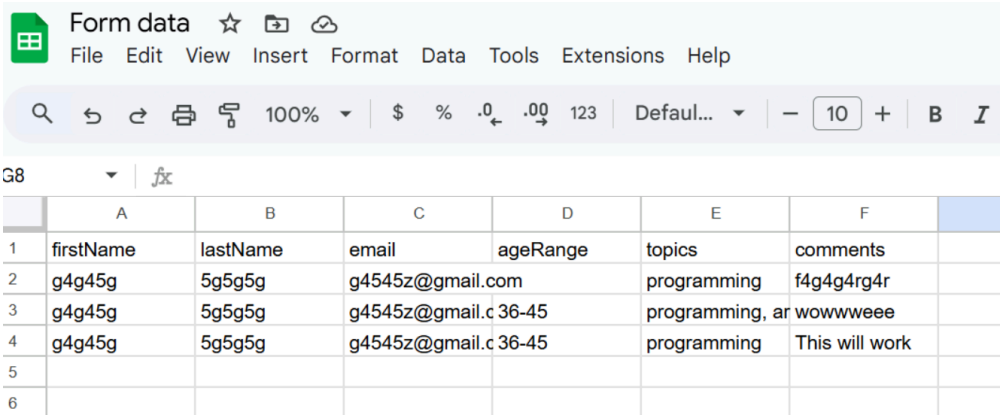
- DOMContentLoaded Event Listener: The entire script is wrapped in a document.addEventListener('DOMContentLoaded', () => { ... });. This is a critical best practice that prevents the script from running until the entire HTML document has been fully loaded and parsed by the browser, avoiding common errors.
- Asynchronous Submission (Newsletter Form): The newsletter form uses the fetch API to submit data in the background without a page reload. This process is initiated by the handleSignupSubmit(event) function, which first calls event.preventDefault() to stop the default browser navigation. This gives JavaScript full control over the user experience, allowing for dynamic feedback like changing the button text to "Submitting..." and displaying a custom "Thank You" modal upon completion.

3. Data Persistence with Google Sheets & Apps Script:

To move beyond a simple test endpoint and create a functional data storage solution without a traditional back-end server, a Google Sheet was leveraged as a free database.

1. The Back-End Bridge (Google Apps Script): A Google Apps Script was created and attached to a Google Sheet. This script contains a special doPost(e) function, which is designed to act as a web API endpoint. When the script is deployed as a "Web App," Google provides a unique URL for it. This URL becomes the new action target for our form.
2. Receiving Data: When the form is submitted, the fetch request from the website sends the form data to the Apps Script URL. The doPost(e) function receives this data in the event object e.
3. Processing and Storing: The script then opens the associated Google Sheet, gets the headers from the first row, and iterates through them. For each header, it finds the matching data in the submitted form parameters (e.parameter). It then assembles this data into a new row and appends it to the bottom of the spreadsheet.
4. Public Data Viewing: To allow anyone to view the submitted data, the Google Sheet's sharing settings are changed to "Anyone with the link can view." This generates a public, read-only URL for the spreadsheet, which can be shared to display all the collected form submissions in a clean, tabular format.





4. Client-Side Validation

To provide immediate feedback and prevent bad data from being sent, client-side validation is performed before any form submission.

- The validateNameFields function: This reusable function takes a form element as an argument. It finds the "first-name" and "last-name" inputs within that specific form.
- .trim() Method: When retrieving the values, the .trim() method is used (e.g., value.trim()). This is a robust way to handle user input, as it removes any leading or trailing whitespace. This ensures that a field containing only spaces is still considered empty and will fail validation.
- Preventing Submission: If validation fails, event.preventDefault() is called to stop the form submission process entirely. The function then immediately calls showModal() to display a specific error message to the user.

4. Custom Modal System

To maintain the immersive, cyberpunk aesthetic of the website, the default browser alert() function is never used. Instead, a custom modal system was built with HTML/CSS and is controlled by JavaScript.

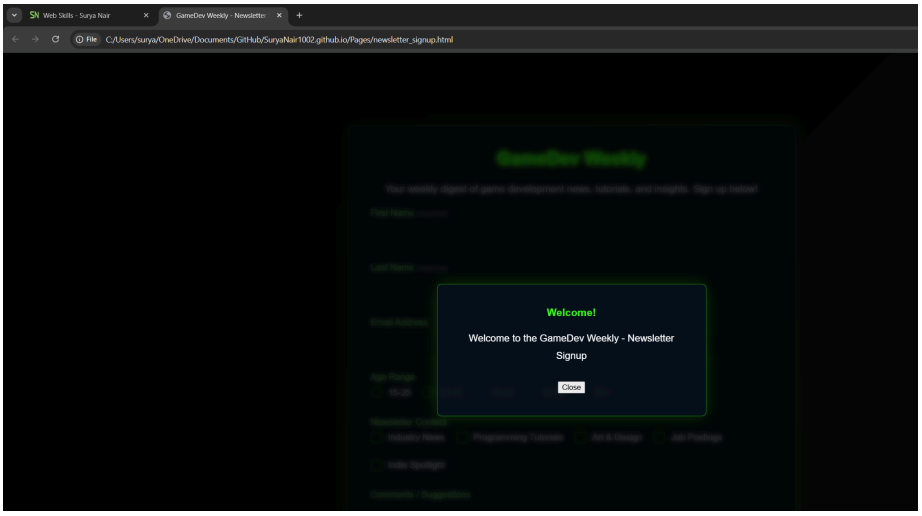
- Rationale: The default alert() is jarring, cannot be styled, and breaks the user's immersion in the site's theme. A custom modal provides a seamless, professional, and thematically consistent way to display important messages.

- Functionality: The showModal(title, message) function dynamically updates the text content of the modal's <h3> and <p> tags and then makes it visible by adding a .visible class to the overlay. The closeModal() function simply removes this class, hiding it again with a CSS fade-out transition. This provides a clean, reusable system for all alerts.

5. onload Event Handling

The newsletter signup page required a welcome message to appear when the page loads. Instead of adding an inline onload attribute to the <body> tag in the HTML, this was handled cleanly within the main script.js file.

- Context-Aware Scripting: Inside the DOMContentLoaded listener, the script checks if the body has the class .signup-page (if (document.body.classList.contains('signup-page'))). This check ensures that the welcome message logic *only* runs when the user is on the newsletter page, demonstrating a best practice for writing a single script that can serve an entire multi-page website without causing unintended side effects on other pages.



Code Reusability and Readability

Core Principles and Architecture

The entire project was built on two fundamental software development principles that guided all architectural decisions: "Don't Repeat Yourself" (DRY) and "Separation of Concerns" (SoC). The SoC principle was implemented by strictly separating the website's structure (HTML), presentation (CSS), and behavior (JavaScript) into their own dedicated files. This makes the codebase significantly easier to navigate and debug. The DRY principle was then applied within this structure by abstracting common functionality into single, reusable locations. For example, all primary button styling is defined by a shared CSS class, and all form validation logic is contained in a single JavaScript function, making the code easier to maintain and less prone to errors.

HTML Formatting for Readability

While HTML is a markup language, consistent formatting is crucial for understanding the document's structure and the relationships between elements. Wherever possible, semantic HTML5 elements like <header>, <main>, <footer>, and <article> were used instead of generic tags. This practice makes the structure of the document immediately understandable to other developers and provides better context for assistive technologies and search engines. Every nested element is indented consistently

using four spaces, and for more complex or non-obvious sections of HTML, comments are used to explain the purpose of the markup.

CSS Formatting and Reusability

The external stylesheet, style.css, serves as the single source for the site's entire visual presentation, and its organization is key to maintainability. The file begins with a large, commented-out table of contents, which acts as a map, allowing a developer to quickly understand the structure and find the section they need without scrolling. The file is further broken into logical sections using large, distinct block comments, making the code able to be skimmed quickly.

Styles are ordered logically from most general to most specific, starting with @import rules, followed by general body styles, layout containers, major site sections, reusable components, and finally, media queries. This ensures that general styles are established first and can be predictably overridden by more specific styles later in the file. To ensure consistency, all custom class names use kebab-case (e.g., .form-group, .project-image), which is the most common naming convention in CSS.

JavaScript Readability and Reusability

The single external script.js file is designed to be efficient, error-free, and easy to understand. A significant challenge in a multi-page site is writing a single script that doesn't produce errors on pages that lack certain elements. This was solved by designing the script to be "page-agnostic." Before attaching an event listener, the script first checks if the target element actually exists on the current page. For example, the code is structured to only attach a listener to the contact form if document.getElementById('contact-form') returns a valid element. This defensive coding practice prevents the script from throwing null reference errors.

Common tasks were abstracted into reusable functions that can be called from multiple places. For example, the showModal(title, message) function is used for the newsletter welcome, form validation errors, and form submission success messages. This is a core practice of DRY programming that makes the code cleaner and easier to debug. To make the purpose of each function immediately clear, a professional commenting style similar to JSDoc is used, explaining what the function does and what parameters it expects. Finally, all code is wrapped in a DOMContentLoaded event listener. This is a critical best practice that ensures the script only attempts to find and manipulate HTML elements *after* the entire document has been parsed by the browser, preventing a whole class of common JavaScript error

Github Link: https://suryanair1002.github.io/

Validation

This section will provide the testing plan and provide evidence for the validation of all files.

Testing Plan

Goal: Ensure the website functions correctly, looks good on different devices, and presents information accurately.

How: Manual testing performed using various browsers and devices

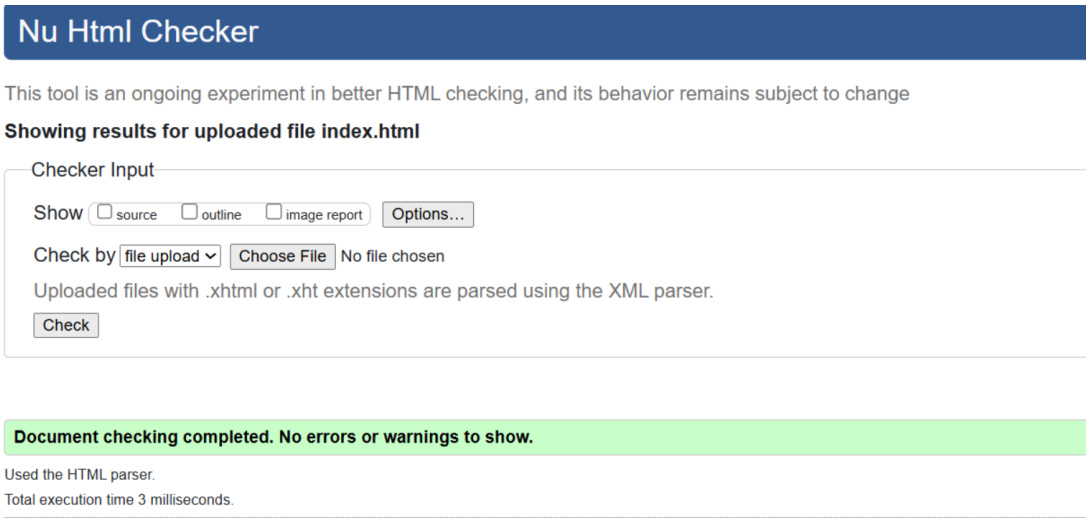
What to Test:

- **Functionality:**
  - All internal navigation links work correctly (Header, Footer, in-page links).
  - All external links (Social media, GitHub, Download links) open correctly (ideally in a new tab)
- **Responsiveness:**
  - Layout adapts correctly at different screen widths (Desktop Monitor, Laptop, Mobile). Check for horizontal scrollbars (should be none) and overflow text/misalignments.
  - Text remains readable and stylish, images resize appropriately, navigation remains usable on small screens.
- **Cross-Browser Compatibility:**
  - Render the site in the latest versions of major browsers (e.g., Chrome, Firefox, Edge, Safari) to check for major layout inconsistencies or visual bugs.

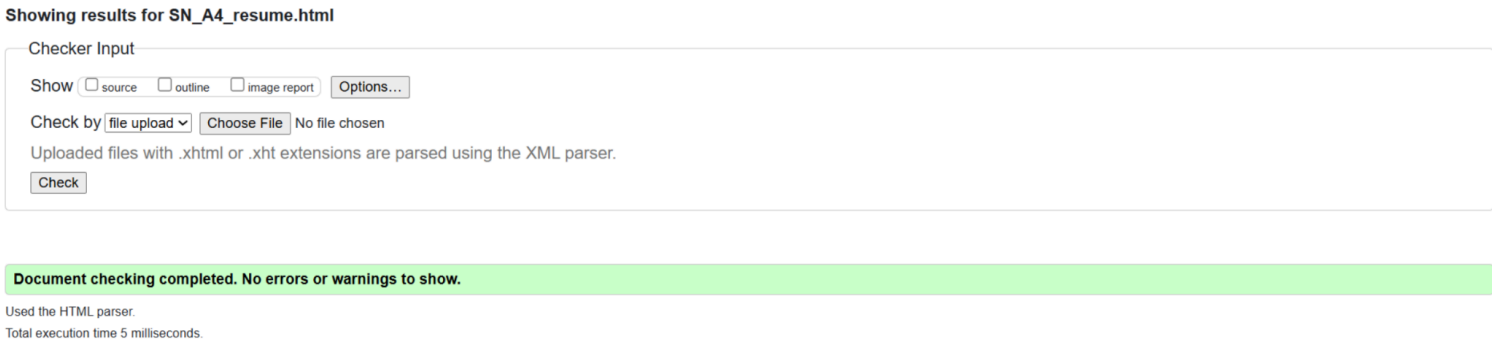
- **Content Accuracy:**
  - Proofread all text content for typos and grammatical errors.
  - Verify project details, skills, and resume information are **consistent.**
  - Does content meet standards expected by an **"online video game"** company as stated in A4
- **Accessibility (Basic Checks):**
  - Ensure sufficient color contrast between text and background.
  - Check if semantic HTML is used correctly (headings in order, landmarks like **<header>, <main>, <footer>**).
  - Verify images have descriptive **alt** text

This section covers the validation of each webpage via the official W3C Markup Validation service as well as the CSS file using W3C CSS validation service.

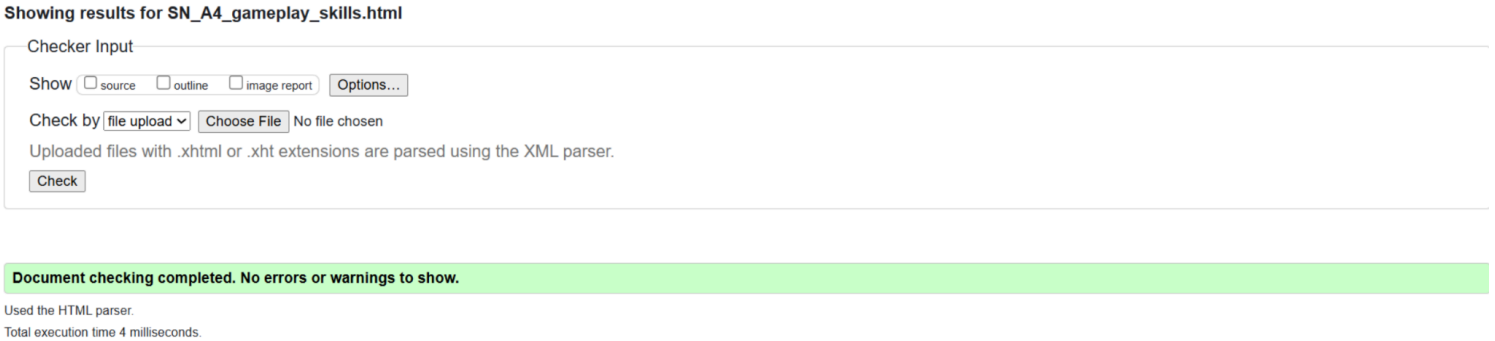
Index.html



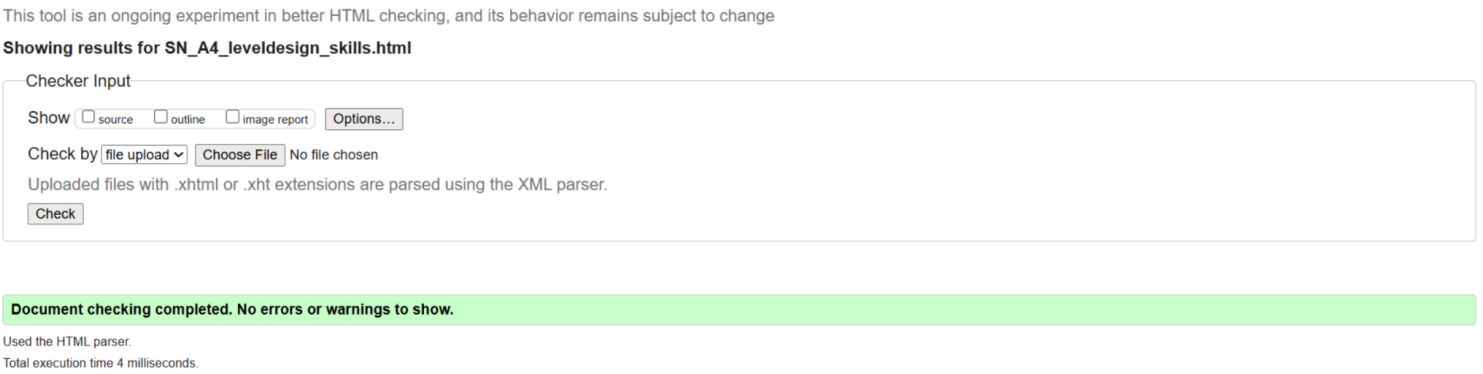
SN_A4_resume.html



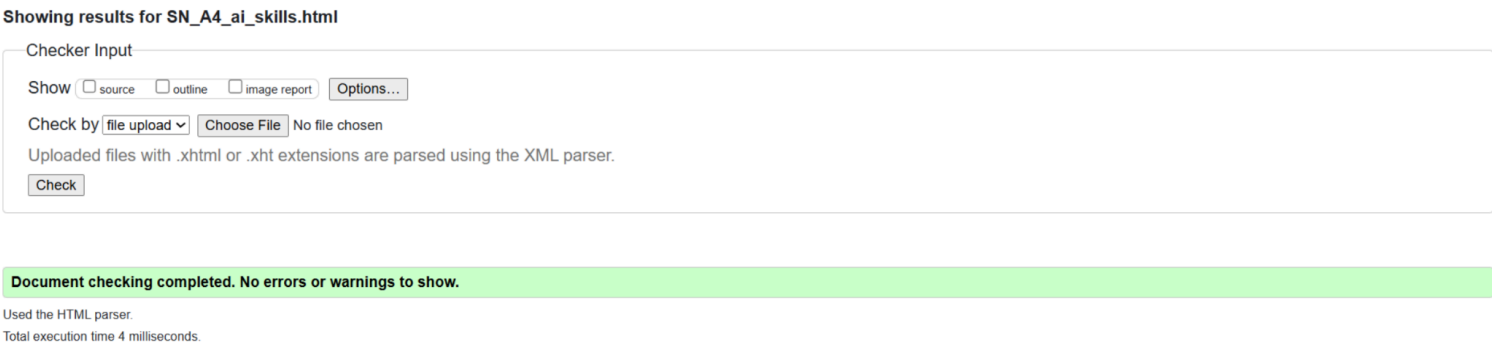
SN_A4_gameplay_skills.html



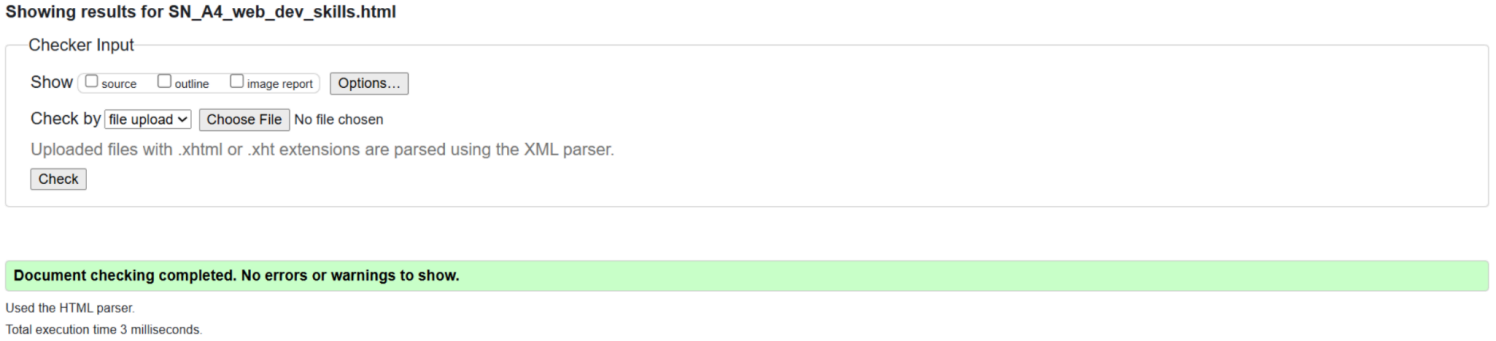
SN_A4_Leveldesign_skills.html



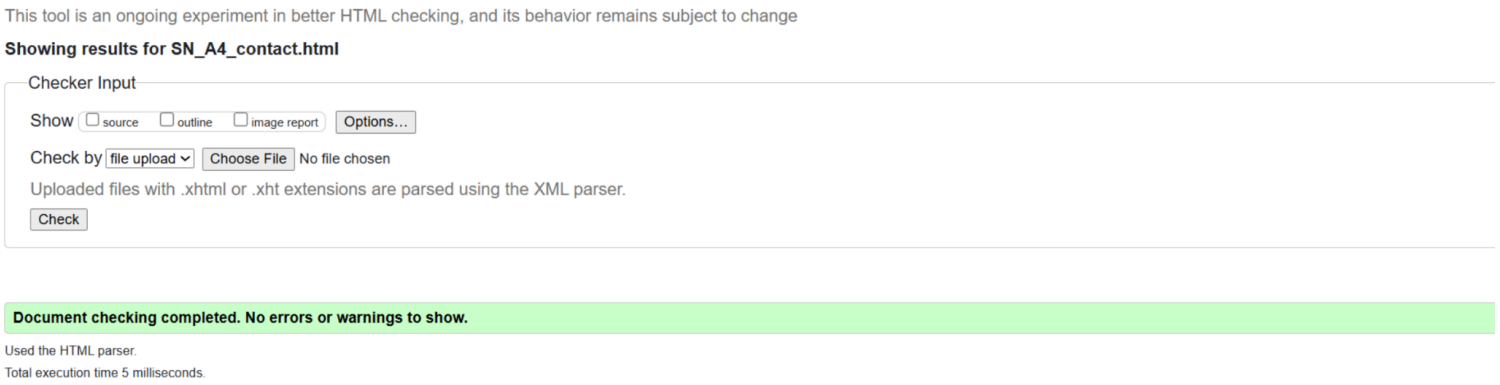
SN_A4_ai_skills.html



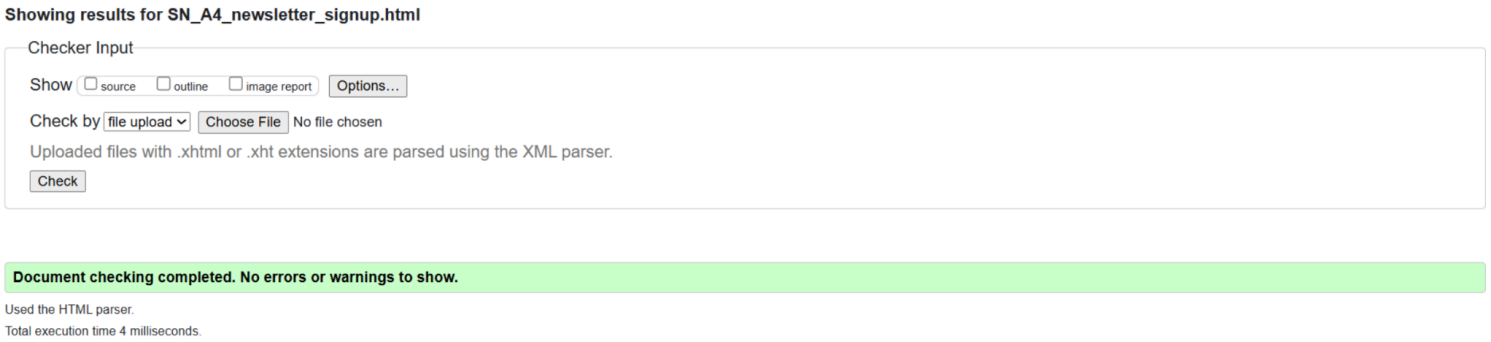
SN_A4_web_dev_skills.html



SN_A4_contact.html



SN_A4_newsletter_signup.html



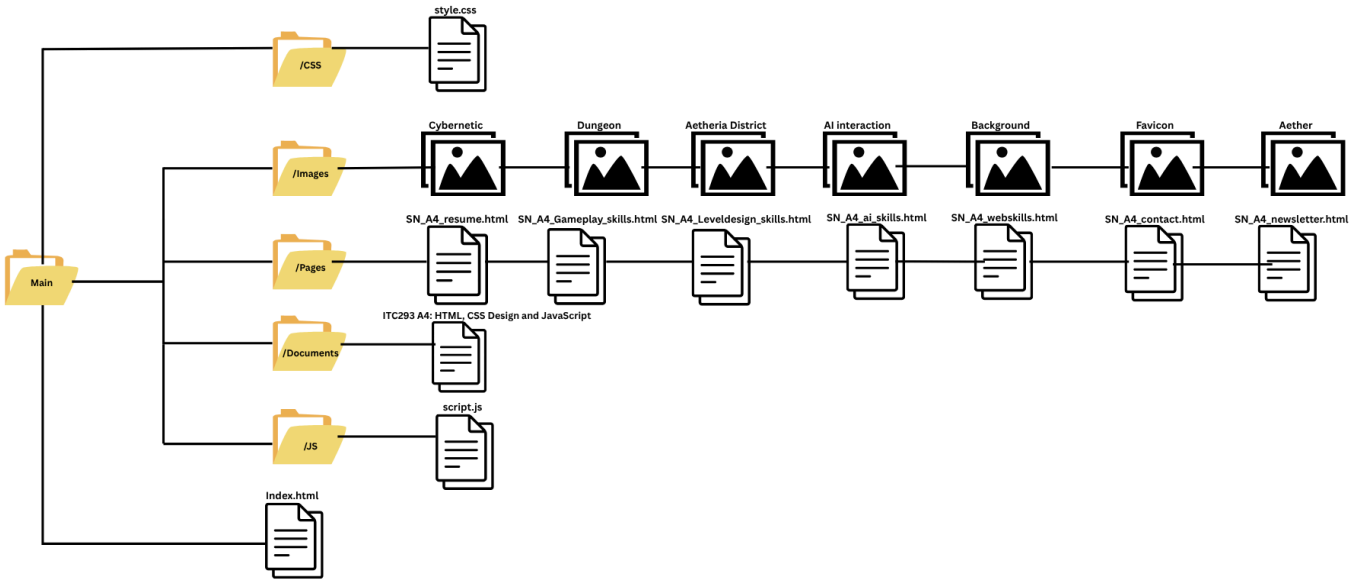
CSS Validation: style.css

File Sizes and Naming

1. File and Folder Structure

The project adheres to the SoC principle not just in its code, but also at the file system level, achieved through a clear and logical folder structure.

- Root Directory: The root of the project contains the primary index.html file and all other top-level HTML pages (resume.html, contact.html, etc.). This provides a flat and easily accessible structure for the main pages of the site.\
- CSS/ Folder: All styling information is contained within this folder. The single style.css file serves as the single source of truth for the entire website's presentation. This organization means that any developer knows exactly where to look for visual styles, and it keeps the root directory uncluttered.
- JS/ Folder: All client-side interactivity and behavior is housed in the script.js file within this folder. This isolates the website's logic from its structure and presentation, making the code much easier to debug and manage.
- Documents & Images/ Folder (Conceptual): dedicated folder for all media assets, such as the project images, a downloadable PDF resume, and any other media. This keeps static assets separate from the core code files.



This structure is highly scalable. As new pages or sections are added, they can simply be placed in the root directory and link to the existing, centralized CSS and JS files, instantly adopting the site's functionality and design without code duplication.

2. File Naming Conventions

A consistent naming convention is used across all files to improve readability and prevent common server-related issues.

- HTML Files: All HTML files use kebab-case (e.g., gameplay_skills.html, newsletter_signup.html). This convention is a web standard for several reasons:
  - Readability: The names are easy to read and immediately describe the content of the page.
  - URL-Friendly: They produce clean, readable URLs without spaces or special characters.
  - Server Compatibility: It avoids issues with case-sensitivity
- Core Assets (style.css, script.js): The core stylesheet and script are given simple, conventional names. This is standard practice and makes their purpose immediately obvious to anyone familiar with web development.
- Image Files: For assets like project images, a descriptive naming convention is used (e.g., cybernetic.png). This is not only helpful for organization but also beneficial for Search Engine Optimization (SEO), as search engines can parse the file names for keywords.

3. File Size and Performance Optimization

A key focus of the development process was to ensure the website is fast and responsive for the user. Several intentional decisions were made to optimize performance by minimizing file sizes and reducing the number of network requests.

- CSS Over Images for Backgrounds: The most significant optimization was the decision to remove animated background effects (initially planned to be a starfield), replacing it with the linear gradient geometric pattern discussed above..
    - HTTP Requests: An image-based starfield would have required the browser to make multiple network requests to download the image files, adding latency to the initial page load. The pure CSS Linear-gradient and repeating-linear-gradient effects require zero extra requests.
    - Page Weight: The CSS code to generate these effects is minuscule in file size compared to even heavily optimized PNG or JPG images. This reduces the total amount of data the user needs to download, which is especially important for users on slower mobile connections.

- Image Optimization (for Project Images): For the raster images that are used (the project screenshots), best practices for optimization would be followed. This includes:
    - Choosing the Right Format: Using .jpg for photographic or complex images and .png for graphics that require transparency.

    - Correct Sizing: Ensuring that the images are not significantly larger in resolution than they need to be for their display size. An image that is 4000px wide being displayed in an 800px container is a major and common cause of slow page loads

Video Presentation

Video Link: Google Drive

Video Presentation - Made with Clipchamp_1749593817612.mp4

Note there is an error where I open an outdated repository leading to a dysfunctional form, however I end the recording and start a new one where it left off to correct the mistake. This means the footage of the dysfunctional form is still in the video for about 10 seconds although I believe it is worthwhile having the error in there as most of this assignment was dealing with various bugs. Most importantly, it was resolved anyway.

The video presentation does not go over the design document extensively owing to its large size and detail, the time limit had already been surpassed at 17 minutes by the end of recording.

Reflection

This project from start to finish was an exhilarating ride; the journey that started at A2 with the initial conception through to technical implementation and overcoming huge roadblocks to create what I believe to be an incredible final result (given my initial skillset). The goal was to create a portfolio that did more than just list skills; it needed to tell a story and immediately communicate a passion for the video game industry. I decided to move away from a traditional, sterile corporate look in an effort to create a unique feel for the site. There would be no point in creating a personal promotion website for it to only look like every other one. Though I'm sure this one has a long way to go before actually being considered a serious contender - it has the right "feel" and foundation. The core design philosophy was that the portfolio itself should feel like a small, interactive game world, reflecting the skills it was built to showcase.

The development process was filled with several key challenges leading to significant learning opportunities and architectural changes. My initial plan was to build the site as a Single-Page Application (SPA) for seamless, animated transitions. However, I soon recognized the significant negative impact this would have on Search Engine Optimization (SEO). For a portfolio that needs to be discoverable, having distinct, indexable URLs for each page is crucial. This led to a major pivot to a traditional Multi-Page Application (MPA), where the trade-off of losing fluid transitions was compensated by a simple, elegant fade-in animation on every page, ensuring the site remained polished and SEO-friendly.

One of the most persistent challenges was the animated background. An initial attempt using linked images failed due to unreliability, and a second attempt highlighted performance issues and CSS layering problems. This journey was a powerful lesson in prioritizing performance, leading to the ultimate solution of removing the images entirely and creating the background effects with pure CSS. The final implementation uses a Linear-gradient for the glossy background, which is infinitely more performant as it requires no extra file downloads. Similarly, the implementation of the contact forms evolved from using a simple "echo" server for testing to a more dynamic, application-like experience. To make the site truly functional, I leveraged Google Sheets as a free database, writing a Google Apps Script to act as a web API. The newsletter form was then updated to use the JavaScript fetch API to send its data to this script in the background, allowing me to prevent the default page reload and provide immediate user feedback with a custom "Thank You" modal. By no means was this simple as it required hours of persistence with little reward for quite some time. I struggled immensely with using github and hosting the site due to absolutely no knowledge of how repositories worked. This led to the corruption and deletion of my files over 3 full times, meaning I had to restart on several occasions. It turned out to be a blessing in disguise as my frustration of losing the site led to an impulsive pivot towards a new fresh look that deviated from my earlier assignments. At this point, I cannot say I have mastered github however I have managed to successfully host the site and learned about push and pull requests alongside how "commits" work (the hard way).

Throughout this project, several key lessons solidified my understanding of web development best practices. The importance of having a clear design document from the outset proved invaluable, even as the plan evolved. I learned that complex visual effects can often be achieved more performantly and reliably with pure CSS than with images or JavaScript, and that mastering CSS layout and stacking context is essential for overcoming major layout challenges. The decision to use single, external

CSS and JavaScript files was one of the most effective choices, as writing reusable functions and page-agnostic scripts resulted in a codebase that is clean, easy to debug, and highly maintainable. Ultimately, every technical decision, from providing custom modals instead of jarring browser alerts to ensuring the site was performant, was weighed against its impact on the end user.

To conclude, this project was an exercise in front-end web development that successfully met its goals while serving as a practical demonstration of modern HTML, CSS, and JavaScript techniques. It offered an immense opportunity to grow my skillset and I am certain that I will continue down this avenue of web design, learning more about the artistic side of digital aesthetics and building mastery over the technical functionality based elements.

List of Changes

General:

- HTML Form
- Several New webpages
- New File Structure Chart
- Updated Navigation map
- Updated Table of Requirements
- New sections within Design Document
- New Icon for website
- Extended commenting
- Updated Wireframes (and storyboards)
- Video Presentation
- Google sheets as a database

HTML:

- HTML form
- Website structure overhaul
- Implemented further classes for styling
- Further links, mailto
- Adjusted Organization of content for a more logical flow

CSS:

- Updated Website theme colours
- Updated Logo
- Navigation bar styling changed
- Background changed
- Page transition fade
- Icons implemented across various pages (e.g. Javascript logo)
- Glowing social media icon links in footer on hover
- Expandable images and modal/lightbox
- Removal of various unnecessary shadow effects
- Improved colour contrast
- Form styling
- Glitch Effect
- Updated Responsive Design
- Updated Print media queries

JS:

- Form Handling and Validation
- AppScript Functionality (for data processing)
- Navigation Highlighting
- Image Lightbox functionality

- Form submission handling

References

Favicon.ico & App Icon Generator. (n.d.). favicon.cc. Retrieved May 1, 2025, from https://www.favicon.cc/

Francis, B. (2022, August 2). How to build a game developer portfolio (that will get you hired). Game Developer. https://www.gamedeveloper.com/business/how-to-build-a-game-developer-portfolio-that-will-get-you-hired-

Interaction Design Foundation. (n.d.). What is User Experience (UX) Design? Retrieved May 4, 2025, from https://www.interaction-design.org/literature/topics/ux-design

MDN Web Docs. (n.d.-a). CSS: Cascading Style Sheets. Retrieved April 30 2025, from https://developer.mozilla.org/en-US/docs/Web/CSS

MDN Web Docs. (n.d.-b). HTML: HyperText Markup Language. Retrieved May 2, 2025, from https://developer.mozilla.org/en-US/docs/Web/HTML

MDN Web Docs. (n.d.-c). JavaScript. Retrieved May 3, 2025, from https://developer.mozilla.org/en-US/docs/Web/JavaScript

World Wide Web Consortium (W3C). (n.d.). W3C CSS Validation Service. Retrieved May 5, 2025, from https://jigsaw.w3.org/css-validator/

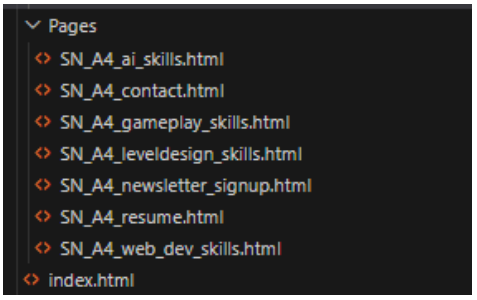
Nielsen Norman Group. (n.d.). Portfolio Design. Retrieved May 1, 2025, from https://www.nngroup.com/topic/portfolios/

W3Schools. (n.d.). HTML Favicon. Retrieved May 2, 2025, from https://www.w3schools.com/html/html_favicon.asp

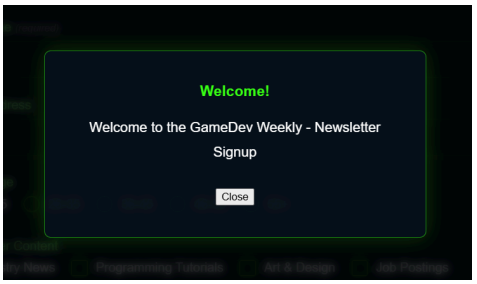
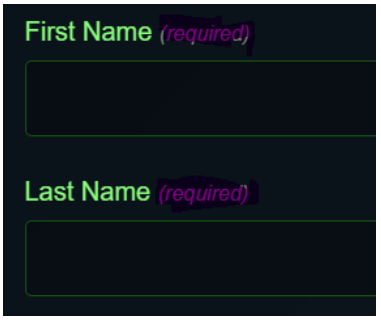
Wilson, J. (2023, October 10). 10 essential tips for creating a standout game developer portfolio. Creative Bloq. https://www.creativebloq.com/features/game-developer-portfolio

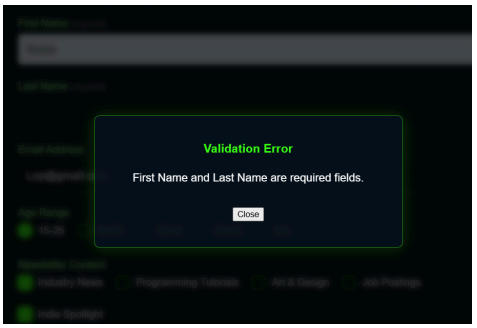
All HTML & CSS was completed using Visual Studio Code (VSC): https://code.visualstudio.com/

Tables of Requirement

General Requirements	Evidence	Status
There ought to be at least six (6) pages on the website in A4. There should be - at a minimum		Completed
a Home page (index.html): it introduces you to possible employers		Completed
Resume page: listing personal details, education, experience (work), qualifications, hobbies, and other relevant information		Completed
at least four pages with examples of your skills related to your company. One of these skills pages must detail your Web Development skills.		Completed
You will write code in CSS, HTML and JavaScript to create an HTML5 form for a fictitious company that delivers services in one of the ten sectors mentioned in the A1 (i.e. decided by your student ID). The form allows the users to sign up for an email newsletter.		Completed
Video Presentation	**In here**	Completed
References	**Design Document**	Completed

HTML Form	Evidence	Status
The form should be styled by using pure CSS code	**See above.**	Completed
The form should open in a new window (or tab) from your Javascriptpage.html	**See Video Presentation. The form**	Completed

page.	**opens from the website's "web skills" page. This was approved as a valid avenue by the coordinator.**	
Form data will be submitted to http://csusap.csu.edu.au/cgi-bin/echo_form using the post method for action to process the data.	**Form data is submitted to external google sheets (attached in design doc). This was approved as a valid alternative to csuasp cgi bin echo.**	Completed
Hosting is mandatory. You may use github.com, https://spaces.w3schools.com/ or any other free hosting sites if CSUSAP has any technical issues.	**Hosted on Github.** **https://suryanair1002.github.io/SuryaNair1002/**	Completed
The welcome message "Welcome to the XYZ(use a fictitious name) Corporation/Ltd.- Newsletter Signup" is displayed through an alert when the signup page is onload.		Completed
The form should include instructive information, such as 'required data'.		Completed
The form includes A text box of maximum size 60 for 'First Name' A text box of maximum size 60 for 'Last Name' An email address box of maximum size 60 for 'Email Address' 5 radio buttons for the following age ranges: 15-25, 26-35, 36-45, 46-55 and 55+ 5 checkboxes for 5 different services/sectors/branches/areas of operation for your assigned company. A text area of a minimum size of 60 cols and 3 rows. A submit button and a reset button.	**See website and/or web page files**	Completed

onsubmit, the form will verify that both the name fields are NOT empty - if a field is empty then an alert message is displayed, and the form is prevented from submitting to the server.		Completed
Ensure you provide enough comments alongside your code.	**See webpage files**	Completed

Design Document Specifications	**Status**
Preliminary Design	Completed
Interface Design	Completed
Navigation Map	Completed
Structure chart	Completed
Storyboards	Completed
Testing Plan	Completed
Reference List	Completed